

University of Oxford

College of Computer Science

Elzein Salah Elzein

22/12/2021

PHD Report Draft

Written By Elzein Salah Elzein

Robotic Disassembly as a Key Enabler of Remanufacturing

	Contents (Section)	Page
1.	Introduction	2
2.	Literature Review	108
3.	Remanufacturing	272
4.	Research Overview	396
5.	Methodology	437
6.	Related Work	480
7.	Discussion	681
8.	Conclusion and Future Work	833
9.	References	850

1. Introduction

1 What is Robotic Process Automation?

Nowadays, there is almost no aspect of our lives that is unaffected by automation. Some examples include washing machines, microwave ovens, autopilot mode for automobiles and airplanes, Nestlé using Robots to sell coffee pods in stores in Japan, Walmart testing drones to deliver products in the US, our bank checks being sorted using **Optical Character Recognition (OCR)**, and ATMs.

The term automation is derived from the Greek words *autos* meaning self, and *motos*, meaning moving. It is believed to have been coined in the 1940s when there was an increased use of automated devices in mechanized production lines in the Ford Motor Company.

Automation, in simple words, is technology that deals with the application of machines and computers to the production of goods and services. This helps in getting work done with little or no human assistance.

With the advent of computers, many software systems were developed to accomplish tasks that were previously done on paper to manage businesses, or not being done at all due to the lack of tools. Some of these are bookkeeping, inventory management, and communications management.

There is also a type of software that ties these systems and people together in workflows, known as **Business Process Management (BPM)** tools. This software has been developed for areas such as record systems, engagement systems, insight systems, and innovation systems. These mostly replicate processes in real-life scenarios.

In the digital world, automation and software development are two different terms. Very often, however, one is confused with the other. If some portion of a workflow can be programmed to be done without human intervention, it can be called automation. For example, in order to pass any invoice in a payment system, Ms. Julia at ABC organization needs to check that goods have been delivered and recorded in an inventory management system. This is a cumbersome job, as it has to be done for each and every invoice. Also, for larger organizations, more people are needed to do this check on computers. However, Jack, an application developer, proposes that he can integrate those two systems using database integration techniques. He will write a procedure that will fetch data from the inventory management system and automate the check of receivables.

Developing an inventory management software system is called software development, while programming a step so that no more human intervention is required is called automation.

In this chapter, you will learn about the basic concepts of automation and Robotic process automation.

Scope and techniques of automation

There are various techniques used and available to automate steps and processes in an organization where software systems are being used to accomplish certain tasks. Before we look at these techniques however, let us see what *can* be automated and what *should* be automated.

What should be automated?

There are a few aspects that have to be taken into consideration for choosing automation candidates. The following processes should be automated:

- Repetitive steps
- Time-consuming steps
- High-risk tasks
- Tasks with a low-quality yield

- Tasks involving multiple people and multiple steps
- And everything else!

We have found out what should be automated. Now the question arises what *can* be automated?

What can be automated?

In order to automate something, it needs to have the following characteristics:

- Well defined and rule-based steps
- Logical
- An input to the task can be diverted to the software system
- Input can be deciphered by software systems with available techniques
- The output system is accessible
- Benefits are more than the cost

Techniques of automation

There are various techniques available for automation and programmers have been using them for years to increase efficiency in enterprises:

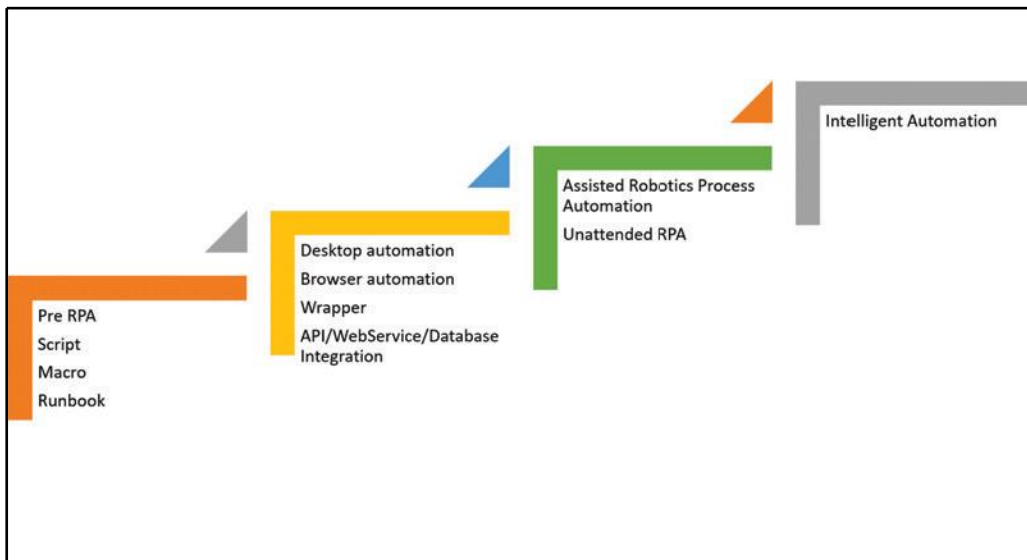
- **Custom software:** Developing new software to perform repetitive tasks.
- **Runbook:** Runbooks are typically used for IT-based operations. They are a compilation of a set of commands or tasks that are performed for maintenance and other types of activities. Runbooks can be offline as well, often referred to as run commands for performing sets of tasks.
- **Batch:** Batch files used to very popular. They used to compile a sequence of commands that could be run by a single click or command. They can also be scheduled to be run at a specific time using the scheduler.
- **Wrapper:** Wraps around existing software or hosts client applications. The wrapper monitors activities in a client app and performs actions based on rules.

For example:

Putting validation on top of a mainframe application using hummingbird

Hosting a website inside a shell, navigation, and actions

- **Browser automation:** Greasemonkey and many other web macro software helped in browser-based automation. It can be used to read from a website and save to a database. It can also write to fields based on rules. Using this technique, a whole website can be changed, and components can be added or removed from the website. Sometimes, it is also referred to as web scripting or web injections.
- **Desktop automation:** Traditionally, desktop automation used to mean that multiple screens on a desktop were woven together to present a single screen, and if there was some data transfer from one screen to another, it could be done automatically. Recently, assisted Robotics process automation has also been considered for desktop automation by some companies.
- **Database/web service integration:** In database integration, we read/write to a client database directly. In web service integration, we communicate with the client system using a web service:



Robotic process automation

Today, automation has reached a stage of maturity where a number of other technologies have developed from it. **Robotic process automation (RPA)** is one such transformational burgeoning area. *Robot* in Robotic process automation means software programs that mimic human actions.

In simple words, RPA involves the use of software that *mimics* human actions while interacting with applications in a computer and accomplishing *rule-based tasks*. This often requires reading from and typing, or clicking on existing applications that are used to perform the given tasks.

In addition, these software Robots also perform complex calculations and decision making on the basis of the data and predefined rules. With the rapid progress of technology and renewed efforts in the area of artificial intelligence, it has become possible to use

State activity: Transitions contain three sections—**Trigger**, **Condition**, and **Action**, which

enable you to add a trigger for the next state or a condition under which an activity is to be executed. with RPA to accomplish tasks that were not possible earlier. Some of the technologies being adopted with RPA are as follows:

- Machine learning
- Natural language processing
- Natural language generation
- Computer vision

With the inclusion of the preceding technologies, sometimes it is also referred to as intelligent automation.

With the advent of RPA, it has become much easier to automate tasks. Now, we need to know only the steps taken by humans and make the Robots mimic the action on a computer screen using mouse and keyboard. This is a big deal because in most cases, the process is already defined and the steps documented. Humans also follow the same operating procedures, which define the steps taken to accomplish the task. Business logic, validation of data, transformation, and use of data is already coded in existing systems that humans use to accomplish a task, a simple example being invoice data entry.

RPA platforms allow the program, called Robots, to interact with any application in the same way a human would do, hence, automating rule-based work by recording those steps for later playback.

An important point that distinguishes RPA from traditional automation is that the software Robot is *trained* using steps that are illustrative rather than using instructions based on code. Thus, a person with little programming experience can be trained on these platforms to automate simple to complex processes.

Also, RPA software, unlike traditional automation, is capable of adapting to dynamic circumstances, for example, when checking an electronic form of new employees in a company. If the pin code is missing in a form, in traditional automation the software would point out the blank field as an exception, and then a human being would search for the relevant pin code and correct the form. In RPA however, the software is capable of performing all the tasks mentioned previously with no human assistance.

From tedious, repetitive, and high volume tasks, to diverse, complicated systems that need to work together lucidly, RPA can handle it all. There is consistency in quality, accuracy, productivity and efficiency, faster delivery of services, and of course, lower operation costs.

With the constant development and integration of RPA with industries, people previously engaged in mundane, repetitive tasks can now move on to engaging themselves in higher value, better quality activities, leaving the tedious tasks to the software Robots.

What can RPA do?

Today, RPA has matured beyond doing mundane repetitive tasks, and is seen as a transformational technology that can bring tremendous value to the organization adopting it. The ability to create full audit trails is significant for improving the quality of work being done and eliminating human error. Once trained, these Robots will perform tasks with the same precision over and over again. These Robots can interact with applications irrespective of the technologies on which the applications are built. They can work with popular ERP applications such as SAP, Oracle, or Microsoft Dynamics, and BPMs such as Pega systems and Appian.

Custom applications built on .NET, Java, the command-line, or mainframe terminal are easy to use with RPA.

With the inclusion of AI technologies, RPA now has the capability to read from images or scanned documents, and it can interpret unstructured data and formats as well. However, most of the implementation is happening with structured and digital data.

Benefits of RPA

Today, RPA is being widely accepted across industries and across the world. The following industries can benefit a lot from RPA:

Business process outsourcing (BPO): With RPA and its benefits of reduced costs, the

- BPO sector can now depend less on outsourced labor.

- **Insurance:** The complexity and number of tasks that must be managed in the insurance sector, from managing policies, to filing and processing claims across multiple platforms, provides an ideal environment for the use of RPA technology.
- **Financial sector:** From day-to-day activities and handling an enormous amount of data, to performing complex workflows, RPA has been helping to transform this sector into an efficient and reliable one.
- **Utility companies:** These companies (such as gas, electricity, and water) deal with a lot of monetary transactions and can leverage RPA to automate tasks such as meter reading, billing, and processing customer payments.
- **Healthcare:** Data entry, patient scheduling, and more importantly billing and claims processing, are important areas where RPA can be used. RPA will help in optimizing patient appointments, sending them automatic reminders of their appointments and eliminating human error in patient records. This leaves workers to focus more on the needs of the patients, and also leads to improved patient experience.

The following are the benefits of RPA:

- **Higher quality services, greater accuracy:** With reduced human error and greater compliance, the quality of work is much better. Also, while it is difficult to trace the point at which the human error occurred, the detection of errors is much simpler in RPA. This is because every step in the automation process is recorded, making it faster to pinpoint errors with ease. A reduction, or removal of, errors also means greater accuracy of data, leading to better quality analytics and hence better decision making.
- **Improved analytics:** Since these software Robots can log each action taken with the appropriate tag and metadata, it is very easy to get business insights and other analytical data. Using analytics on the collected data such as transaction received time, transaction complete time, and predictions can be made for the incoming volume and ability to complete the tasks on time.

Reduced costs: Nowadays, it is commonplace to hear that one Robot is equivalent to three human **full-time executives (FTE)**. This is based on the simple fact that one FTE works for eight hours a day, while a Robot can work for 24 hours without a break. Increased availability and productivity means the cost of operations is reduced tremendously. The speed of the work being performed coupled with multitasking results in further reductions in cost.

- **Increased speed:** Robots are very fast and sometimes the speed of execution has to be reduced to match the speed and latency of the application on which these Robots work. Increased speed can result in better response times and an increase in the volume of the tasks being performed.
- **Greater compliance:** As mentioned earlier, a full audit trail is one of the highlights of RPA and can result in greater compliance. These Robots do not deviate from the defined set of steps to be taken while doing a task and hence it will certainly result in better compliance..
- **Agility:** Reducing and increasing the number of Robot resources requires managing the volume of the business process. This is just a click away. More Robots can be deployed to perform the same task easily. Redeployment of resources does not require any kind of coding or reconfiguration.
- **Comprehensive insights:** In addition to the audit trail and time stamping, Robots can tag transactions to use them later, in reports for business insight. By using these business insights, better decisions can be made for the improvement of the business. This recorded data can also be used for forecasting.
- **Versatility:** RPA is applicable across industries performing a wide range of tasks—from small to large businesses, simple to complex processes.
- **Simplicity:** RPA does not need prior programming knowledge. Most platforms provide designs in the form of flowcharts. This simplicity enables easy automation of business processes, leaving the IT professionals relatively free to carry out higher value work. Also, since automation is carried out by people from within the department or area of work, no requirements are lost in translation between the business unit and the development team, which may have happened otherwise in traditional automation.
- **Scalability:** RPA is highly scalable, up as well as down. Whether one requires an increase or reduction in the virtual workforce, Robots can be quickly deployed at zero or minimum costs while maintaining consistency in the quality of work.

- **Time savings:** Not only does the virtual workforce complete large volumes of work in a shorter span of time with precision, but they help save time in another way too. If there is any change—say, a technology upgrade it is much easier and faster for the virtual workforce to adapt to the changes. This can be done by bringing about modifications in the programming or introducing new processes. For humans, it is difficult for them to learn and get trained in something new—breaking from the old habit of performing repetitive tasks.
- **Non- invasive:** RPA, as we know, works at the user interface just like a human would. This ensures that it can be implemented without bringing changes to the existing computer systems. This helps in reducing risks and complexities that would arise in the case of traditional IT deployments.
- **Better management:** RPA allows for managing, deploying, and monitoring Robots through a centralized platform. This also lessens the need for governance.
- **Better customer service:** Since Robots can work around the clock, capacity increases. This leaves humans to focus on customer service and satisfaction. Also, better quality of services delivered to customers at faster speeds greatly boosts customer satisfaction.
- **Increased employee satisfaction:** With repetitive, dreary tasks now being taken over by the virtual workforce, employees are not just relieved of their workload, but can also engage in better quality work that requires the use of human capabilities and strengths such as emotional intelligence, reasoning, or tending to customers. Thus, RPA doesn't take away work, it just frees humans from tedious, mind-numbing work, giving us an opportunity to engage in much more satisfying jobs.

The applicability of RPA is across industries such as banking and financial services, insurance, healthcare, manufacturing, telecom, travel, and logistics. There is also penetration in industries such as consumer products, food and beverages, and entertainment.

There are horizontals irrespective of industries' domain that have seen a higher adoption, such as in finance and accounting, human resources, and procurement.

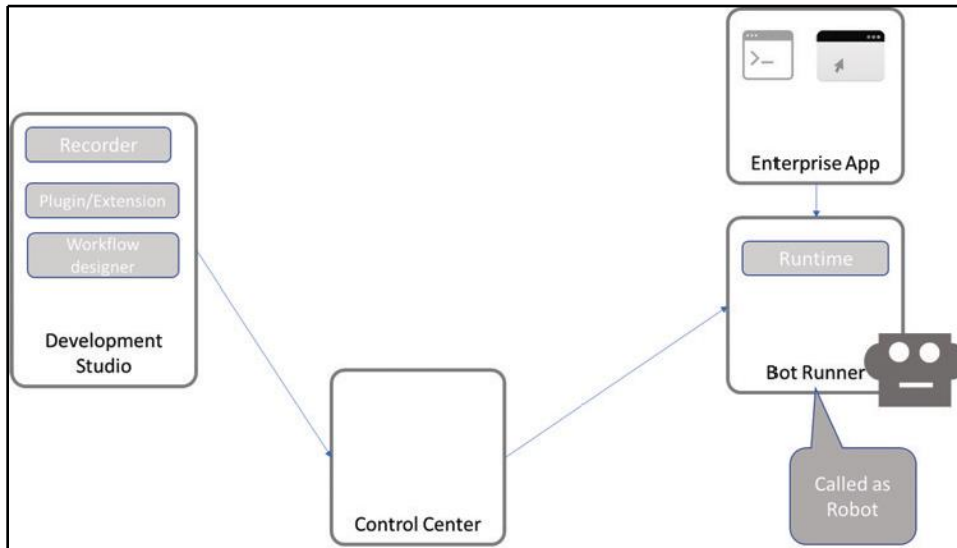
Most successful implementations are observed in what is being called *industry-specific processes* or *domain-specific processes*, for example, *claims processing* in the insurance industry.

Components of RPA

Any Robotics process automation platform provides some basic components, which together build the platform.

The following are the basic or core components of RPA:

- **Recorder**
- **Development Studio**
- **Plugin/Extension**
- **Bot Runner**
- **Control Center:**



Recorder

The recorder is the part of the development studio that developers use to configure the Robots. It is like the macro recorder in Excel, the bot recorder in any platform, records steps. It records mouse and keyboard movements on the UI and this recording can be replayed to do the same steps again and again. This enables rapid automation. This component has played a very big role in the popularity of RPA. We will see the application and usefulness of this component in [chapter 2, Record and Play](#).

Development studio

The development studio is used by developers to create Robot configuration or train the Robots. Using the development studio, a set of instructions and decision-making logic is coded for Robots to execute. Some platforms provide flow-charting capabilities such as Visio, so it becomes very easy to plot steps in a process, whereas some other platforms require coding. In most studios, in order to do commercial development, developers need to have a fair amount of knowledge of programming, for example, loops, if else, variable assignment, and so on. We will study the UiPath Development Studio in detail in [Chapter 2, *Record and Play*](#).

Extensions and plugins

Most platforms offer many plugins and extensions to ease the development and running of bots. In many applications, such as Java SAP, it is not easy to individually identify controls of the UI through traditional techniques. RPA vendors have developed plugins and extensions to help with these issues. We will get to understand importance of plugins and extensions of the UiPath platform later in the book.

Bot runner

This is also referred to as the Robot, other components make it run.

Control center

The objective of the control room is to provide Robot management capabilities. It monitors and controls a Robot's operation in a network. It can be used to start/stop Robots, make schedules for them, maintain and publish code, redeploy Robots to different tasks, and manage licenses and credentials.

RPA platforms

The burgeoning RPA vendor market has been showing continual and steady growth. While the largest market is the US, followed by the UK, the market in **Asia Pacific Countries (APAC)** is also showing considerable progress. Successful pilot projects and increased customer satisfaction among the early adopters of RPA will encourage new players to adopt this technology. There is growing demand for RPA, especially in industries that need large- scale deployments. The major markets for RPA are banking and finance, healthcare and pharmaceuticals, telecom and media, and retail.

A few key vendors, their client market, and company specifications are mentioned in the following sections.

Automation Anywhere

Automation Anywhere helps to automate business processes for companies. They focus on RPA, cognitive data (machine learning and natural language processing), and business analytics. Their bots are capable of handling both structured as well as unstructured data. The system has three basic components:

1. *A development client* for the creation of a bot
2. *A runtime environment* for the deployment of a bot
3. *A centralized command system* for handling multiple bots, analyzing their performance:
 - **HQ:** San Jose, California, USA
 - **Est:** 2003
 - **CEO:** Mihir Shukla
 - **Some key clients:** Deloitte, Accenture, AT&T, GM, J P Morgan Chase
 - **Source of revenue by region:** Its highest source of revenue is the USA, which accounts for more than half its revenue, followed by APAC, then UK and continental Europe
 - **Source of revenue by industry:** The **Banking, Financial services, and Insurance (BFSI)** accounts for more than half of its revenue, followed by healthcare, telecom, media, and others

UiPath

UiPath is an RPA technology vendor who designs and delivers software that helps automate businesses. The RPA platform consists of three parts:

- *UiPath Studio* to design the processes
- *UiPath Robot* to automate tasks designed in UiPath Studio
- *UiPath Orchestrator* to run and manage the processes:
 - **HQ:** Bucharest, Romania
 - **CEO:** Daniel Dines
 - **Key Clients:** Atos, AXA, BBC, Capgemini, CenturyLink, Cognizant, Middlesea, OpusCapita, and SAP
 - **Source of revenue by region:** North America, Continental Europe, the UK, and APAC
 - **Source of revenue by industry:** BFSI, healthcare, telecom and media, and retail

Blue Prism

Blue Prism aims to provide automation that enterprises can use according to their needs. Blue Prism aims to do this by providing automation that is scalable, configurable, and centrally managed. It sells its software through its partners, some of which are Accenture, Capgemini, Deloitte, Digital Workforce Nordic, HPE, HCL, IBM, TCS, Tech Mahindra, Thoughtonomy, and Wipro:

- **HQ:** United Kingdom
- **Est:** 2001
- **CEO:** Alastair Bathgate

- **Key Clients:** BNY Mellon, RWE npower, and Telefonica O2
 - **Source of revenue by region:** More than half of its revenue source comes from the UK, followed by North America, Continental Europe, and APAC
- Source of revenue by industry:** BFSI, health, and pharmaceuticals, retail and consumer, telecom and media, manufacturing, public sector, travel, and transportation

WorkFusion

WorkFusion offers automation that is based on RPA and machine learning. It delivers software as a solution for automating high volume data. WorkFusion enables man and machine to work in tandem while managing, optimizing, or automating tasks:

- **HQ:** New York, USA
- **Est:** 2011
- **CEO:** Max Yangkelivich, Andrew Volkov
- **Key Clients:** Thomson Reuters, Infogroup, Citi, and Standard Bank
- **Source of revenue by region:** North America provides more than 80% of WorkFusion's revenue, followed by Europe, APAC, and MEA
- **Source of revenue by industry:** Around 90% of its revenue comes from the BFSI sector, followed by the retail and consumer sectors

Thoughtonomy

Thoughtonomy delivers software that helps automate business and IT processes. It uses Blue Prism and other automation software and customizes it:

- **HQ:** London, UK
- **Est:** 2013
- **CEO:** Terry Walby
- **Key Clients:** Atos, Fujitsu, CGI, Unite BT, and Business Systems
- **Sources of Revenue by region:** Around 70% of revenue comes solely from the UK. The rest comes from Continental Europe, North America, APAC, and the **Middle East and Africa (MEA)**
- **Sources of Revenue by industry:** A major part of its revenue comes from third-party clients, followed by BFSI, public sector, telecom, healthcare, retail, and consumer sectors

KOFAX

Kofax's Kapow RPA platform is capable of automating and delivering processes that are repetitive and rule-based. It uses Robots for extracting and consolidating information. The software platform consists of a management console to deploy and manage bots, Robot performance, and a monitoring system. This software can also group together high priority tasks that should be completed first by the Robot during times of high workload. Kofax's software, however, doesn't have machine learning:

- **HQ:** Irvine, California
- **CEO:** Paul Rooke
- **Key Clients:** Arrow Electronics, Delta Dental of Colorado, Pitt Ohio, Audi
- **Sources of revenue by region:** North America accounts for almost half of its revenue, followed by Continental Europe, APAC, and LATAM (Latin America)
- **Sources of revenue by industry:** BFSI, retail, consumer, travel, transportation, public sector, manufacturing, and healthcare

About UiPath

Headquartered in Bucharest, UiPath is an RPA vendor that provides software to help organizations automate their business processes. The company aims to do away with repetitive and tedious tasks, allowing humans to engage in more creative and inspiring activities.

UiPath was founded by Daniel Dines, who is the CEO. It has offices in London, Bucharest, Tokyo, Paris, Singapore, Melbourne, Hong Kong, and Bengaluru. With clients spread across the world, from North America to the United Kingdom, Continental Europe to Asia Pacific countries, the company has shown remarkable growth in the last year, both in terms of revenue and its workforce.

Today, its software is being widely used to automate business processes. However, the IT sector is also gradually embracing UiPath's software. Major clients of UiPath in the industry include BFSI, Telecom and media, healthcare, retail and consumer, and manufacturing.

With UiPath automation software, one can configure software Robots to mimic human action on the user interface of computer systems. The basic components of the UiPath RPA platform

are in line with what was explained in *Components of RPA*, these components are necessary for enterprise deployment. The components of the UiPath platform are UiPath Studio, UiPath Robot, and UiPath Orchestrator, see the following sections.

UiPath Studio

UiPath Studio helps users with no coding skills to design Robotic processes in a visual interface. It is a flowchart-based modeling tool. Thus, automation is faster and more convenient. Multiple people can contribute to the same workflow. The presence of a visual signal that points out errors in the model, and a recorder that performs what the user executes, make modeling much easier.

UiPath Robot

UiPath Robot runs the processes designed in UiPath Studio. It works in both attended (working only on human trigger) and unattended environments (self-trigger and work on their own).

UiPath Orchestrator

UiPath Orchestrator is a web-based platform that runs and manages Robots. It is capable of deploying multiple Robots, and monitoring and inspecting their activities.

The future of automation

Throughout the history of human civilization, there has been many major turning points in innovation and discoveries that have instilled awe as well as fear in the minds of people, so much so that the word *Luddite* (used for people who were strongly opposed to the introduction of textile mills during the *First Industrial Revolution* for fear of losing their livelihoods) has now become synonymous with all people who are against new technologies, be it industrialization, automation, or computerization.

The buzzword today is the **Fourth Industrial Revolution**—the current age where technology is embedded within societies and even the human body—be it Robotics, 3D printing, nanotechnology, Internet of Things, or autonomous vehicles. This will fundamentally change the way we live, work, and interact with one another.

Technological changes and innovations are taking place today at an unprecedented pace and scope and are having an impact on many disciplines. Technological innovation has reached a stage where machines have now entered the realm of what was once considered exclusively human. For these reasons, there is a wide section of people who fear this age of Robots. While the arguments over how much of our lives will be taken over by Robots are endless, what cannot be denied is that Robots are here to stay.

There are various advantages of automation today; there are also fears surrounding its advancement, which are not completely unfounded.

As mentioned earlier, this time automation is capable of impacting a wide range of disciplines. Thus, unlike in the past where only blue collar jobs were at risk of being replaced by machines, this time even white collar jobs are believed to be at risk. While this is not untrue, reports suggest that only around 5% of the total jobs may be *totally* replaced by automation. For other jobs, automation will only replace a part of the job and not completely take over.

There are, of course, those jobs in the 5% category that run the risk of being completely automated. These are the jobs that are routine, repetitive, and predictable. A few examples are telemarketing, data entry operation, clerical work, retail sales, cashiers, toll booth operators, and fast food jobs.

However, like in the past, people should be able to find a way to adapt to the changes. With each generation, humans become smarter, more adaptable to change, and also progressive. Also, with automation mostly taking over routine and tedious tasks, humans are provided the opportunity to make better use of their capabilities—be it reasoning, emotional intelligence, or their creativity. What we can do is not fret over the inevitable rather prepare for it. One way of doing so is to start changing the pattern of education. The next generation should be taught how to recognize and adapt to changes quickly. An important aspect of their education should be to *learn how to learn*.

Summary

In this chapter, we acquired a basic understanding of RPA and the ability to differentiate it from other types of automation. We have also learned globally about the benefits of RPA and various platforms available on the market. In the next chapter, we will start learning UiPath and the recording tool, which is a quick and efficient way of implementing RPA.

2 Record and Play

The facility of recording user steps on a computer and playing them back has made **Robotic Process Automation (RPA)** highly successful. Without this feature, the adoption of the technology might have been very slow and it would have been seen as another automation/scripting tool.

In the previous chapter, we gained a basic understanding of Robotic process automation. In this chapter, we will see how to use the recorder as the first step of automation in our journey. Before that, let us understand the UiPath tool and learn how to install it (you can only use the recorder after installing it). We will cover:

- The UiPath stack and components of the platform.
- How to download and install UiPath components.
- Understanding the Project Studio in detail. The Project Studio is the place where developers spend most of their time configuring the Robots.
- The recorder, with two step-by-step examples to quickly master record and play.

UiPath stack

In order to make the UiPath platform fully operational at an enterprise level, there are various components that need to be in place. There are three basic components in UiPath:

1. UiPath Studio
2. UiPath Robot
3. UiPath Orchestrator

The UiPath platform is available in two variations:

1. **Enterprise Edition:** This edition is suitable for large companies starting their RPA projects and looking to scale their Robot deployments in the future. It is integrated with UiPath Orchestrator (we will discuss UiPath Orchestrator later). This version can be updated by visiting the UiPath website and by downloading the newest version of the UiPath platform installer. Running the installer automatically replaces all the old files without modifying any of your settings.
2. **Community Edition:** This is suitable for individual developers and small organizations with fewer employees. The Community Edition is always up-to-date, and it automatically updates itself as soon as a new version is available.



The Community Edition can be used to learn UiPath free of cost.

Now, let us understand all three components in detail.

UiPath Studio

UiPath Studio is the development environment of UiPath. It is the primary tool to develop UiPath Robots.

It can be used to configure steps of a task or launch a full recorder to record a sequence of steps. The recording facility in the Studio is a game-changing feature for RPA tools. Its simplicity lets even nontechnical business users design/record steps of a process.

This studio lets the user configure Robots, that is, develop steps to perform tasks visually. Most of the configuration and coding in UiPath is visual. By using the drag-drop facility from the toolbox, you may write a whole sequence of workflows to perform a set of tasks by Robots. These steps look like a data flow diagram and are very easy to understand. It is one of the simplest visual flow diagramming tools. Most of the time, in an enterprise environment you will receive process maps to understand the flow of work, which you will use to develop Robots. The studio gives the same look and feel as a workflow. The designer gives you full control of the

execution order and actions taken, also known as activities. An activity or action includes clicking a button, writing and reading a file, and so on.

UiPath Robot

UiPath Robot is a Windows service that can open interactive/non-interactive window sessions to execute processes or a set of steps, developed or recorded using UiPath Studio. Sometimes, it is also called an execution agent as it executes automation projects, or a runtime agent as it executes instructions generated by developing or recording processes in UiPath Studio. The most acceptable nomenclature is *Robot*.

These Robots can be controlled by Orchestrator, which is part of the Enterprise Edition. There is an option at installation to de-link these Robots from Orchestrator and work independently on the desktop. In most of our examples, we will refer to the Community Edition, which does not have Orchestrator, and the installed Robot will work independently in user mode.

When installed in user mode, these Robots have the exact same rights as the user. If you opted for Orchestrator, you can control Robots irrespective of whether it is installed on a user machine, in user mode, or on a server.

Types of Robots

The following are types of Robots:

- **Attended:** It operates on the same workstation as a human to help the user accomplish daily tasks. It is usually triggered by user events. *You cannot start a process from Orchestrator on these type of Robots, and they cannot run under a locked screen.*
- **Unattended:** It can run unattended in virtual environments and can automate any number of processes. In addition to the Attended Robot's capabilities, this Robot is responsible for remote execution, monitoring, scheduling, and providing support for work queues.
- **Free:** It is similar to Unattended Robots, but can be used *only* for development and testing purposes, not in a production environment.

These types of Robots are built to suit different automation needs, such as back or front office scenarios.



A **Front Office Robot** is another name for an Attended Robot. These Robots monitor actions along with humans. These Robots take over the work for programmed steps on a certain event such as pressing a shortcut key. There is an interaction between humans with the Robots/system to do the required work. On the other hand, Back Office Robots are Robots that have the capabilities to run unattended. They can run without interaction with humans.

UiPath Orchestrator

UiPath Orchestrator is a server-based application that lets you orchestrate your Robots, hence the name Orchestrator. It runs on a server and connects to all the Robots within the network, whether Attended, Unattended, or Free. It has a browser-based interface that enables the orchestration and management of hundreds of Robots with a click. Orchestrator lets you manage the creation, monitoring, and deployment of resources in your environment, acting in the same way as an integration point with third-party applications.

Orchestrator's main capabilities:

- It helps in creating and maintaining the connection between Robots It
- ensures the correct delivery of the packages to Robots
- It helps in managing the queues
- It helps in keeping track of the Robot identification It
- stores and indexes the logs to SQL or Elasticsearch

Behind the scenes, Orchestrator Server uses:

- IIS Server
- SQL Server
- Elasticsearch
- Kibana

We have the option to store information or logs in an SQL database or Elasticsearch (which is based on Lucene). If the size of the data is small, then SQL will be preferred. However, if we have a large chunk of data and we also need some analysis on that data, it becomes tedious. Here, text search engine tools such as Lucene come into play.



Lucene is a free and open source information retrieval software library, originally written in Java. It is a full-text search library that makes it easy to add search functionality to an application or website.

Now, you may be wondering what the role of Elasticsearch is.

Elasticsearch is built on top of the Lucene engine. It is a JSON-based architecture that can support the REST API model. Elasticsearch sends all the queries to the Lucene engine for analyzing the text, and the query is returned back to Elasticsearch. This result, in turn, is sent back to the client in JSON format.

There is another tool that is frequently used with Elasticsearch, known as Kibana, which is an open source data visualization plugin for Elasticsearch. It provides visualization capabilities on the content indexed on an Elasticsearch cluster. It is used to create bar charts, line charts, scatter plots, pie charts, and maps with large volumes of data.

The following modules exist in UiPath Orchestrator and help in managing the Robots:

- Robots
- Processes
- Jobs
- Schedules
- Assets
- Queues

Downloading and installing UiPath Studio

To learn UiPath, you need the software. Fortunately, UiPath has provided multiple options to learn and use the platform. You may get a free trial for 60 days, which is the fully working Enterprise Edition, or you may opt for the Community Edition, which is free for learning purposes. Commercial use, however, is not allowed.



For commercial use, you need to buy licences from UiPath. To buy licences, please contact sales@UiPath.com. You can also contact me if you face any difficulty in obtaining commercial licences.

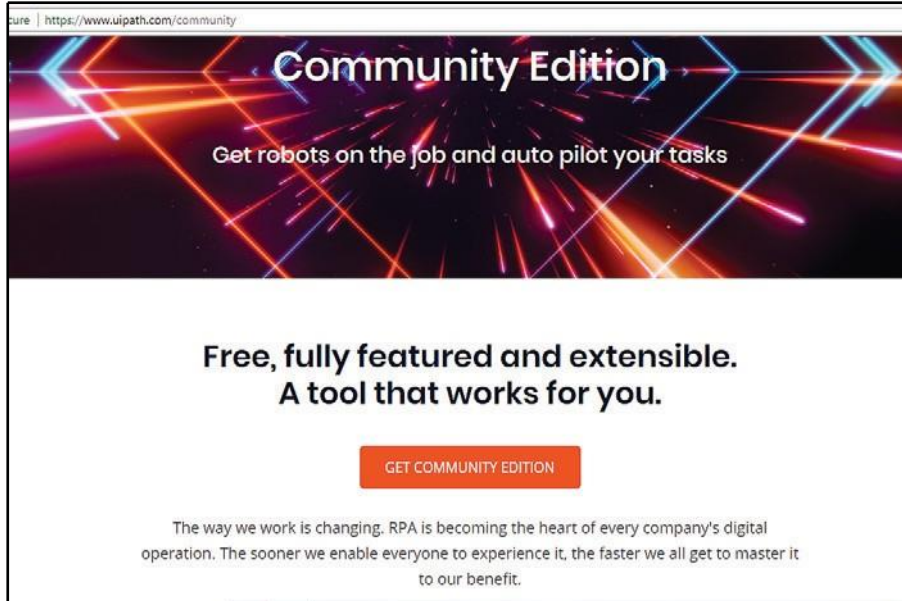
UiPath Community Edition is free to use in academia, nonprofits, and small businesses with an annual turnover of less than \$1 million or 250 workstations. (This may change over time, so please check license agreements at the time of downloading).

The UiPath Community Edition has the following features:

- Auto update
- No server integration
- Community forum for support
- Online self-learning
- No complex installation required
- Online activation is mandatory

To get your Community Edition of UiPath Studio, type the following link in your browser:
<https://www.UiPath.com/community>:

1. A **Community Edition** page opens. Click on **Get Community Edition**:



2. On the next page, you must register yourself in order to download the Community Edition. So, use the correct details and remember them because the same email will be used to activate the software. Fill in the following details: **First Name***, **Last Name***, and **Email***. Filling in the **Twitter User** field is not mandatory, but it is good to provide it:

For individuals and small professional teams.

First Name*

- Please complete this required field.

Last Name*

- Please complete this required field.

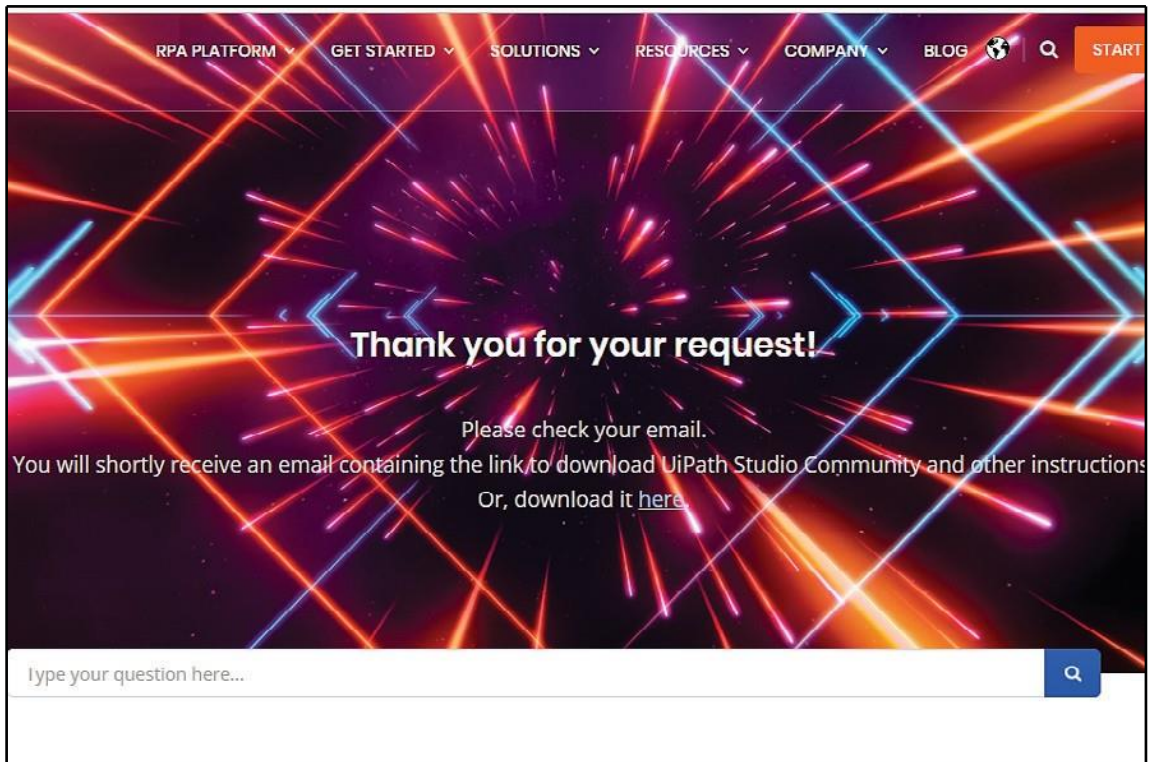
Email*

Twitter User

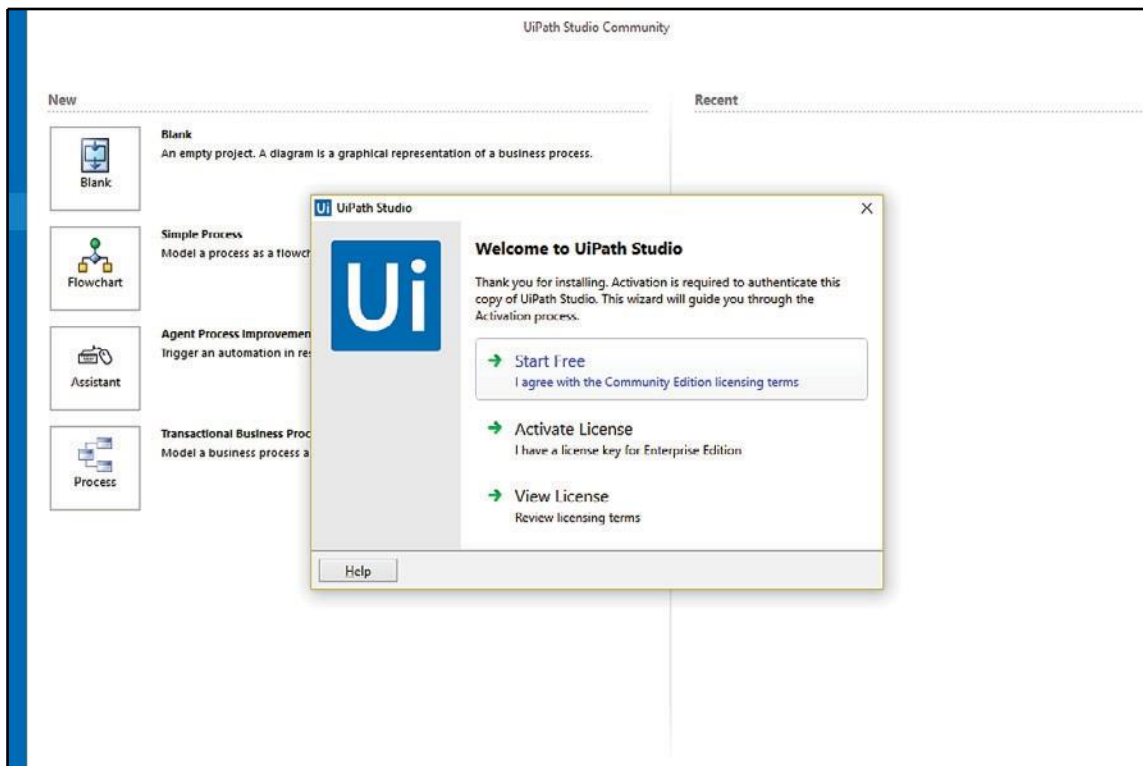
REQUEST COMMUNITY EDITION

Click on **REQUEST COMMUNITY EDITION**.

3. You will be directed to a page that requests you to check your email for downloading the link. Click on the link to download UiPath Studio. You may also directly download UiPath Studio. Just click on the word **here** in **download it here**, as shown in the following screenshot:



4. Once the download is complete, open the downloaded file, `UiPathStudioSetup.Exe`.
5. The installation will then begin. Once the installation is complete, a welcome message will be displayed. Click on the **Start Free** option:



6. Then, as requested, enter your **Email Address** once again and click on **Activate**. Please remember to use the same email ID that you used to download the software. This email ID will be bound to the computer. The activation will happen online. An offline activation option is not available for the Community Edition.
7. A message will then be displayed on the screen informing you of the successful installation. Close this window.



For more convenient use, you can pin it to your taskbar immediately; otherwise, you may have to unnecessarily search for `UiPath.exe` in your computer every time you wish to use it.

Your UiPath Studio is now ready for use!

Learning UiPath Studio

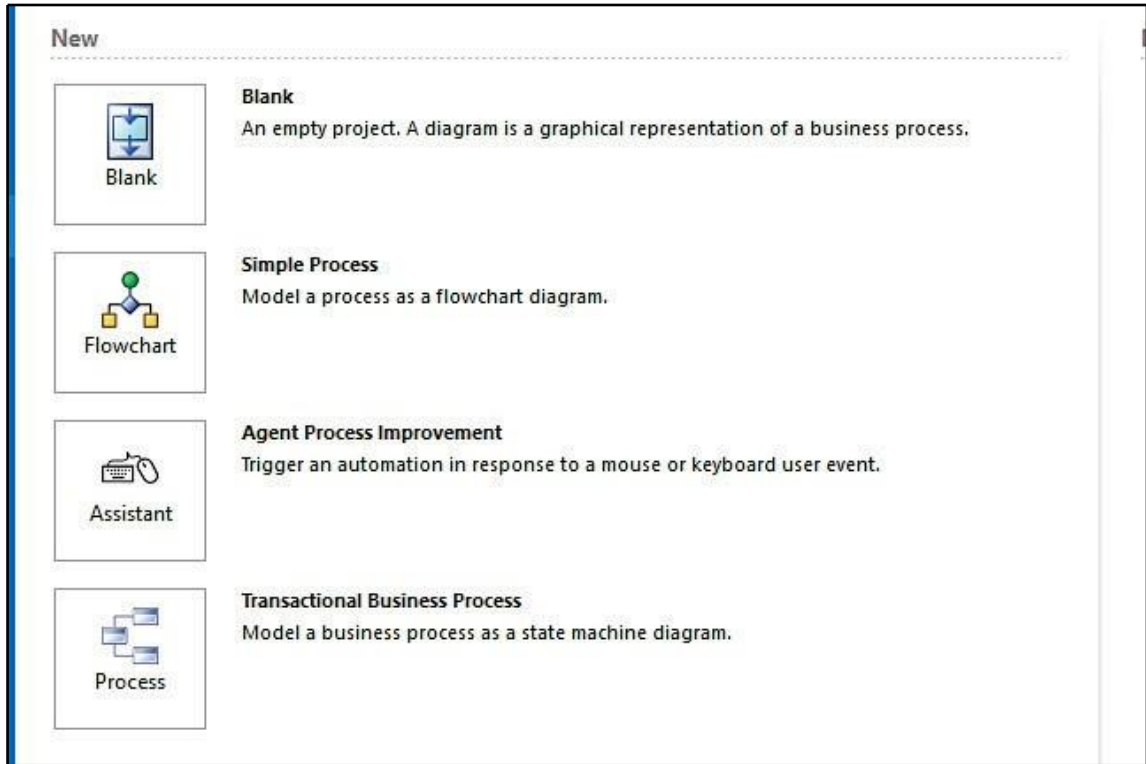
The **UiPath Studio** platform helps to design Robotic processes with a visual interface. Automation in UiPath Studio requires no or very little prior programming knowledge. It is a Flowchart-based modeling tool. Thus, automation is faster and more convenient. The presence of a visual signal that points out errors in the model, along with the recorder, which performs what users execute, makes modeling much easier.

We will study UiPath Studio in detail now. First and foremost, let us understand the types of project available and which should be used when.

Projects

The main types of project supported by UiPath Studio are as follows:

- **Sequence:** This is suitable for simple actions or tasks. It enables you to go from one activity to another, without interfering with your project. It consists of various activities. Creating sequences is also useful for debugging purposes. One activity from a particular sequence can easily be tracked. The Basic type of project can be started using the **Blank** option in the start tab and then adding the sequence in the diagram from the toolbox.
- **Flowchart:** This is suitable for dealing with more complex projects. It enables you to integrate decisions and connect activities. To start this kind of project, choose the **Flowchart - Simple Process** option from the new project menu.
- **Assistant:** This is suitable for developing attended or Front Office Robots: sometimes these Robots are called assistants. To start this kind of project, choose the **Assistant - Agent Process Improvement** option from the new project menu.
- **State machine:** This is suitable for very large projects that use a finite number of states in their execution, triggered by a condition. To start this kind of project, choose the **Process - Transaction Business Process** option from the new project menu:



Please remember, the four types of project mentioned in the preceding screenshot are only available in the **Start** tab of the studio. However, if you click on the **New** option in the DESIGN tab, you only get three options:

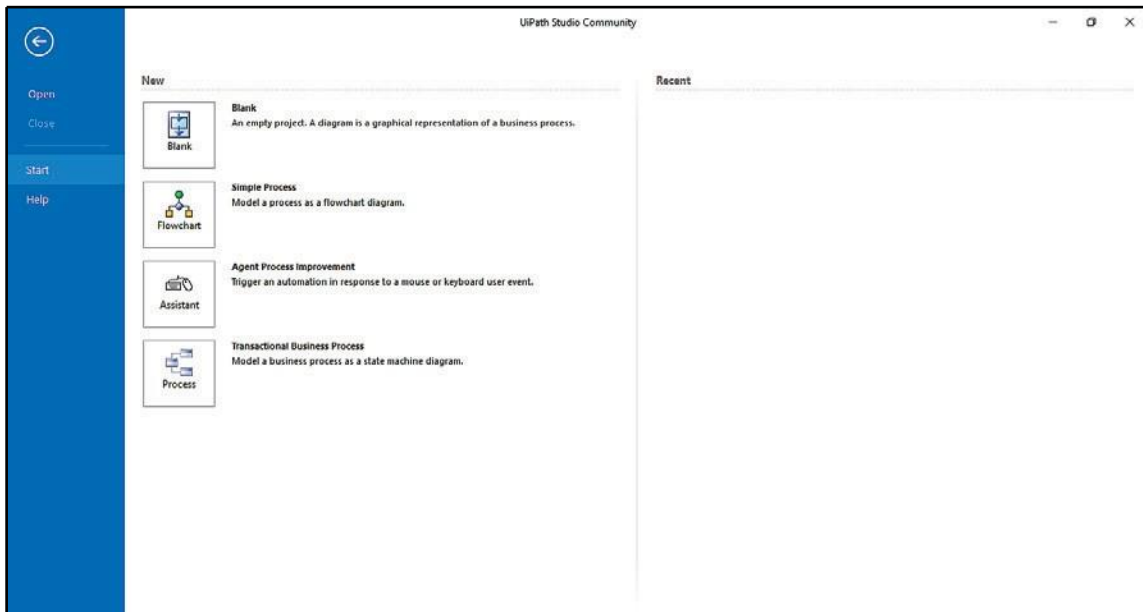
- Sequence
- Flowchart
- State Machine
-

The preceding options selected from the DESIGN tab's **New** menu become part of an existing project and are referred to as a diagram.

UiPath Studio basically helps in automating various tasks through the designing of projects. A project is a graphical representation of any rule-based business process. It is usually in Flowchart form. One can design projects by customizing and defining the various steps, known as activities, ranging from a simple click to entering particular data.

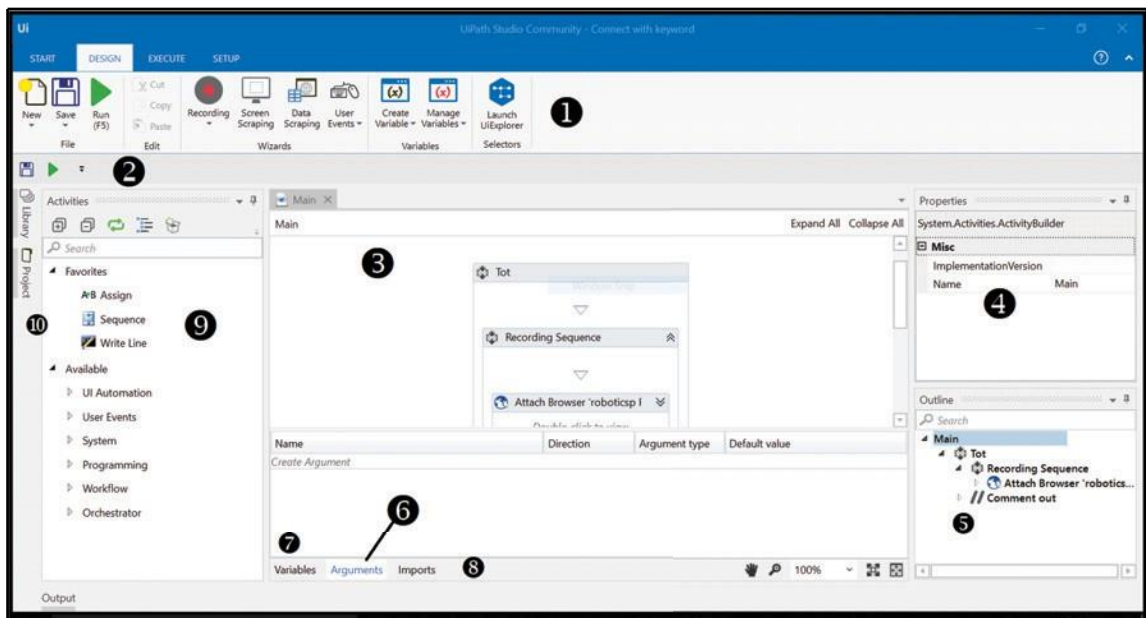
The user interface

When you first open UiPath Studio, you are directed to the page shown in the following screenshot:



Start tab of UiPath Studio

You can either open an old project or create a new one. Let us say we are making a new project. We click on **Blank** and name it. We will then be directed to a screen, which will display the following:

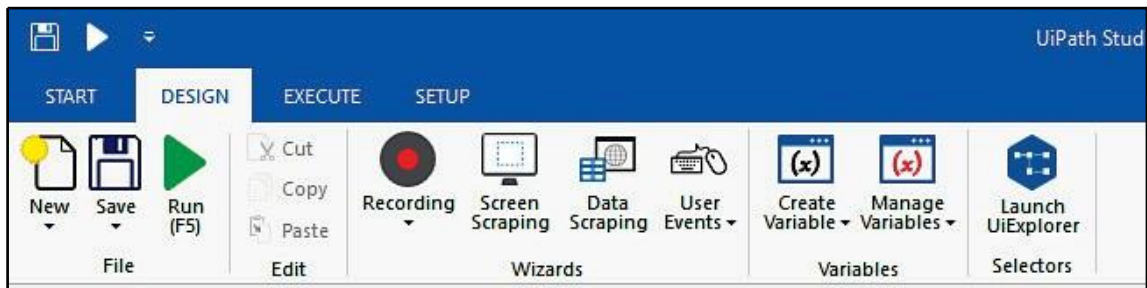


1. The Ribbon
2. Quick Access Toolbar
3. Designer panel
4. Properties panel
5. Outline panel
6. Arguments panel
7. Variable panel
8. Import panel
9. Activity panel
10. Library panel
11. Project panel
12. Output panel

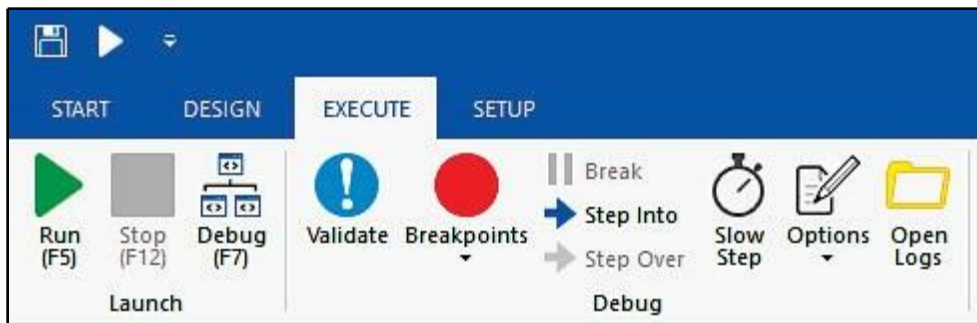
The Ribbon

This panel located at the top of the user interface and consists of four tabs:

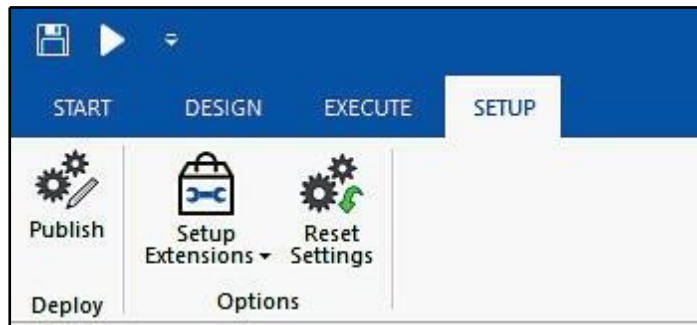
1. **START**: This is used to start new projects or to open projects previously made.
2. **DESIGN**: This is to create new sequences, Flowcharts, or state machines, or to manage variables:



3. **EXECUTE**: This is used to run projects or to stop them, and also to debug projects:

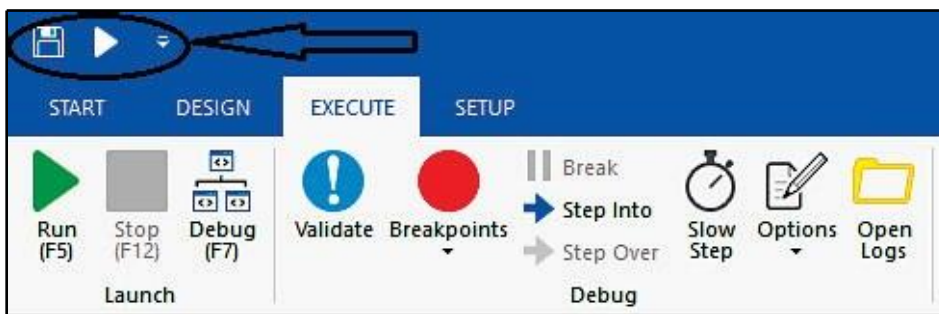


4. **SETUP:** This panel is for deployment and configuration options; it has three tools available:
- **Publish:** This is used to publish a project or create a shortcut for it and schedule tasks
 - **Setup Extensions:** This is used to install extensions for Chrome, Firefox, Java, and Silverlight
 - **Reset Settings:** This is used to reset all settings to defaults:



The Quick Access Toolbar

This panel gives the user a shortcut to the most used commands. One can also add new commands to this panel. This is located above the Ribbon on the user interface. The Quick Access Toolbar has been circled in the following screenshot and is indicated by the arrow:



It can be moved above or below the Ribbon. By default, there are two buttons available, **Save** and **Run**, which are also available in the **DESIGN** tab of the Ribbon.

Designer panel

This is the panel where one defines the steps and activities of the projects. It is where a developer does most of the things to record activities or manually drop activities on the canvas. In UiPath, this is equivalent to the code windows of Microsoft Visual Studio. When we develop a Robot, this is the window where we will be organizing various activities in a flow or chain to accomplish a task.

The project a user makes is clearly displayed on the Designer panel and the user has the option of making any changes to it.

Properties panel

The panel located on the right-hand side of the user interface is for viewing the properties of the activities and for making any changes, if required. You need to select an activity first and then go to the **Properties** panel to view or change any of its properties:

Properties

UiPath.Core.Activities.InputDialog

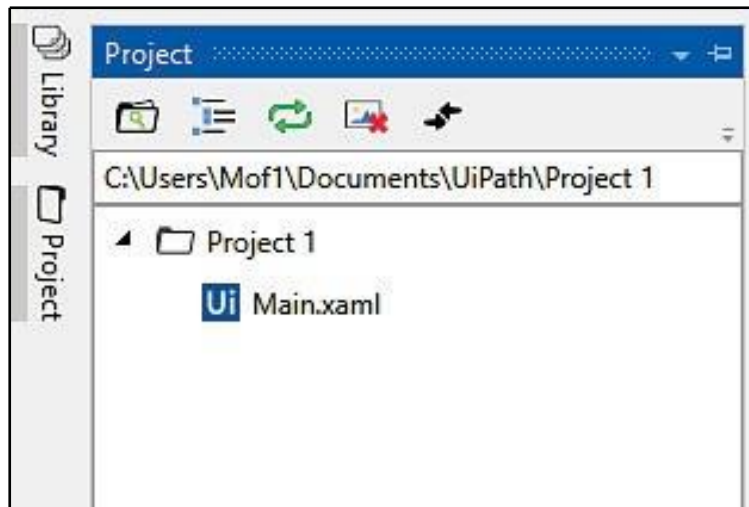
Common	
DisplayName	Input dialog
Input	
IsPassword	☐
Label	The label of the form field ...
Options	An array of options to ch ...
Title	"Hello World!" ...
Misc	
Private	☐
Output	
Result	The value inserted by th ...

Activities panel

Located on the left-hand side of the user interface, this panel contains all the activities that can be used in building the project. The activities can easily be used in making a project by simply dragging and dropping the required activity into the required location in the Designer panel.

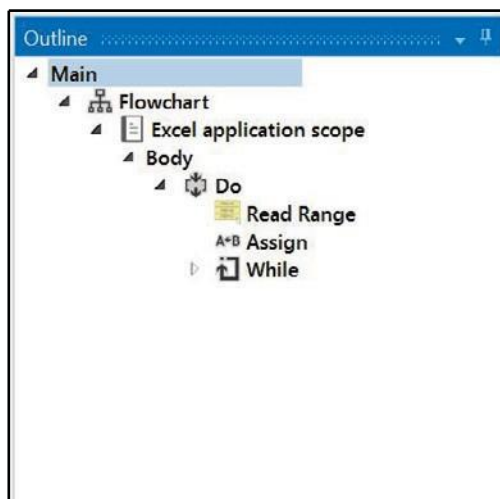
Project panel

With the **Project** panel, you can view the details of your current project and open it in a **Windows Explorer** window. It is located on the extreme left-hand side of the design panel, below the **Library** panel:



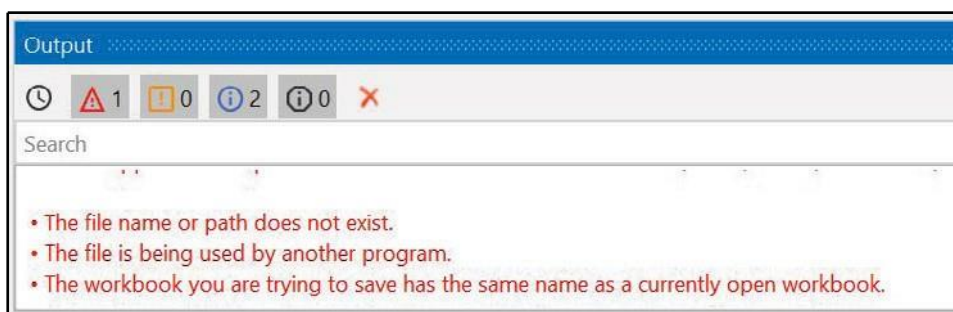
Outline panel

As the name suggests, this panel gives a basic outline of the project. The activities that make up the workflow are visible in this panel. Using this, you may see a high-level outline of the project and you can drill down to see deeper. This panel is especially helpful of large automation projects, where one may otherwise have a tough time going through it:



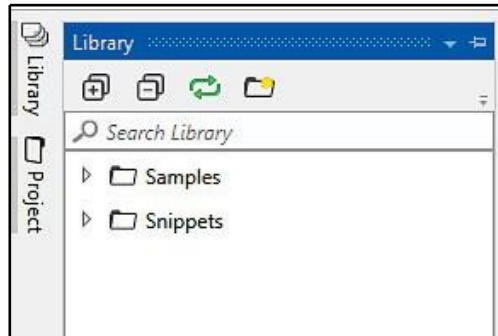
Output panel

This panel displays the output of the *log message* or *write line* activities. It also displays the output during the debugging process. This panel also shows errors, warnings, information, and traces of the executed project. It is very helpful during debugging. The desired level of detail can be changed in **Execute** | **Options** | **Log activities**:



Library panel

With this panel, you can reuse automation snippets. It is located on the extreme left-hand side of the Designer panel:



Variable panel

This allows the user to create variables and make changes to them. This is located below the Designer panel.

In UiPath Studio, variables are used to store multiple types of data ranging from words, numbers, arrays, dates, times, and timetables. As the name suggests, the value of the variable can be changed.

An important point to note is that variables can only be created if there is an activity in the Designer panel.

To create new variables, you can go to the **DESIGN** tab on the Ribbon and click on **create variable**, then choose the type of variable. Otherwise, one can simply go to the Variable panel located below the Designer panel and create a variable. Also, if one renames a variable in the Variables panel, the variable is renamed in every place it is used in the workflow. The Scope of the variable shows where the variable is located.

Argument

While variables pass data from one activity to another in a project, arguments are used for passing data from one project to another. Like variables, they can be of various types—String, Integer, Boolean, Array, Generic, and so on.

Since arguments are used to transfer data between different workflows, they also have an added property of *direction*. There are four types of direction:

- In
- Out
- In/Out
- Property

These depend on whether we are giving or receiving data to or from another workflow.

Task recorder

The task recorder is the main reason for RPA's success. With the task recorder, we can create a basic framework for automation. The user's actions on the screen are recorded by the recorder and turned into a recording sequence in the current project. That's how Robots are able to mimic human actions.

The recording is collection of execution steps that has to be taken, on the applications in the scope, in order to accomplish a task.

These steps can be recorded one by one (manually) by pointing it on the screen or many steps in a go that is, automatically.

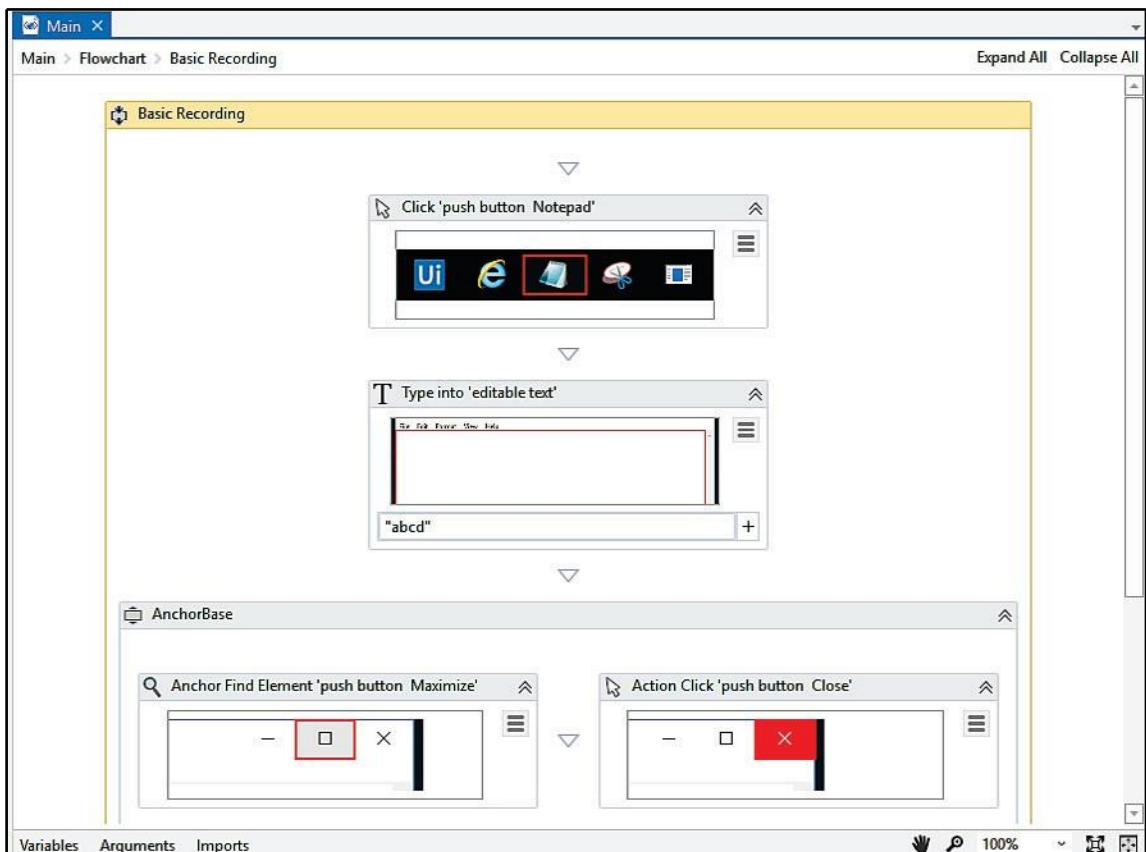
There are four types of recording in UiPath Studio:

- Basic
- Desktop
- Web
- Citrix

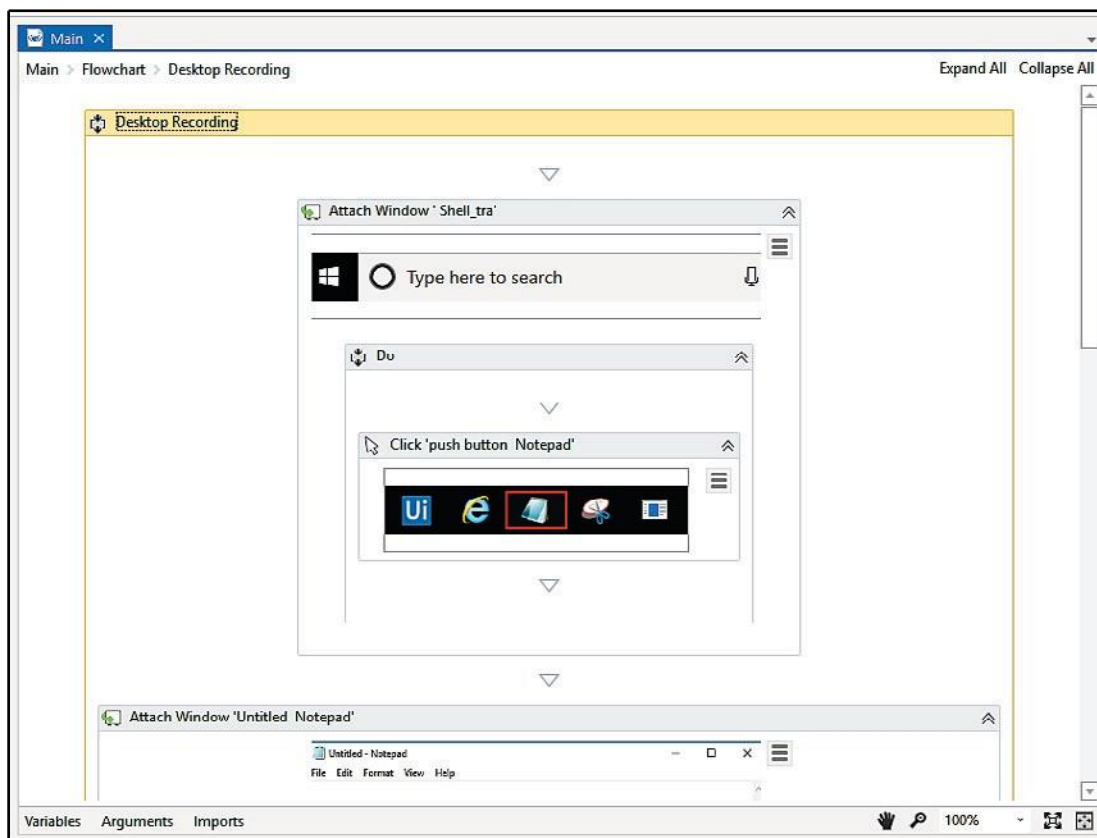
We will talk about them later. The user can modify the recorded sequence even after the recording is over. This is especially helpful in cases where small changes have to be made to the recording sequence. The option to modify an existing recorded sequence thus ensures that there is no need to record the entire process again.

There are four basic types of recording:

- **Basic recorder:** Basic recorder is used to record activities on the desktop. This type of recorder is used for single activities and simple workflows. The actions here are self-contained and not contained in separate windows, as shown in the following screenshot:



- **Desktop recorder:** The desktop recorder, like the basic recorder, is used to record activities on the desktop. However, it is used to record and automate multiple actions and complex workflows. Each activity here is contained in an **Attach Window** component, as shown in the following screenshot. The **Attach Window** component is especially important to ensure that other windows of the same application do not interfere in the workflow. UiPath uses the name of the app, the title of the window, and the currently opened file to locate and identify the correct window. However, there may be cases where, for example, two untitled Notepads are open on the screen. Without **Attach Window**, UiPath may select the wrong Notepad, thus causing errors:



- **Web recorder:** The web recorder, as the name suggests, is used to record actions on web applications and browsers.
- **Citrix recorder:** Citrix is used to record virtual machines, VNC, and Citrix environments. This recording allows only keyboard, text, and image automation.

Some actions are recordable while others are not:

- **Recordable actions:** Left-click on buttons, check boxes, drop-down lists, and other GUI elements. Text typing is also recordable.
- **Actions that cannot be recorded:** Keyboard shortcuts, mouse hover, right-click. Modifier keys such as *Ctrl* and *Alt* cannot be recorded.

There are two types of recording:

- **Automatic recording:** This is for recording multiple actions in one go. This is a very good feature for preparing a solid foundation for automating a task. It can be invoked with the **Record** icon available in basic, desktop, and web recorders. The Citrix recorder does not support automatic or multiple step recording. A few types of action cannot be recorded using automatic, for example, hotkeys, right-click, double-click, and a few more. For all these activities, you should use a single step recorder, also known as a manual recorder.
- **Manual recording:** This type of recording is used to record each step one at a time and hence offers more control over the recording. Also, it can record all actions that cannot be recorded using automatic recording such as keyboard shortcuts, mouse hover, right-click, modifier keys, such as *Ctrl* and *Alt*, finding text from apps, and many other activities.

While the desktop, basic, and web recorders can automatically record multiple actions and manually record single actions on the screen, the Citrix recorder can only record a single action (manual recording).

Shortcut keys:

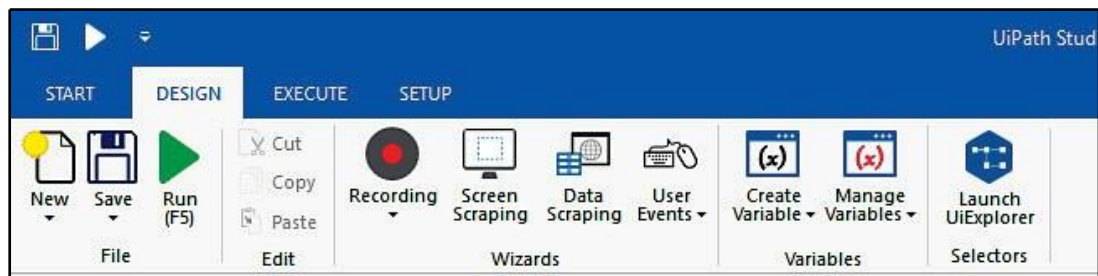


- **F2 key:** This pauses the recording for 3 seconds. The countdown menu is also shown on the screen.
- **Right-click:** Exits the recording.
- **Esc key:** Exits the recording. If one presses the *Esc* key again, then the recording will be saved.

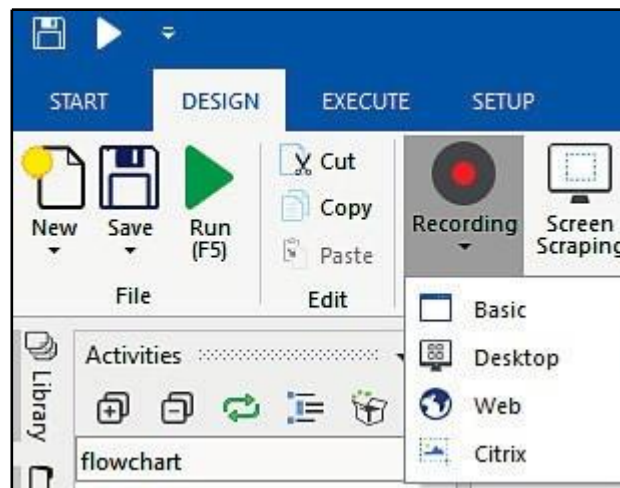
Let us now explore the functions of these recordings. The operations that can be completed with the help of recording are as follows:

- Click (clicking a UI element: button, image, or icon)
- Type (typing any value into the available text field)
- Copy and paste

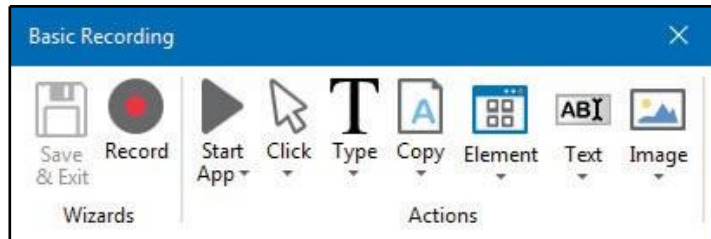
We can see a **Recording** icon at the top of the user interface on the **DESIGN** tab of the Ribbon, as shown in the following screenshot:



After clicking on this **Recording** icon, a list of the recording types are displayed, as shown in the following screenshot:



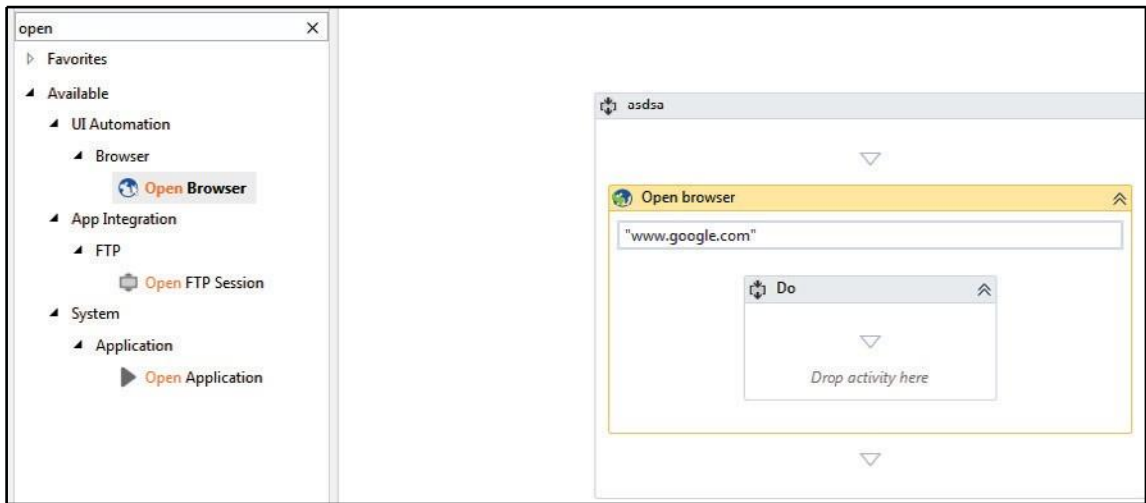
Clicking on each type of recording will result in the display of a recording panel with features specific to the type of recording. When clicking on **Basic** from the recording options, then the recording panel that appears looks as follows:



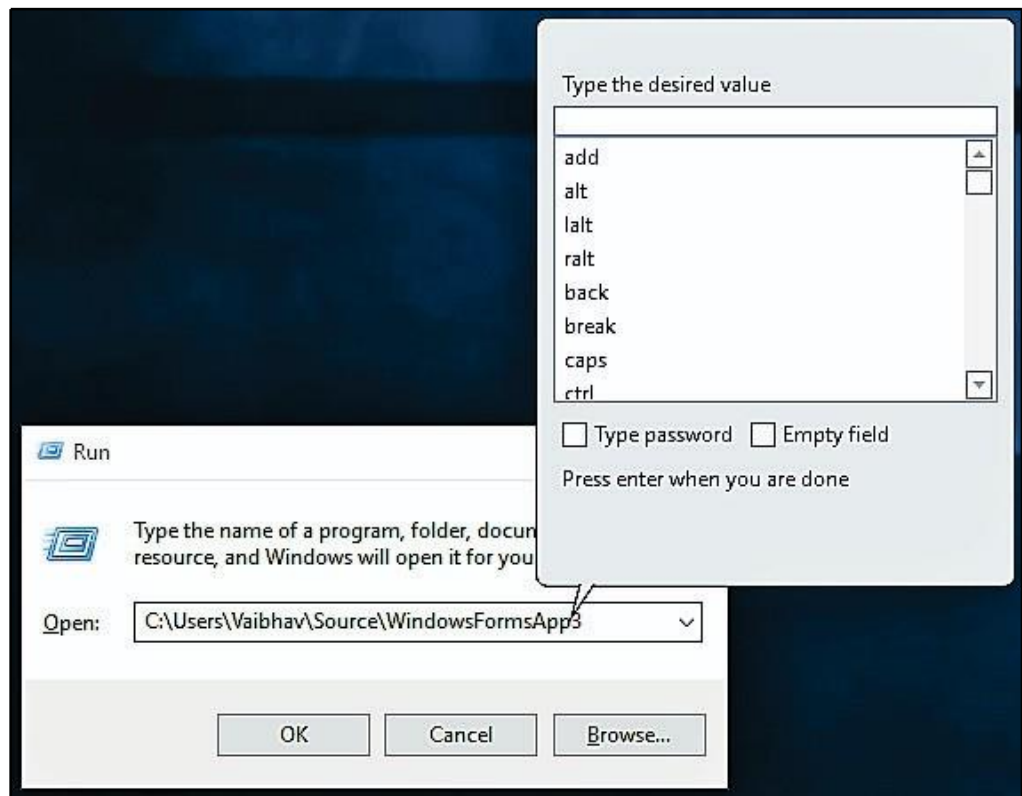
The panel that appears in the previous screenshot contains features specific to **Basic** Recording. For example; **Start App**, **Click**, **Type**, **Copy**, and so on.

- **Start App:** This is used to start an application. When we left-click on this option, we are asked to point to an application that we want to open. When we are done, we can click on the **Save & Exit** option. The following screenshot shows the recorded sequence.

As we can see in the screenshot, an open `explorer.exe` program appears. This is the title of the application. Below it, the path of this application is shown. As mentioned previously, the features that appear in the panel are specific to the type of recording. In case of web recording, there is an option of **Open Browser** rather than **Open Application**:

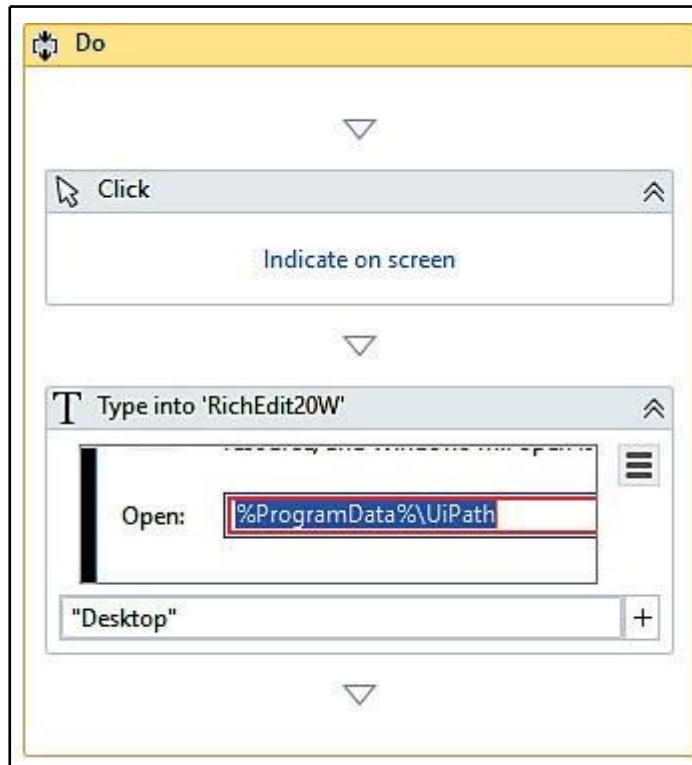


- **Click:** Another option is **Click**, which is used to click on a UI element. This feature is used as a mouse input. That is, it is used for clicking, checking, or selecting an item. When we click on this option, we are asked to indicate the location of the UI element we want to click. We can change the type of click to right-click or double-click in the **Click Type** property from the **Properties** panel.
- **Type:** Another option shown in the recording panel is **Type**. As the name suggests, it is used for typing something inside the indicated element. Say, for example, you want to type something in Command Prompt. All you need to do is to indicate the area where you want to type. Then, you need to type your input in the popup that appears for typing. Checking the empty field box (shown in the following screenshot) ensures that text written in the past (if any) will be emptied, leaving you with only the current text you have typed:



After you are done typing, do not forget to press the *Enter* key. When the *Enter* key is pressed, the step is recorded. You can then click on **Save & Exit** to view the recording sequence.

The recording sequence is shown in the following screenshot. You can change the text you have written (by changing the value of the **Type** in the block). You can write the desired text in double quotes (" "), or you can simply use a variable to store the data:



- There are three more options in the recording panel:
 - Element
 - Text
 - Image

These three are UI elements; the same keyboard and mouse options can be performed on them as explained in the preceding screenshot.

- You can see by clicking on the **Element** option that the click and type options are available as shown in the following screenshot:



Similarly, in case of **Text** and **Image** options, there are basically two events that come into play:

- Clicking any text or image indicated as the UI element
- Get text or get image

Advanced UI interactions

Advanced UI interactions are input and output interactions. In other words, it refers to the types of input methods and output techniques that are used while automating.

Input methods

The input that we give in the form of text can be of three types:

1. Default
2. Simulate
3. Window message

Default is the generated method, while the other two are available in the **Properties** panel.

There are two checkboxes for these two methods. The Default method is the slowest process and is the best way to test whether our input option is working or not.

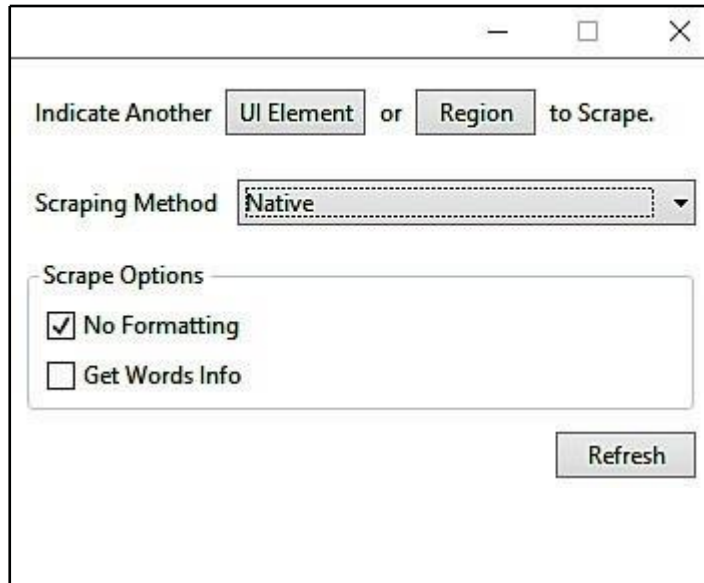
The other two methods work in the background. Out of these three methods, the simulate type is the fastest method and is mostly preferred because in the window message input type, it types only the lowercase characters.

Output methods

These are the methods we use for getting our output, which can be in the form of text or images. The available methods are:

- Native
- Full text
- OCR
-

Native is, by default, the generated method to extract data from the window. When you indicate to any element, the scraping window appears, and here all of the options can be found. We can choose any one that displays better results. OCR is preferred when the other two fail to extract data:



As shown in the screenshot, the scraping methods are **Native**, **Full text**, and **OCR**.

In OCR, there are two types of **OCR engine**: One is Google OCR and the other is Microsoft OCR. We can choose whichever displays better results. Also, we can adjust the scale mentioned in the properties of the OCR. This scale can be used to improve the efficiency of the OCR.

Step-by-step examples using the recorder

In this section, we will illustrate two examples of using the UiPath recorder:

1. Emptying the trash folder in Gmail
2. Emptying Recycle Bin

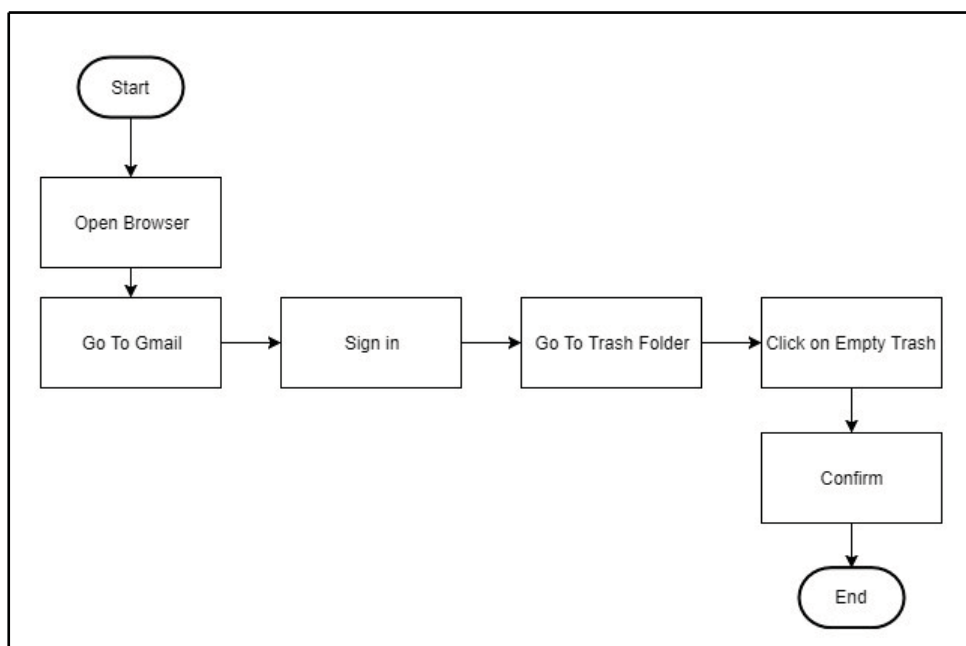
The first one is to show a recording of a web-based application, and the second is Windows-based. These are very simple examples, and show how a simple task can be automated quickly.

Emptying trash in Gmail

This is an example of how we can empty a folder in Gmail with the help of a UiPath Robot, solely on the basis of recording.

To do this, we are going to record all the actions that have to be performed to empty this Trash folder so that our Robot understands the sequence to be performed.

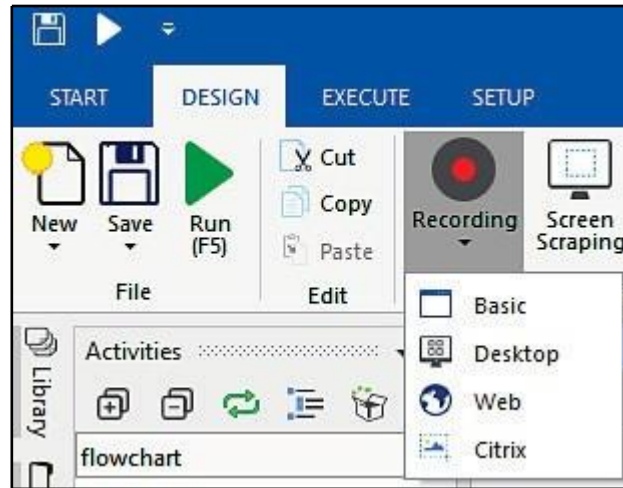
We can see the process flow of this simple activity in the following diagram:



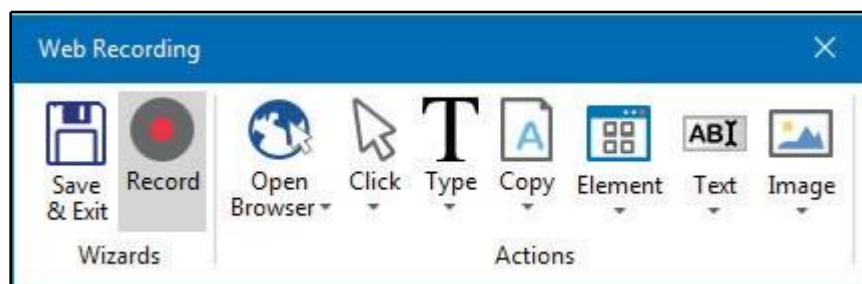
Process flow for emptying Gmail trash

We need to see all process flows, however small or big, as shown in the previous diagram. This makes developing RPA much easier and, organized.

First and foremost, we begin with a blank project in UiPath Studio and then choose **Web** recorder from the **Recording** drop-down list:



We have to click on the **Recording** option and select the type of recording. As discussed before, we will use **Web** recording for this process since we are working on a website. Just click on the **Recording** icon at the top of the page. From the four types of recording that appear, choose **Web** recording. A **Web Recording** panel will appear, as shown in the following screenshot:



Notice **Open Browser** between **Record** and **Click**; this is available with web recorder to record steps in browser-based applications.



Preparation: Open your favorite browser, navigate to <https://gmail.com>, and keep this browser open.

The following are the six steps in our process flow:

1. **Open Browser:** Although we have already opened Gmail in the browser, we did not record that step. Here, we will note that step in the recorder using the **Open Browser** button in the recorder. A drop-down menu will appear. Again, choose **Open Browser** from the drop-down menu. It will ask to highlight the browser, highlight the already opened browser and click on the top of the browser.
2. **Go to gmail.com:** You will be prompted to enter the **URL** of the website to navigate to. Type <https://gmail.com> or gmail.com and press **OK**:



Please remember the first step will merely make note of the steps in the recording but will not do anything on the screen. From the next step onwards, we will use the already opened gmail.com.

3. **Sign In:** Start recording by clicking on the **Record** icon of the recording panel.

Go to the already open Gmail and click on the Email or Phone field. UiPath will pop up a prompt for typing the email:



Type `Email` in the box provided by the UiPath recorder and press *Enter*. The Gmail textbox will automatically fill up with your typed content.

Click on the **NEXT** button of the Gmail interface; it will also get recorded.

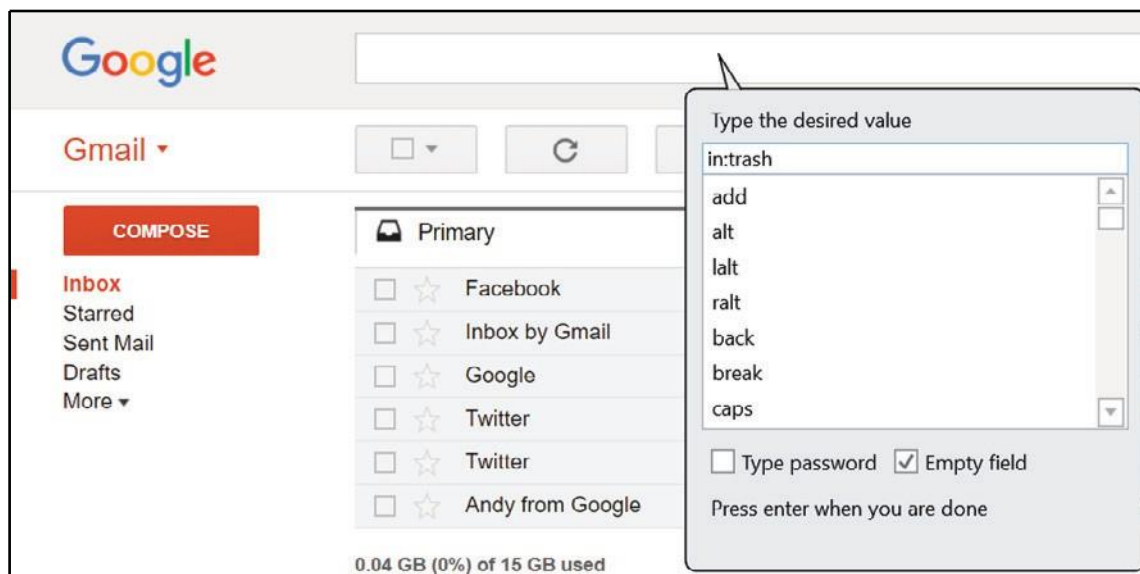
Now, you have recorded an entry in the password field. For simplicity, you may type the password in the prompt provided by UiPath.



In a real-world environment, you will select the **Type password** checkbox if you are entering a password (details on this will be discussed in later chapters).

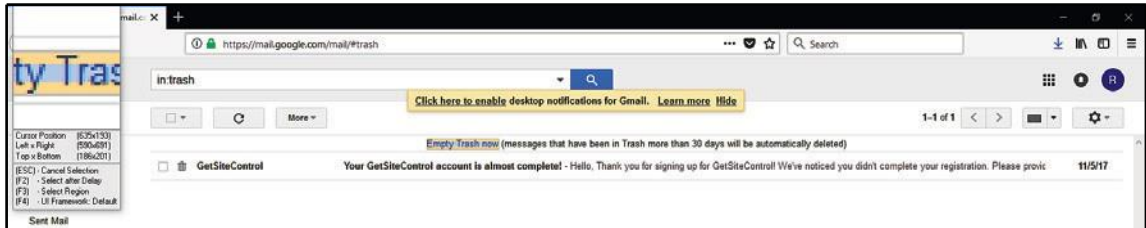
Type your password in the text field of the popup that appears. Then, click **NEXT** to log in to your account. Clicking on the **NEXT** button will also get recorded.

4. **Locate Trash Folder:** In this step, we have to click on the search box of Gmail and type `in:trash` in the UiPath prompt and hit *Enter*:



Now, click on the Search button beside the search box. It will also get recorded automatically and the Trash folder will appear.

5. **Click on Empty Trash now:** Once you are done with clicking on the Trash action, You can see a link showing **Empty Trash now**. Hover mouse on this link and it will get highlighted, click on it to delete all the messages in the Trash folder:



6. **Confirm:** When you click on **Empty Trash now**, a confirmation dialog will appear asking your permission for the action. Just confirm your action by clicking on the **OK** button.



After clicking on any button, the recorder may display a dialog for using the **Indicate Anchor**. In that case, just click on the **Indicate Anchor** button and indicate the element adjacent to the button you want to click. This is used to confirm the location of the element on which you are performing the action.

In the indicate anchor wizard, we have to indicate the adjacent button, that is, the **Cancel** button, so that the recorder will identify that the button is adjacent to **Cancel**.

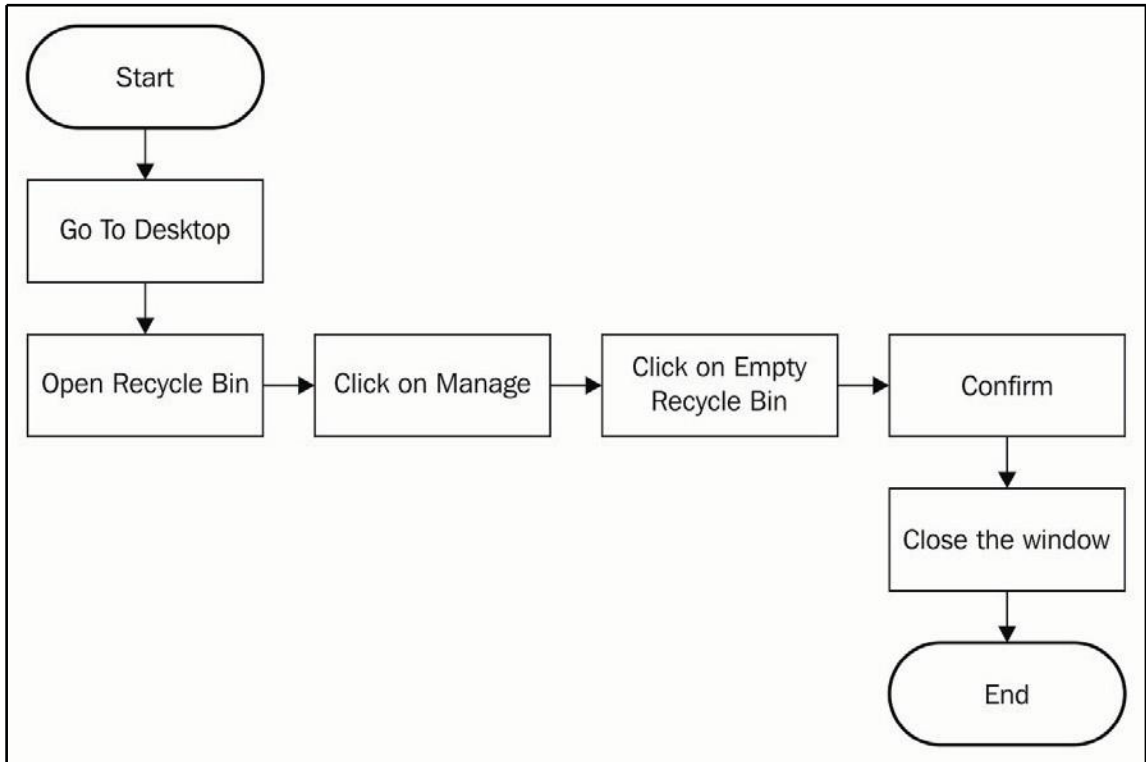
Now recording is complete, press *Esc* to get to the recording dialog. Click on the **Save & Exit** button.

Then, in UiPath Studio, you can see a recording sequence in the Designer panel. Rename it to **emptying trash folder**. This will help in easy recognition of the purpose of the sequence.

Now run it by pressing the *;F5* key; it should perform the same task again. You have created your first Robot, which empties trash from your Gmail!

Emptying Recycle Bin

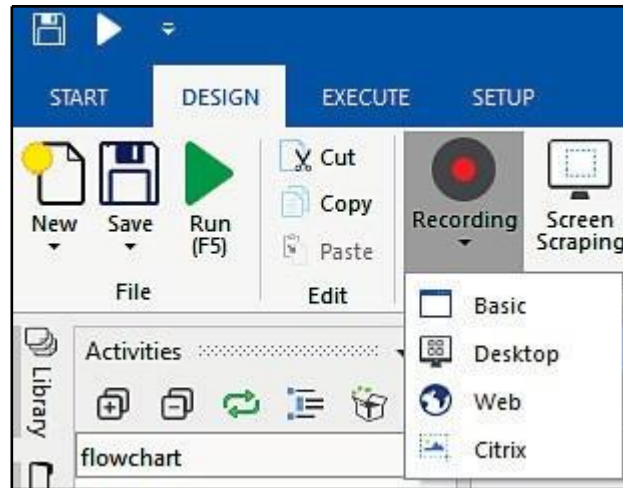
We are going to automate emptying the Recycle Bin. There are various steps that are involved. Let's map the process of how to empty the Recycle Bin:



Steps to empty Recycle Bin

This diagram is simpler and more detailed than in the Emptying trash in Gmail example; we need to do exactly the same steps in order to perform this task.

Open UiPath Studio and choose a blank project. Since we are working in the recorder, and since we are working on the desktop and not a web application, we are required to choose the **desktop** recorder:

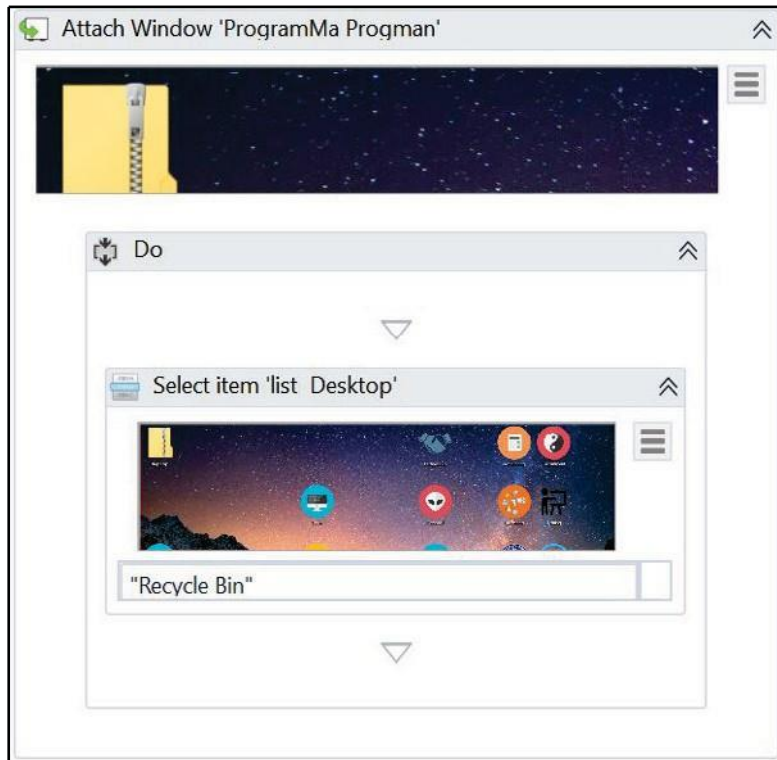


Start the recorder and simply perform the following steps:

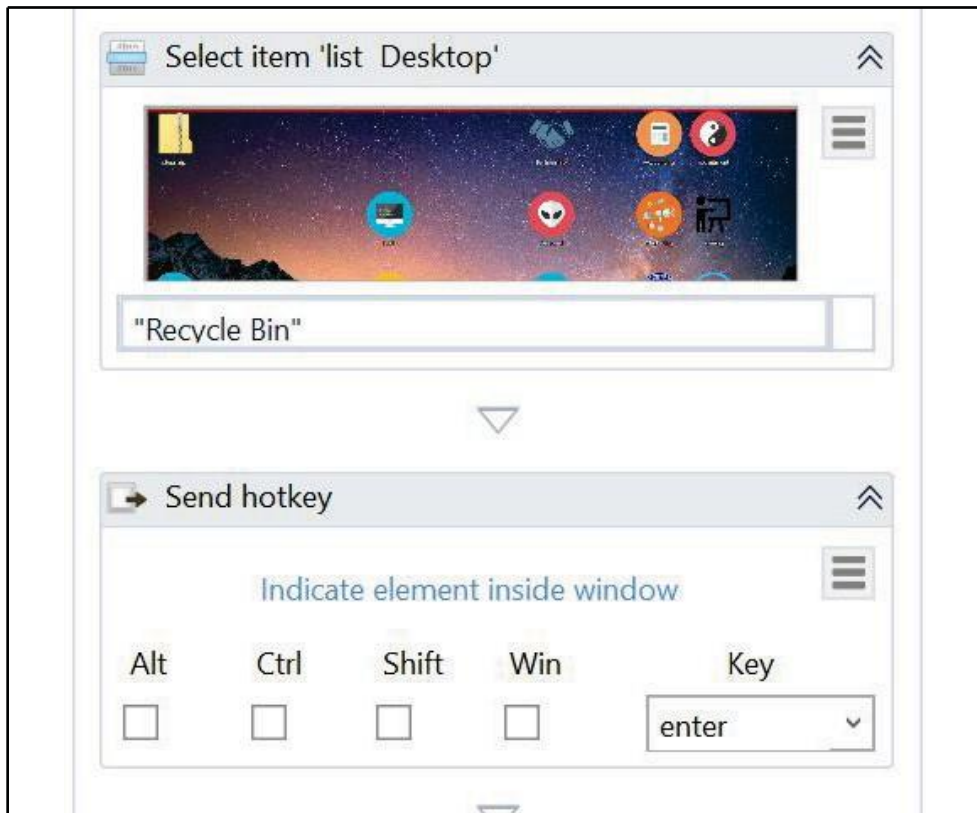
1. Go to the desktop by pressing the *Windows + D* keys.
2. Open Recycle Bin by clicking on Recycle Bin and then pressing *Enter* key.
3. Click on the **Manage** tab of the *Recycle Bin* folder.
4. Click on the **Empty Recycle Bin** button.
5. Confirm by clicking on the **Yes** button in the dialog box.
6. Close the Recycle Bin folder by pressing the **cross** button.
7. Press the *Esc* key and **Save & Exit** the recorder.

Now your recording is ready to view, let's examine each step recorded:

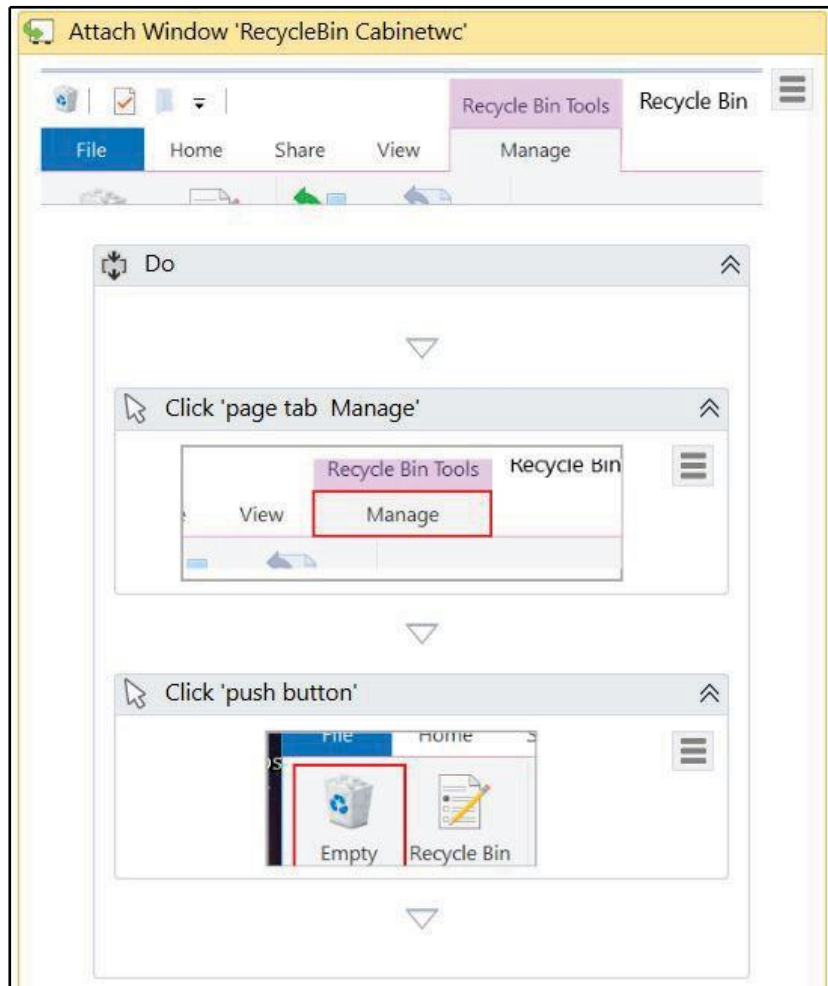
1. Go to the desktop by pressing *Windows + D* keys: This step is not recorded! Never mind, it is not needed. Please note that the recorded steps attach themselves to an application, and execute commands for that application, so the next step (Open Recycle Bin) will be executed on the desktop whether you are there or not.
2. Open Recycle Bin by clicking on Recycle Bin and then pressing the *Enter* key—We can see the recorded step in the following screenshot:



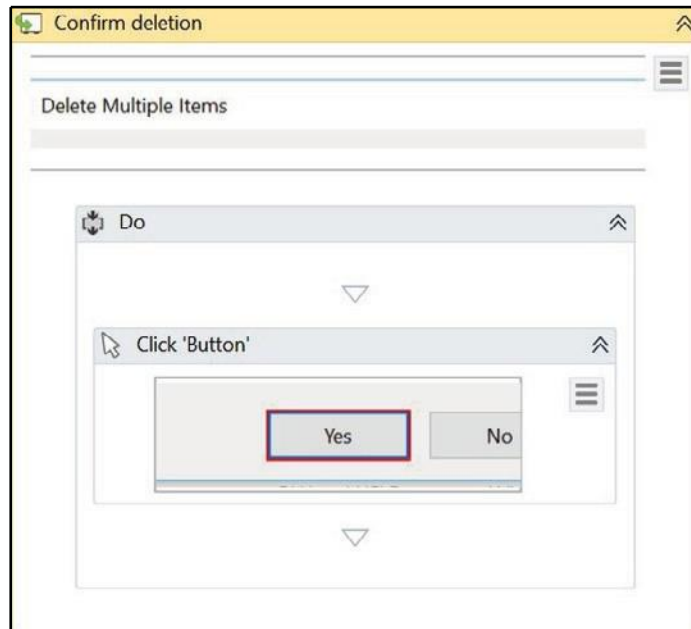
Please note that only selecting the Recycle Bin is recorded, not the *Enter* key. We should manually add that step. Search for `Send hotkey` in the **Activities** window and insert it into the workflow just below the **Select item 'list Desktop'** step, as shown in the following screenshot:



3. Click on the **Manage** tab of the Recycle Bin folder: This is recorded as it is and so is the fourth step, click on the **Empty Recycle Bin** button:



4. Confirming by clicking on the **Yes** button on the dialog box is also recorded smoothly:



In the last step, closing the Recycle Bin folder by pressing the **cross** button, you may have to indicate an anchor.

Save it and press *F5* to run it. Voila! It runs like a charm. You see how easy it is to record steps taken on a computer and automate them.



In some scenarios, the second step of opening Recycle Bin may get recorded as a single-click instead of Selection; in that scenario, you may not manually insert **Send hotkey** for *Enter* but instead, change the single click Recycle Bin activity from single to double. For that, open your recording sequence and find the click Recycle Bin activity.

Now click this activity, and you will see that its properties contain the click activity, and we have to change the **ClickType**; from single to double.

Summary

In this chapter, we have learned about components of the UiPath platform and their functions. In the next chapter, readers will examine the project that we generated with the recorder, explain the structure of the program flow (workflow), and understand the use of sequences and the nesting of activities. Readers will learn how to use the building blocks of a workflow Flowchart and control flow (looping and decision making).

3 Sequence, Flowchart, and Control Flow

So far we have learned what RPA is, and we have seen how simple it is to *train* a UiPath Robot by recording the activities of a task and running it. Automating mundane routine tasks is very easy with UiPath using the recorder. Before we start automating complex tasks, let us learn how to control the flow of activities from one to another.

In this chapter, we will learn about methods to lay down activities in an orderly fashion and how to control the flow. These are basic to any kind of programming. We will learn about putting activities in Sequences, Flowcharts, and loops. We will also look at logical control using if-else.

To train robots to process a type of transaction, it is very important to have the instructions in a methodical flow. Mostly, these instructions are executed in Sequences. Let us understand this in detail.

This chapter will cover the following topics:

- Sequencing the workflow
- Activities
- What Flowcharts are and when to use them
- Control Flow, various types of loops, and decision making
- Step-by-step example using a Sequence and Flowchart

Sequencing the workflow

UiPath provides four types of projects:

- Sequences
- Flowcharts
- User Events
- State Machines
-

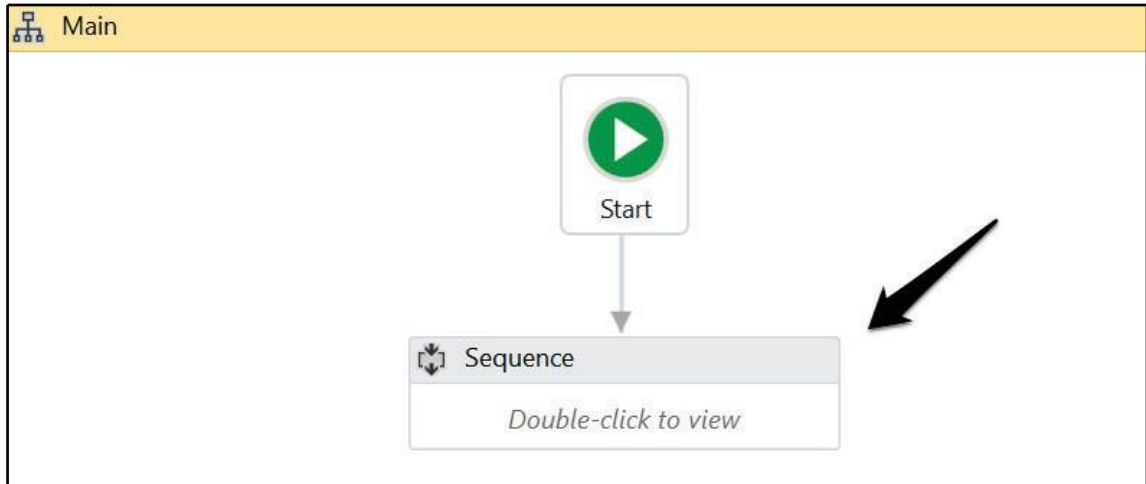
These are used to design a project on the basis of type and convenience. These four types of projects are useful in dealing with different kinds of processes. A Flowchart and Sequence are mainly used for simple automation. User Events are beneficial for implementing front office robots, while State Machines are used for dealing with complex business processes.

What is a Sequence?

A Sequence is a group of logical steps. Each step represents an action or a piece of work. A Sequence is used for processes that happen in linear succession, that is, one after the other. Among the three types of projects in UiPath, Sequences are the smallest. An illustration of how to build a Sequence in UiPath Studio is presented in the following section.

In the following example, we will make a simple project that asks for the name of the user and then displays his or her response:

1. Open UiPath Studio and click on **Blank** to start a fresh project. Give it a meaningful name. On the Designer panel, drag and drop a **Flowchart** activity from the **Activities** panel.
2. Search for `Sequence` in the **Activities** panel and then drag and drop it into the **Flowchart**, as shown in the following screenshot:

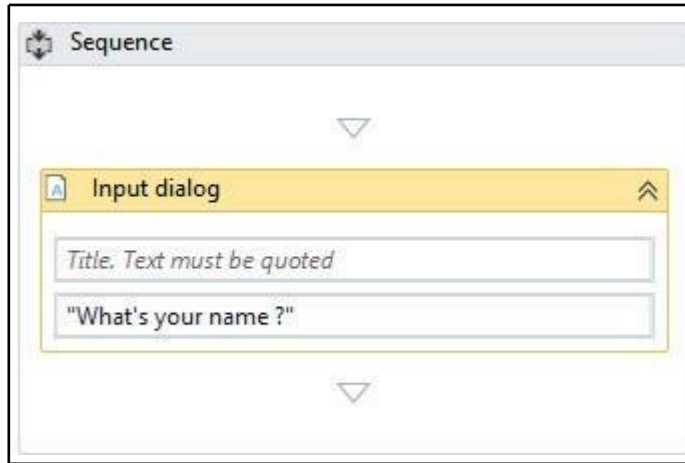


3. Double-click on the **Sequence**. We now have to add the steps that we want to perform. Consider each step as an action.

We can add many steps inside a **Sequence**. For the sake of simplicity, we will add two steps:

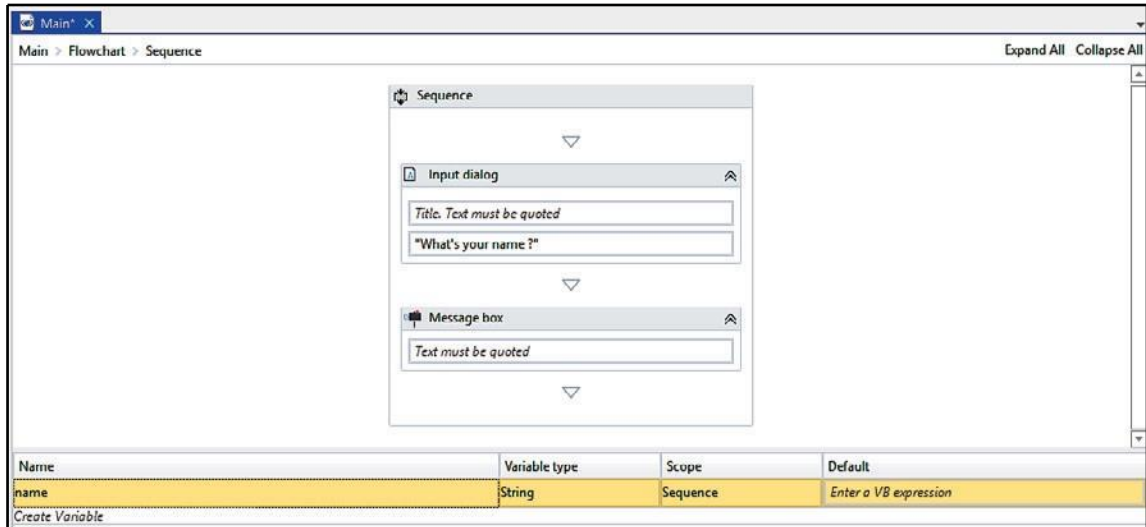
1. Ask for the username in an **Input dialog**
2. Display the username in a **Message box**

4. Search for **Input dialog** in the Search panel of the **Activities** panel. Drag and drop the **Input dialog** activity inside the **Sequence** (the **Input dialog** activity is a dialog box that appears with a message or a question, in response to which the user is required to put in his or her reply):

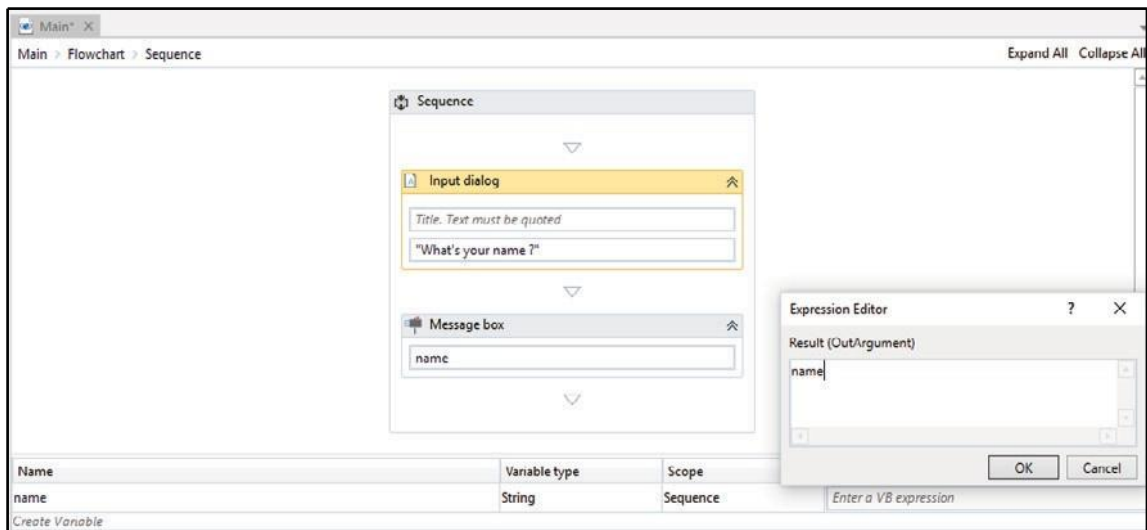


Write the appropriate message on the **Label** of this **Input dialog** to ask for the user's name. In our case, we have put in "What's your name?".

5. Drag and drop a **Message box** activity into the **Sequence**. (A **Message box**, as the name suggests, displays a given text. In this case, we will use it to display the text/reply that the user has given in the **Input dialog** box on being asked his/her name.)
6. Next, create a variable and give it the desired name. This variable will receive the text that the user has entered in the **Input dialog** box in response to our question, that is, the user's name:



7. We now have to specify the **Result** property (in the **Properties** panel) of the **Input dialog** box. On specifying the variable name there, it will receive the text that the user entered. Click on the dotted icon that appears on the right side of the **Result** property. Now, specify the variable name:

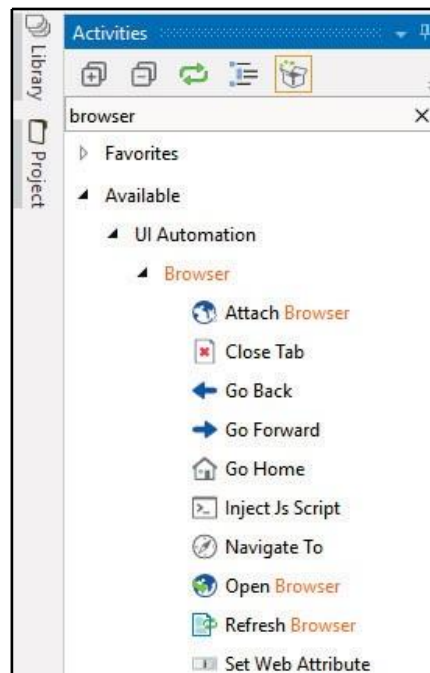


8. Specify the variable name that we have created in the Text area of the **Message box** (the Text area of the **Message box** is used to input text that will be displayed in the **Message box**). We just need to connect the **Sequence** to the **Start** icon. This can be done by right-clicking on the **Sequence** activity and choosing the **Set as Start node** option.
9. Hit the **Run** button and see the result.

Activities

In UiPath Studio, an **activity** represents the unit of an action. Each activity performs some action. When these activities combine together, it becomes a process.

Every activity resides on the **Activities** panel of the main Designer panel. You can search for a particular activity and use it in your project. For example, when we search for **browser**, all the browser activities will appear in the **Activities** panel, as shown in the following screenshot:



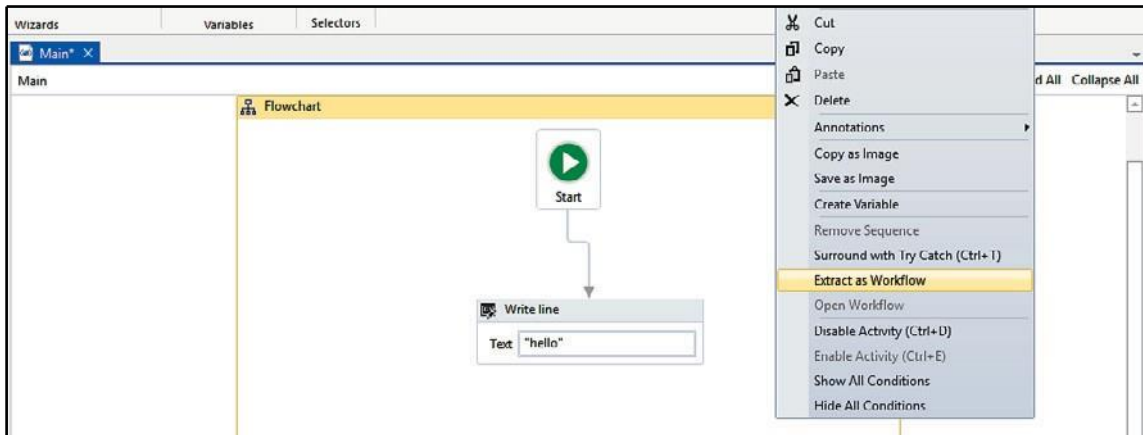
Using activities with workflows

We have seen how we can easily search for a particular activity. Now, let us see how to use them in a workflow:

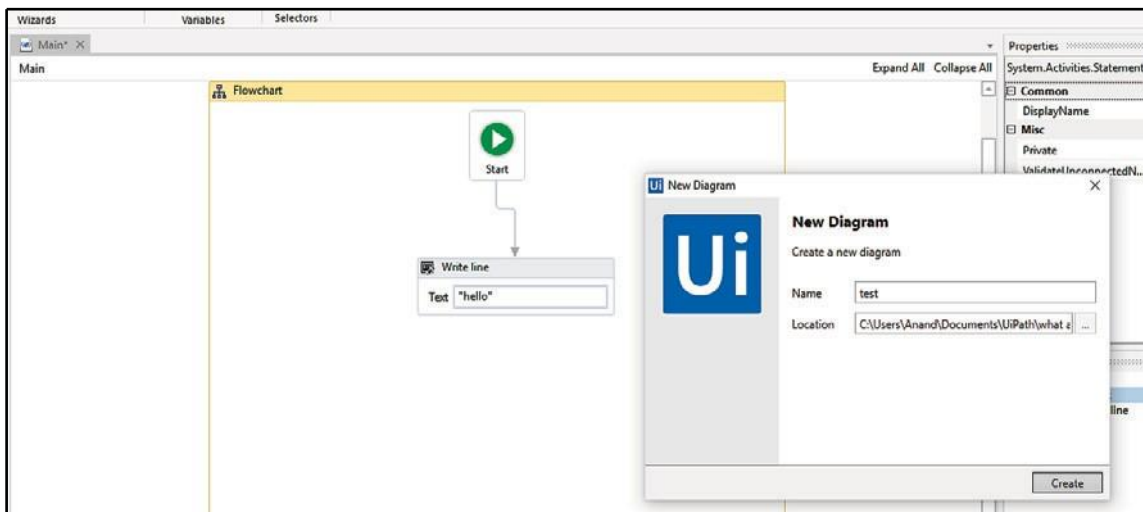
- 1 Search for **Flowchart** in the same way that we have searched for the browser activities in the **Activities** panel search bar. Drag and drop the **Flowchart** activity inside the Designer panel.
- 2 The **Flowchart** appears in the Designer panel and we have a given **Start** node. The **Start** node specifies where the execution begins.
- 3 We are ready to use different activities in our **Flowchart**. You can use any activity/activities inside the **Flowchart**. For the sake of simplicity, let us just use a **Write line** activity.
- 4 Drag and drop the **Write line** activity inside the **Flowchart**. Set its text property by providing a string value. Connect this **Write line** activity with the **Start** node by right-clicking on the **Write line** activity and selecting **Set as Start Node**.

You may be wondering what the role of a workflow is. Suppose you have a big project that consists of hundreds of activities. How will you debug it? It is a developer's nightmare to deal with such a situation. Here, the workflow comes into play. To build such a big project, a developer will simply divide it into smaller modules and extract it as a workflow. Now, each workflow can be tested separately. Thus, it is very easy to find bugs and errors. Creating different workflows and combining them into a logical **Sequence** will enhance our code quality, maintainability, reliability, and readability.

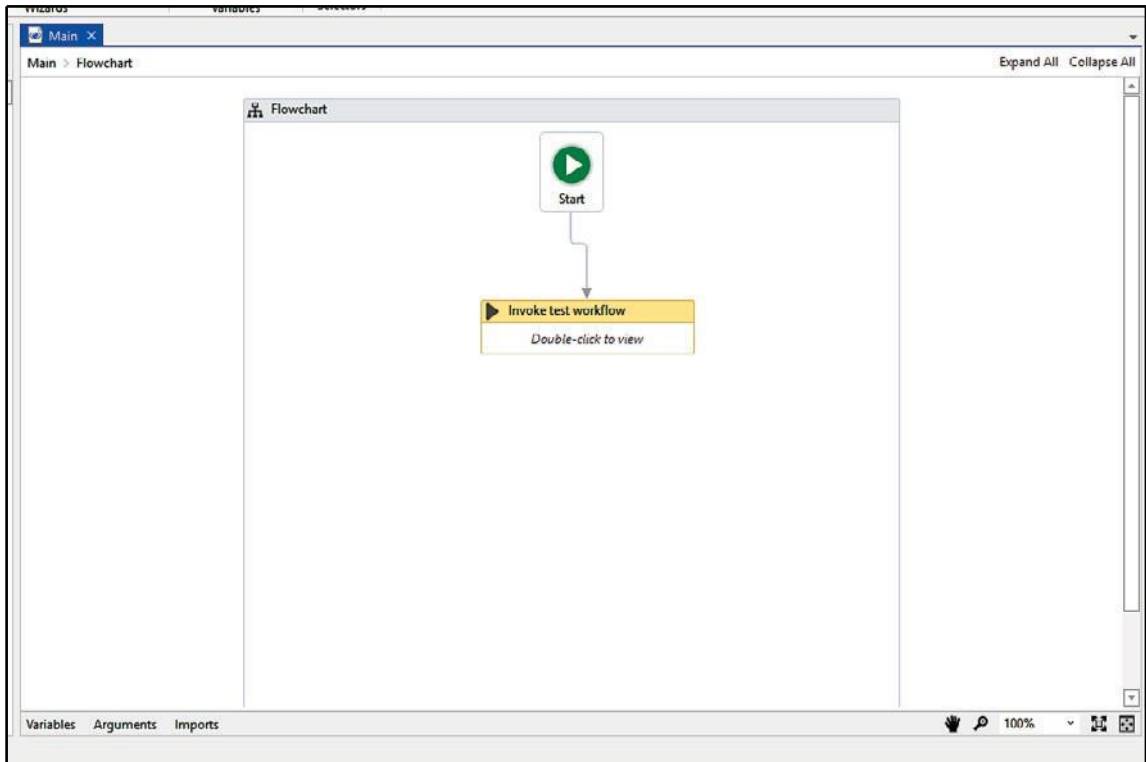
We have made a smaller module and now it is time to extract it as a workflow. Right-click on the main Designer panel and choose **Extract as Workflow**:



A window will pop up asking for the name. Give it a meaningful name and click on **Create**. This will be the name of your workflow:



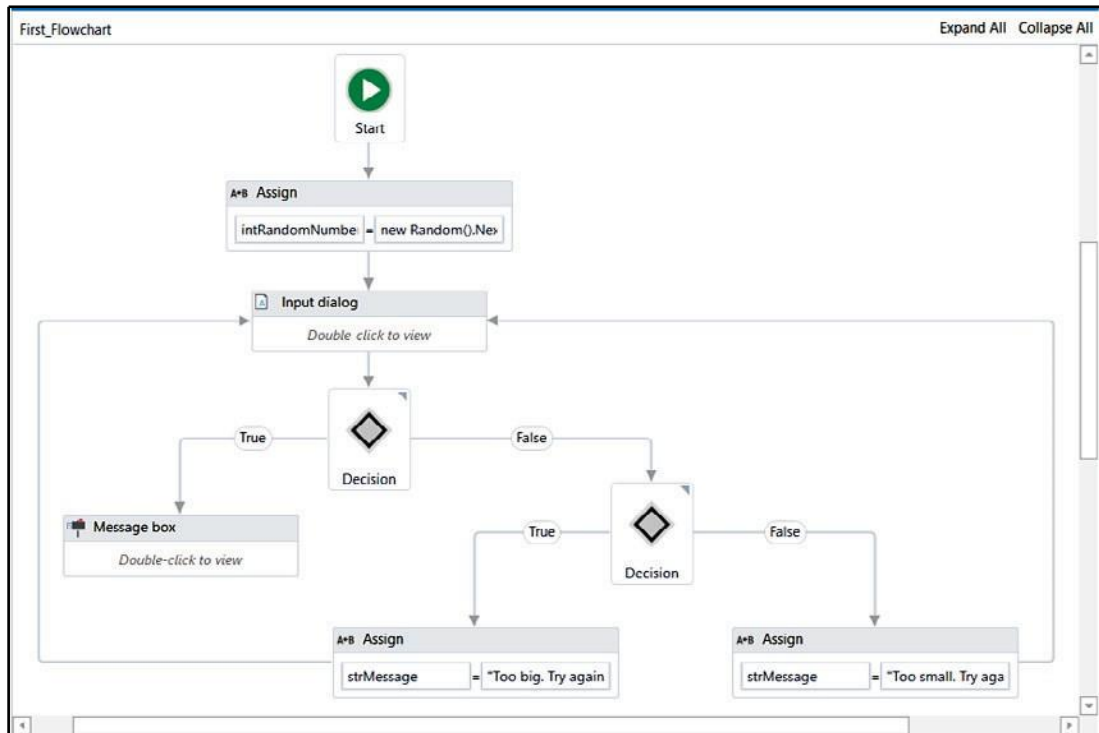
We have just used activities and extracted them in a workflow. If you check the main Designer panel, it looks like the following screenshot:



It automatically generates the **Invoke test Workflow** activity. Now, when we run the program, it will invoke the workflow that we have extracted (double-click on the **Invoke test workflow** activity to see which workflow it is going to invoke and where it is generated).

What Flowcharts are and when to use them

A Flowchart is generally used for complex business processes. It provides decision-making facilities and can be used for both small and large projects. Here, we can add activities in different ways:



A Flowchart provides multiple branching logical operators to make decisions. A Flowchart is able to run in reverse. Also, it can be used inside Sequences. A Flowchart facilitates reusability for distinct projects. Once we create it to use in a project, it can be used for a different but similar project.

A Flowchart's branches are set to true/false by default. However, its names can be manually changed from the **Properties** panel. For example, enter two numbers and check whether their sum is less than 20.

Perform the following steps:

1. First, add a **Flowchart** from the **Activities** panel into the Designer panel.
2. Add a **Sequence** activity within the **Flowchart**.
3. Take two **Input dialog** activities (for entering the numbers to be added) inside the **Sequence** activity.
4. Create the variables *x* and *y* to save the values.
5. Next, add a **Message box** activity to perform a mathematical operation. In our case, the sum of the two numbers is less than 20:

$$x + y < 20$$

6. Now, add a **Flow Decision** activity to check the mathematical operation.
7. If true, the **Flow Decision** will flow toward the true branch. Otherwise, it will flow towards the false branch.

Control flow, various types of loops, and decision making

Control flow refers to the order or the particular manner in which actions are performed in an automation. UiPath provides numerous activities for performing the decision-making process. These activities, present in the **Activities** panel, are put into the workflow either using the double-click method or the drag and drop method.

Different types of control flow activities are as follows:

- The Assign activity
- The Delay activity
- The Break activity
- The While activity
- The Do activity
- The For activity
- The If activity

- The Switch activity

The Assign activity

The **Assign** activity is used to designate a value to the variable. The Assign activity can be used for different purposes, such as incrementing the value of a variable in a loop, or using the results of a sum, difference, multiplication, or division of variables and assigning it to another variable.

The Delay activity

The **Delay** activity, as the name suggests, is used to delay or slow down an automation by pausing it for a defined period of time. The workflow continues after the specified period of time. It is in the hh:mm:ss format. This activity plays a significant role when we need a waiting period during automation, perhaps say, a waiting period required for a particular application to open.

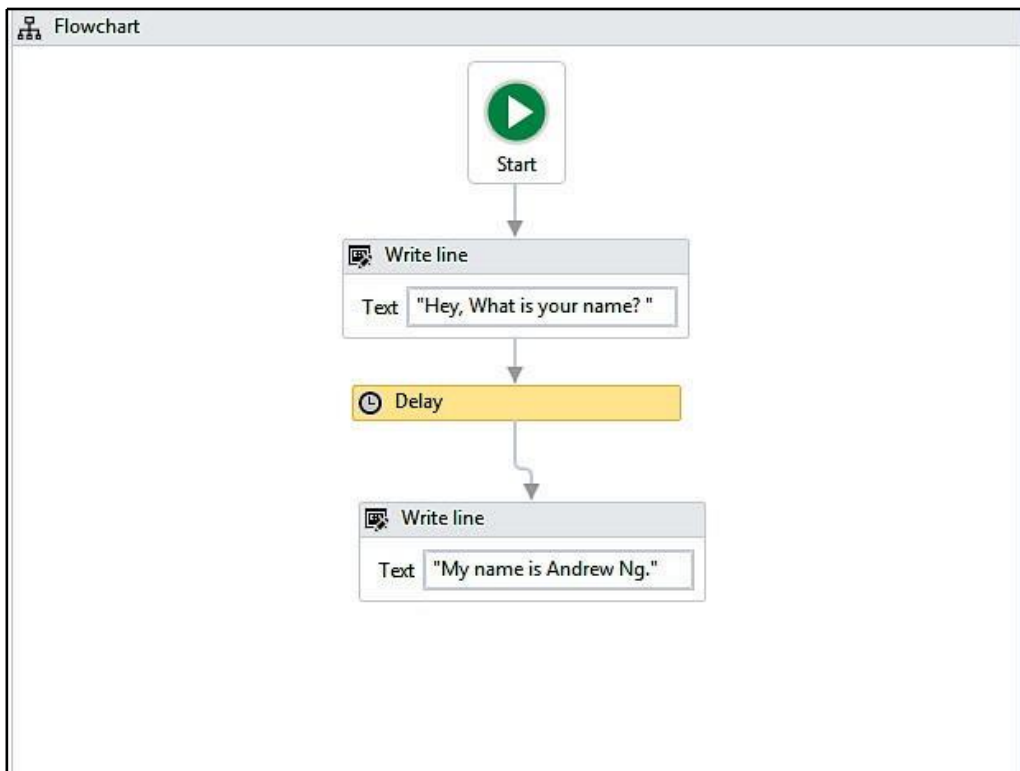
Example

To better understand how the **Delay** activity works, let us see an example of an automation that writes two messages to the **Output** panel with a delay of 50 seconds.

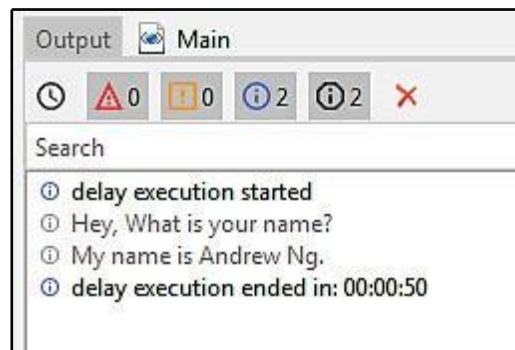
Perform the following steps:

1. First, create a new **Flowchart**.
2. Add a **Write line** activity from the **Activities** panel and connect it to the **Start** node.
3. Select the **Write line** activity. Now, type the following text into the **Text** box:
"Hey, what is your name?".
4. Next, add a **Delay** activity and connect it to the **Write line** activity.
5. Select the **Delay** activity and go to the **Properties** panel. In the **Duration** field, set 00:00:50. This is a 50-second delay between the two logged messages.
6. Take another **Write line** activity and connect it to the **Delay** activity. In the

Text field, write "My name is Andrew Ng.":



7. After clicking on the **Run** button, the **Output** panel shows the message that delays it by 50 seconds:



The Break activity

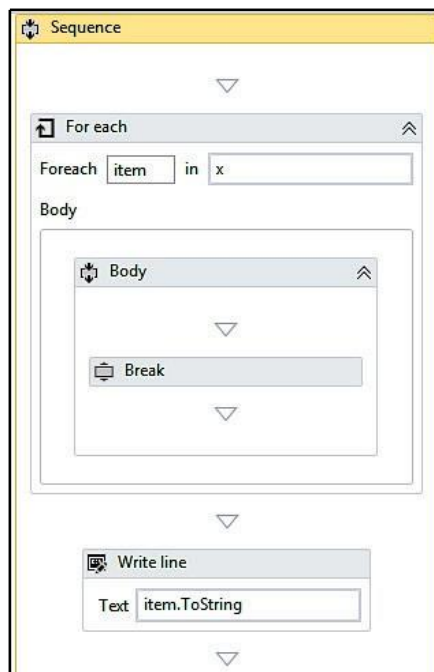
The **Break** activity is used to break/stop the loop at a particular point, and then continue to the next activity according to the requirement. It cannot be used for any other activity apart from the For each activity. It is useful when we want to break the loop to continue to the next activity in the For each activity.

Example

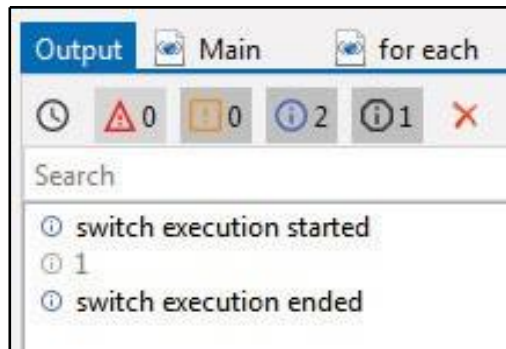
In this example, we will use the Break activity to execute only one iteration.

Perform the following steps:

1. Add a **Sequence** activity to the Designer panel.
2. Next, add a **For each** activity inside the **Sequence** (as mentioned in the preceding section, to use the **Break** activity, we need the **For each** activity):



3. Create two variables; an integer variable named `item`, and an array integer variable named `x`. Then, set them to the text field.
4. Now, assign a default value to the integer variable `x`.
5. Add a **Break** activity inside the body of the loop.
6. Under the **For Each** activity, add a **Write line** activity.
7. In the **Write line** activity, type `item.ToString` in the text field.
8. When we click the **Run** button, it will display one element, as shown in the following screenshot. This is due to the **Break** activity, which has stopped execution after the first iteration:



Before going on to the other control flow activities, we will learn about loops, which are an important aspect of automation. One of the most frequently mentioned advantages of automation is its ability to perform functions that are repetitive and to perform them without errors. Loops are meant precisely for such functions. Let us say, one wants to repeat a certain part of a workflow for different cases or when a certain criterion is fulfilled. In such a scenario, a loop comes in very handy. A loop can simply be created by connecting the end of the workflow to the point where we want the workflow to resume.



One thing that needs to be kept in mind while making such loops is to make sure there is also an exit point. Otherwise, the loop will continue infinitely!

The While, Do while, and For each activities mentioned among the various control flow activities are examples of loops. Let us now see where they are used and how they work.

The While activity

The While activity is used in automation to execute a statement or process based on a certain condition. If found true, the loop is executed; that is, the process is executed repeatedly. The project only exits from the loop when the condition does not hold true. This activity is useful while iterating through an array of elements.

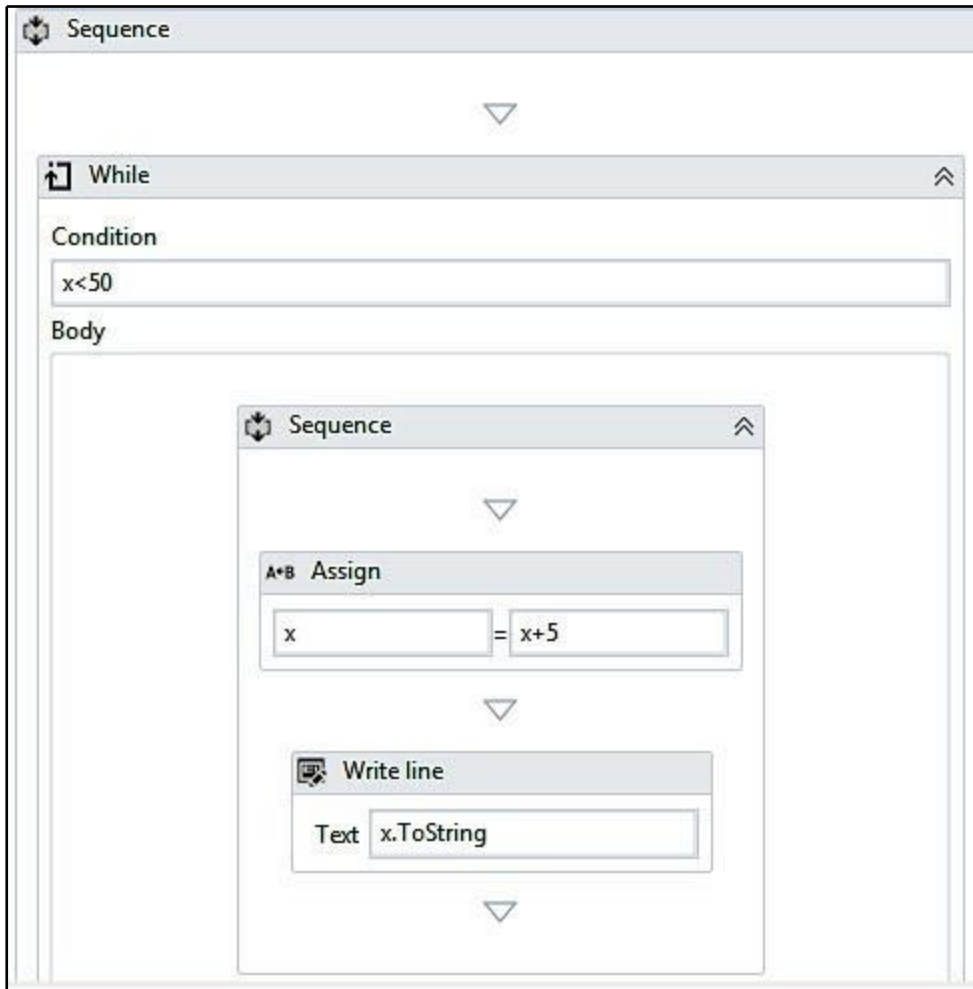
Example

In the following example, we will see how an integer variable will increase from 5 to 50 in increments of 5.

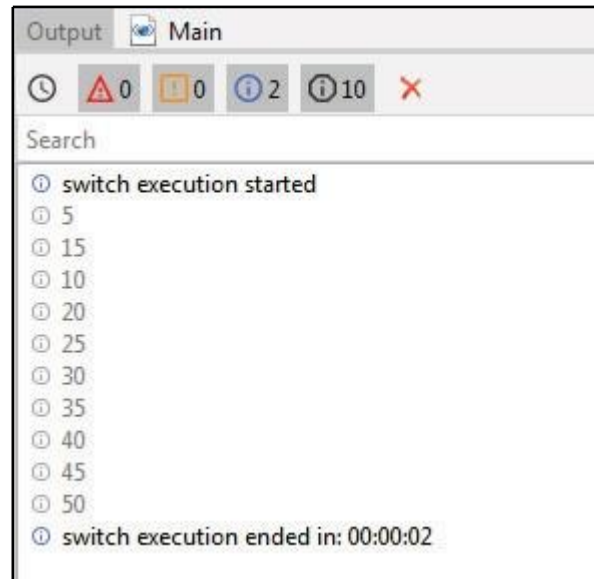
Perform the following steps:

1. On a **Blank** project, add a **Sequence** activity.
2. Now, create an integer type variable `x`. Set its default value to 5.
3. Next, add a **While** activity to the **Sequence**.
4. In the condition field, set `x<50`.
5. Add an **Assign** activity to the body section of the **While** loop.
6. Now, go to the **Properties** panel of the **Assign** activity and type in the text field integer variable for value field integer `x+5`.

7. Drag and drop a **Write line** activity and specify the variable name `x` and apply `ToString` method on this variable:



8. Now, click the **Run** button. The output will display in the **Output** panel, as shown in the following screenshot:



The Do while activity

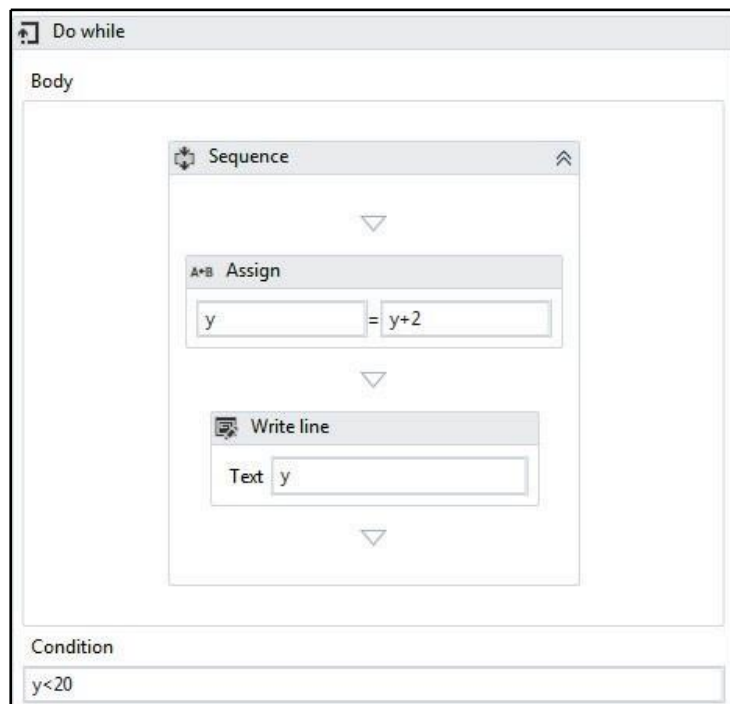
The **Do while** activity is used in automation when it is required to execute a statement based on the fulfillment of a certain condition. How it differs from the **While** activity is that it executes a statement, then checks whether the condition is fulfilled. If the condition is not fulfilled, it exits the loop.

Example

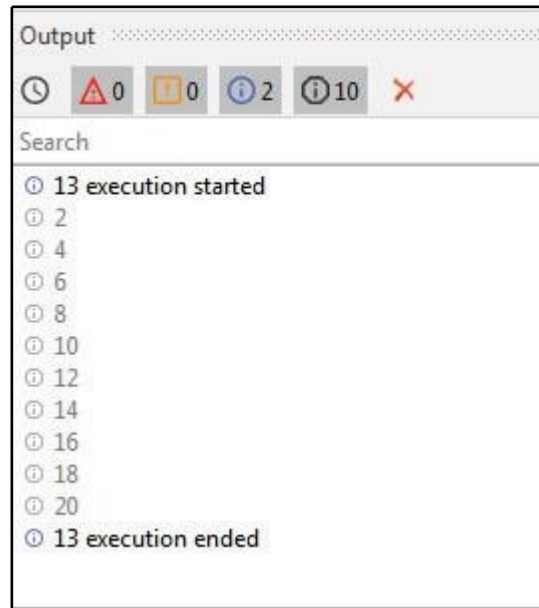
Let us take an example to understand how the **Do while** activity works in automation. Take an integer variable. Starting with this variable, we shall generate all multiples of 2, less than 20.

Perform the following steps:

1. Add a **Sequence** to the Designer panel.
2. Add a **Do while** activity from the **Activities** panel.
3. In the body section of the **Do while** activity, add an **Assign** activity.
4. Now, select the **Assign** activity. Go to the **Properties** panel and create an integer variable y . Set its default value to 2.
5. Set $y+2$ in the value section of the **Assign** activity to increment the result each time by 2 until the loop is executed.
6. Add a **Write line** activity inside the **Assign** activity.
7. In the text field of the **Write line** activity, type y .
8. In the condition section, set the condition $y < 20$. The loop will continue until the condition holds true:



9. On clicking the Run button, the output displayed will be as follows:



The For each activity

The **For each** activity works by iterating each element from the collection of items or list of elements, one at a time. In the process, it will execute all the actions that are available inside the body. Thus, it iterates through the data and processes each piece of information separately.

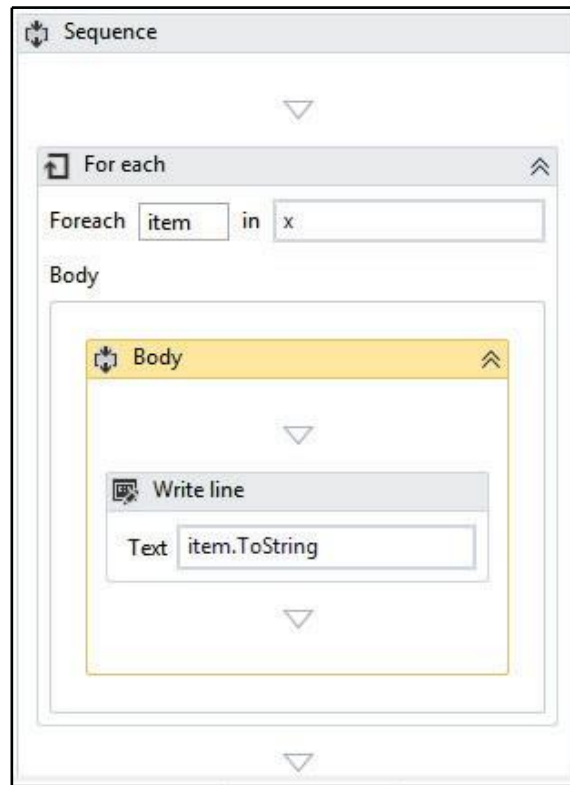
Example

In the following example, we shall use the **For each** activity to go through a collection of even numbers and display each element one at a time.

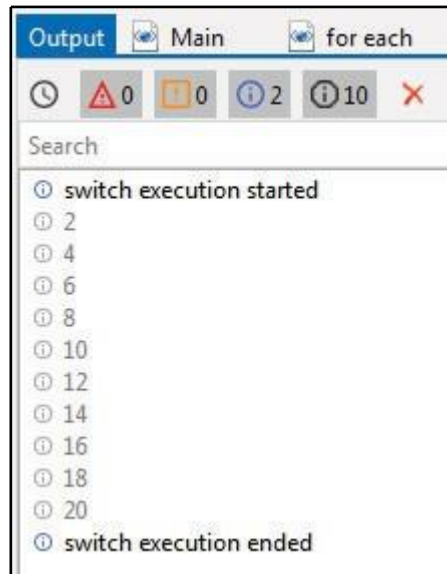
Perform the following steps:

1. Start with a **Blank** project in UiPath.
2. Add a **Sequence** activity to the Designer panel.

3. Next, add a **For each** activity within the **Sequence** and create an integer type array variable, `x`.
4. In the default value of the variable, put in `(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)`.
5. Add a **Write line** activity to the Designer Panel (this activity is used to display the results).
6. In the **Text** field of the **Write line** activity, type `item.ToString` to display the output:



7. Now, run the program. You will see that each number of the array is displayed one by one because of the use of the **For each** activity:



The Control flow also facilitates decision-making mechanisms that can help in taking a decision on a particular activity's step. For example, suppose we are using a loop and we want to display only a desired value, then we can filter out all our desired values by implementing the If activity, and making a decision based on the basis of the If activity's result, that is, true or false. The decision-making process will require some time to break the action after executing the desired element. This is followed by the Break activity, which will play a significant role. If you want to choose an execution from the task, then the activity needs to be switched in order to make such a decision.

The If activity and the Switch activity are the Control flow's decision-making activities.

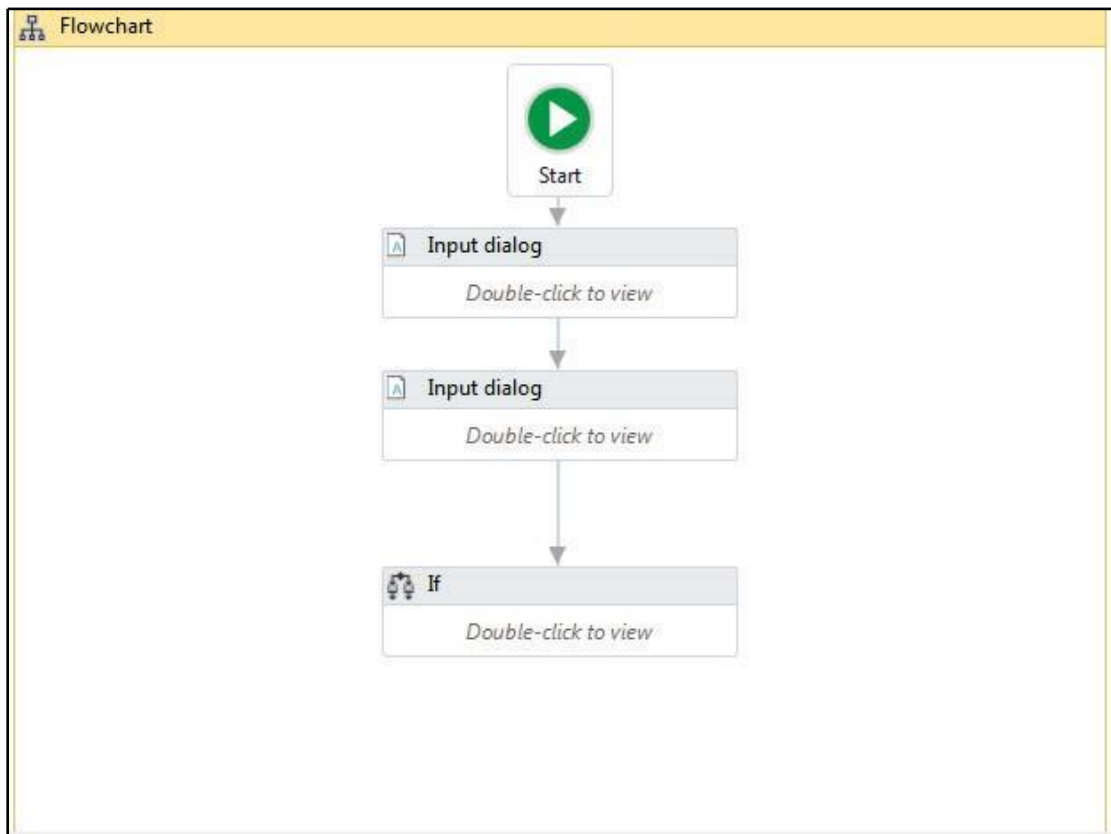
The If activity

The If activity consists of a statement with two conditions: true or false. If the statement is true, then the first condition is executed; if not, the second condition is executed. This is useful when we have to take decisions on the basis of statements.

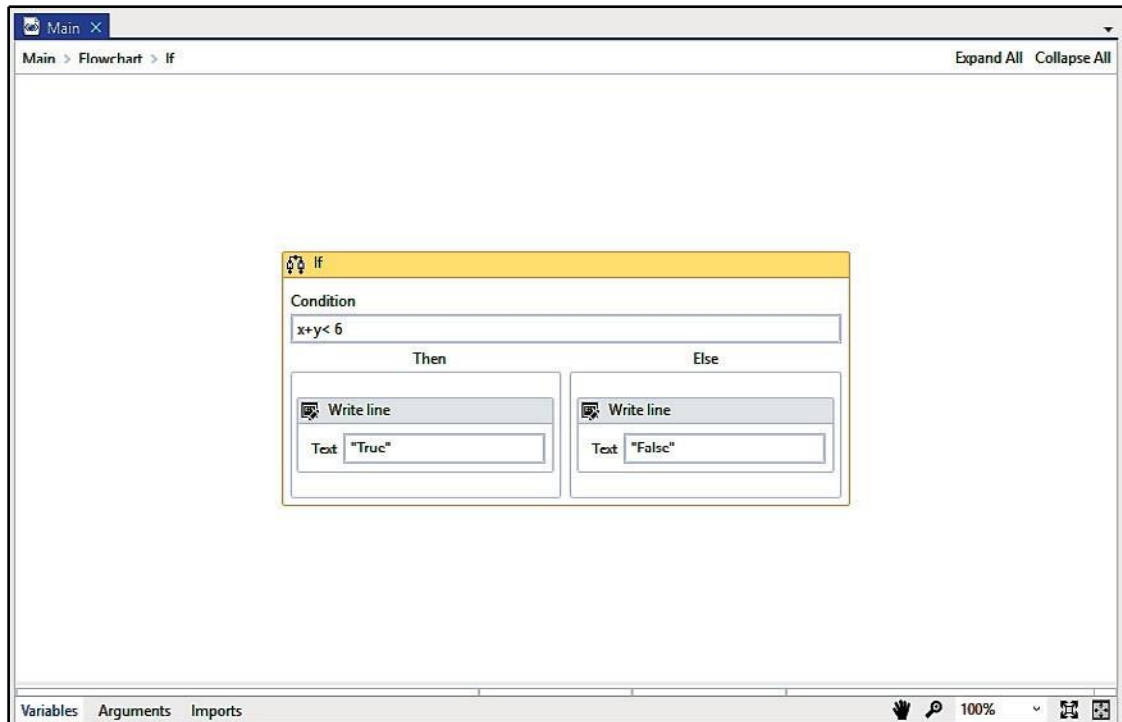
To better understand how the **If** activity works, let us see an example that checks whether the sum of any two numbers is less than 6.

Perform the following steps:

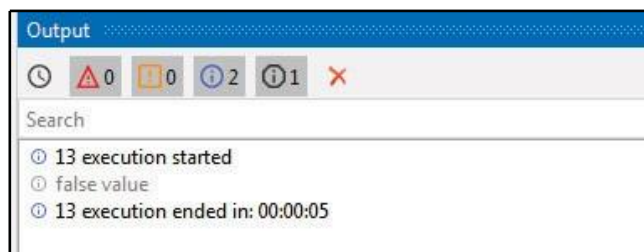
1. Add a **Flowchart** from the **Activities** panel.
2. Add two **Input dialog** activities. Create two integer variables, *x* and *y*.
3. In the **Properties** panel, change the label name and title name of both the **Input dialog** activities.
4. Now, specify these name of these two variables in the **Result** property of both the **Input dialog** activities.
5. Now add the **If** activity to the Designer panel:



6. In the condition part, $x+y < 6$, check whether it is true or false. Add two Write line activities and type "True" in one and "False" in the other:



7. Click the **Run** button to check the output. If the condition holds true then it will show the true value; otherwise, it will show the false value, as shown in the second screenshot (in our case, we put in the values of x and y as 9 and 4, respectively, thus getting a sum of 13, which is not less than 6; hence, the output shows it as false value):



The Switch activity

The **Switch** activity can be used to make a choice. When we have various options available and want to execute one option, we frequently use the Switch activity.

By default, the Switch activity takes an integer argument. If we want to take a desired argument, then we can change it from the Properties panel, from the TypeArgument list. The Switch activity is very useful in the categorization of data according to one's own choice.

Example

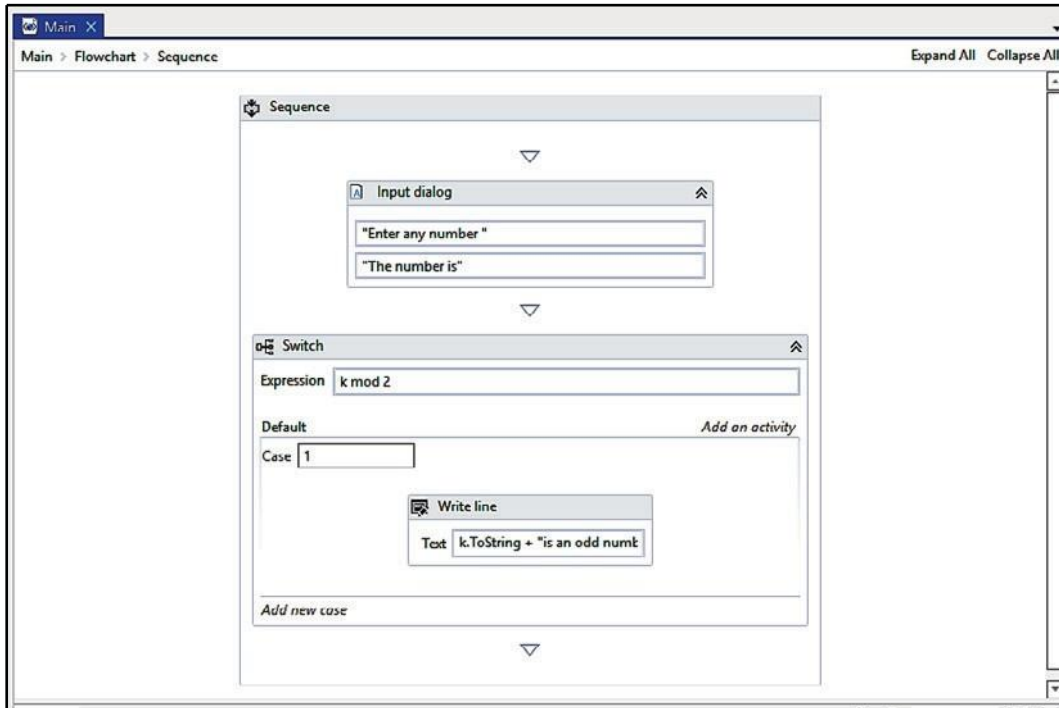
Let's see an example where we have to check whether a given number is odd or even.

We know that all odd numbers, when divided by 2, leave a remainder of 1. On the other hand, even numbers, on being divided by 2, leave a remainder of 0. Hence, we will have two cases getting a remainder of 1 or 0.

Perform the following steps:

1. Add a **Sequence** activity.
2. Add an **Input dialog** activity inside the **Sequence**.
3. Now, create an integer type variable `k`.
4. Specify the newly created variable's name in the **Result** property inside the **Properties** panel.
5. Add the **Switch** activity under the **Input dialog** activity.
6. In the **Expression** field, set `k mod 2` to check whether the number is divisible by 2 or not.
7. Add a **Write line** activity to the **Default** section and type the `k.ToString` + "
is an even number"
in the text field.

8. Now, create **Case 1**, add the one other **Write line** activity to it, and type `k.ToString + " is an odd number"` in the text field:



Step-by-step example using Sequence and Flowchart

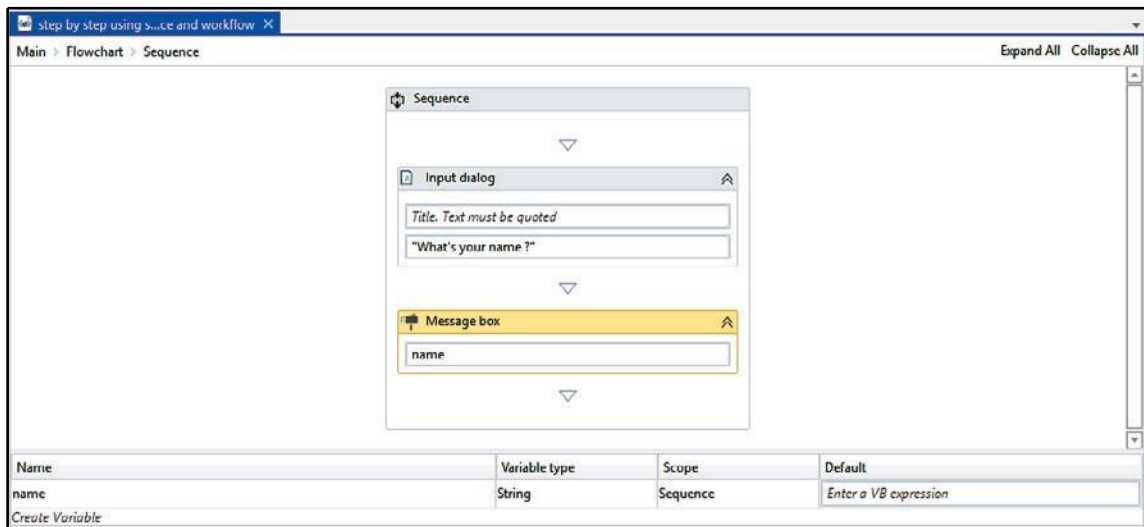
A Sequence and a Flowchart are similar concepts. They are both used to contain logical steps or actions. One should know when to use each of them. Where they differ from each other is that a Sequence is generally used to contain multiple steps to perform an action. A Flowchart, on the other hand, is suitable for a particular task. When we have lots of steps of a similar kind, we contain them in a Sequence. There may be different Sequences doing their jobs. We can easily put similar Sequences into a workflow; each workflow represents a task. It is very easy to test a separate workflow alone. Let us try to understand them better with an example.

How to use a Sequence

There may be different Sequences doing their jobs. We can easily put similar Sequences into a workflow; each workflow represents a task. It is very easy to test a separate workflow alone. Let us try to understand them better with an example.

Perform the following steps:

1. Drag and drop a **Flowchart** onto the Designer panel. Drag and drop a **Sequence** activity. Connect the **Sequence** activity with the **Start** node.
2. Double click on the **Sequence** activity. Drag and drop an **Input dialog** activity and a **Message box** activity. Specify a message in the **label** property of the **Input dialog** activity.
3. Create a variable of type **String**. Give it a name. Also, specify this newly created variable's `name` in the content property of the **Message box** activity:



Hit the **Run** button or press *F5* to see the result.

We can see clearly that we have used two activities inside the **Sequence** that are logically related (one for inputting the name and the other for popping it up). Here, the **Sequence** contains two activities. Of course, you can put as many activities as you want inside the **Sequence**; it will be executed according to the order in which you have defined it.

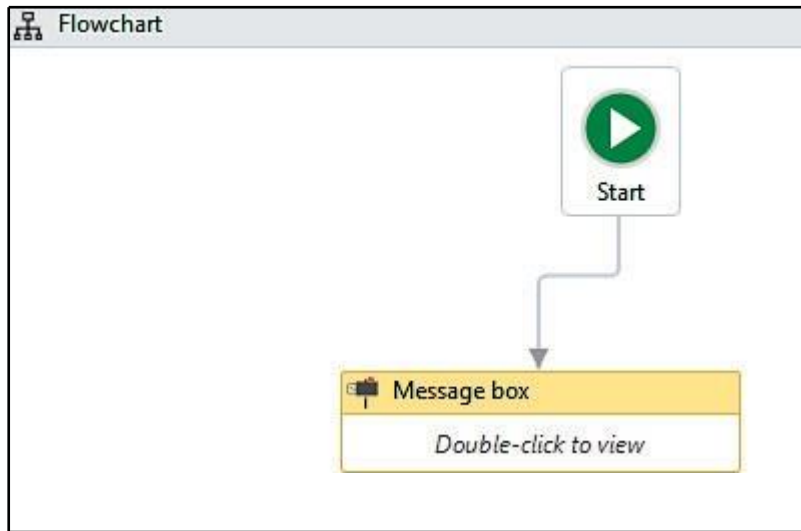
How to use a Flowchart

Previously, we have seen how to use a Sequence and activities. We shall now learn how to use a **Flowchart**. A **Flowchart** is a container. It can contain activities inside it.



In order to use email activities in example please install **UiPath.Mail.Activities**. You can find it by clicking on Manage package icon or pressing *Ctrl + P* and search for mail in all package. You will learn more on packages in [Chapter 6, Taking Control of the Controls](#).

Let us drag and drop a **Message box** activity inside the **Flowchart**. Double click on the **Message box** and type "Hello World!" in the area where the text is to be quoted. Press *F5* to see the result):



So, when the program has only a few steps, we can use activities directly inside the **Flowchart**. However, it becomes more complex when we have a large number of steps. That is why it is necessary to arrange the related activities into Sequences and then group the Sequences into a **Flowchart**.

Let us take an example to see how to use Sequences in the **Flowchart**.



We are not going to implement the actual code for sending the email. It will be covered later in detail. The aim of this session is to clearly understand where and how we use workflows and Sequences.

Perform the following steps:

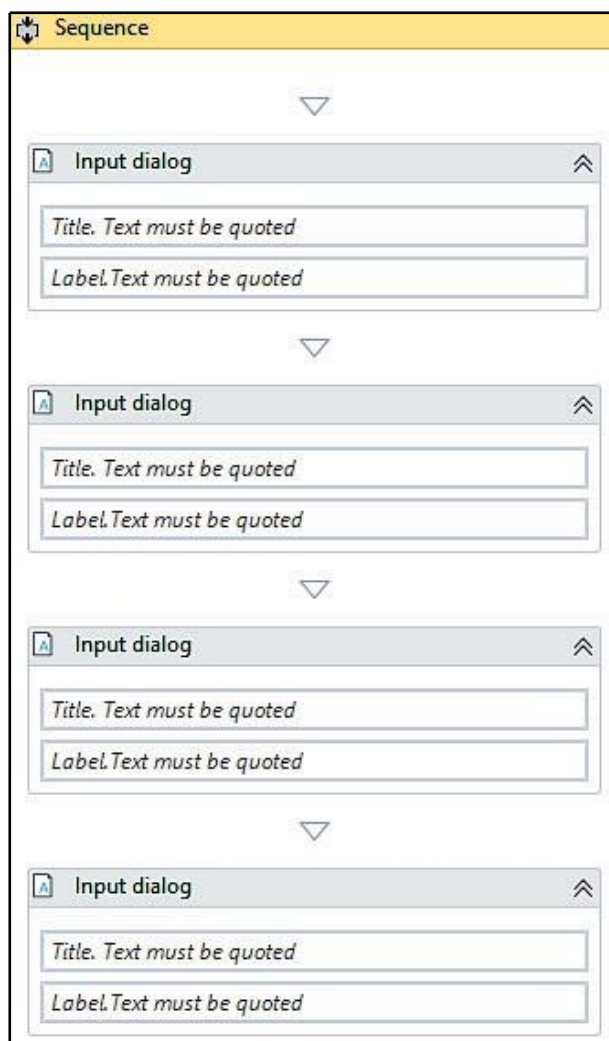
1. Drag and drop two **Flowchart** activities on the main **Flowchart**. Rename them as `Send Mail` and `Message`.

We have two different workflows. The **Send Mail** workflow will send the mail to an email address. The **Message** workflow has the message body of that email and will ask the user for a name, message, sender, and receiver.

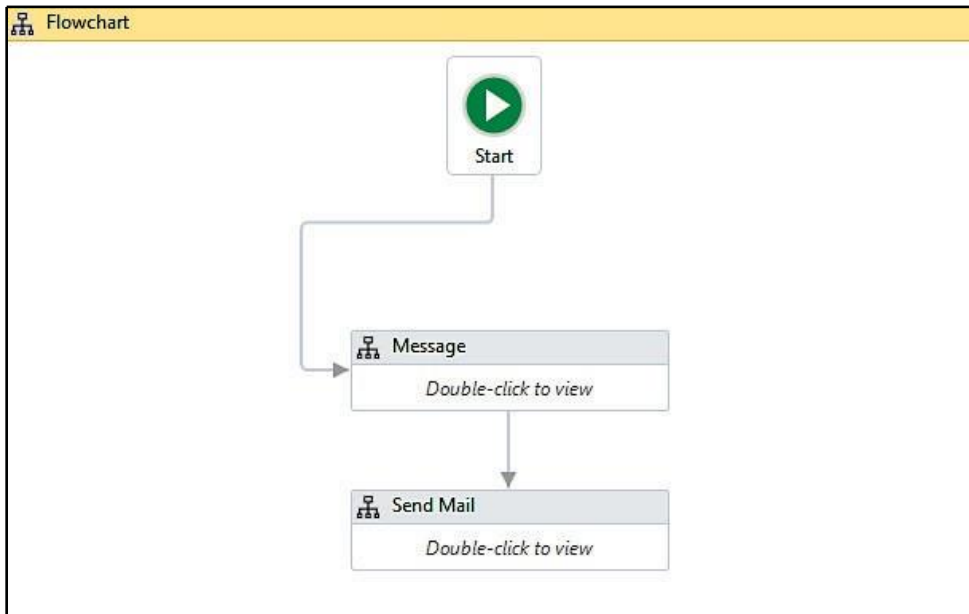
2. We have to implement the desired steps in both workflows.

For that, we are using a **Sequence** inside the **Flowchart**. Double click on the **Flowchart**. Drag and drop a **Sequence** activity inside both Flowcharts. Connect the **Sequence** to the **Start** node by right-clicking on the **Sequence** and selecting the **Set as Start node** option.

3. Double click on the **Sequence** in the **Message** Flowchart. Drag and drop four **Input dialog** activities for the name, message, sender, and receiver (in this **Sequence**, we are not going to set any property of the **Message box** since the purpose of this lesson is to clearly understand where and how we use workflows and Sequences):



4. Double click on the **Send Mail Flowchart**. Double click on the **Sequence**. You can drag and drop the email activities here. (We are not going to drag any mail activity although you are free to do so. There is another chapter for that).
5. That's it. Now, go to the main **Flowchart**. Connect the **Message Flowchart** to the **Start** node. Also, connect the **Send Mail** activity to the **Message Flowchart**:



6. Run the program and visualize it.

Step-by-step example using Sequence and Control flow

In this session, we are going to discuss Control flow with an example. We will see how to use Control flow in a Sequence. There are the various Control flow activities, as mentioned before.

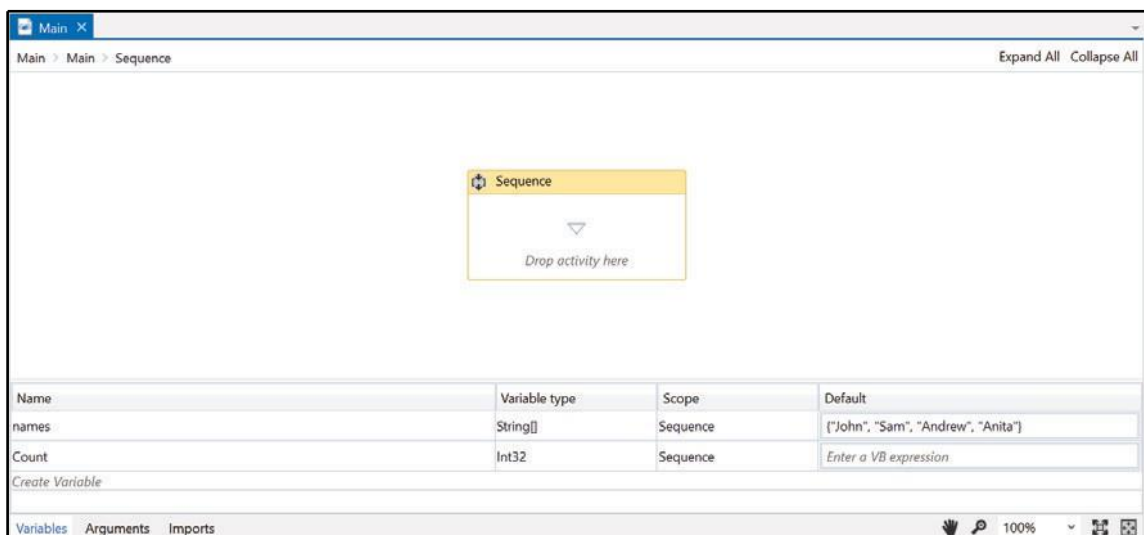
Consider an array of names. Say we have to find out how many of them start with the letter *a*. We will then create an automation where the number of names starting with *a* is counted and the result is displayed.

Perform the following steps:

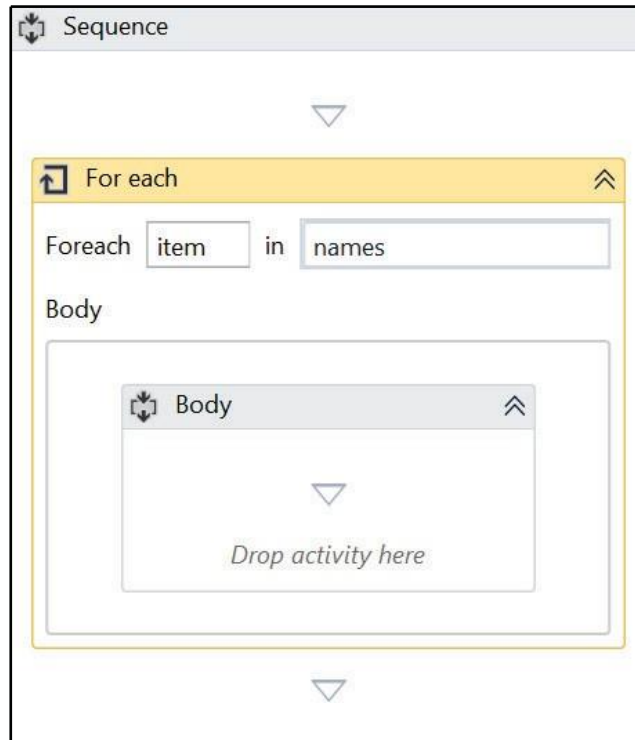
1. Drag and drop a **Flowchart** activity from the **Activities** panel.
2. Drag and drop a **Sequence** activity inside the **Flowchart**. Connect the **Sequence** to the **Start** node by right-clicking on the **Sequence** activity and selecting the **Set as Start node** option.
3. Double click on the **Sequence** activity. Create a variable. Give it a name (in our case, we will create an array of type string and name the variable as `names`). Set the variable type to **Array of [T]**. When asked for the type of array, select **String**.

Also, initialize the array in the **Default** section of the variable by giving it a default values. For example, `( "John", "Sam", "Andrew", "Anita" )`.

4. Create a variable of type integer **Count** for storing the result. Set the variable type to **Int32**:

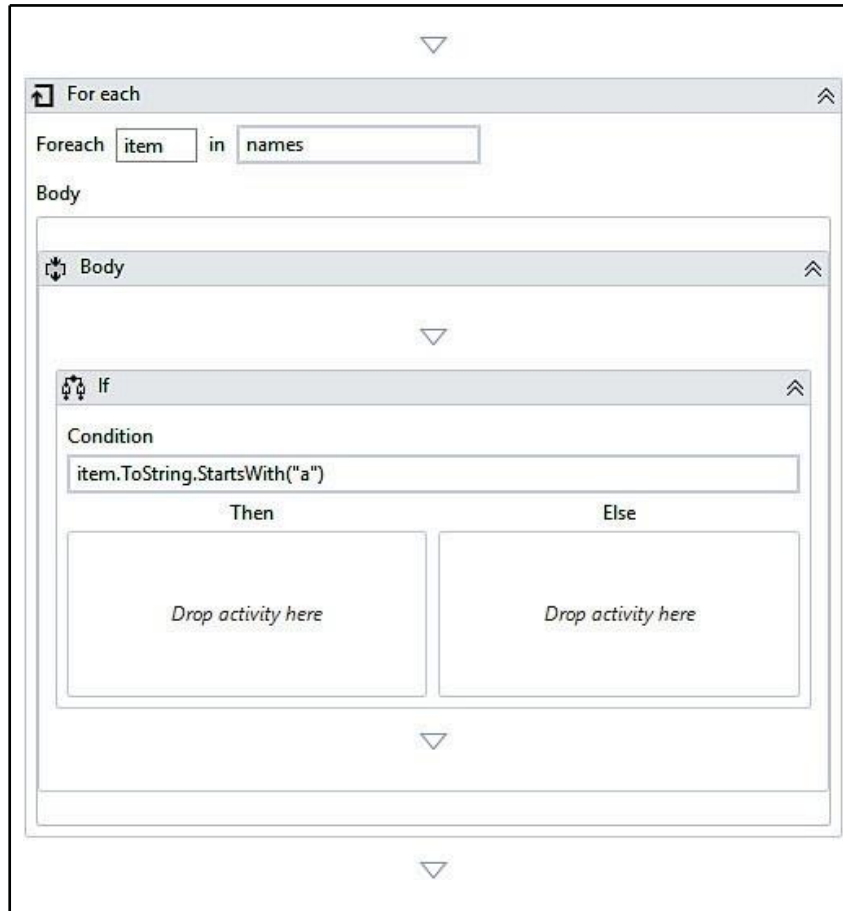


5. Drag and drop a **For each** activity inside the **Sequence**. Also, specify the array name in the expression box of the **For each** activity. The **For each** activity is used to iterate over the array. It will pick up one name from the array each time until it reaches the end:

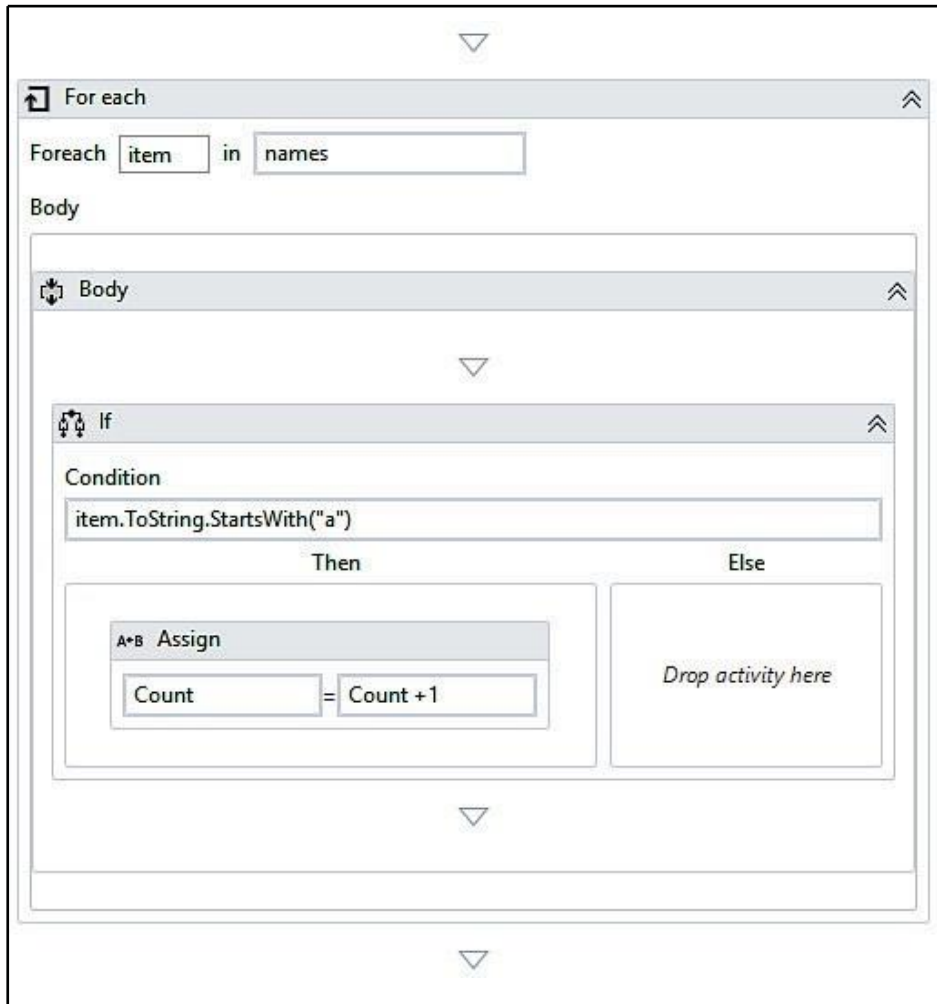


6. Drag and drop the **If** activity from the **Activities** panel and place it inside the **For each** activity at the location where **Drop activity here** is mentioned. Specify the condition in the expression box of the **If** activity. The **If** activity is used to check for a particular condition/expression. If that expression is satisfied, the **Then** block will be executed. Otherwise, the **Else** block will be executed.

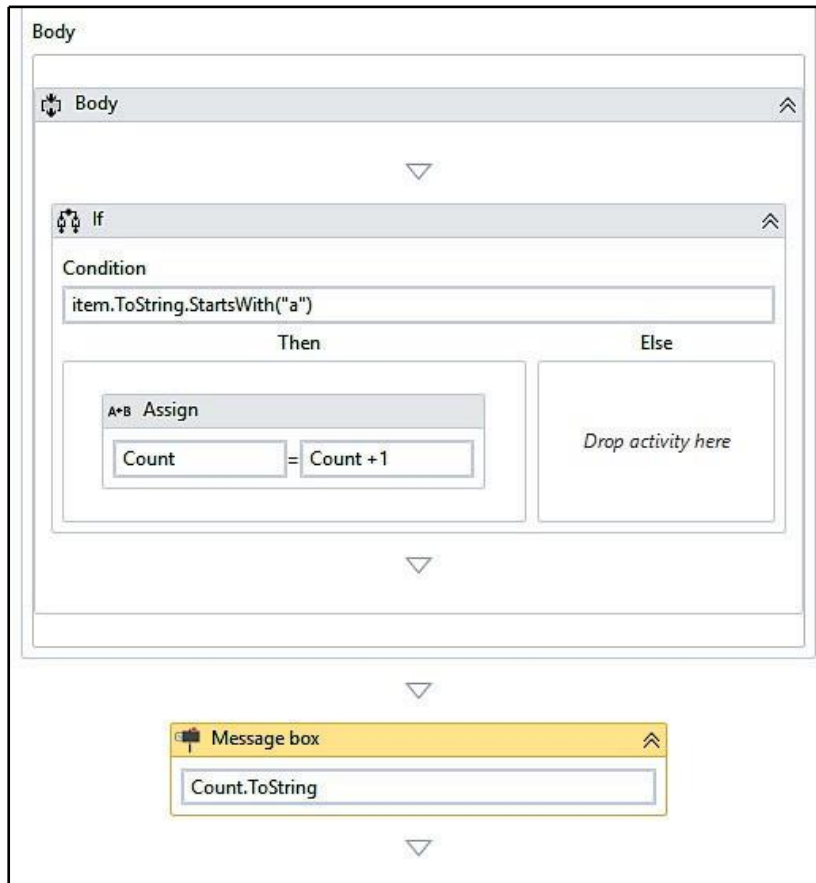
We have specified the expression as `item.ToString.StartsWith('a')`. This expression specifies the name present in the `item` variable starts with the letter 'a'. The **For each** activity iterates over the array, picks up one name at a time, and stores it as a variable, `item`:



7. Now, we are going to use the `Count` variable and increment it each time a name from an array starts with the letter `a`. For this, we have to use the **A+B Assign** activity. Drag and drop the **A+B Assign** activity inside the **If** activity. Set the **To** property to `Count` (variable name) and the **Value** property to `Count+1` (to increment its value) of the **A+B Assign** activity:



8. Just drag and drop a **Message box** activity inside the **Sequence** activity. Specify the count variable in the expression box of the **Message box** activity. But remember, the variable that we have created is of type **Int32**, so, it cannot be used with the **Message box** activity without converting it to a string. To convert it to a string, we have the `'.ToString'` method available in UiPath Studio. Just apply it to the variable and select `'.ToString'`:



Hit the **Run** button or press *F5* and see the result.

Summary

In this chapter, we examined the project that was generated by the recorder and saw an explanation of the structure of the program flow (workflow). We understood the use of Sequences and the nesting of activities. We learned how to use the building blocks of a workflow, Flowchart, and Control flow (looping and decision making).

In the next chapter, we will learn techniques to use memory with variables, and we will also learn about using data tables to store and easily manipulate data in memory. The next chapter will also show how disk files (CSV, Excel, and so on) are used to persist data.

2. Literature Review

4 Data Manipulation

So far we have learned about the basics of RPA and how to organize steps in a workflow using a Flowchart or Sequence. We now know about UiPath components and have a thorough understanding of UiPath Studio. We used a few simple examples to make our first robot. Before we proceed further, we should learn about variable and data manipulation in UiPath. It is not very different from other programming concepts.

However, here we will look at the specifics of UiPath data handling and manipulation.

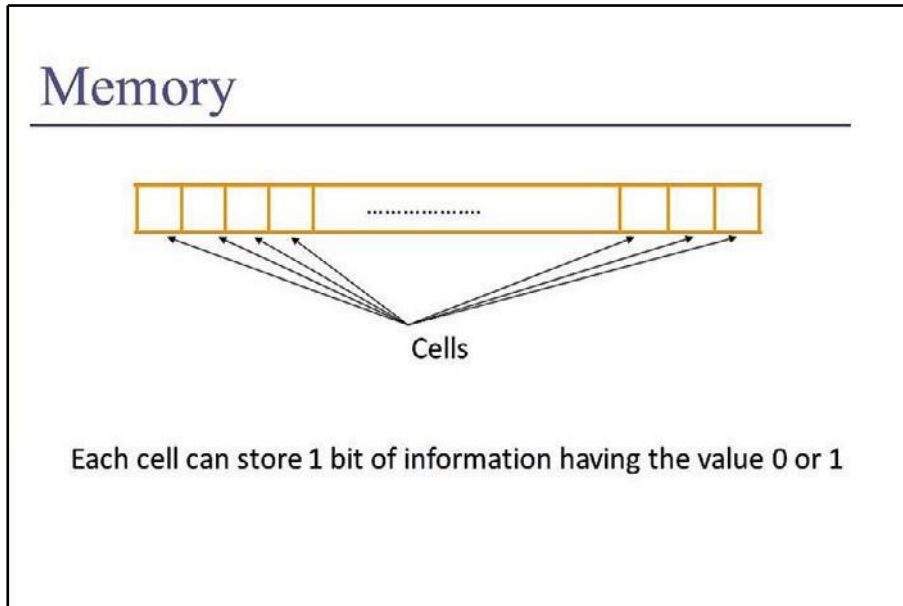
This chapter will mainly deal with data manipulation. Data manipulation is the process of changing data—whether it is adding, removing, or updating it. Before learning about data manipulation, we shall see what variables, collections, and arguments are, what kind of data they store, and what their scope is. We will then carry out various examples of data manipulation. We will also learn to store and retrieve data.

In this chapter, we will cover:

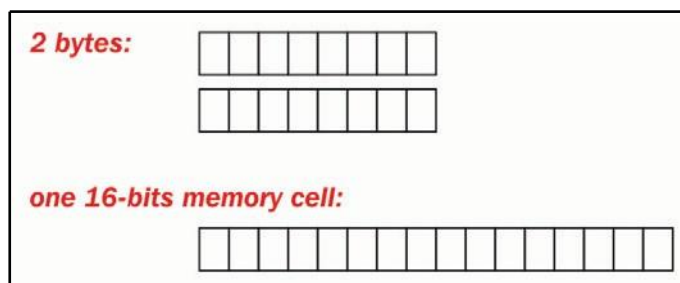
- Variables and the scope of a variable in the workflow
- Collections, how to store data in arrays, and how to traverse them
- Arguments, why we need them, and how to use them
- Clipboard usage
- Data scraping
- File management with a step-by-step example
- Data table usage with an example

Variables and scope

Before discussing variables, let us take a look at **Memory** and its structure:



Memory consists of millions of memory **Cells** and each memory cell stores data in the form of 0s and 1s (binary digits). Each cell has a unique address, and by using this address, the cell can be accessed:



When data is stored in memory, its content gets split into further smaller forms (binary digits). As shown in the preceding diagram, **2 bytes** of data consists of several memory cells.

A variable is the name that is given to a particular chunk of memory cells or simply a block of memory and is used to hold data.

You can declare any desired name and create a variable to store the data.

It is recommended, however, that we use meaningful variable names. For example, if we wish to create a variable to store the name of a person, then we should declare



Name: Andy

It is a good practice to create meaningful variable names. This becomes very useful in debugging the program.

As we discussed, a variable is used to store data. Data is present around us in different types—it can be an mp3 file, text file, string, numbers, and so on. That is why variables are associated with their respective data types. A particular type of variable can hold only that type of data. If there is a mismatch between the data and the variable type, then an error occurs. The following table shows the type a of variable available with UiPath:

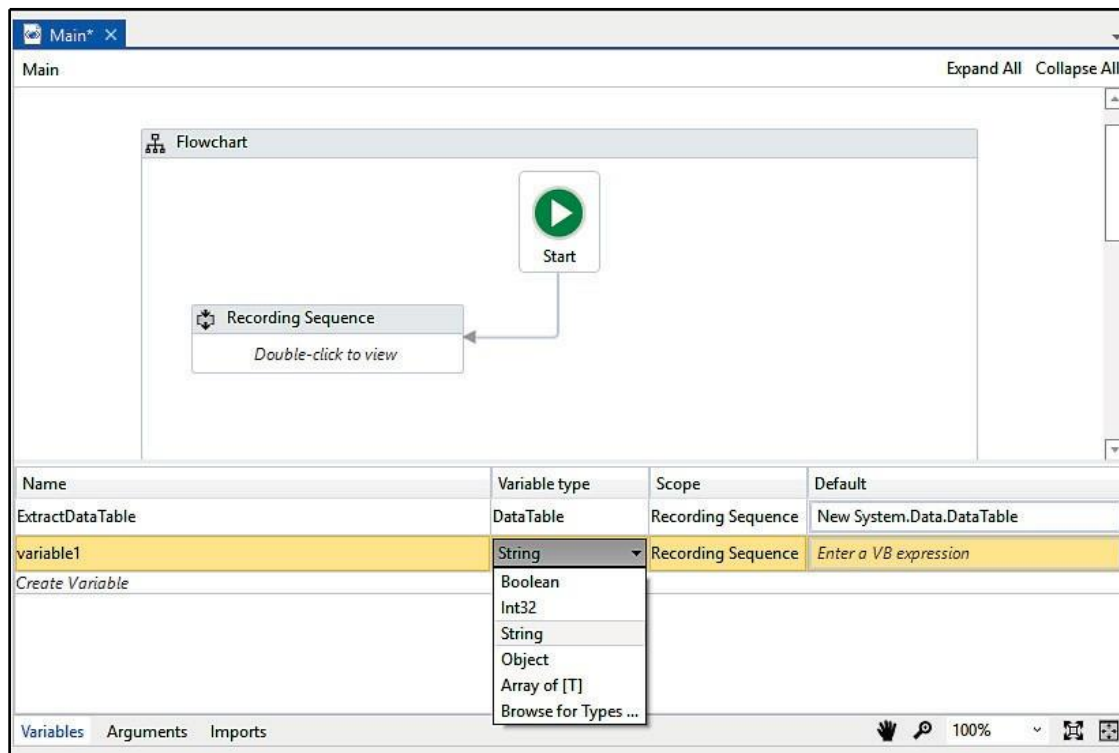
Type	Content
Integer	Whole numbers
String	Text of any kind: "The Quick Fox @4598"
Boolean	True or false
Generic	Anything

In UiPath, we can declare a variable in the **Variables** section. Just give it a meaningful name and select the appropriate type from the drop-down list.



By a meaningful variable, it is implied that the variable name should not be ambiguous. Try to make it as descriptive as possible so that the person reading the code understands the purpose of the variable. Good examples are `daysDateRange`, `flightNumber`, and `carColour`. Bad examples are `days`, `dRange`, `temp`, and `data`.

We can also specify the scope of a variable. The **Scope** is the region under which the data has its effect or availability. You can choose the **Scope** of the variable according to your requirements; try to limit it as far as possible. Please refer to the following screenshot to understand **Variables** panel:



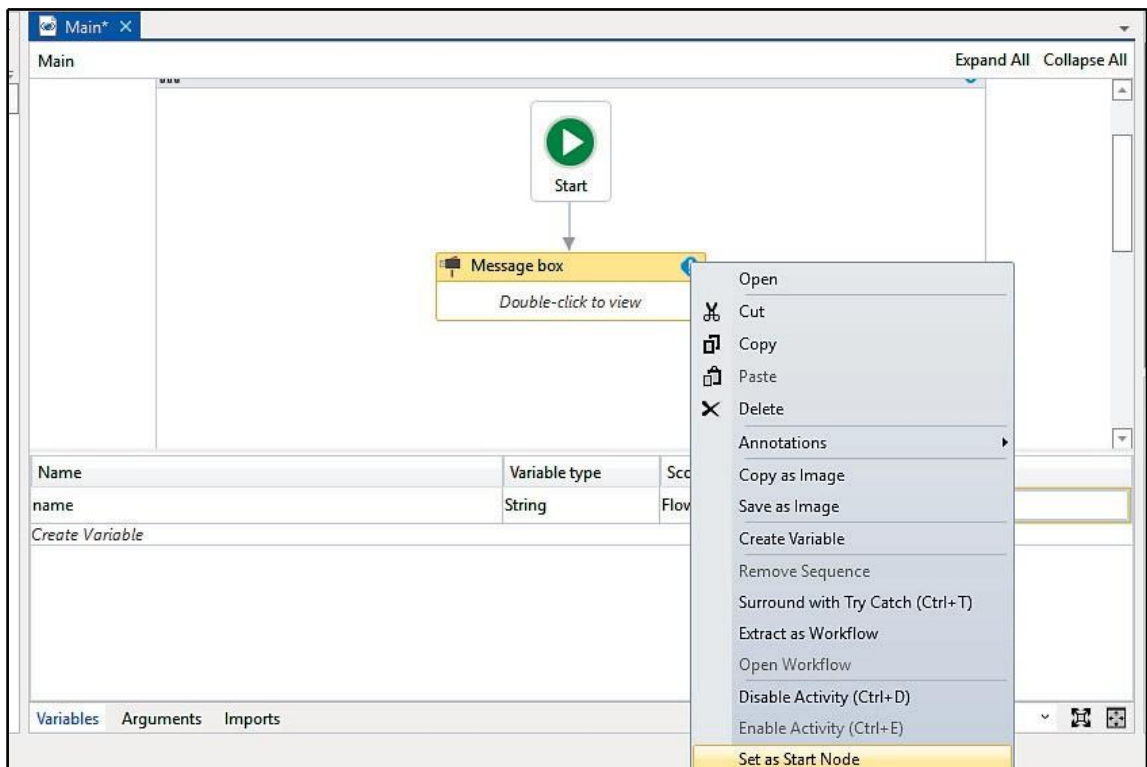
Creating a variable



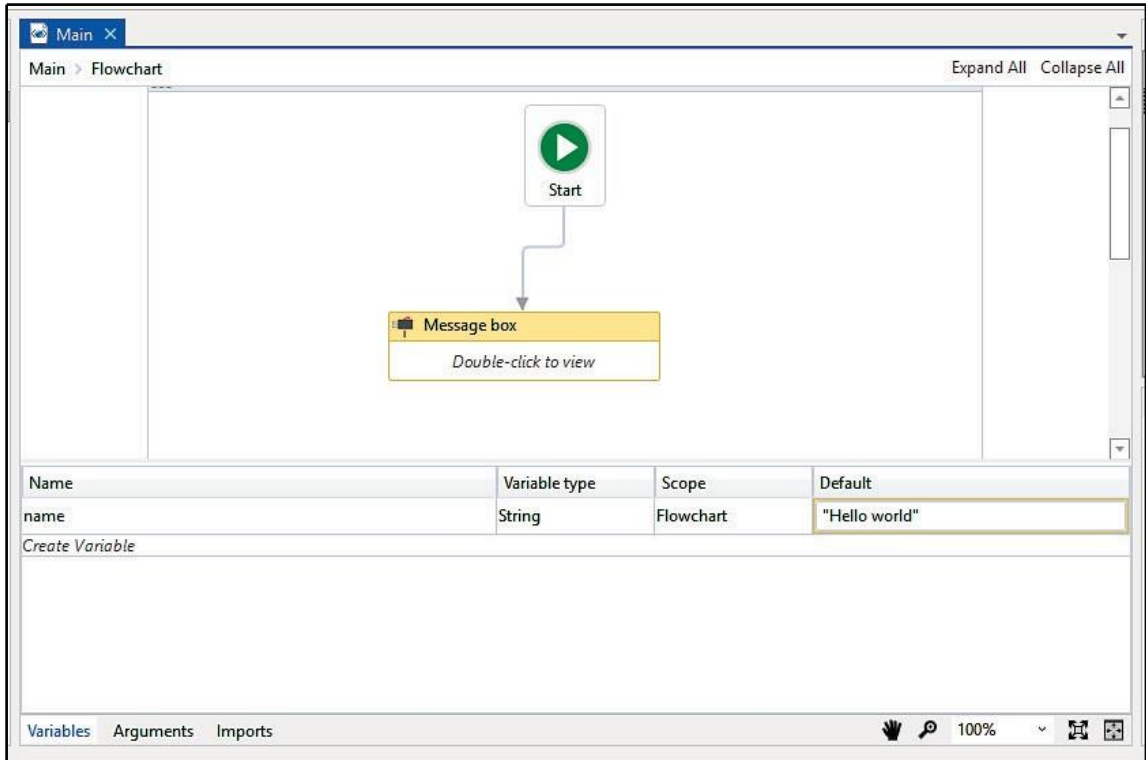
For security reasons, it is not a good practice to set the **Scope** of your variable to fullest as it may accidentally be accessed by another region or could be modified.

Let us take an example of creating a variable and then displaying a **Message box** using that variable:

- 1 We have declared a variable as `name` in the **Variables** section and set its **Default** value to "Hello world". By default, the type of the variable is **String** (we can change its type according to our needs).
- 2 Search for `Message box` in the **Activities** panel. Drag and drop that **Message box** template into a **Flowchart**.
- 3 Right-click on the message template and select **Set as Start node**:



- 4 Double-click on the **Message box** template and specify the variable name that we created earlier. At this stage, we are ready to run our application by simply clicking on the **Run** button:



A dialogue box will pop up with the "Hello world" text displayed on it.

Collections

There are different types of variables. Variables can be classified into three categories:

- **Scalar:** These are variables that can only hold a single data point of a particular data type, for example; Character, Integer, Double, and so on.
- **Collections:** These are variables that can hold one or more data point of a particular data type. For example; array, list, dictionary, and so on.
- **Tables:** These are a tabular form of the data structure which consists of rows and columns.

In this section, we are going to see how collections work and how we can store values in the collection variables.

In a collection, we can store one or more data points, but all the data must be the same. Consider an example. An array is a collection in which we can store different values of a particular data type. It is a fixed data type, meaning if we store five values inside the array, we cannot add or remove any value/values in that array.

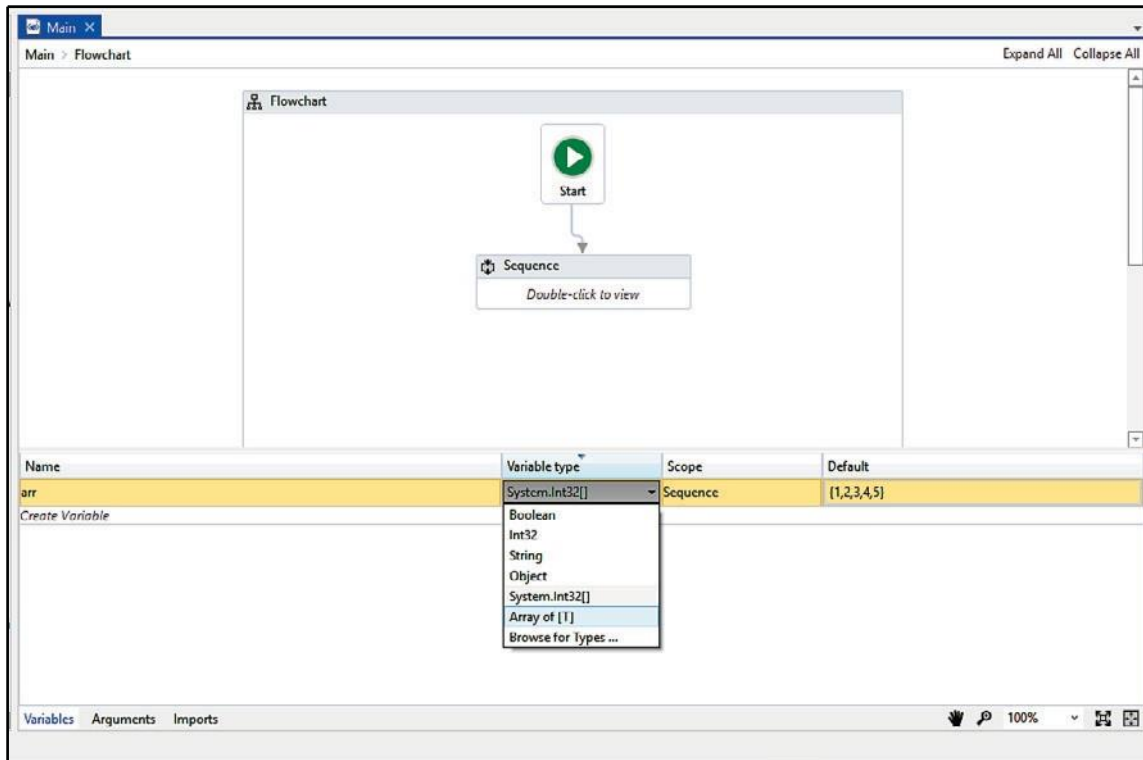


The object is a data type in which you can store any type of data. Hence, if we take an array of objects then we can store different types of data in an array. This is an exceptional case.

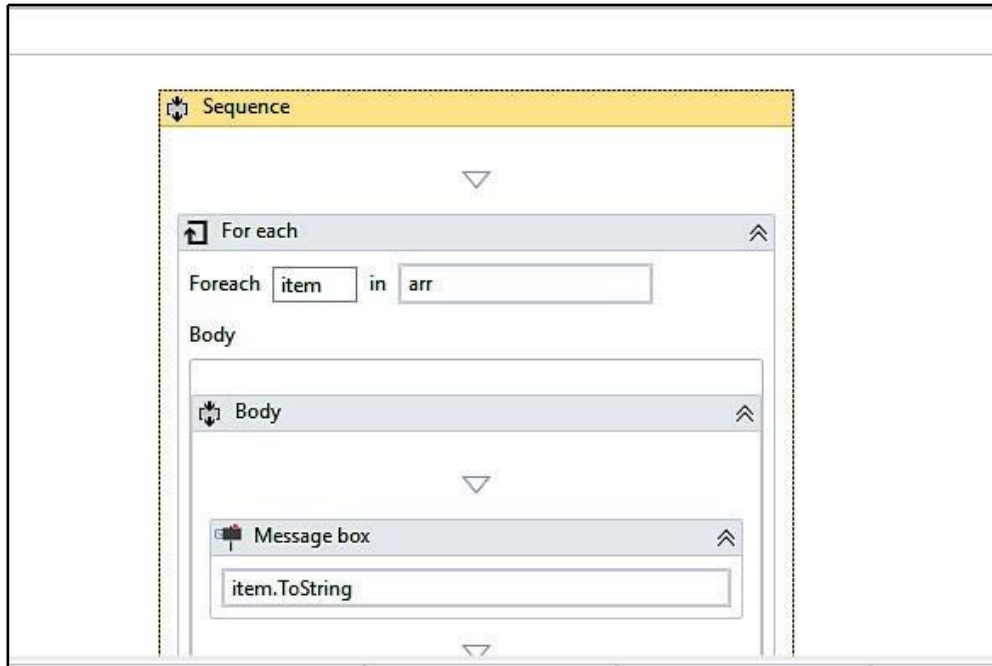
Let us see how we can use an array with an example. In this example, we are going to take an array of integers, initialize it, and then iterate through all the elements of the array:

1. Drag and drop a **Flowchart** activity onto the main Designer panel, and drag and drop a **Sequence** activity inside the **Flowchart**. Set the sequence as **Start** node.
2. Create a variable in the **Variables** panel and give it a meaningful name (in this example, we have created a variable named `arr`, which is an array of integers). Choose the data type as an array of integers.

3. We have initialized the array as {1, 2, 3, 4, 5} in the **Default** section. You can initialize it with the **int32** data type:



4. Drag and drop a **For each** activity from the **Activities** panel inside the **Sequence**, and drag and drop a **Message box** activity inside the **For each** activity.
5. Specify the array name in the expression text box of the **For each** activity.
6. Specify the `item` variable that is auto-generated by the **For each** activity, inside the **Message box** activity. But hold on, we have to convert the `item` variable into the **String** type because the **Message box** activity is expecting the string data type in the text box. Just press the dot (.) along with the `item` variable and choose the **ToString** method:



Hit the **Run** button to see the result. All the values will pop up at once.

In this example, we have seen how easily we can initialize the array and iterate through it.

Arguments – Purpose and use

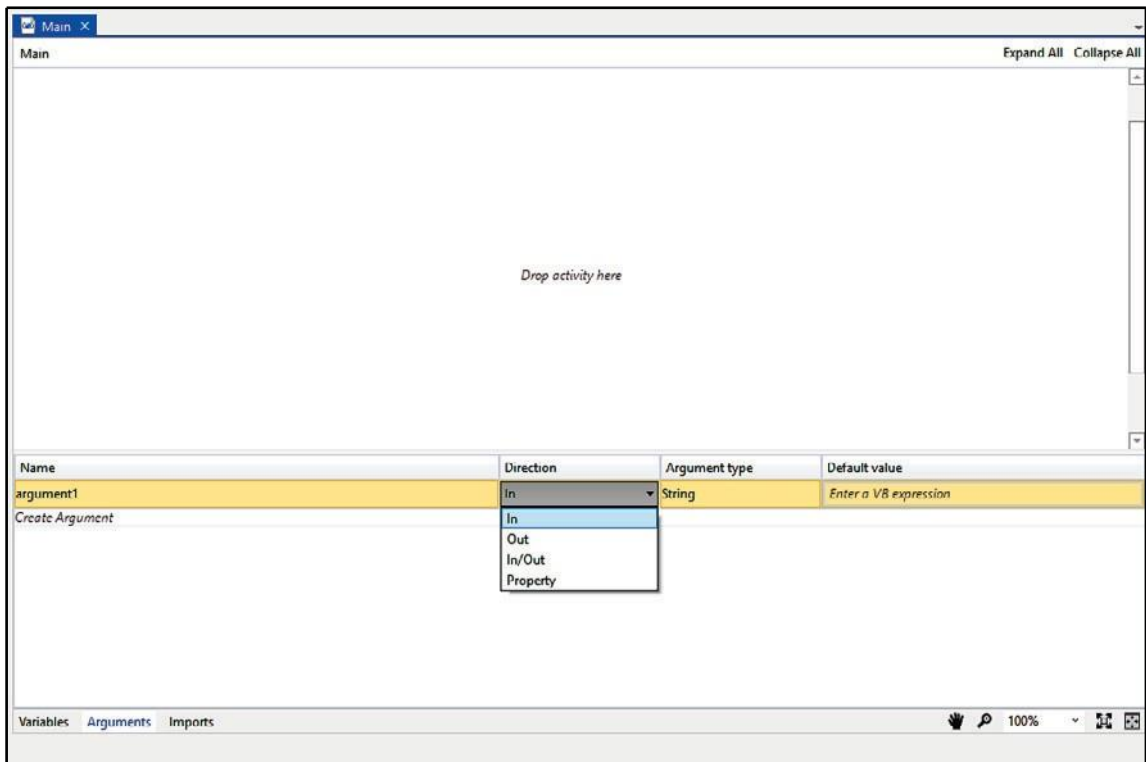
An **Argument** is simply a variable that can store a value. You can create an argument in the Argument section of the main Designer panel.

But remember, they are not limited to variables. An argument has a larger scope than a variable and is used to pass values between different workflows. You might be wondering why we need this. Suppose we have a big project to build; we break down the project into different workflows because smaller workflows can be easily tested separately. It is very easy to build smaller workflows and combine them, thus turning them into the real solution of the project.

These Arguments are used for interacting with different workflows by exchanging data between them. That is why the direction property is associated with Arguments. We can choose the direction on the basis of our requirement—either giving the value to some workflow or receiving the value from another workflow.

We can easily create arguments in the **Arguments** panel. We can also specify the direction:

- **In:** When we have to receive the value from another workflow.
- **Out:** This is the current value if we have to send the value to a workflow.
- **In/Out:** This specifies both; it can take or receive the value.
- **Property:** This specifies that it is not being used currently:



Data table usage with examples

A data table is a tabular form of data structure. It contains rows and each row has columns, for example:

Student name	Roll number	Class
Andrew Jose	1	3
Jorge Martinez	2	3
Stephen Cripps	3	2

The preceding illustration is an example of a data table that has three rows and three columns. You can build a data table in UiPath also.

A data table is used for various purposes. Say, for example, you have to build a table dynamically. You can use a data table as your preferred choice. A data table is also extensively used to store tabular data structures. In data scraping, data tables are widely used. Data scraping is a method in which we can dynamically create tabular data records of search items on the web.

We shall build two projects in which we will use a data table:

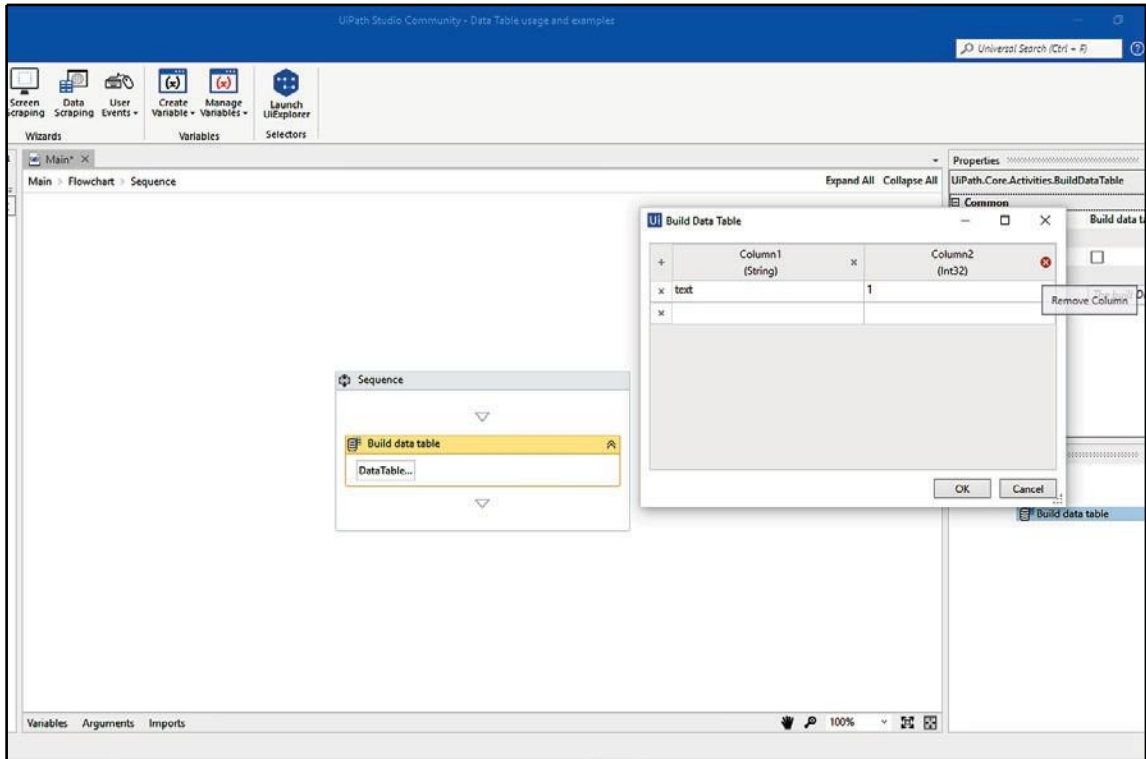
- Building a data table
- Building a data table using data scraping (dynamically)

Building a data table

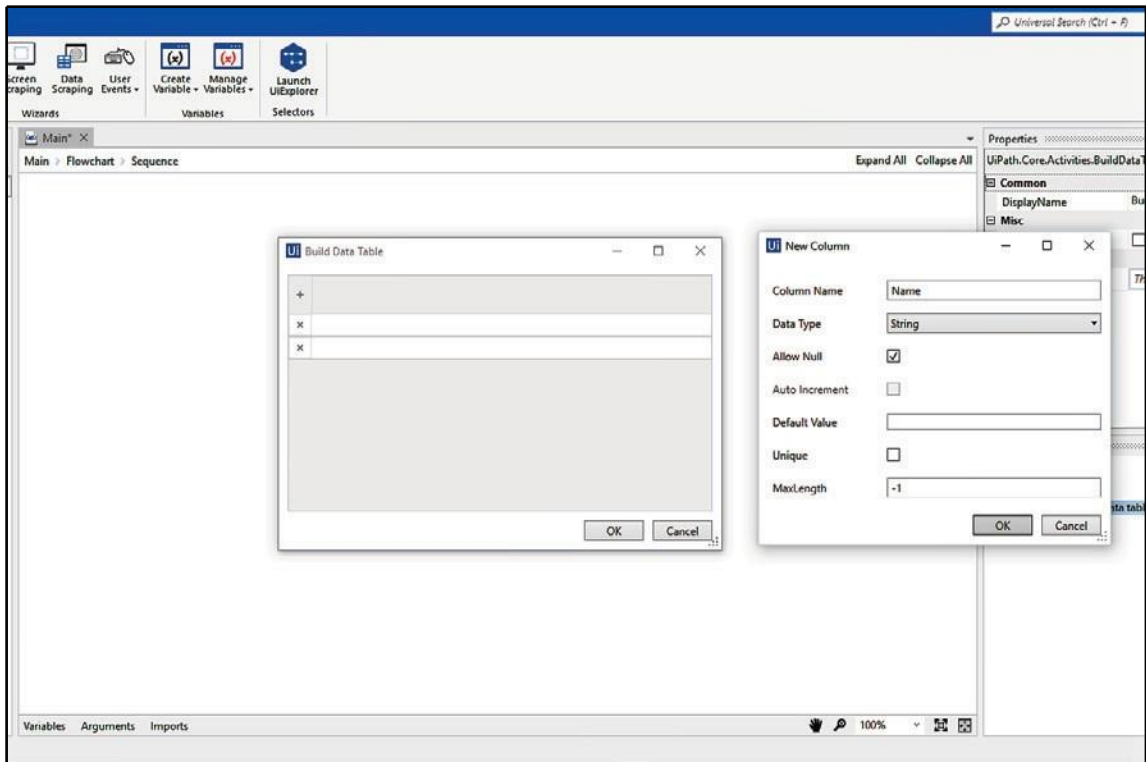
Let us see, how to build a data table can be built. First, create an empty project. Give it a proper name:

1. Drag and drop a **Flowchart** activity on the Designer panel. Also, drag and drop a **Sequence** activity and set it as the **Start** node.
2. Double click on the **Sequence** and drag and drop the **Build Data Table** activity inside the **Sequence** activity.

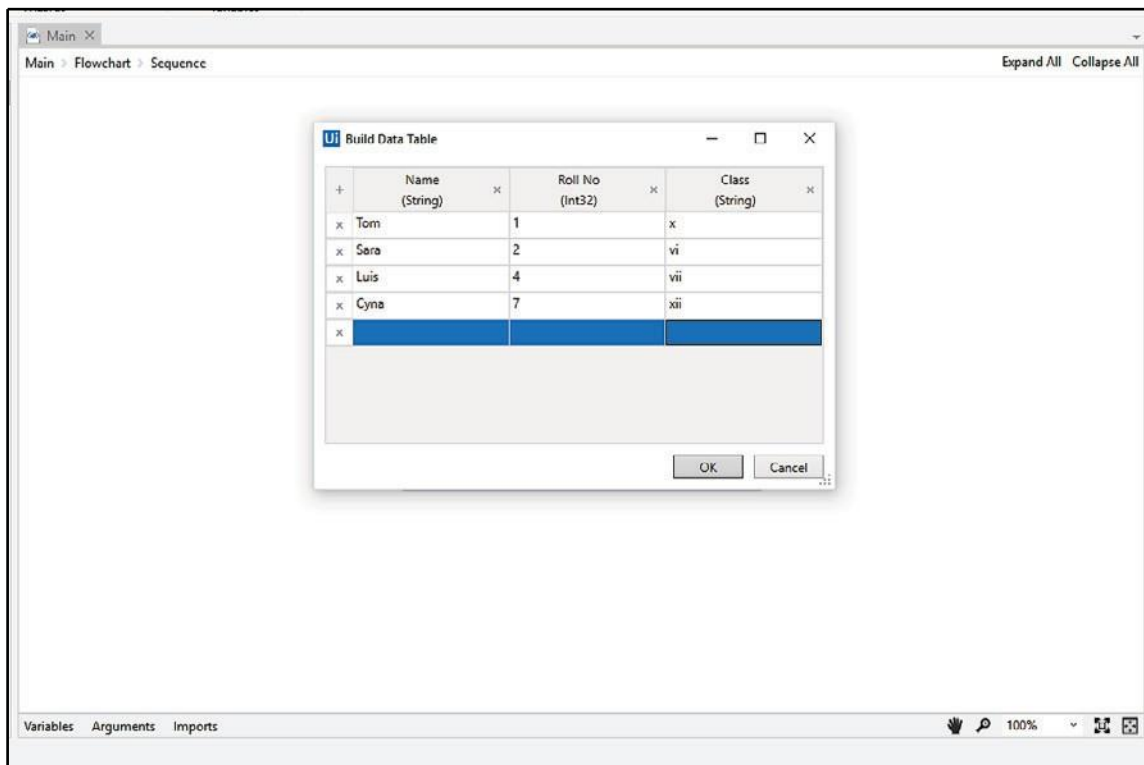
3. Click on the **Data Table** button. A pop-up window will appear on the screen. Remove both the columns (auto generated by the **Build Data Table** activity) by clicking on the Remove Column icon:



- Now, we will add three columns by simply clicking on the + symbol. Specify the column names and select the appropriate data types from the drop-down list. Click on the **OK** button. We will add column **Name** of **String** Data Type, **Roll_No** of **Int32** type and finally **Class** of string type:

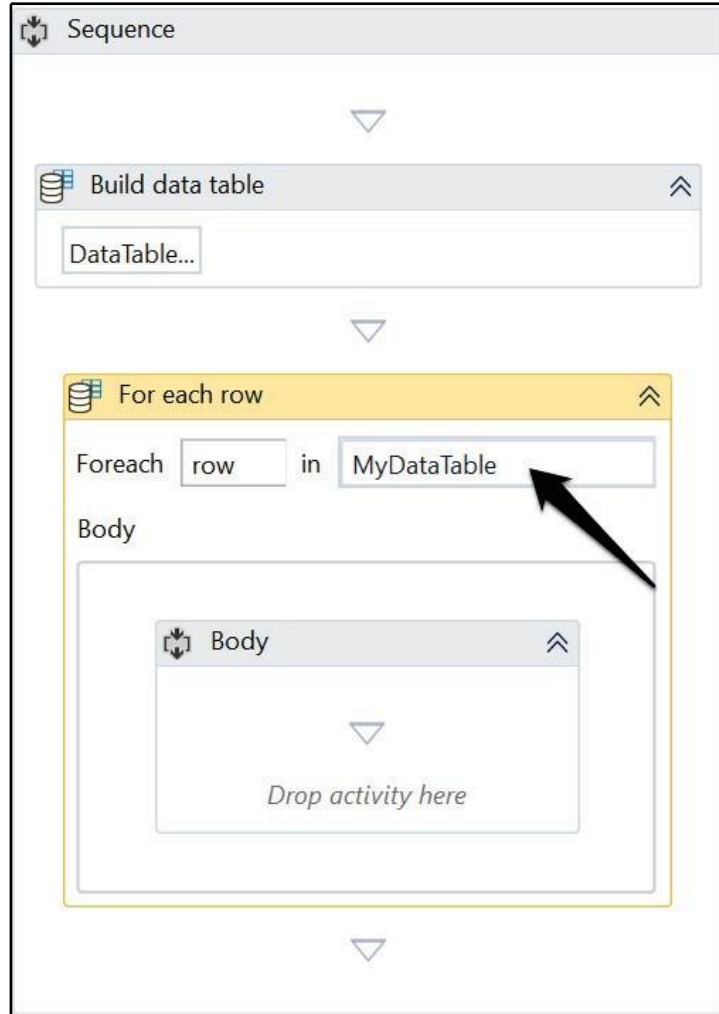


Now enter some random values just to insert the data into the rows:

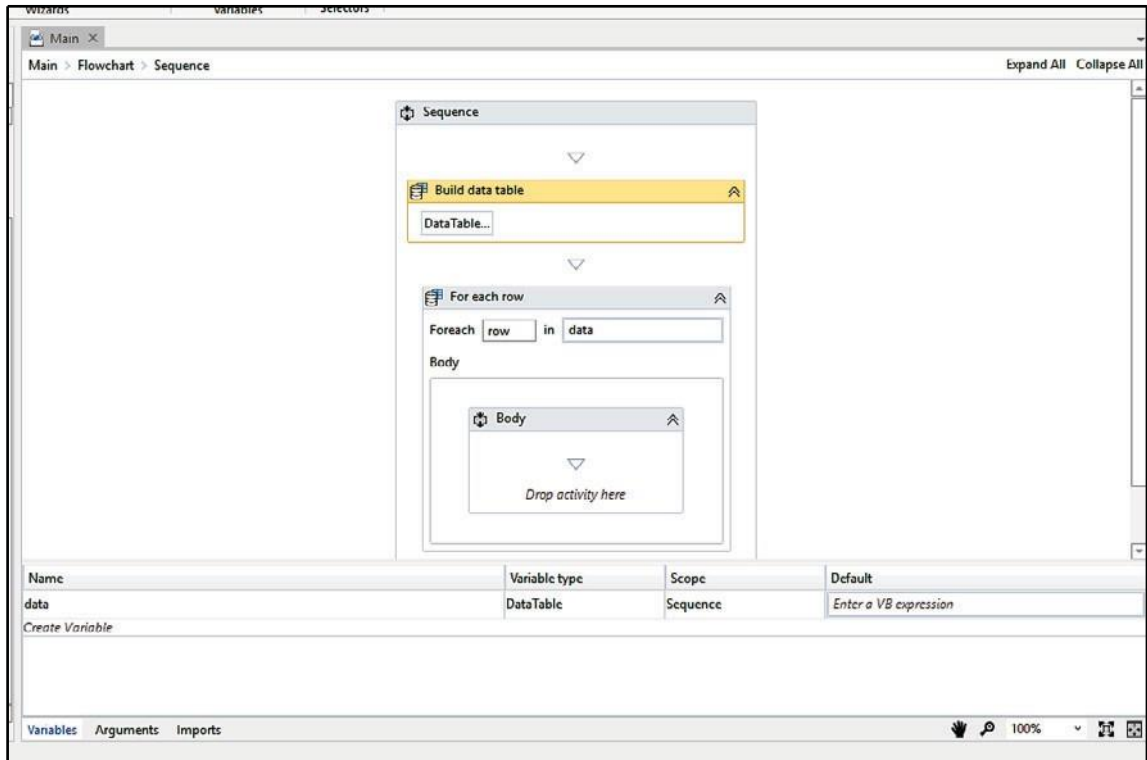


Click on the **OK** button and our data table is ready. We have to iterate over the data table's rows to make sure everything works correctly.

5. In order to store the Data Table created by **Build Data Table** activity, we have to create a data table variable `MyDataTable` of **DataTable** type and in order to store the result of the data table that we have dynamically built. Also, specify assign the **Output** property of the **Build Data Table** activity with this variable. Specify the data table variable's name there.
6. After our data table is ready, we will iterate the data table's rows to make sure everything works correctly. Drag and drop the **For each row** activity from the **Activities** panel inside the **Sequence** activity. Specify the data table variable's name (`MyDataTable`) in the expression text box of the **For each row** activity:

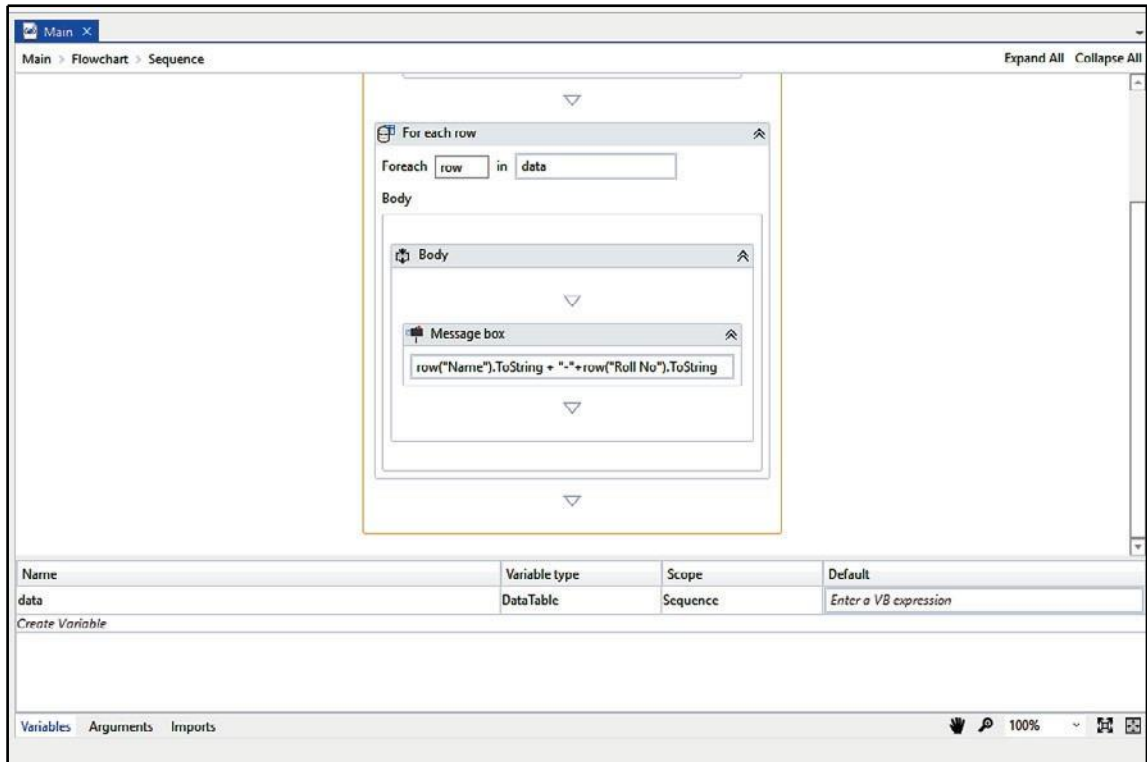


6. Drag and drop the **For each row** activity from the **Activities** panel inside the **Sequence** activity. Specify the data table variable's name in the expression text box of the **For each row** activity:



For each and **For each row** are two different activities. **For each** is used to iterate over the collections, while the **For each row** activity is used to iterate over the data table rows.

7. Drag and drop a **Message box** activity inside the **For each row** activity. In the **Message box** activity, inside the message box we have to write following string: `row("Name").ToString + "-" + row("Roll_No").ToString + "-" + row("Class").ToString`. `row` the variable which holding data for the data row in each iteration:

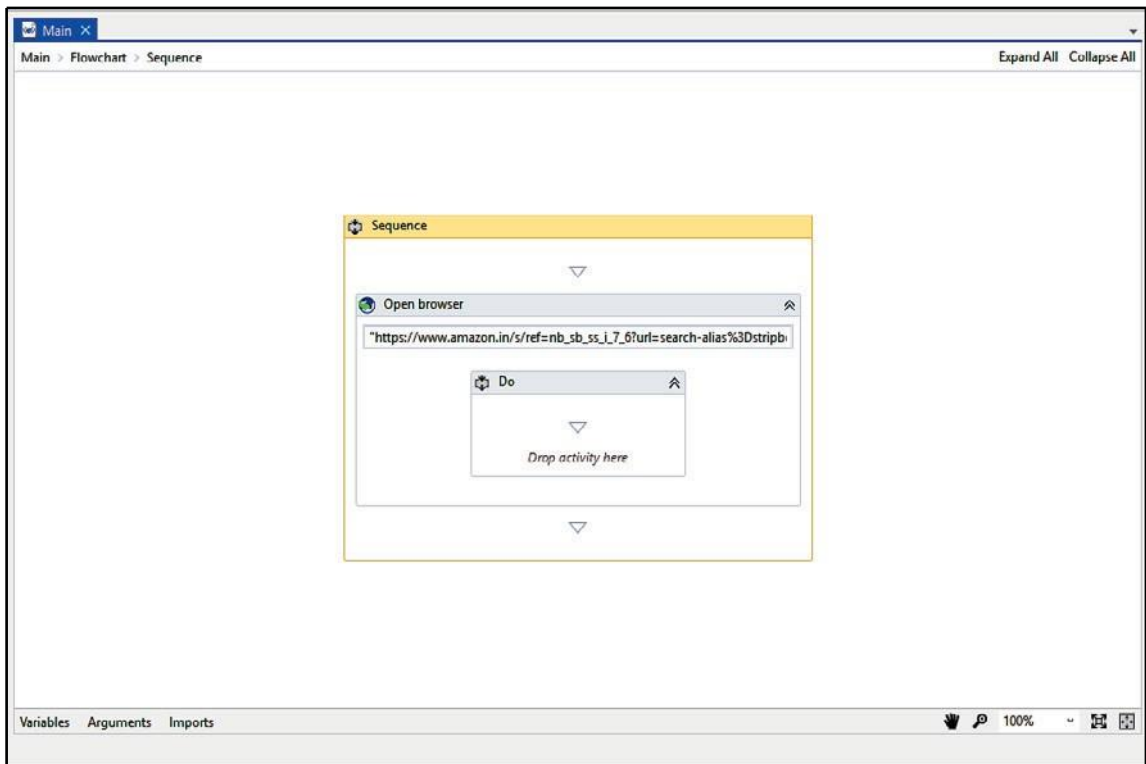


This row variable contains all the columns of a particular row. Hence, we have to specify which column value we want to retrieve by specifying the column name. Instead of the column name, we can also specify the column index (the column index always starts from zero). Hit the **Run** button to see the result.

Building a data table using data scraping (dynamically)

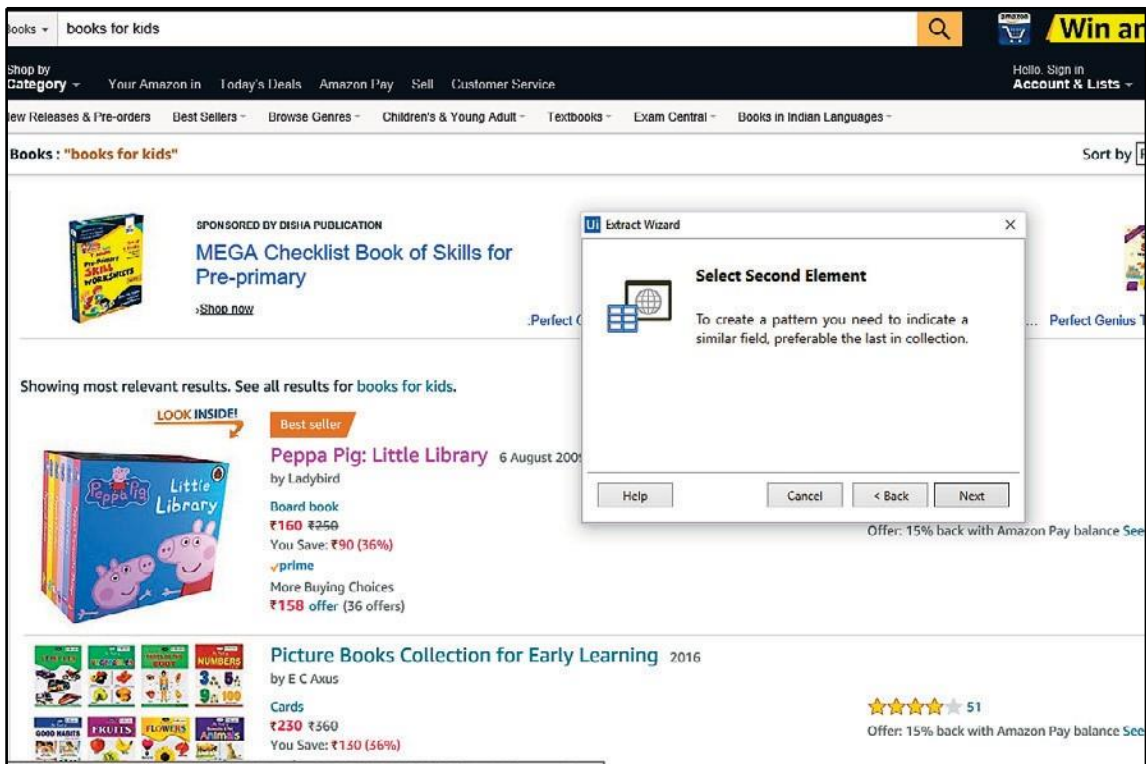
Using data scraping, we can build the data table at runtime. Let us consider an example of extracting data from Amazon's website. Perform the following steps:

1. Drag and drop the **Flowchart** activity from the **Activities** panel, and drag and drop the **Sequence** activity inside the **Flowchart** activity.
2. Double-click on the **Sequence** activity.
3. Drag and drop the **Open Browser** activity inside the **Sequence** activity. Specify the URL in the text box:

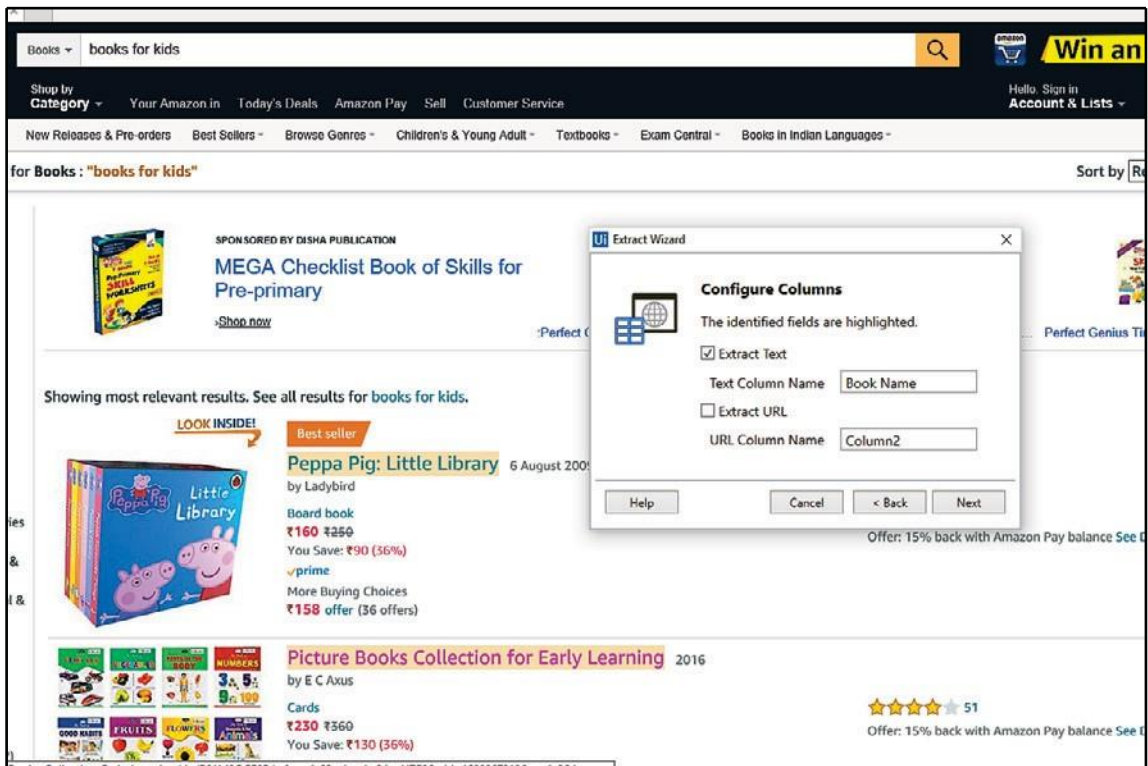


(URL: https://www.amazon.in/s/ref=nb_sb_ss_i_7_6?url=search-alias%3Dstripbooks&field-keywords=books+for+kidss&prefix=books+%2Cstripbooks%2C322&crid=20WJE9AMZYS06)

4. Click on the **Data Scraping** icon on the top left corner of UiPath Studio. A window will pop up. Click on the **Next** button.
5. Now, there will be a pointer pointing to the UI elements of the web page. Click on the name of the book:

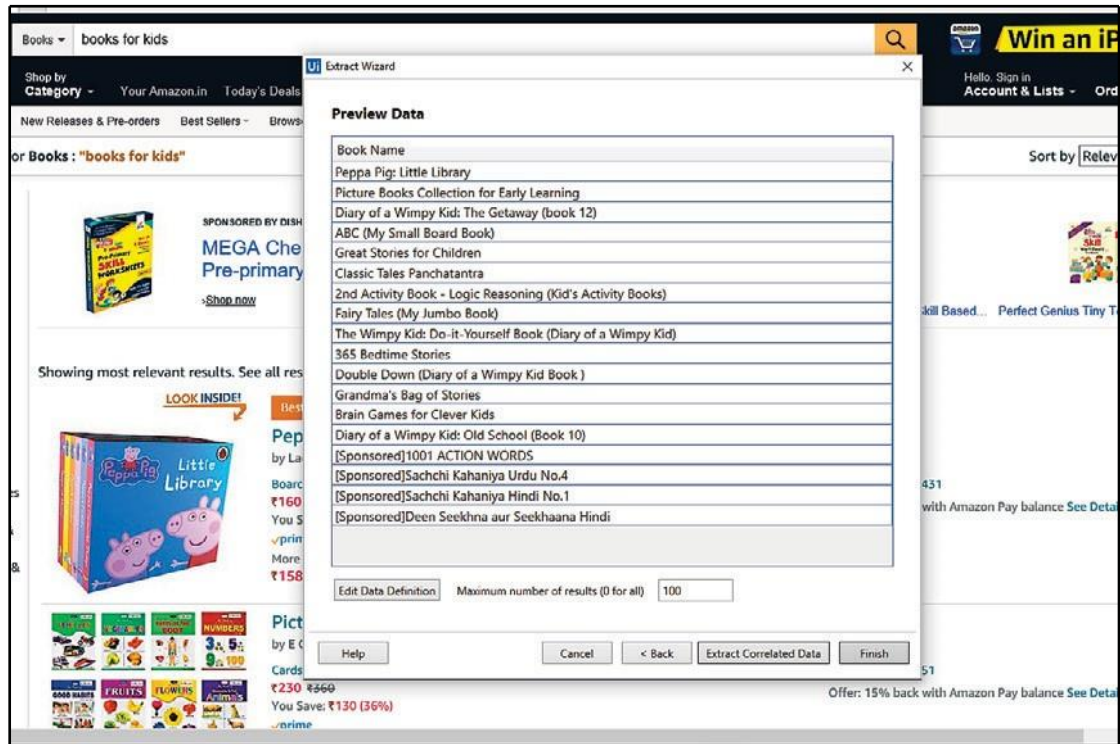


It will ask you to point to a second similar element on the web page:

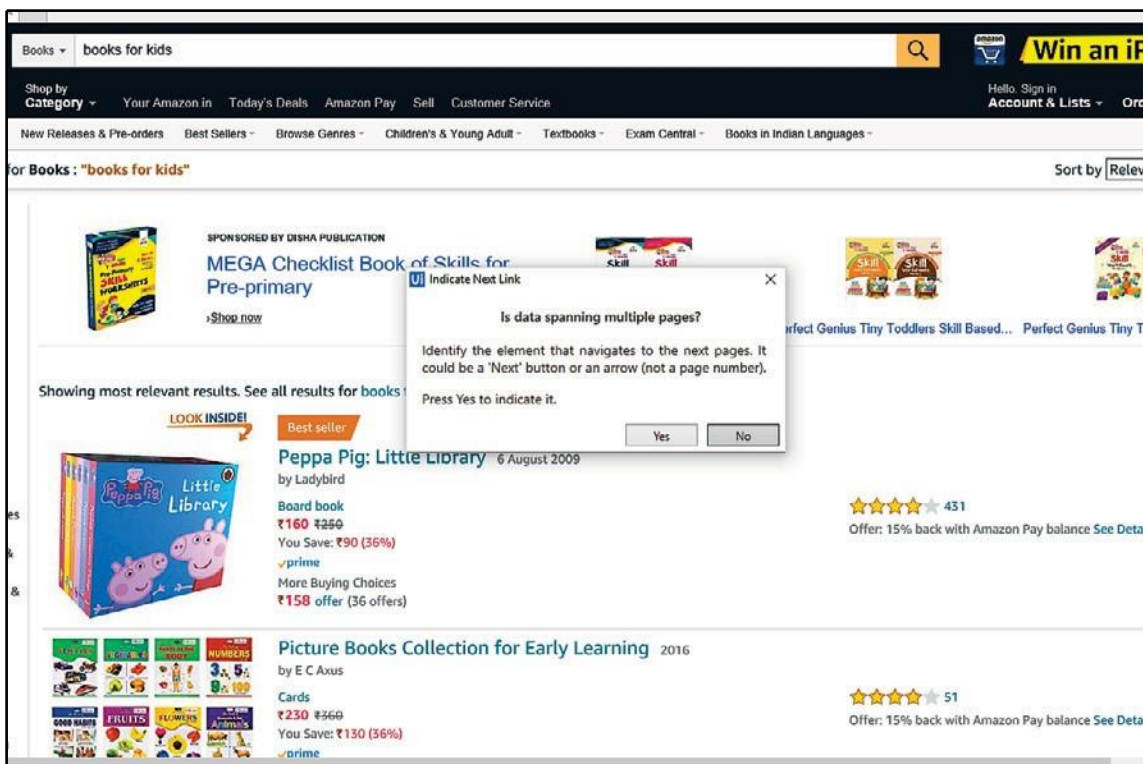


6. Point to a second similar element on that web page. Specify the name that you want to give for that extracted data column. (It will become the column name of the extracted data). Click on the **Next** button.
7. A list of names will appear in a separate window.

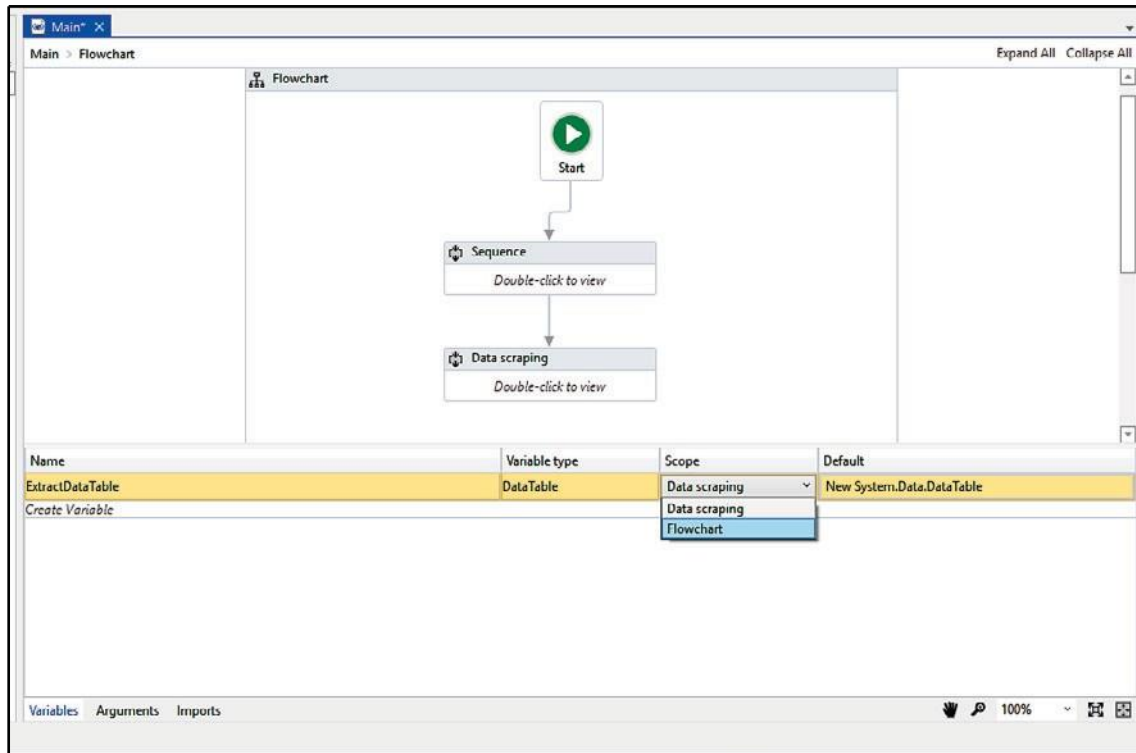
If you want to extract more information, then click on the **Extract correlated data** button and repeat the same process once again (just as we extracted the name of the book from Amazon's website). Otherwise, click on the **Finish** Button:



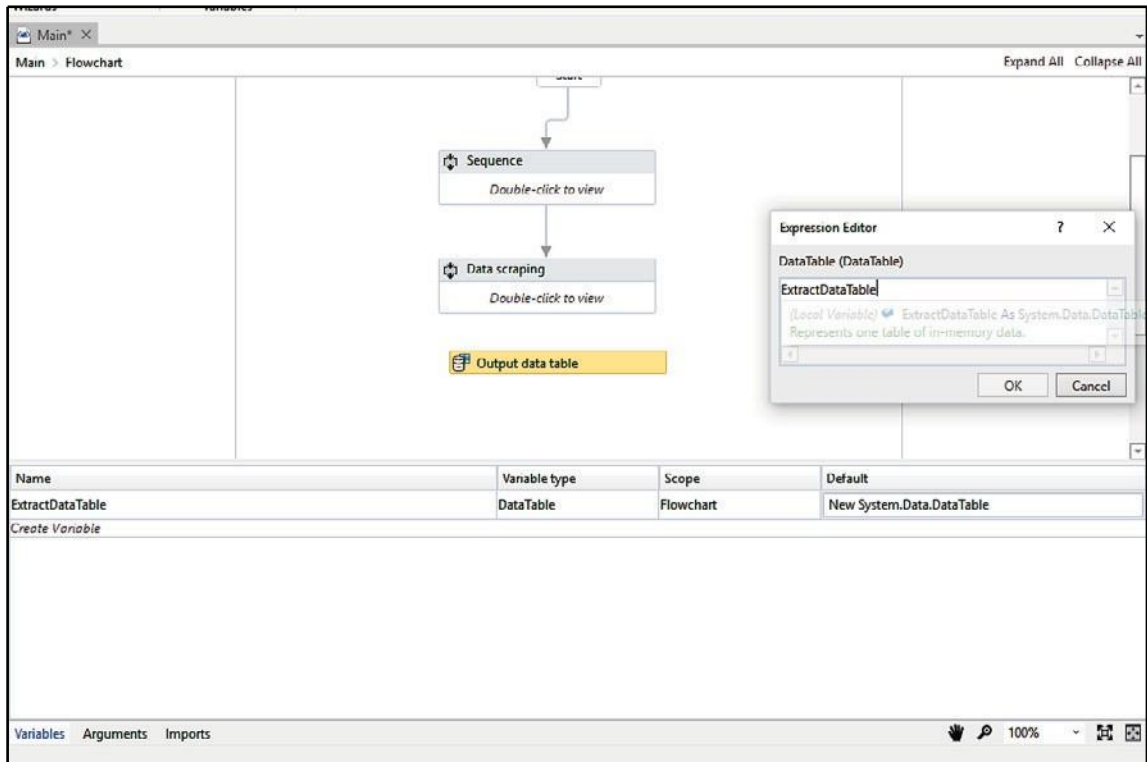
8. It will ask you to locate the next page's button/link. If you want to extract more information about the product and it spans across multiple pages, then click on the **Yes** button and point to the next page's button/link. Then, click on it. If you want to extract only the current page's data, click on the **No** button, (you can also specify the number of rows that you want to extract data from: By default it is 100):



9. Data scraping generates a data table. (In this case, `ExtractedDataTable` is generated.) Change the scope of `ExtractedDataTable` to the **Flowchart** so that it is accessible within the **Flowchart** activity:

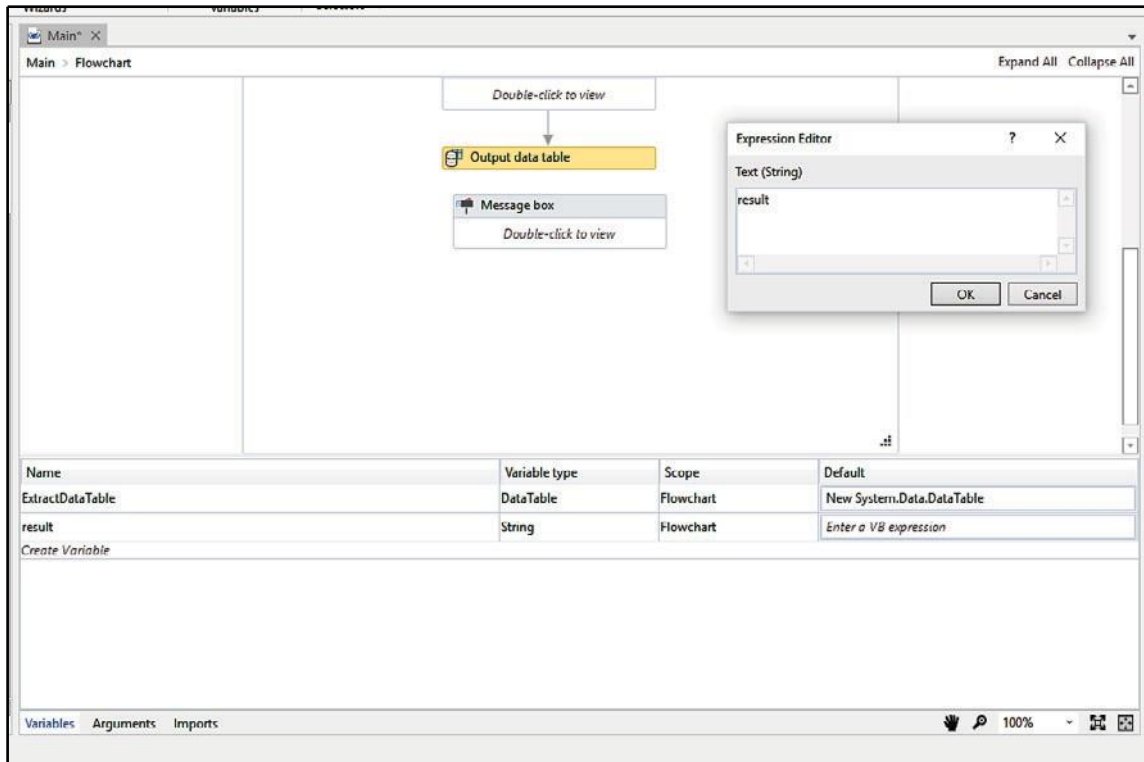


10. Drag and drop the **Output data table** activity on the **Flowchart**. Set the **Output** property of the **Output data table** activity as: `ExtractedDataTable`:



11. Connect the **Output data table** activity to the **Data Scraping** activity. Drag and drop the **Message box** activity on the Designer window. Also create a string variable to receive the text from the **Output data table** activity (in our case, we have created a `result` variable).

Specify the text property of the **Output data table** activity as the `result` variable to receive the text from the **Output data table**:



12. Connect the **Message box** activity to the **Output data table** activity. Double-click on the **Message box** and specify the text property as the `result` variable (the variable that you created to receive the text from the **Output data table** activity).
13. Hit the **Run** button and see the result.

Clipboard management

Clipboard management involves managing the activities of the clipboard, for example, getting text from the clipboard, copying selected text from the clipboard, and so on.

Let us see an example of getting text from the clipboard.

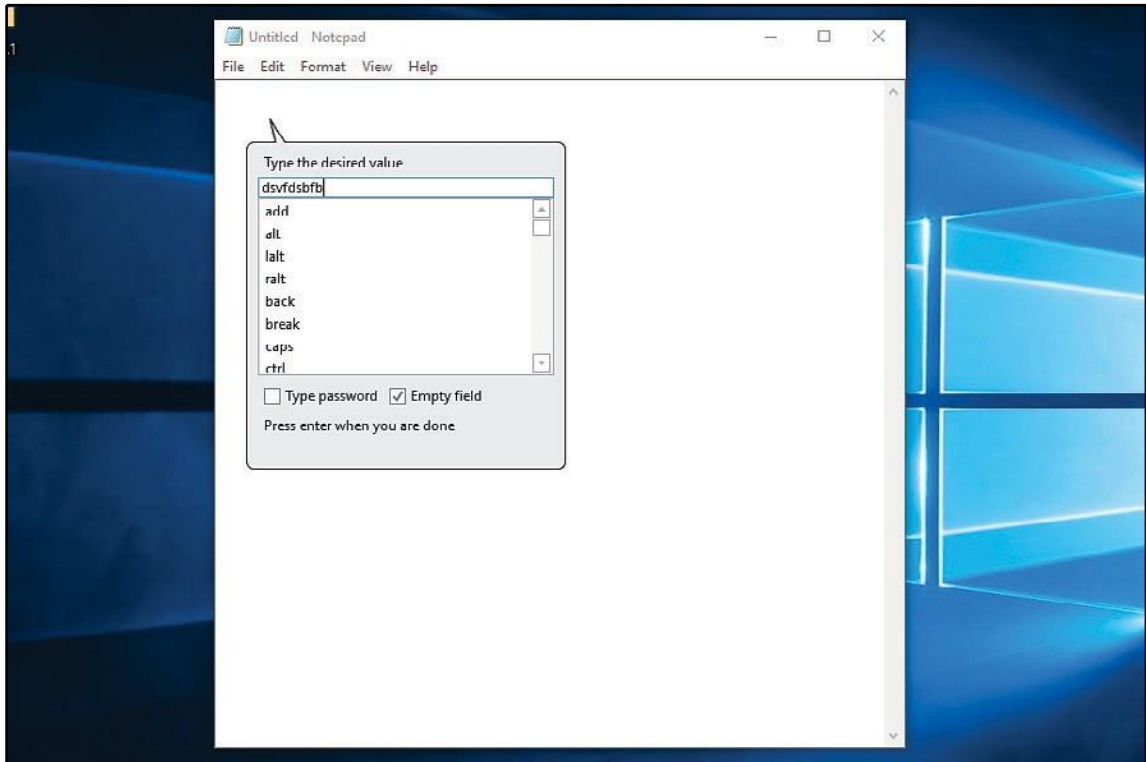
In this example, we will use Notepad. We will open Notepad, write some data into it, and then copy the data to the clipboard. We will then extract the data from the clipboard:

1. Drag and drop a **Flowchart** activity from the **Activities** panel.
2. Click on the **Recording** icon on the top of UiPath Studio. A drop-down menu will appear with the options, **Basic**, **Desktop**, **Web**, and **Citrix**, indicating the different types of recording. Select **Desktop** and click on **Record**.
3. Click on **Notepad** to open it. A Notepad window will pop up:

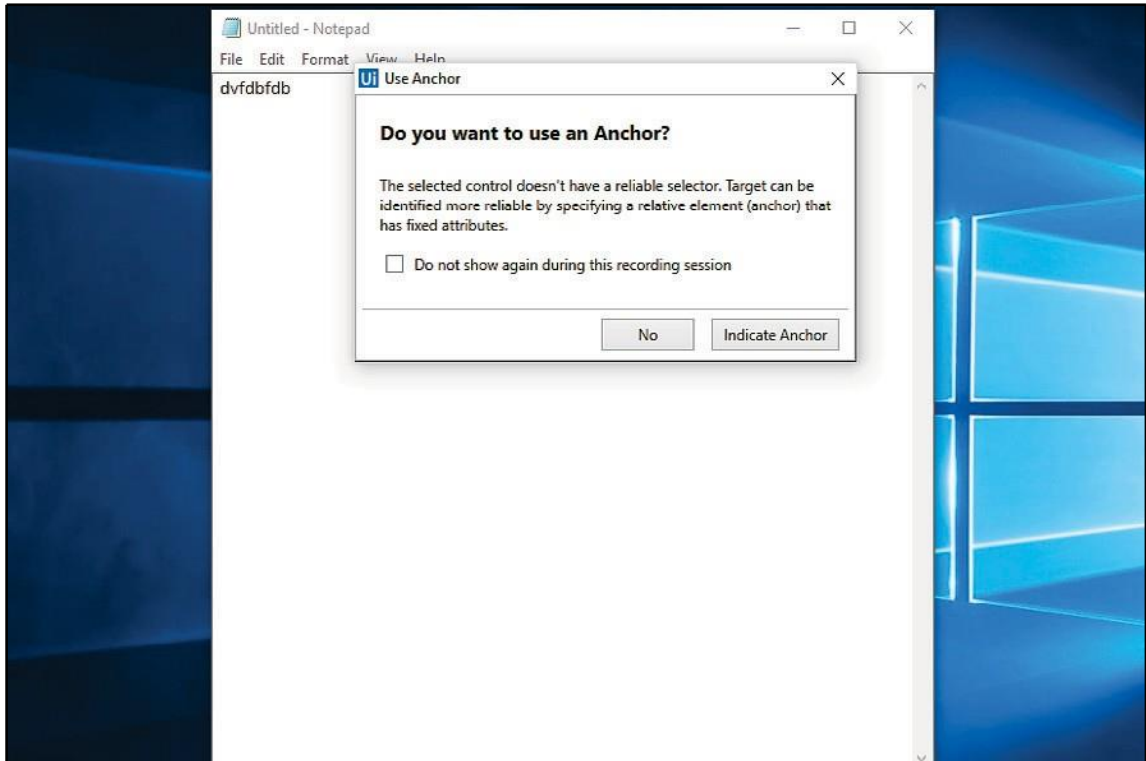


4. Click on the text area of Notepad. Type into the dialog box and check the empty field. (Checking the empty field will erase all existing data in Notepad before writing any new data.) Press *Enter*.

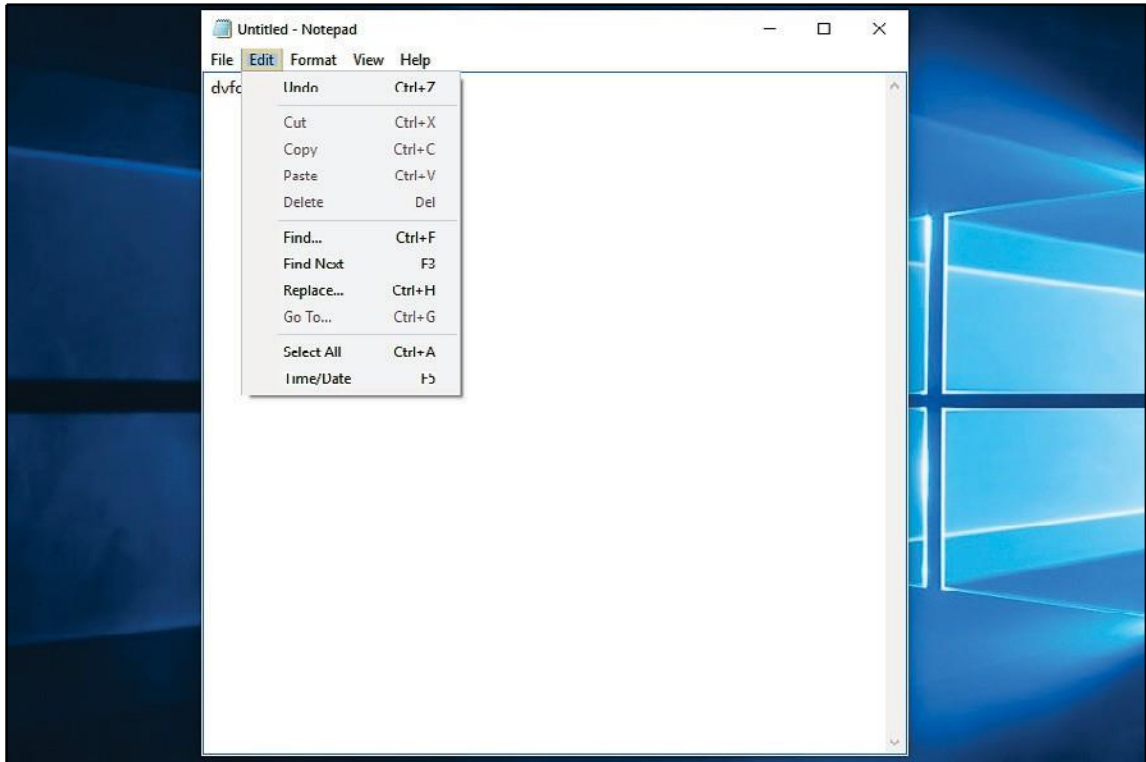
Data will be written on the Notepad text area:



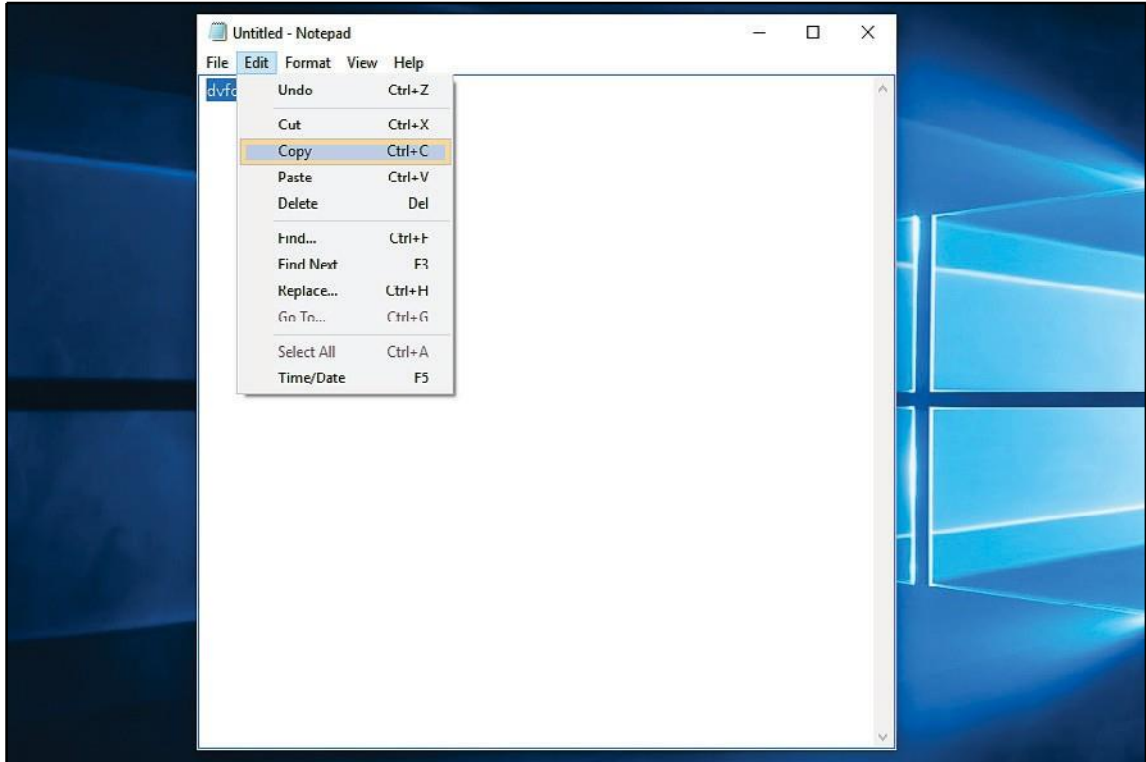
5. Click on the **Edit** button. A pop-up window will appear asking you whether you want to use an anchor. (An anchor is a relative element of the current `{focused}` element.) As you can see clearly, the anchor element of the **Edit** button can be the **File** or **Format** button. In this case, we have chosen the **Format** button:



6. Then, it will automatically start recognizing the **Edit** button. Choose the **Select all** option from the drop-down list:



7. Once again, click on the **Edit** button. It will again ask you to indicate the anchor element. Indicate the anchor button and the **Edit** button will be highlighted, giving you a drop-down box. Select the **Copy** option:

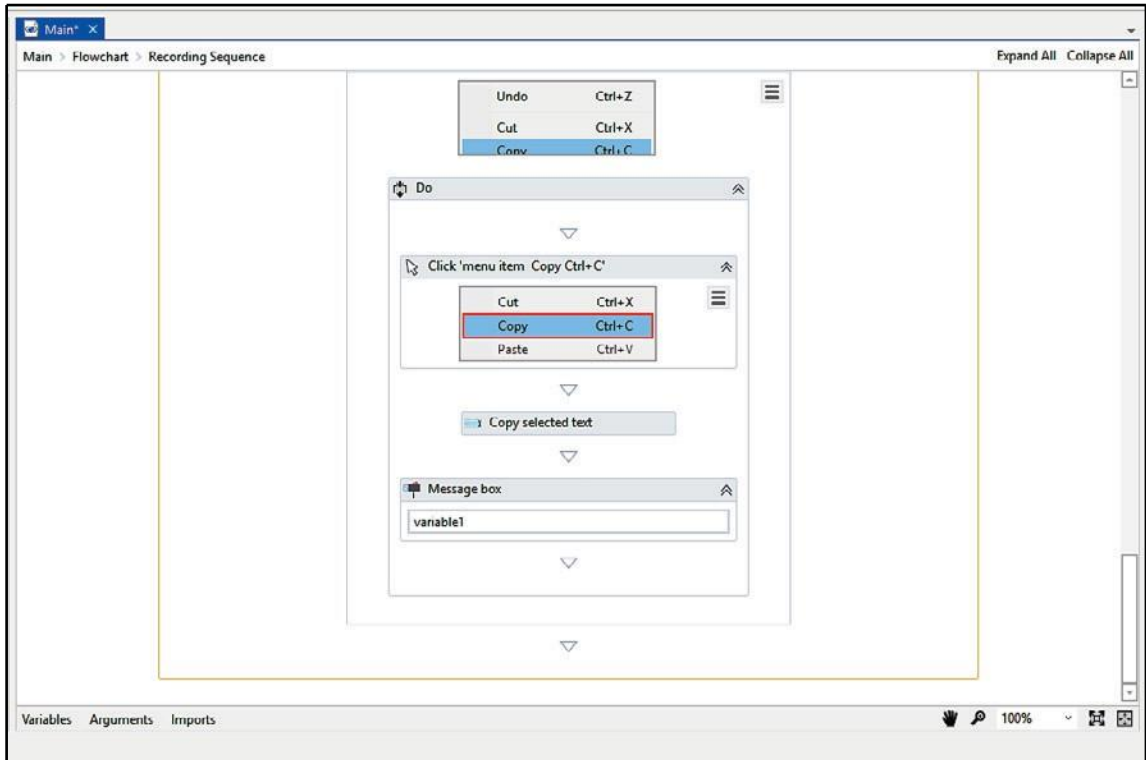


This copied text is now stored in the clipboard.

We can use the **Get from clipboard**, and **Copy selected text** activities to copy the text that is stored in the clipboard.

We will use the **Copy selected text** activity.

8. Double-click on the **Recording sequence** that is generated by the recording. Scroll down and drag and drop the **Copy selected text** and **Message box** activities inside the **Recording sequence**:



9. Create a variable of type **String** to store the output value of **Copy selected text**. This variable will receive the required text from the clipboard with the **Copy selected text** activity. Now, specify the newly created variable in the **Output** property of the **Copy selected text** activity. This will be the required selected text that we have copied into the clipboard.
10. Specify the string variable in the text property of the **Message box** activity.
11. Hit the **Run** button to see the result.

File operation with step-by-step example

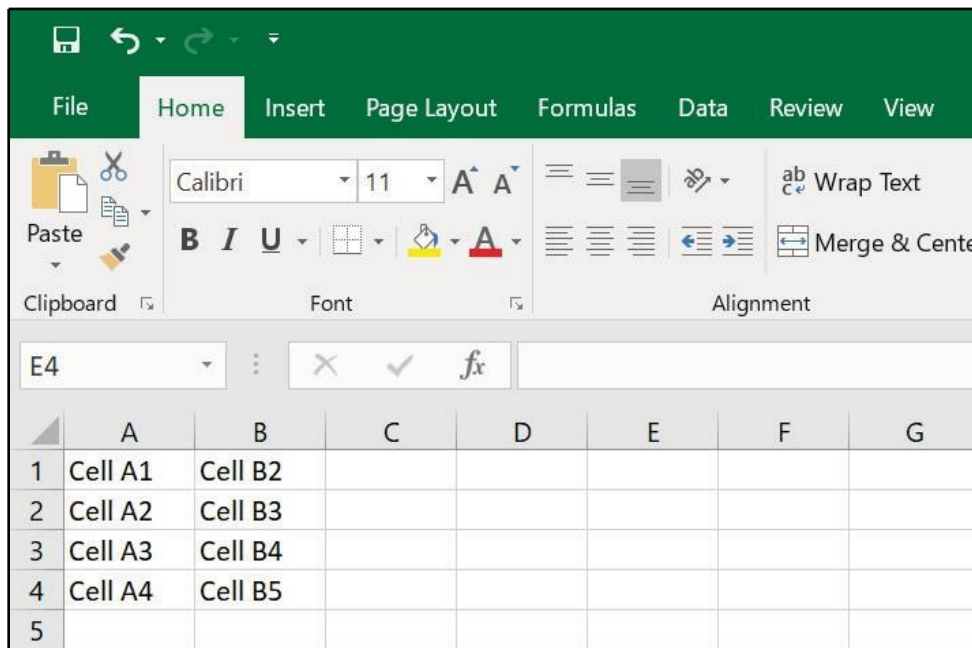
In this module, we are going to operate on Excel file. The following are the methods that are frequently used with an Excel file:

- Read cell Write
- cell Read range
- Write range
- Append range
-

Once you get familiar with these methods, it will become very easy for you to use other methods too.

Read cell

This is used to read the value of a cell from an Excel file. We have a sample Excel file that we will use in this example:



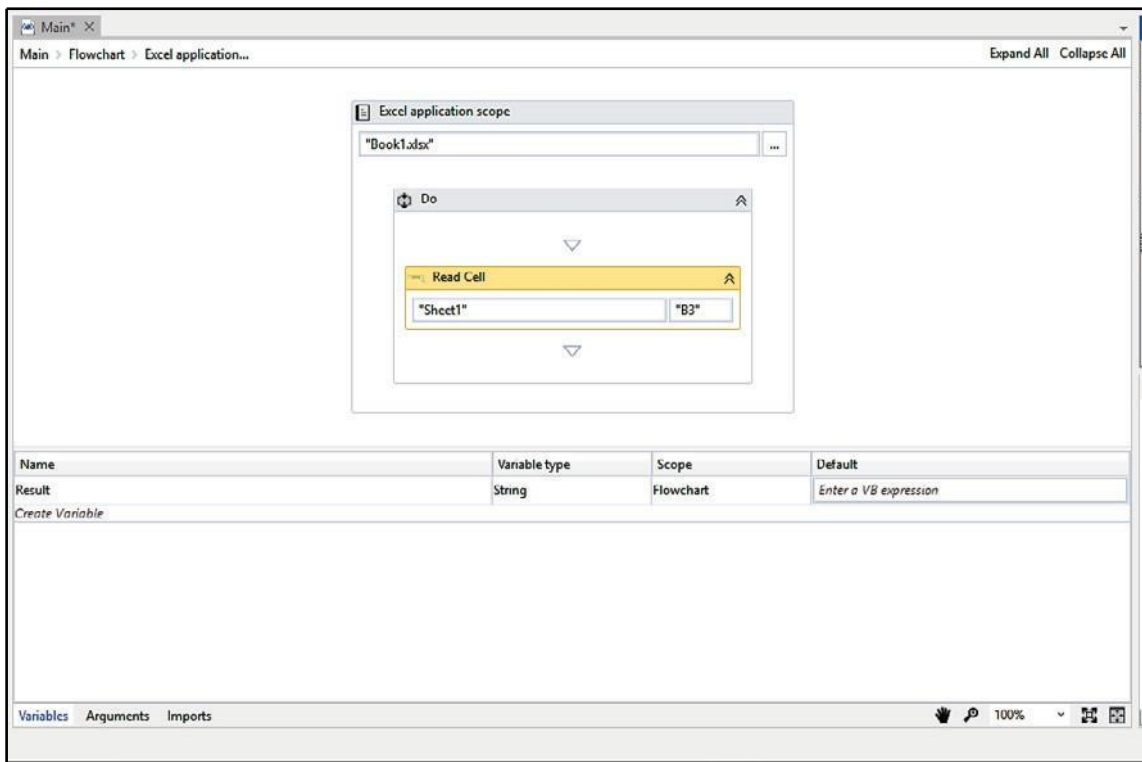
Suppose we have to read the value of the **B3** cell:

1. Drag and drop a **Flowchart** activity on the main Designer panel. Also, drag and drop an **Excel application scope** inside the **Flowchart**. Connect it to the **Start** node. Double click on Excel application scope.



It is a good practice to use the **Excel application scope** when using Excel activities inside our project.

2. Drag and drop the **Read Cell** activity inside the **Excel application scope** activity. Specify the range value in the cell text box of the **Read Cell** activity. Create a variable of type string to hold the result produced by the **Read Cell** activity. In our case, we have created a `Result` variable. Specify the **Output** property of the **Read Cell** activity by providing the variable's name that we have created:



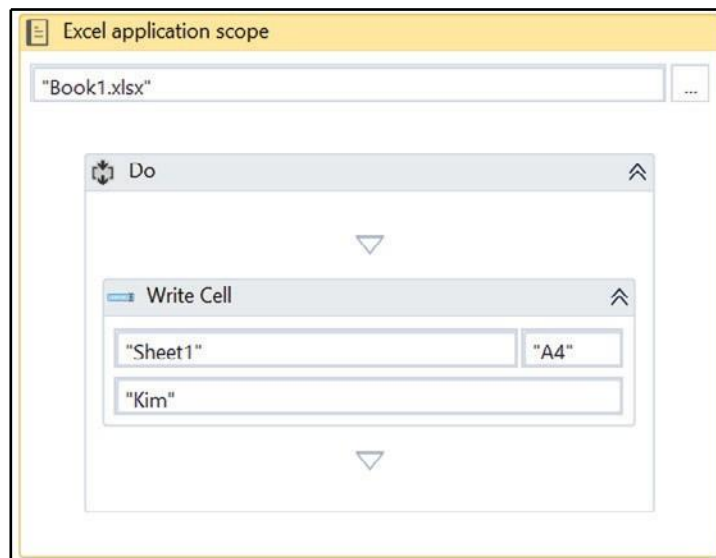
3. Drag and drop a **Message box** activity inside the **Excel application scope** activity and specify the string variable's name (which we created earlier) in the expression box of the **Message box** activity.

That's it. Press *F5* to see the result.

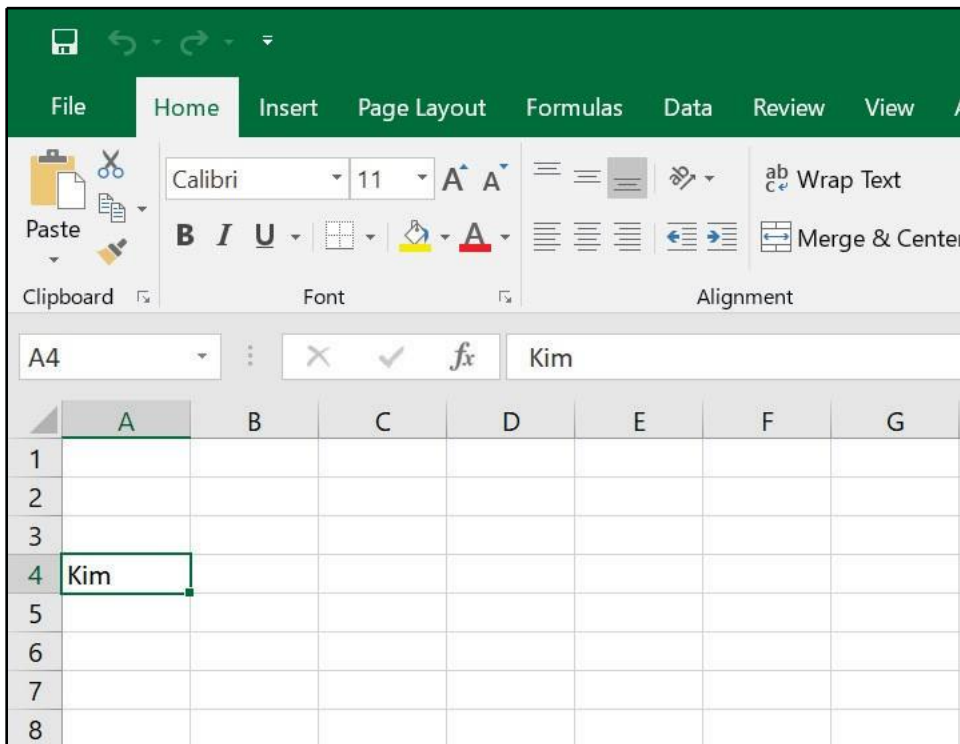
Write cell

This activity is used to write a value in a cell of an Excel file:

1. Drag and drop a **Flowchart** activity on the main Designer panel. Also, drag and drop an **Excel application scope** inside the **Flowchart** activity. Connect it to the **Start** node.
2. Drag and drop a **Write Cell** activity inside the **Excel application scope**. Specify the cell value in which we want to write in the **Range** property of the **Write Cell** activity. Also, specify the value of the **Value** property:



Press *F5* and see the result. Open the Excel file to see the changes:

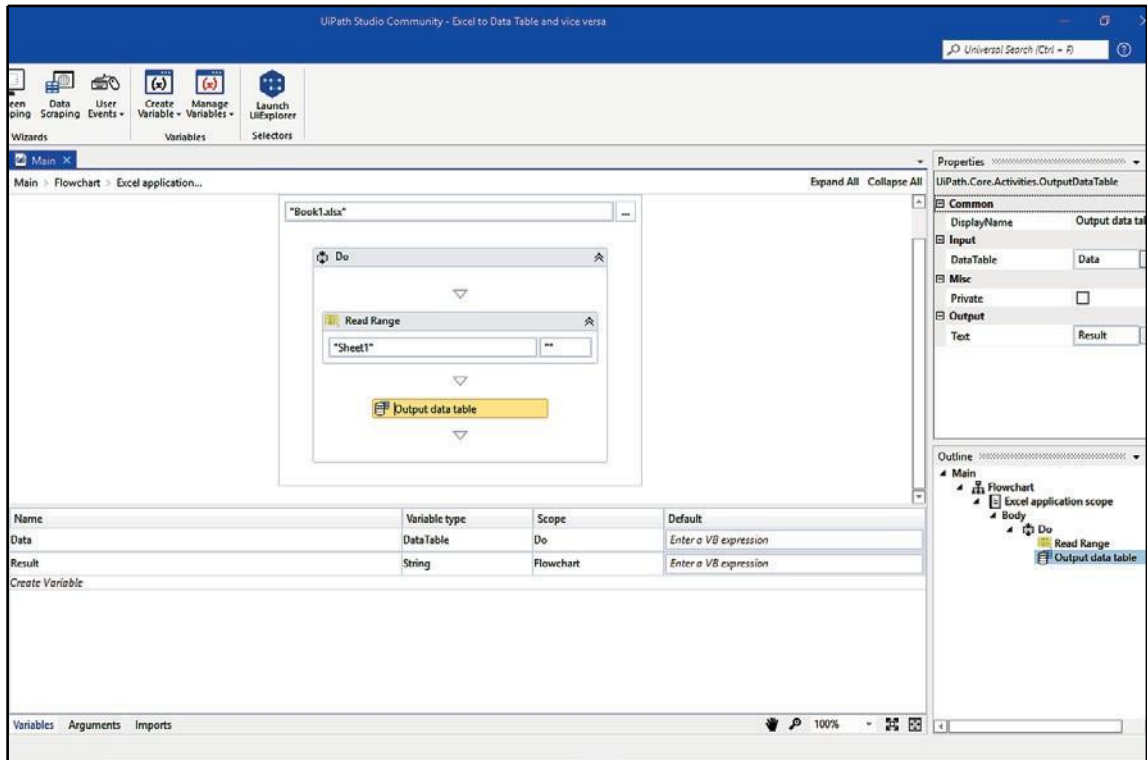


Read range

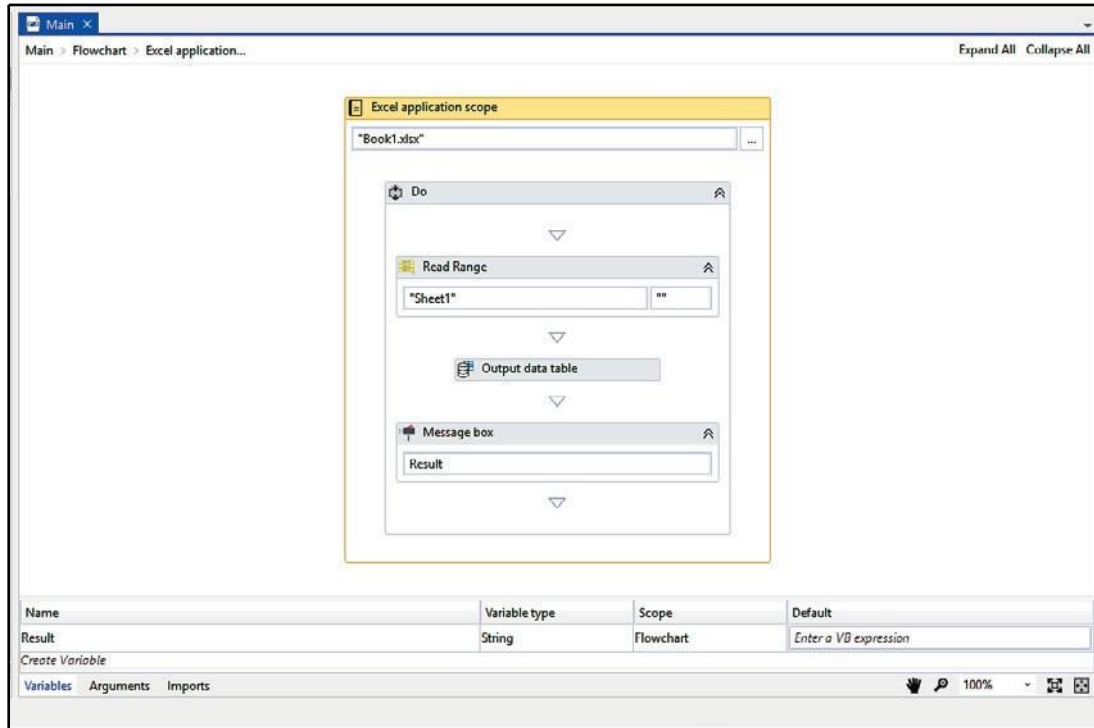
This is used to read the value up to the specified range. If the range parameter is not specified, it will read the entire Excel file:

1. Drag and drop a **Flowchart** activity on the main Designer panel. Also, drag and drop an **Excel application scope** inside the **Flowchart** activity. Connect it to the **Start** node.
2. Drag and drop a **Read Range** activity inside the **Excel application scope** activity. The **Read Range** activity produces a data table. We have to receive this data table in order to consume it. We need to create a data table variable and specify it in the **Output** property of the **Read Range** activity.

3. Drag and drop an **Output Data Table** activity inside the **Excel application scope** activity. Now, we have to specify two properties of the **Output Data Table** activity: **Data Table** property and text property. The **Data Table** property of the **Output Data Table** activity is used to convert the data table into a string format. The text property is used to supply its value in a string format. We have to receive this value in order to consume it. For this, let us create a variable of type string. Give it a meaningful name (in our case, it is **Result**):



4. Drag and drop a **Message box** activity inside the **Excel application scope** activity. Also, specify the string variable's name that we created earlier inside the **Message box** activity:

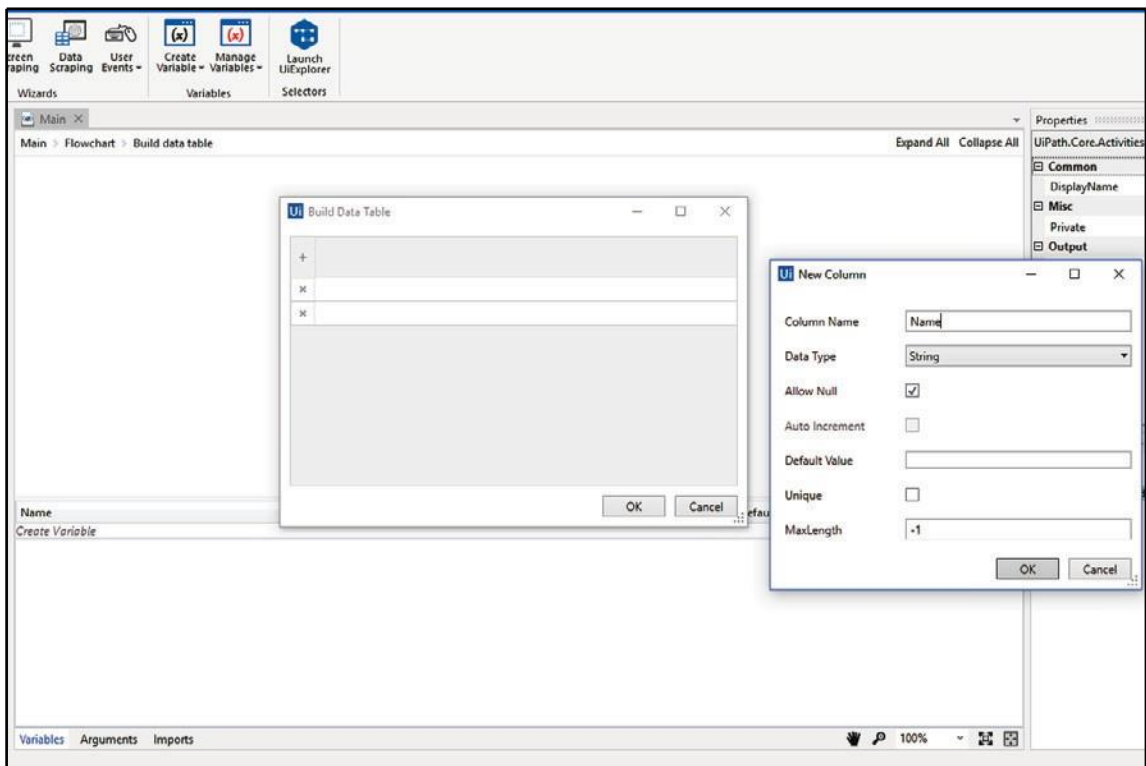


That's it. Press *F5* to see the result. A window will pop up displaying your Excel file data.

Write range

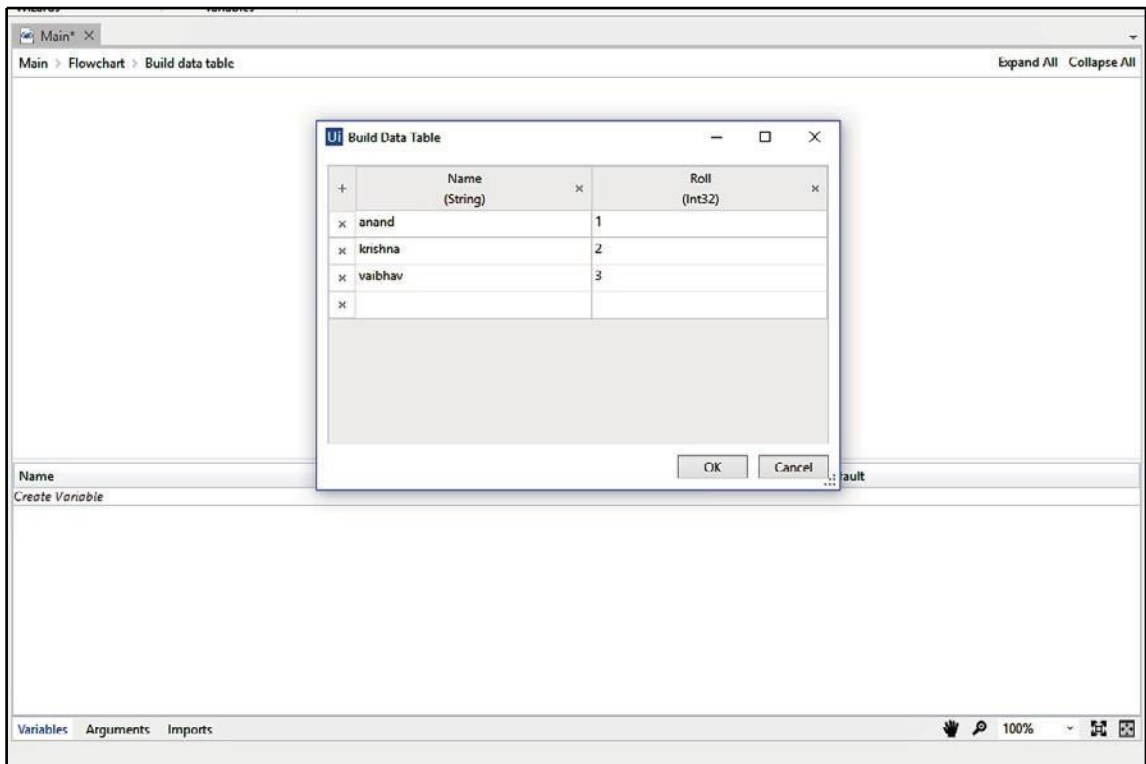
This is used to write a collection of rows into the Excel sheet. It writes to the Excel file in the form of a data table. Hence, we have to supply a data table:

1. Drag and drop a **Build data table** activity from the **Activities** panel. Double-click on this activity. A window will pop up. You will notice that two columns have been generated automatically. Delete these two columns. Add your column by clicking on the **+** icon and specify the column name. You can also select your preferred data type. You are free to add any number of columns:



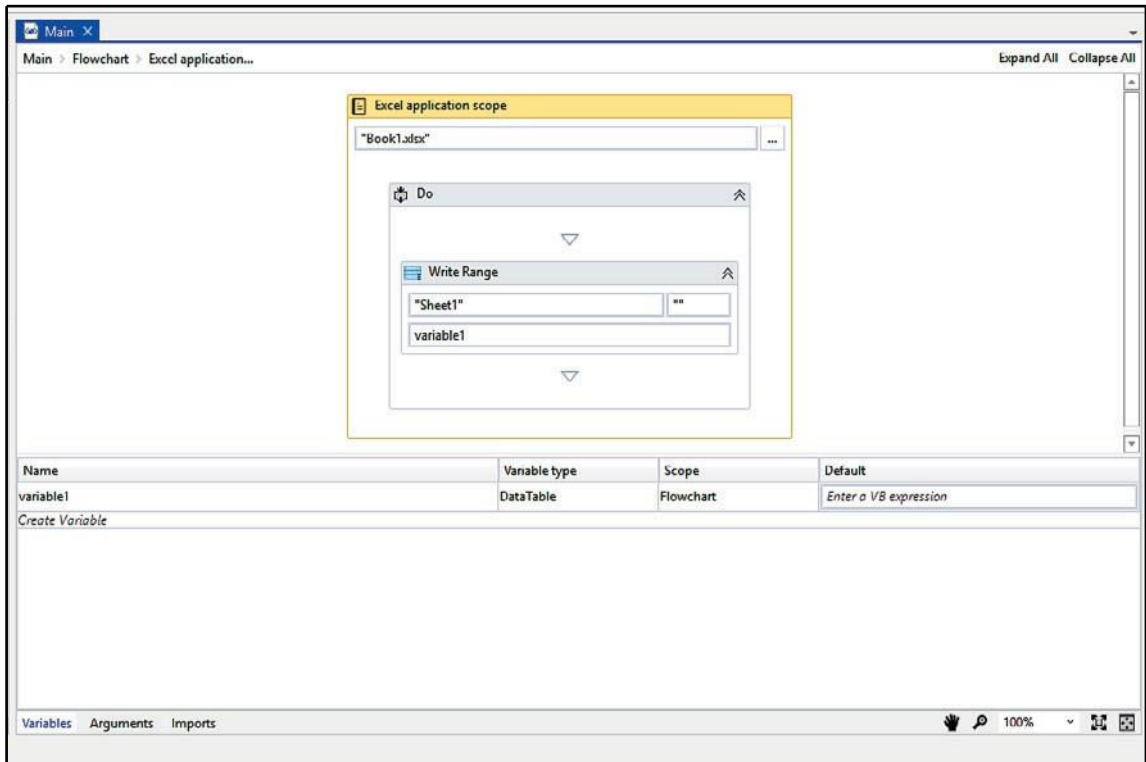
2. In this project, we are adding two columns. The procedure for adding the second column is almost the same. You just have to specify a name and its preferred data type. We have added one more column (Roll) and set the data type to **Int32** for the data table. We have also initialized this data table by providing some values in its rows.

Create a variable of type data table. Give it a meaningful name. Specify this data table name in the **Data table** property of the **Build data table** activity. We have to supply this variable in order to get the data table that we have built:

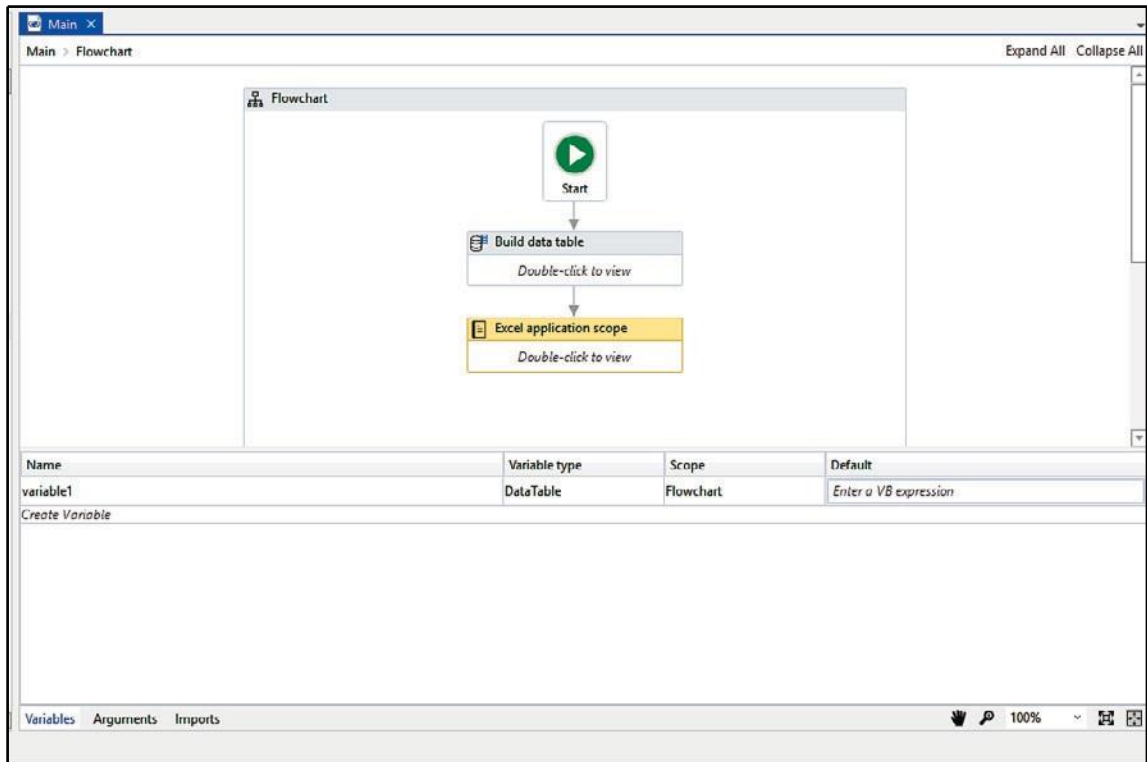


Our data table has been built successfully.

3. Drag and drop an **Excel application scope** inside the main Designer panel. You can either specify the Excel sheet path or manually select it. Connect this activity to the **Build Data Table** activity. Inside the **Excel application scope** activity, just drag and drop the **Write Range** activity:



4. Specify the data table variable name that we created earlier and set it as a **Data table** property inside the **Write Range** activity. We can also specify the range. In this case, we have assigned it as an empty string:



That's it. Hit the **Run** button or press *F5* to see the result.

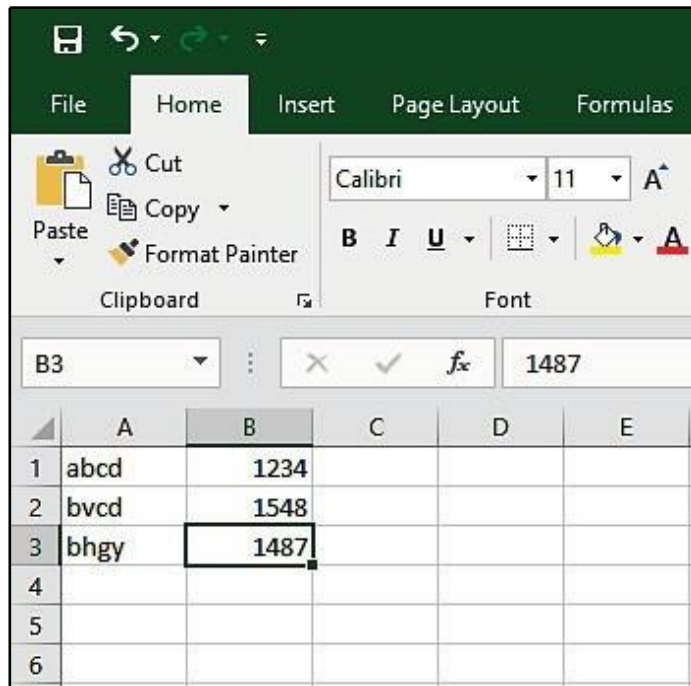
Append range

This is used to add more data into an existing Excel file. The data will be appended to the end.

1. Drag and drop the **Flowchart** activity on the main Designer window. Also, drag and drop the **Excel application scope** inside the **Flowchart** activity. Connect it to the Start node.

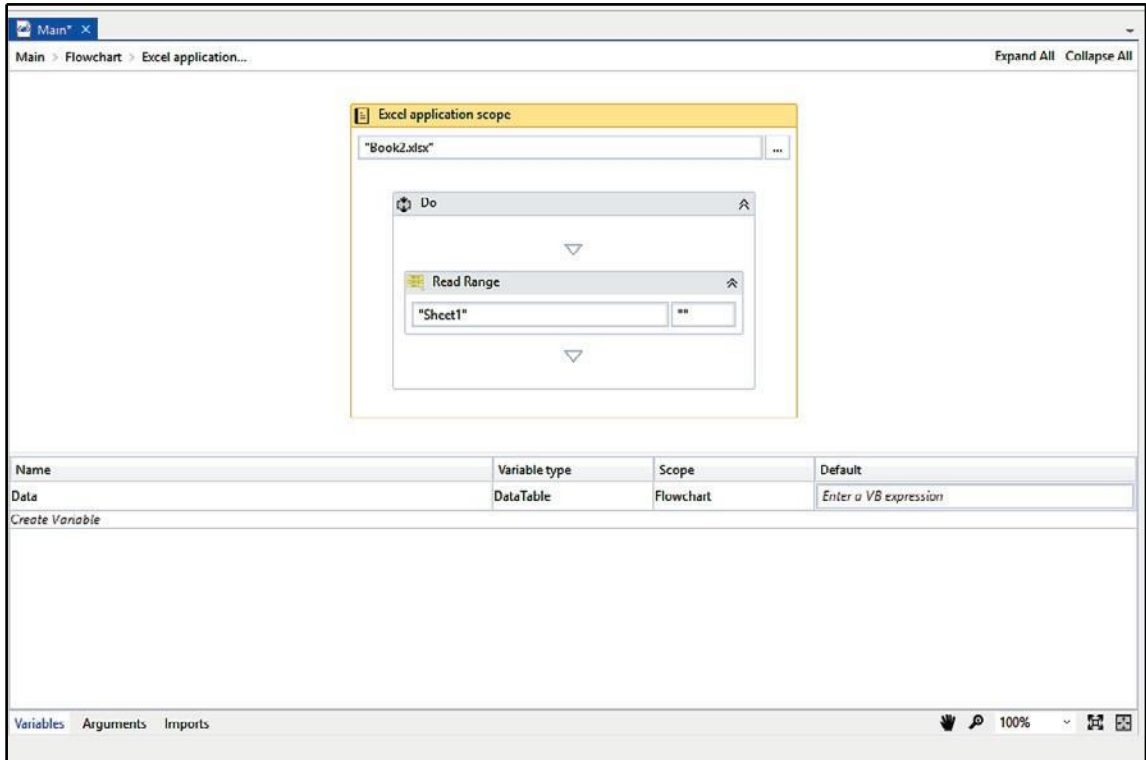
The **Append Range** activity requires a data table. In this program, we are going to use another sample Excel file, which has some raw data. Then, we will read this Excel file and append the data to another Excel file.

First, we have to read its contents:

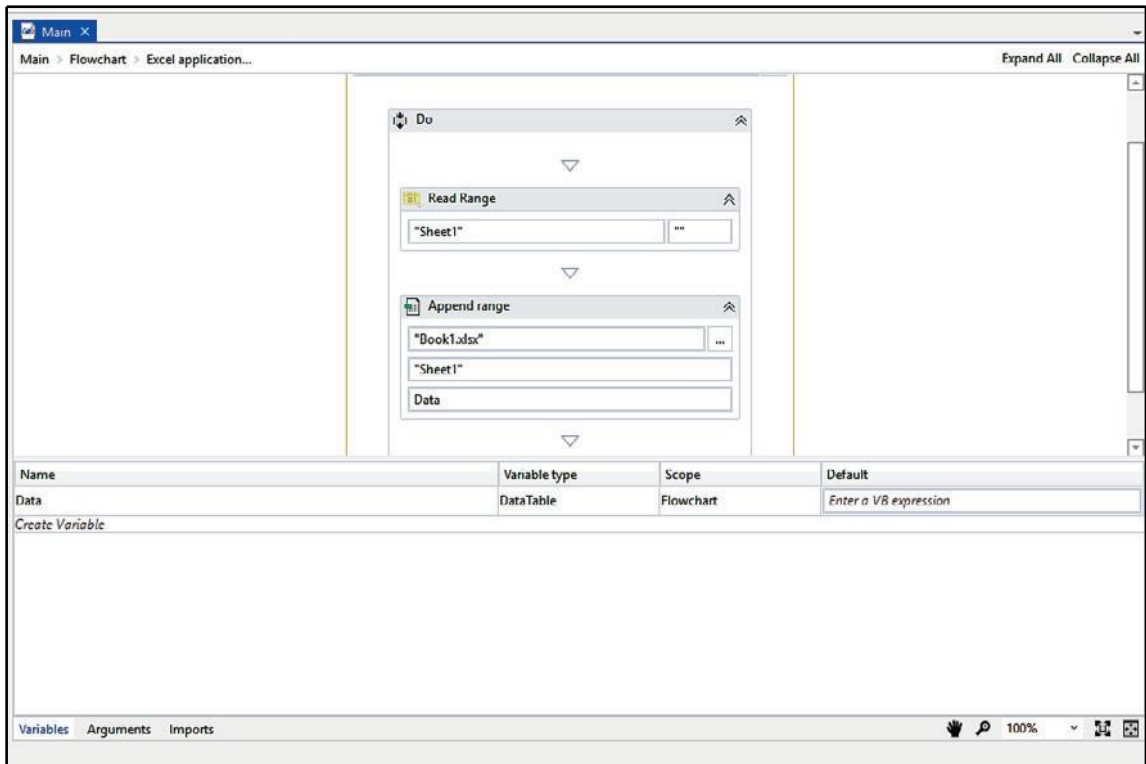


	A	B	C	D	E
1	abcd	1234			
2	bvcd	1548			
3	bhgy	1487			
4					
5					
6					

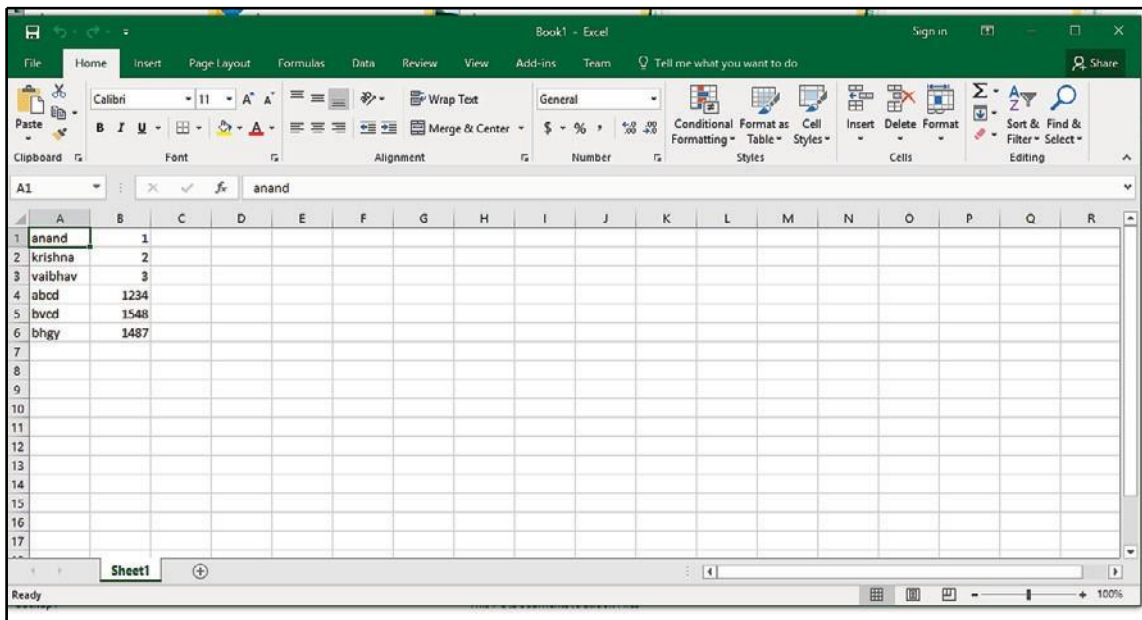
2. Drag and drop the **Read Range** activity inside the **Excel application scope** activity. The **Read Range** activity produces a data table. We have to receive this data table in order to consume it. Create a data table variable and specify it in the **Output** property of the **Read Range** activity:



3. Drag and drop the **Append Range** activity inside the **Excel application scope** activity. Specify the Excel file path in the **Append Range** activity (in which we want to append the data). Also, specify the data table (which is generated by the **Read Range** activity):



That's it. Press *F5* to see the result:



We can clearly see that the data has been appended successfully to the Excel sheet.

CSV/Excel to data table and vice versa (with a step-by-step example)

In this section, we will see how to extract data from an Excel file into a data table and vice versa. We will achieve this by:

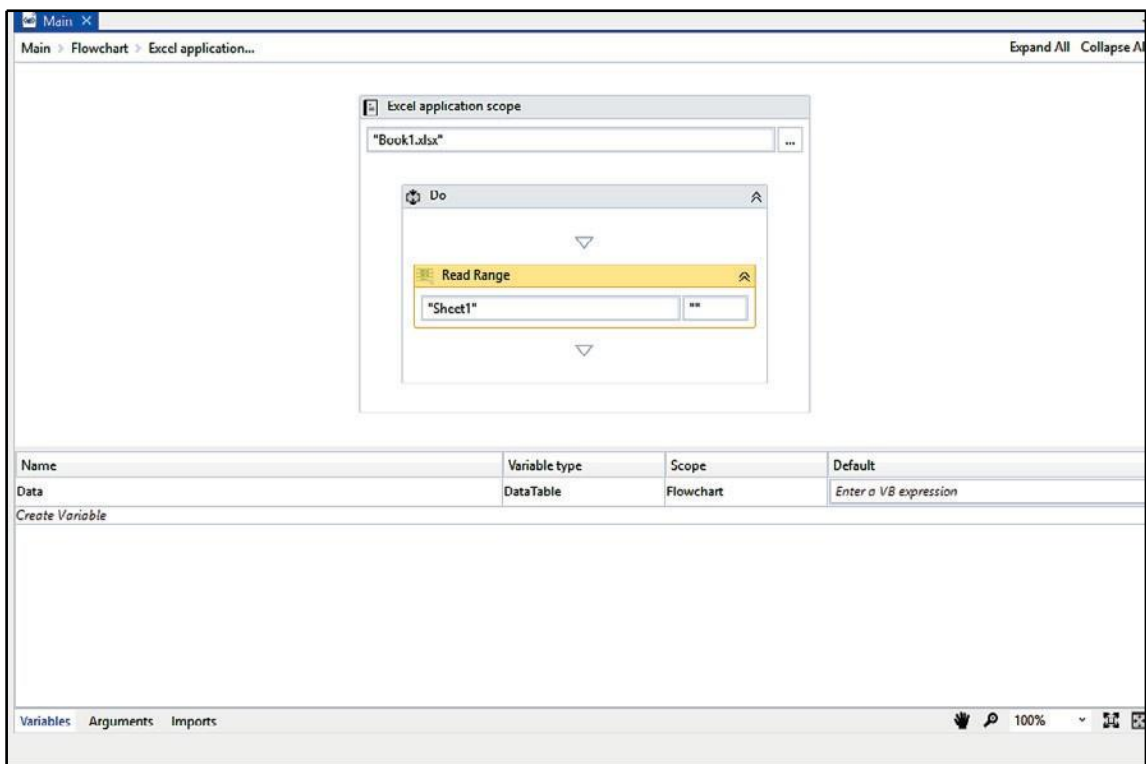
- Reading an Excel file and creating a data table using data from the Excel file
- Creating a data table and then writing all its data to an Excel file

Reading an Excel file and creating a data table by using data from the Excel file

We have an existing Excel file and we are going to use it in our project:

1. Drag and drop the **Flowchart** activity on the main Designer window. Also, drag and drop the **Excel application scope** inside the **Flowchart**.
2. Double-click on the **Excel application scope**. You have to specify the path of your workbook/Excel file. Drag and drop the **Read Range** activity from the **Activities** panel inside the **Excel application scope**.

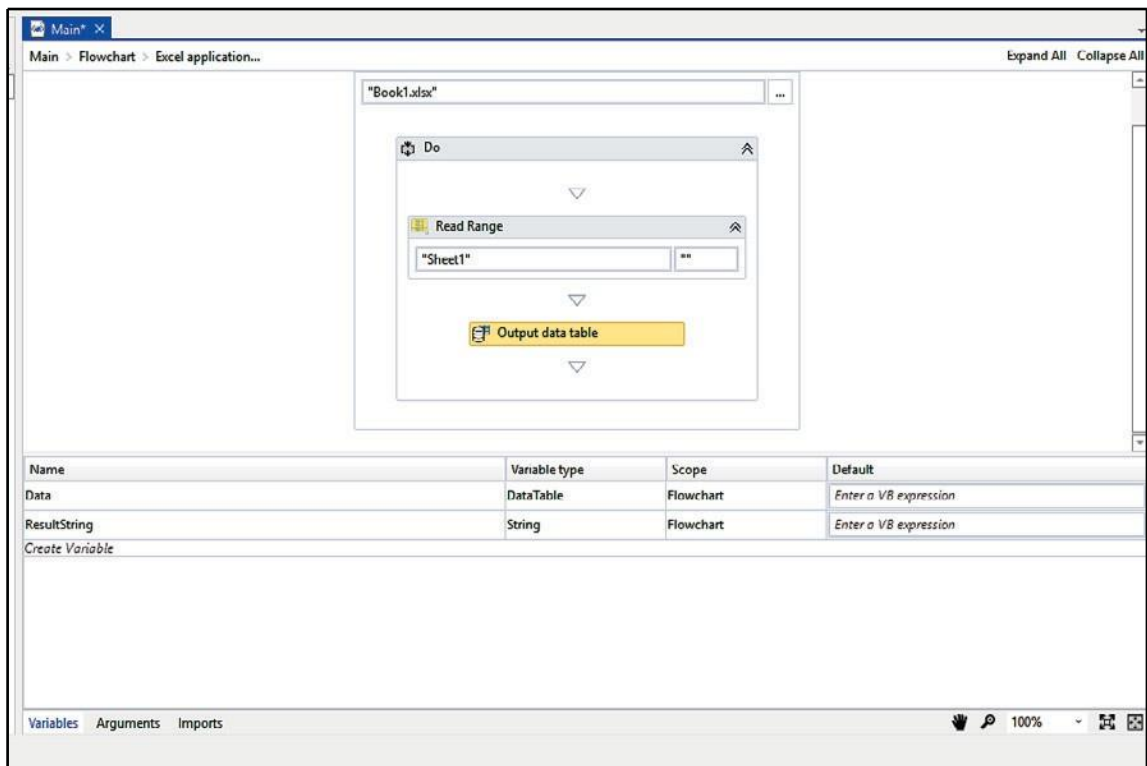
The **Read Range** activity will read the entire Excel sheet. We also have the option of specifying our range. Create a variable of type data table and specify it in the **Output** property of the **Read Range** activity. This variable will receive the data table produced by the **Read Range** activity:



3. Drag and drop the **Output Data Table** activity inside the **Excel application scope** activity. Now, we have to specify two properties of the **Output Data Table** activity: the **Data Table** property and the text property. The **Data Table** property of the **Output Data Table** activity is used to convert the **Data Table** into string format.

The text property is used to supply its value in a string format. We have to receive this value in order to consume it. For this, let us create a variable of type string.

Give it a meaningful name:



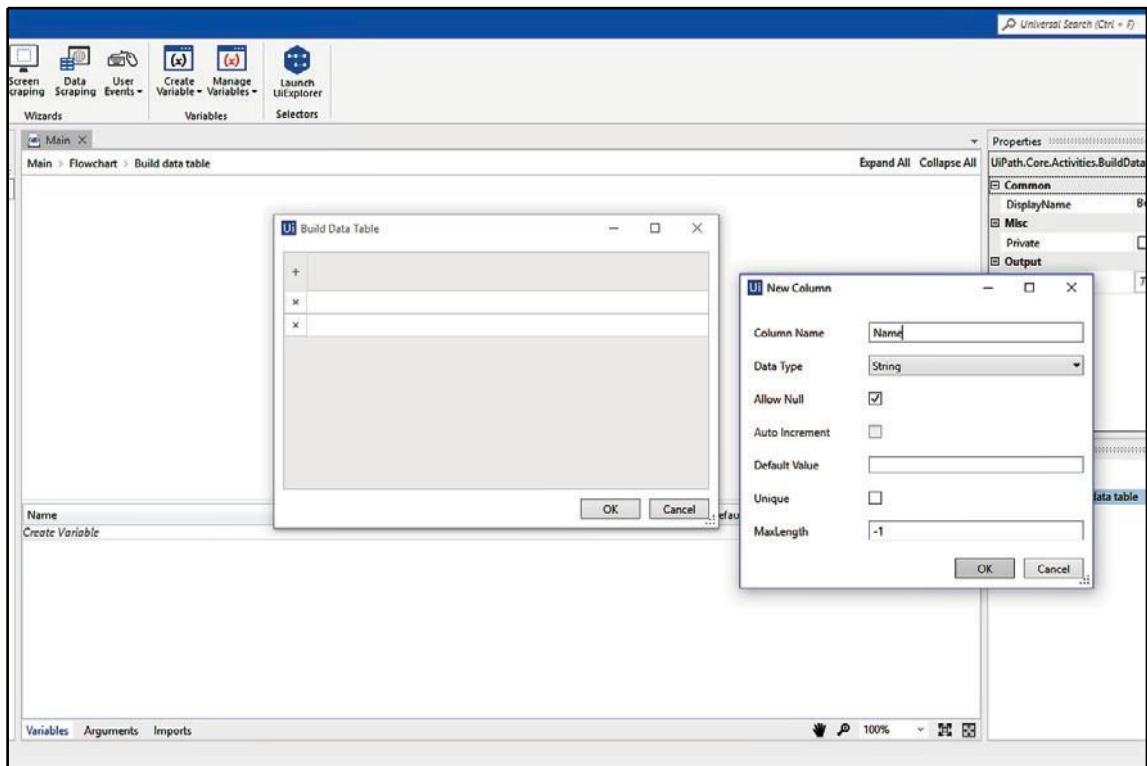
4. Drag and drop a **Message box** activity inside the **Excel application scope** activity. Also, specify the string variable's name that we created earlier inside the **Message box** activity.

That's it. Press *F5* to see the result. A window displaying the Excel file data will pop up.

Creating a data table and then writing all its data to an Excel file

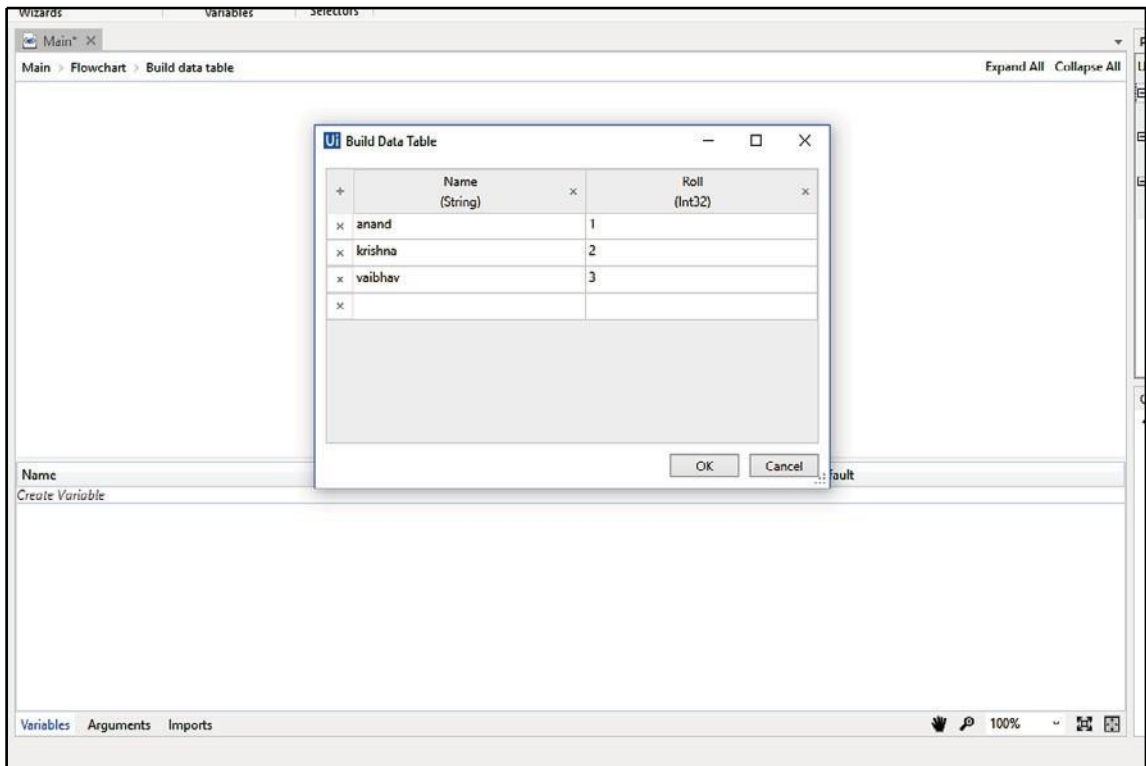
In this project, we will build a data table dynamically and then write all its data to an Excel file:

1. Drag and drop a **Build data table** activity from the **Activities** panel. Double-click on this activity. A window will pop up. Two columns have been generated automatically; delete these two columns. Add your column by clicking on the + icon and specify the column name. You can also select your preferred data type. You are free to add any number of columns:



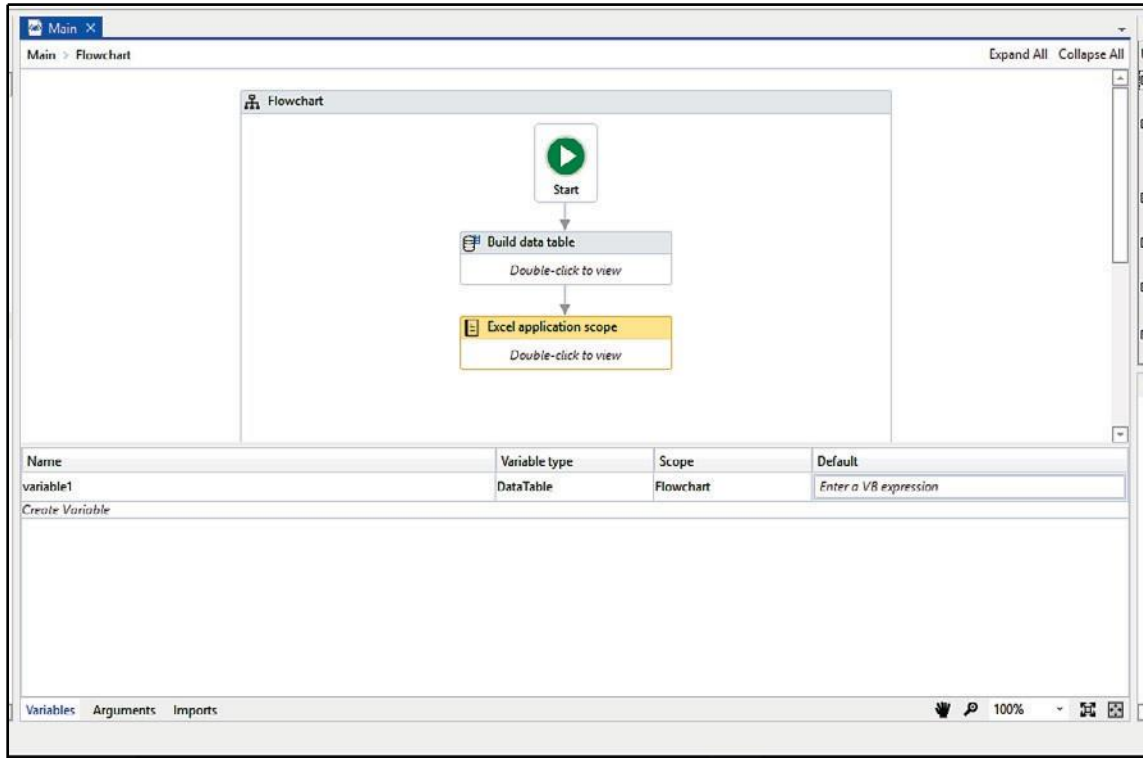
2. In this project, we are adding two columns. The procedure for adding the second column is almost the same. You just have to specify a name and its preferred data type. We have added one more column (Roll) and set the data type to **Int32** in the data table. We have also initialized this data table by giving some values to its rows.

Create a variable of type **Data Table**. Give it a meaningful name. Specify this data table's name in the **Data Table** property of the **Build data table** activity. We have to supply this variable in order to get the data table that we have built:

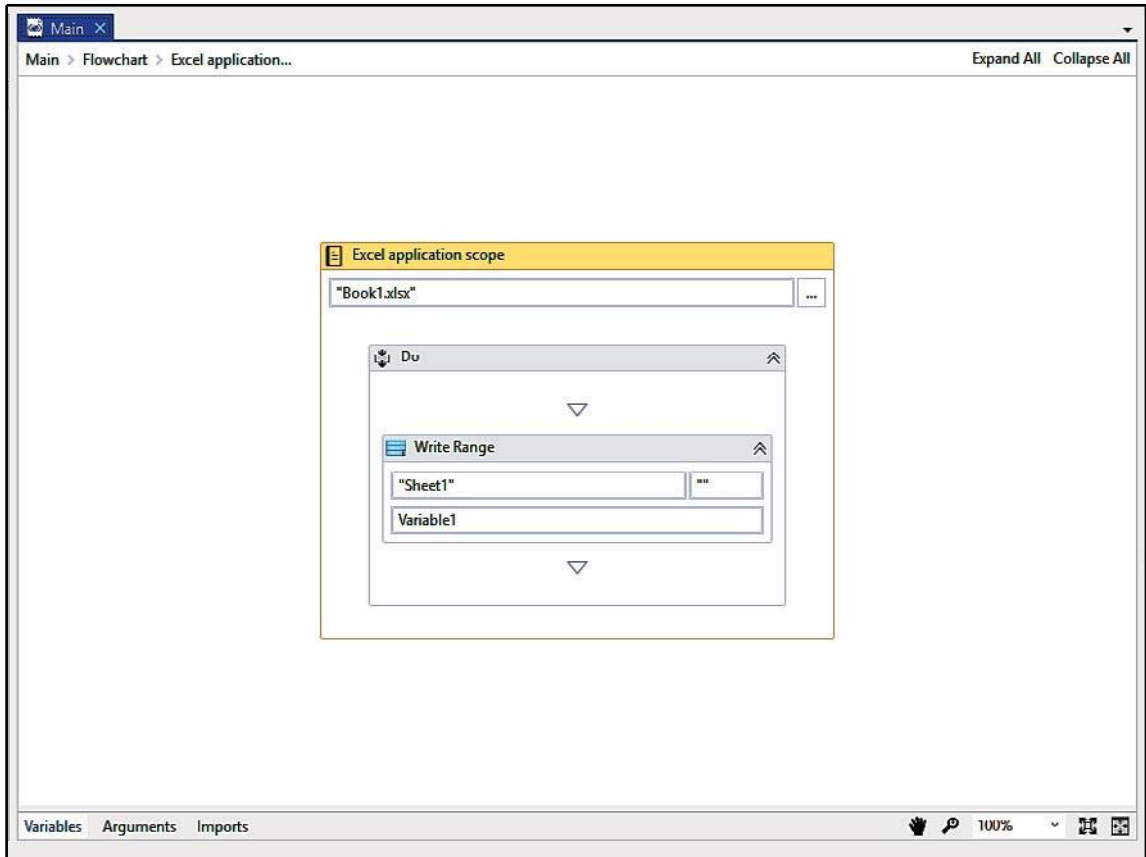


Our data table has been built successfully.

3. Drag and drop the **Excel application scope** inside the main Designer window. Specify the Excel sheet's path or manually select it. Connect this activity to the **Build Data table** activity:



4. Inside the **Excel application scope** activity, drag and drop the **Write Range** activity. Specify the data table variable name that we created earlier and set it as a **Data table** property inside the **Write Range** activity. We can also specify the range. In this case, we have assigned it as an empty string:



5. That's it. Hit the **Run** button or press *F5* to see the result.

Summary

In this chapter, you have learned techniques for using memory with variables. You also have learned about data tables and easy ways to manipulate data in memory.

Apart from using a variable or collection to store data, we have learned to store and manipulate data in a more persistent way using such files as CSV and Excel.

In the next chapter, we will learn to handle controls within applications in a better way.

5 Taking Control of the Controls

By now, you should be able to make fairly complex workflows and take various paths using control flows. You should now know how to store and evaluate variables to make a decision. I hope you are using a recorder extensively, as we will revisit the recorder in this chapter to learn more about it.

In this chapter, we will go into detail on how we can interact with the controls in the UI. Sometimes, you may need to click on a particular button or extract information from a textbox. Either we take some action on a control, or we read/write. We will go into detail on how to do this accurately. In this chapter, readers will learn about the various *selectors* available in UiPath to extract and take action on controls:

- Finding and attach windows
- Find controls
- Techniques to wait for a control
- Acting on controls—mouse and keyboard activities
- Working with UiExplorer
- Handling events

We will then discuss the recorder in a new context.

Extraction is a primary feature of RPA, enabling UI automation. Behind the scenes, many technologies are at work on the seamless extraction of information from the UI. When typical RPA techniques are not successful, OCR technology is used to extract information. We will learn about using OCR and other techniques in the following topics:

- Screen Scraping
- When to use OCR
- Types of OCR available
- How to use OCR

Finding and attaching windows

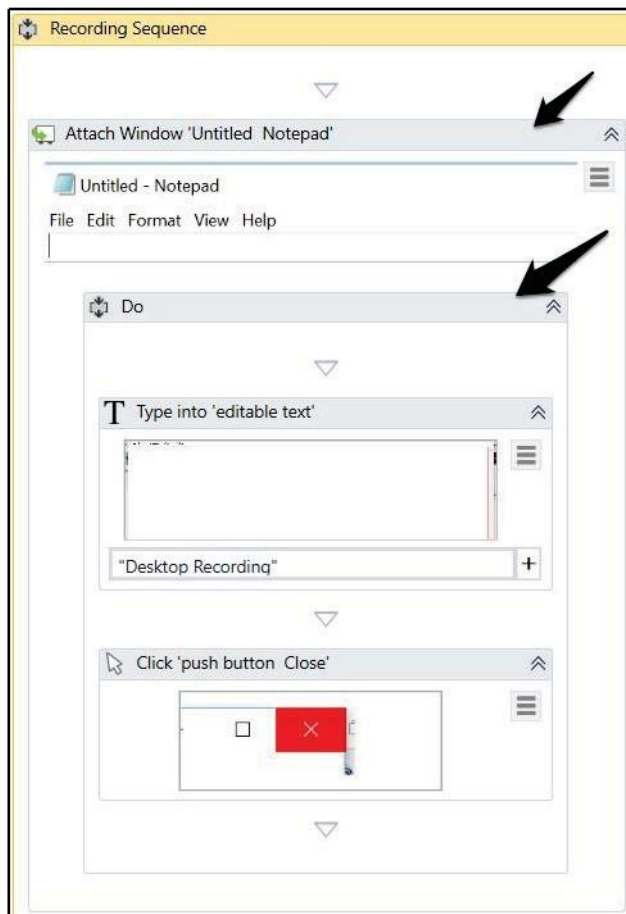
In this section, we are going to use the Attach Window activity.

The Attach Window activity can be found in the **Activities** panel. This activity is generally used to attach an already opened window. It is also auto-generated when we record actions using the Basic or Desktop recorder. You will get a much clearer idea after going through the example in the following section.

Implementing the Attach Window activity

In this example, we shall use the **Attach Window** activity manually. Here, we are going to attach a Notepad window and then write some text into it:

1. Create a blank project and give it a meaningful name.
2. Drag and drop a **Flowchart** activity on the Designer panel. Also, drag and drop a **Click** activity inside the Designer panel. Set this **Click** activity as the **Start** node.
3. Double-click on the **Click** activity and then click on **Indicate on screen**. Locate the Notepad icon.
4. Drag and drop the **Attach Window** activity on the main Designer panel. Connect the **Attach Window** activity to the **Click** activity.
5. Double-click on the **Attach Window** activity. Click on **Click Window on Screen** and indicate the Notepad window. The Notepad window is now attached to the previous activity:



6. For the sake of completeness, we are going to add a **Type into** activity. Just drag and drop the **Type into** activity, inside the **Attach Window** activity. Click on the **Indicate element inside window** and locate the Notepad window where you want to write the text. Write the text in the Text property of the **Type into** the activity.
7. Hit the **Run** button.

Finding the control

There are many activities which can be used to find controls on screen/ applications. These activities are used to find or wait for an UI element.

Following are the activities that help in finding the controls:

- Anchor base
- Element Exists
- Element scope
- Find children
- Find element
- Find relative element
- Get ancestor
- Indicate on screen

We will discuss all these controls one by one.

Anchor base

This control is used for locating the UI element by looking at the UI element next to it. This activity is used when we have no control over the selector. That means when we do not have a reliable selector, then we should use the **Anchor base** control to locate the UI element.

We can use the **Anchor base** control as explained in the following section:

1. Drag and drop a **Flowchart** activity on the Designer panel of a blank project. Also, drag and drop an **Anchor base** control from the **Activities** panel. Connect the **Anchor base** control with **Start**.
2. Double-click on the **Anchor base** control:



3. There are two activities that we have to supply to the **Anchor base** control: **Anchor** and action activities.
4. Drag and drop the **Anchor base** activity (for example; **Find Element** activity) in the **Anchor** field and **Action** activity (for example; **Type into**) in the **Drop Action Activity Here** field of the **Anchor base** control.

The **Anchor base** activity will find the relative element nearby the element on which you want to perform the Action, and the Action activity will perform the appropriate action that you have specified.

Element Exists

This control is used to check the availability of the UI element. It checks if the UI element Exists or not. It also returns a Boolean result if the UI Element Exists, then it returns true: otherwise, it returns false.

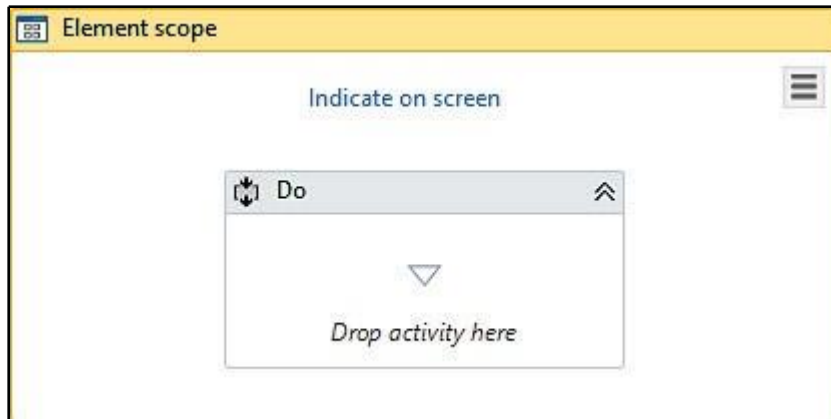
You can use this control to check for the UI element. In fact, it is good practice to use this control for UI elements whose availability is not confirmed or those that change frequently.

Just drag and drop the **Element Exists** control from the **Activities** panel. Double-click on it. You can see there is an **Indicate on screen** option. Click on it to indicate the UI element. It returns a Boolean result, which you can retrieve later from the Exists property. You just have to supply a Boolean variable in the **Exists** property in the **Properties** panel.

Element scope

This control is used to attach a UI element and perform multiple actions on it. You can use a bunch of actions within a single UI element.

Drag and drop the **Element scope** control and double-click on this control:



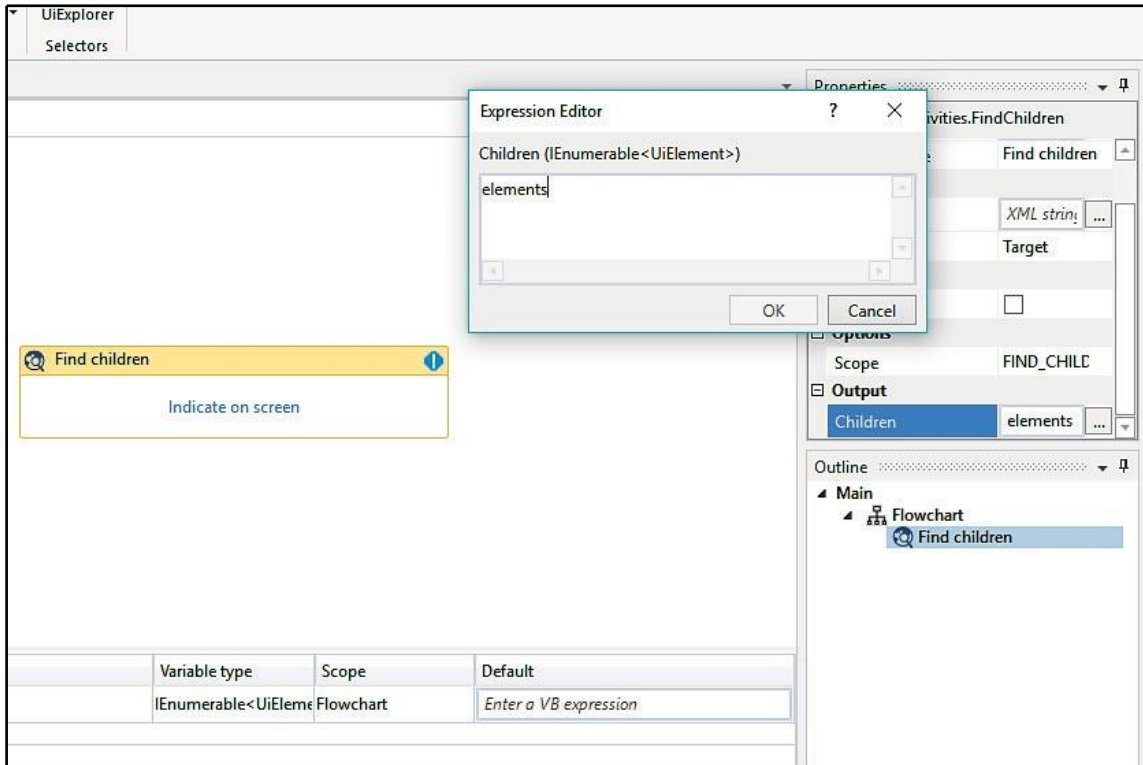
You can clearly see that you have to indicate the UI element by clicking on **Indicate on screen** and specifying all the actions that you want to perform in the **Do** sequence. You can add many activities inside the **Do** sequence.

Find children

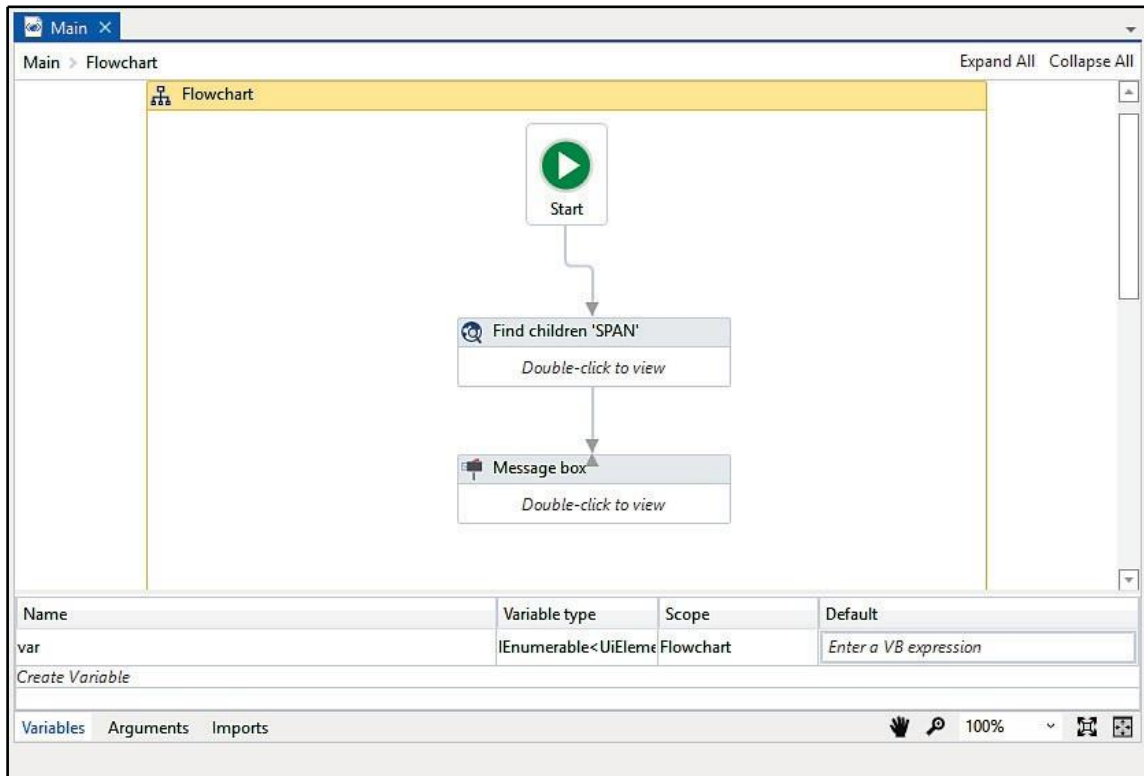
This control is used to find all the children UI elements of a specified UI element. It also retrieves a collection of children UI elements. You can use a loop to inspect all the children UI elements or set up some filter criteria to filter out the UI elements.

Drag and drop the **Find children** control from the **Activities** panel.

Double-click on it to indicate the UI element that you want to specify. You can indicate it by clicking on **Indicate on screen**:



You have to supply a variable of type `IEnumerable<UIElements>` in the children property, as mentioned in the preceding screenshot. This variable is then used for retrieving the UI elements:



Find element

This control is used to find a particular UI element. It waits for that UI element to appear on the screen and returns it back.

You can use this control in the same way that you used the other controls. Just drag and drop this control, and indicate the UI element by clicking on **Indicate on screen**.

You can specify the variable of type UI element in the Found element property of the Find Element control to receive the UI element as output.

Find relative element

This control is similar to the Find element control. The only difference is that it uses the relative fixed UI element to recognize the UI element properly.

This control can be used in scenarios where a reliable selector is not present.

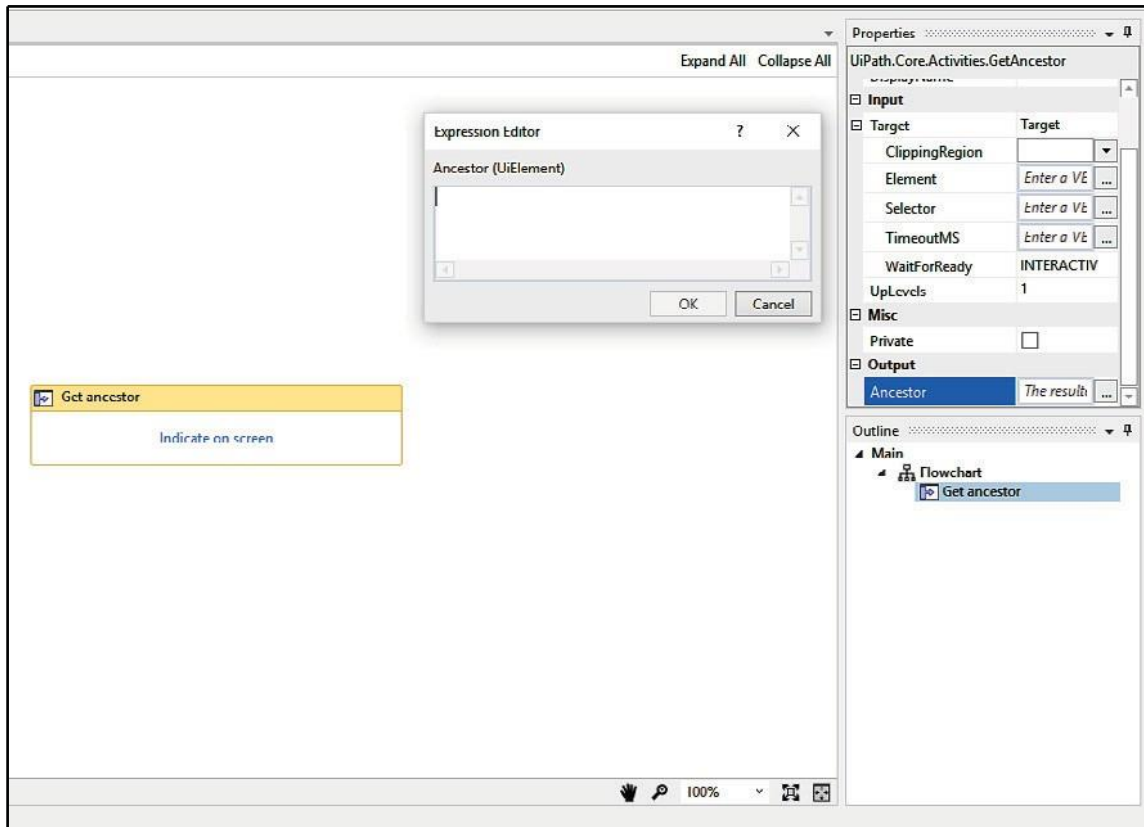
Just drag and drop this control, and indicate the UI element by clicking on **Indicate on screen**.

You can also look for its selector property after indicating the UI element for better analysis.

Get ancestor

This control is used to retrieve the ancestor of the specified UI element. You have to supply a variable to receive the ancestor element as output. You can specify the variable name in the **Ancestor** property of the **Get ancestor** control.

After receiving the ancestor element, you can retrieve its attributes, properties, and so on for further analysis:

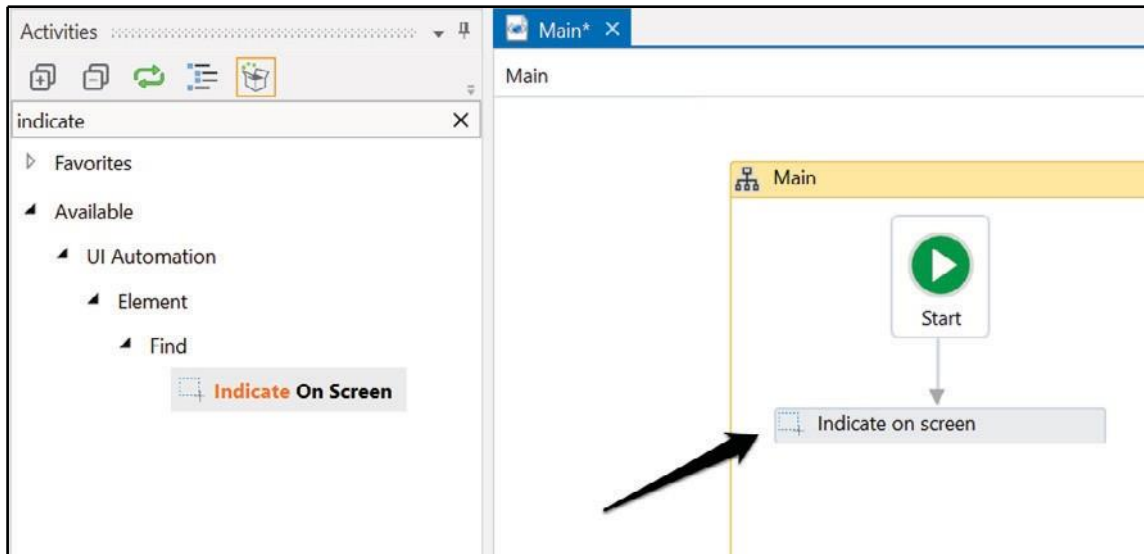


Just drag and drop this control and indicate the UI element by clicking on **Indicate on screen**.

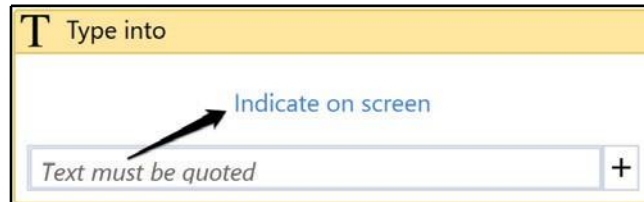
Indicate on screen

This control is used to indicate and select the UI element or region at runtime. It gives flexibility to indicate and select the UI element or region while running the workflow.

You just have to drag and drop this control in your project:



Do not confuse this with **Indicate on screen** written inside any activity like **Type into**. In previous examples, we have used **Indicate on screen** inside various controls (as shown in the following screenshot). This button is used to locate the region or UI element before the execution of the workflow, while the **Indicate on screen** control executes its process after the execution of the workflow:



Techniques for waiting for a control

There are three techniques through which we can wait for a control. They are:

1. Wait Element Vanish
2. Wait Image Vanish
3. Wait attribute

Wait Element Vanish

This activity is used to wait for a certain element to disappear from the screen. Let us see an example where the **Wait Element Vanish** activity is in use:

1. Create a **Blank** project and give it a meaningful name.
2. Drag and drop a **Flowchart** activity on the Designer panel. Also, drag and drop the **Wait Element Vanish** activity on the Designer panel. Set this activity as the **Start** node.
3. Double-click on the **Wait Element Vanish** activity, then indicate on the screen which element needs to vanish.

Wait Image Vanish

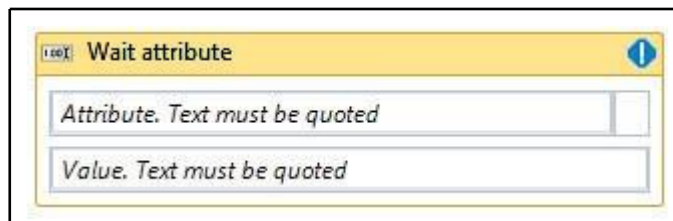
The **Wait Image Vanish** activity is similar to the **Wait Element Vanish** activity. This activity is used to wait for an image to disappear from the UI element.

The only difference between the **Wait Element Vanish** and the **Wait Image Vanish** activities is that the former is used to wait for an element to disappear, while the latter is used to wait for an image to disappear.

Wait attribute

This activity is used to wait for the value of the specified element attribute to be equal to a string. We have to specify the string explicitly:

1. Drag and drop a **Flowchart** activity on the Designer panel. Next, drag and drop the **Wait attribute** on the Designer panel. Now, right-click on the **Wait attribute** activity and set it as the **Start** node.
2. Double-click on the **Wait attribute** activity. We have to specify three values: attribute, element, and text property. We also have to specify the element on which we have to supply the value:



Hit the **Run** button and see the result.

Act on controls – mouse and keyboard activities

While working in UiPath Studio, we have to work with various types of controls, such as Find control, mouse control, keyboard control, and so on, to automate tasks. In this section, we are going to implement the mouse and keyboard activities.

Mouse activities

Those activities that involve interaction with the mouse fall under the category of mouse activities.

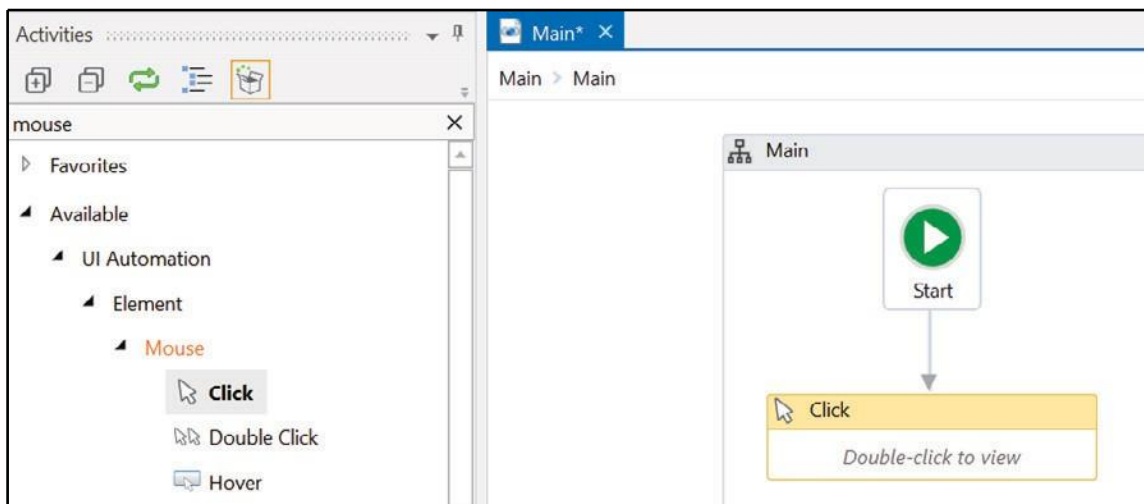
There are three mouse activities in UiPath Studio:

- Click activity
- Double-click activity
- Hover activity

The Click activity

When we have to click on a UI element on the screen, we generally use the Click activity. It is very easy to use the **Click** activity, as illustrated by the following example:

- 1 Drag and drop a **Flowchart** on the Designer panel. Search for `mouse` in the search bar of the **Activities** panel. Drag and drop the **Click** activity. Right-click on the **Click** activity and select **Set as Start Node**.
- 2 Double-click on the **Click** activity. Click on **Indicate on screen** and indicate the UI element you want to click on:



Hit the **Run** button to see the result.

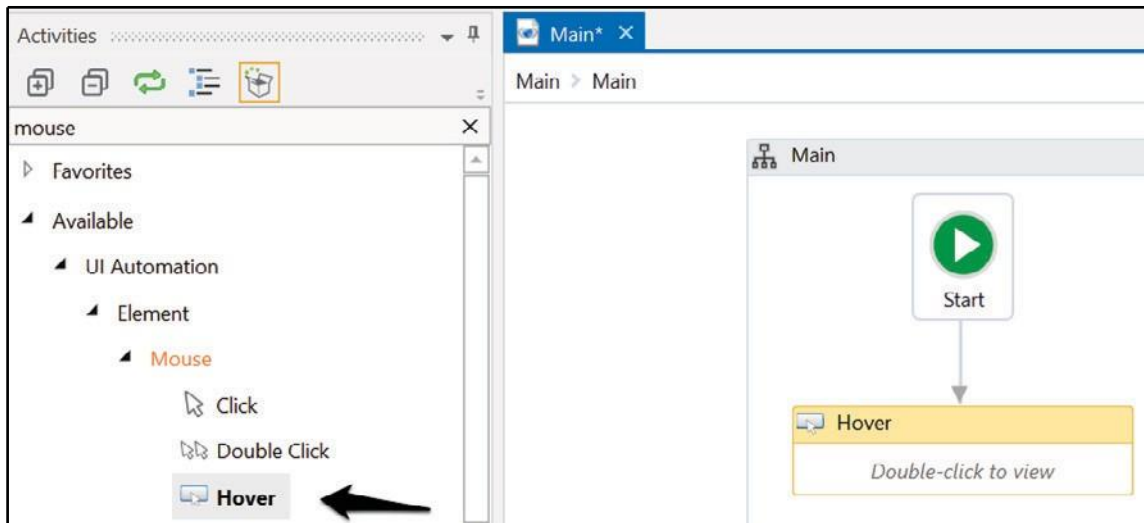
The Double-click activity

We have already seen the Click activity. The Double Click activity is similar to the Click activity. It just performs the double-click action. Using the Double click activity in your project is almost the same as click. You have to use the Double click activity instead of the Click activity and indicate the UI element, as we have done in the previous example.

The Hover activity

The **Hover** activity is used to hover over a UI element. Sometimes, we have to hover over a UI to perform an action. The Hover activity can be used in this case:

- 1 Drag and drop a **Flowchart** on the Designer panel. Search for `mouse` in the search bar of the **Activities** panel. Drag and drop the **Hover** activity. Right-click on the **Hover** activity and select **Set as Start Node**:



- 2 Double-click on the **Hover** activity. Click on **Indicate on screen** to indicate the UI element you want to hover on. That's it. We are done.

Hit the **Run** Button to see the result.

Keyboard activities

While automating tasks, we have to deal with keyboard activities a lot of a time. Keyboard activities generally involve an interaction with a keyboard.

In UiPath Studio, the following are keyboard activities:

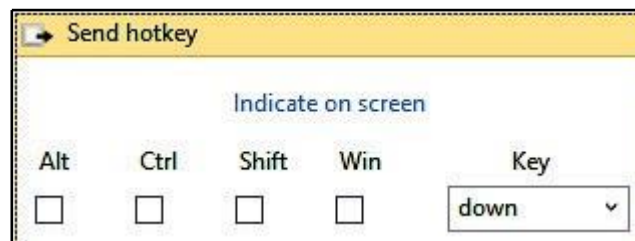
- Send hotkey
- Type into
- Type secure text

Send hotkey

This activity is used to send keystrokes from the keyboard as an input to the screen. Let us use an example.

In the following example, we will use the **Send hotkey** activity to scroll the Flipkart main page:

- 1 Drag and drop a **Flowchart** on the Designer panel. Search for **Keyboard** in the search bar of the **Activities** panel. Drag and drop a **Send hotkey** activity. Right-click on the **Send hotkey** activity and select **Set as Start Node**.
- 2 Double-click on the **Send hotkey** activity. Click on the **Indicate on screen** and indicate the required page (in our case, <https://www.flipkart.com>). You can assign any key by marking the checkboxes. You can also specify the key by selecting a key from the drop-down list. In our example, we have chosen the **down** key:



Hit the **Run** button to see the results.

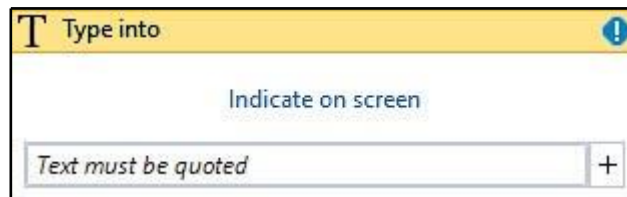


To see the proper result, just scroll down the website. When this activity does its work, you will be able to distinguish the change in the web page's position.

Type into activity

This activity is used to type the text into the UI element. It also supports special keys.

The **Type into** activity is quite similar to the **Send hotkey** activity. We have to send the keystrokes along with the special keys. Special keys are optional:



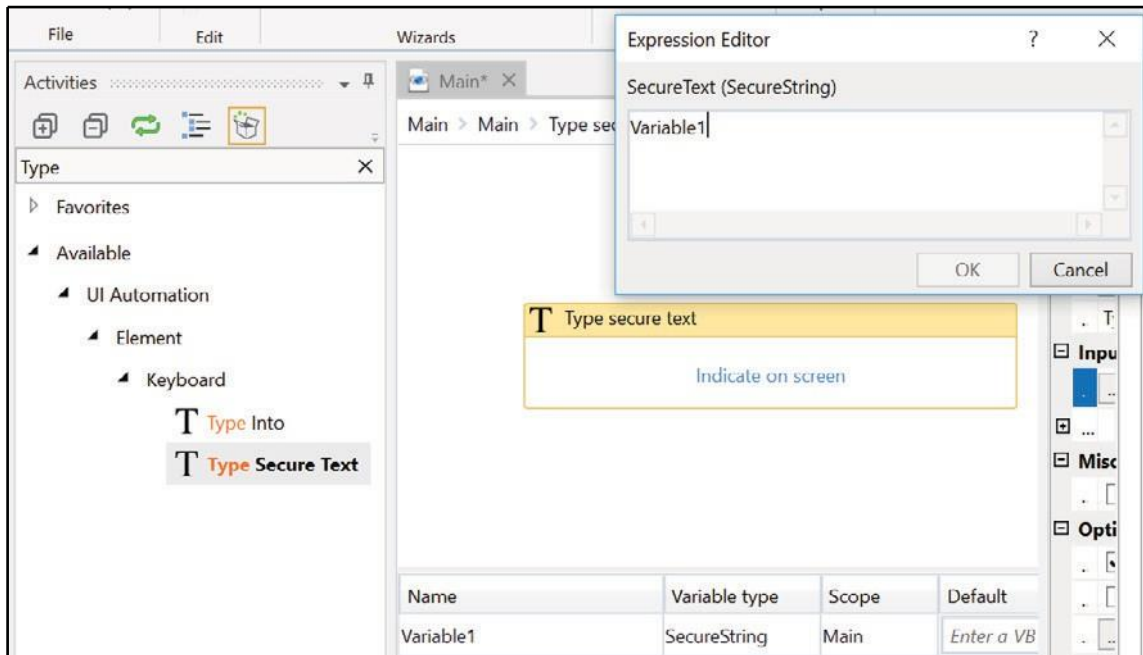
You can use this activity by simply dragging and dropping the **Type into** activity, and specifying the keystrokes and the special keys by clicking on the + icon and choosing the key from the drop-down list (if you wish to send special keys also). You also have to **Indicate on screen** the area where you want the text to be typed.

Type secure text

This activity is used to send secure text to the UI element. It sends the string in a secure way:

- 1 Drag and drop a **Flowchart** on the Designer panel. Search for **Keyboard** in the search bar of the **Activities** panel. Drag and drop the **Type secure text** activity. Right-click on the **Type secure text** activity and select **Set as Start Node**.

- 2 Create a variable of type **SecureString**. Now, double-click on the **Type secure text** activity and specify the variable's name in the **SecureText** property of the **Type SecureText** activity. You also have to indicate on the screen by clicking on **Indicate on screen**:



We have not assigned a value to the `variable1` of type `SecureString`. In enterprise scenarios, you will be using the **Get credential** activity. The **Get credential** activity is used when we have to use Orchestrator. We shall learn about Orchestrator later in the book.

Working with UiExplorer

UiExplorer is a more advanced version of the selector. It is a tool that gives us the flexibility to customize the selector.

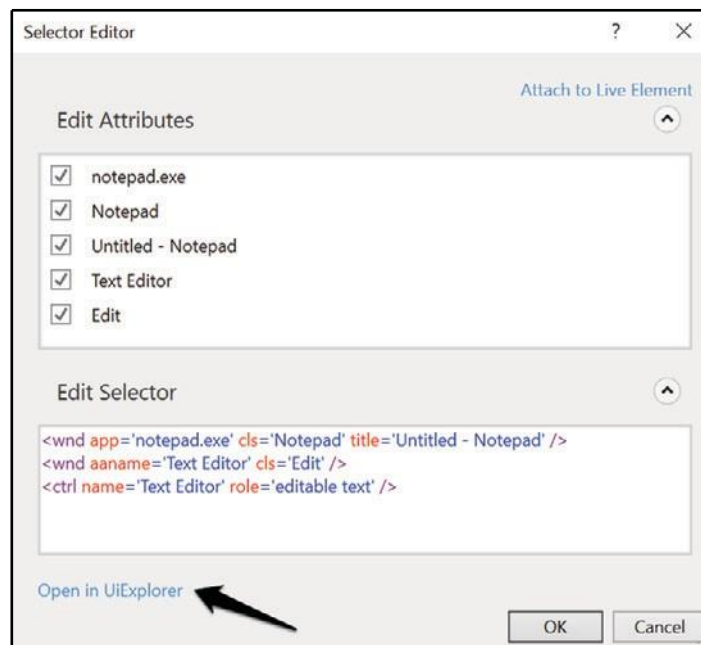
Let us try to understand the concept of UiExplorer with an example. In this example, we are going to type some text into a Notepad window. It is very easy to automate this task. You just have to use the **Type into** activity and **Indicate on screen** the area to be typed into and provide the text to be typed. Suppose you have opened a Notepad window, written some text into it, and then saved this file. If you want to write some text into it again, then UiPath Studio gives you an error.

There is nothing wrong with the implementation. What actually happens is that when you write some text in Notepad, UiPath Studio recognizes the file, app, type, title, and class, and saves this information for future recognition.

You have saved the file by providing a name. Hence, the title has been changed by the system (as the name of the Notepad window has changed). When you made the second attempt to write some text, UiPath Studio failed to recognize that instance of the Notepad window.

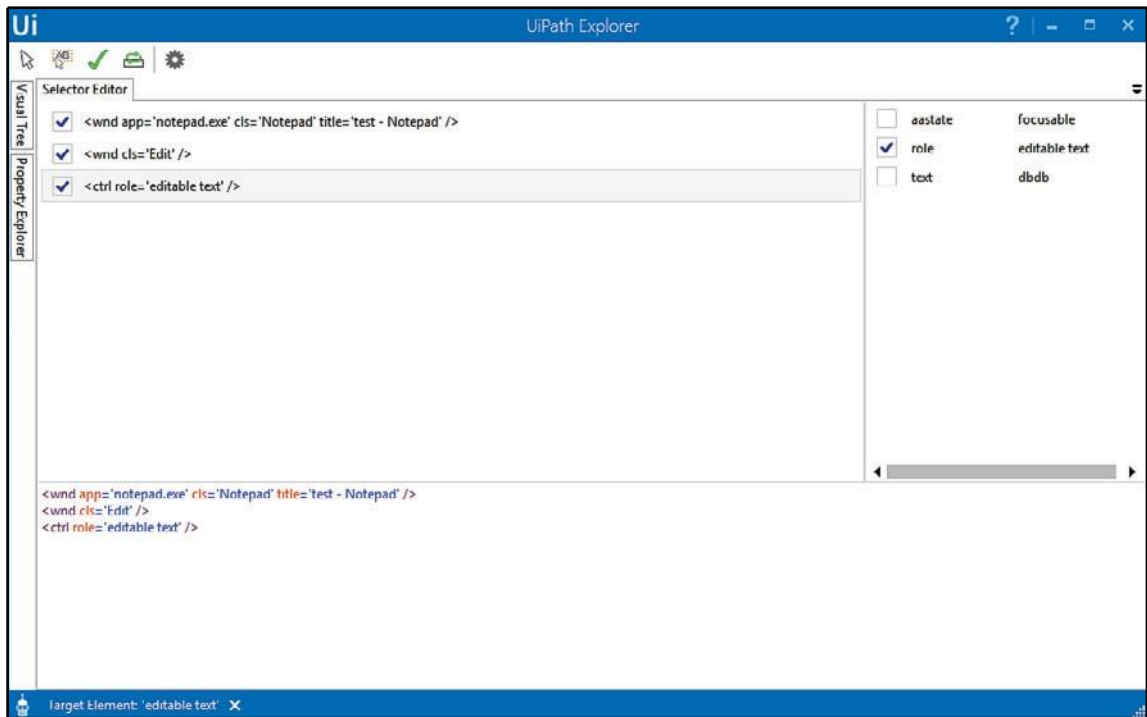
We can correct it by using UiExplorer. We have automated the task of writing some text in the Notepad window. Double-click on the **Type into** activity.

Click on the right side of the **Selector** property, expand **Target** property to find **Selector** property. A window will pop up. Click on the **Open in UiExplorer** button:



A window will pop up. You can see the **Selector Editor** window. Analyze all the text written there. You will notice the title: **Untitled-Notepad**.

You just have to edit this title. Just specify **test-Notepad** between the quotes:



The problem was when you opened the Notepad window, UiPath Studio saved the title attribute as **Untitled-Notepad**. You saved the file and its title changed to **test-Notepad**. When you tried to write some text next time, it did not recognize the title as it had been changed from **Untitled-Notepad** to **test-Notepad**.

You just have to edit the title attribute to remove the error.



UiExplorer is used to customize the selector and to view attributes and their associated values. View it carefully and inspect the attribute that should be changed.

Handling events

An event occurs when some action is performed. There are different types of events:

- Element triggering event
- Image triggering event
- System triggering event
-

Element triggering events

This type of event deals with clicking and keypress events.

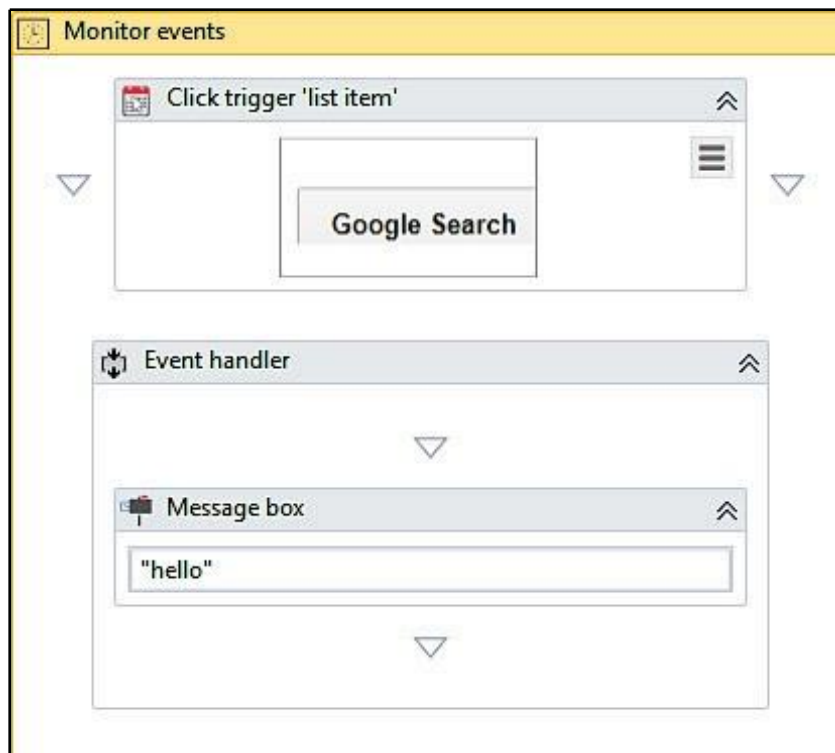
Click trigger

This event occurs when a specified UI element is clicked.

Before using the Click trigger, we have to use the **Monitor events** activity. Without **Monitor events**, the Click trigger cannot be used.

Double-click on **Monitor events**. Drag and drop the Click trigger inside **Monitor events**. Also, drag and drop the activity in the **Event handler** section of **Monitor events**. In this case, we have used the **Message box** activity and also specified the string value.

Inside the Click trigger, you have to indicate the UI element that you want to click on:



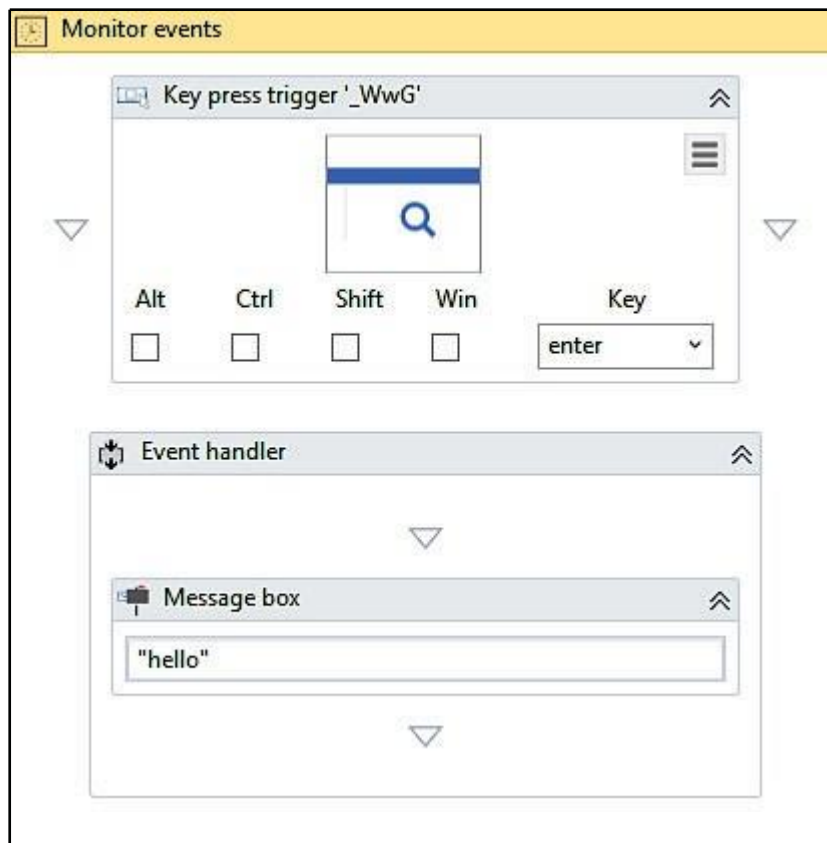
When the Click action is performed on the specified button, then the event handler will be called and the activities inside the event handler will be executed.

Key press trigger

This event is similar to the Click trigger. A Key press trigger event occurs when keystrokes have been performed on some particular UI element. It calls the **Event handler** when it is triggered.

While using **Key press trigger** event you have to specify the key or combination of keys.

Indicate the UI element on which you want to perform the action:



When the keys are pressed on the specified UI element, the event handler will be called.

Image triggering events

The Click image trigger is an image triggering event.

Click image trigger, as the name suggests, is used for when we click an image. You just have to use the Click image trigger event inside the Monitor event and indicate the image. Upon clicking the indicted image in the Click image trigger event, the event handler will be called.

System triggering events

The following are System triggering events:

- Hotkey trigger
- Mouse trigger
- System trigger

Hotkey trigger

This event is raised when special keys are pressed. As we have already looked at triggering events, you can use the Hotkey trigger event on your own. You have to use this event inside the Monitor event.

Specify the special key or combination of keys. Also, provide the event handler that will be called when the event occurs.

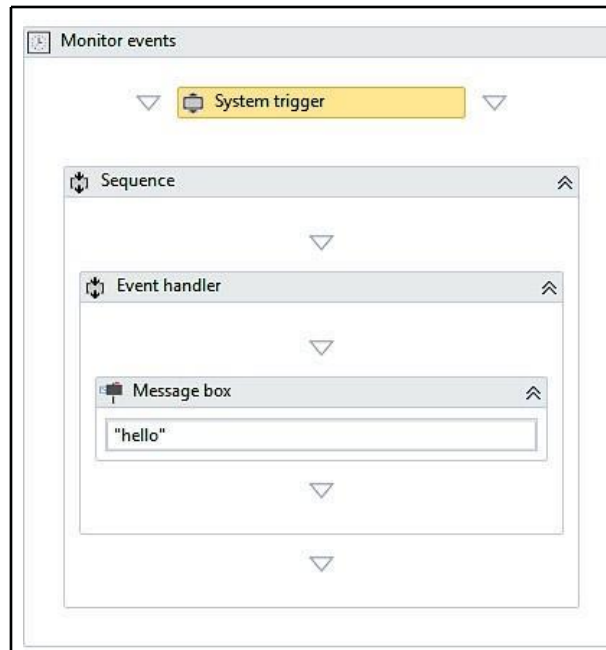
Mouse trigger

This event is fired when the mouse button is pressed. Use this event inside the Monitor event and specify the Mouse button: Either the left mouse button, middle mouse button, or the right mouse button.

System trigger

This event is used when you have to use all of the keyboard events, all of the mouse events, or both.

In the following screenshot, we have dragged and dropped the **System trigger** event into **Monitor events**. You can specify the trigger input property:



Revisit recorder

You have already learned about Task recording in Chapter 2, *Record and Play*. In this section, we will explore recording a bit more.

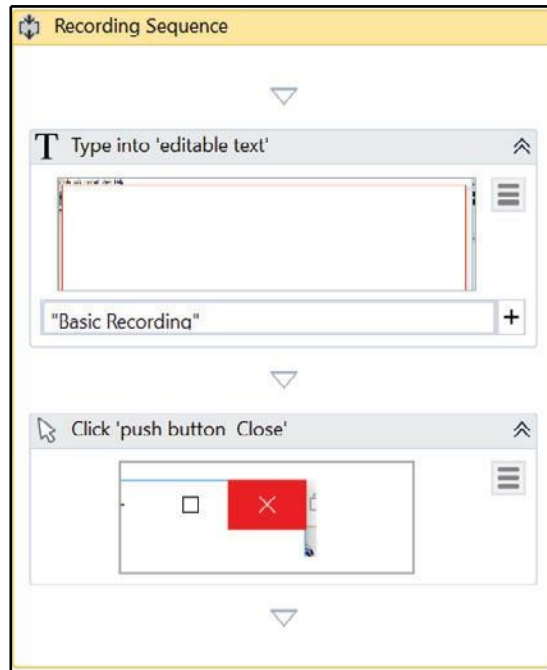
As we discussed earlier, there are four types of recording in UiPath Studio:

- Basic recording
- Desktop recording
- Web recording
- Citrix recording

Basic recording

This is used to record the actions of applications that have a single window. Basic Recording uses a full Selector. It works better for applications performing a single action. It is not suitable for applications with multiple windows.

There are two types of selectors, partial selectors and full selectors. A Full selector has all the attribute to recognize a control or application. The Basic recording uses full selectors.



Please note that, in the preceding image that there are different activities but those activities are not wrapped inside containers, it is generated by Basic recorder. Basic recording generates different activities and places them directly in the sequence with full selector.

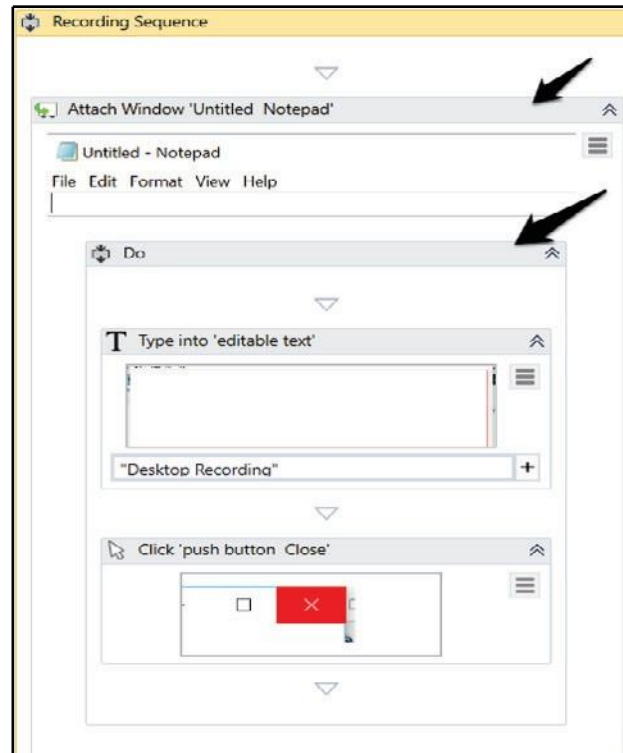
You have already seen how to automate tasks using the Basic recorder; now, let us cover other recorders.

Desktop recording

This is similar to Basic recording with the added advantage of working with multiple actions. It is most suitable for automating Desktop applications.

Desktop recorder generates Partial selectors. The Partial selectors, have a hierarchical structure. They are split into parent child views for recognizing the UI element properly.

Please note in the preceding image there is a **Attach Window** activities and other activities are nested under it. This flow is generated **Desktop** recorder:



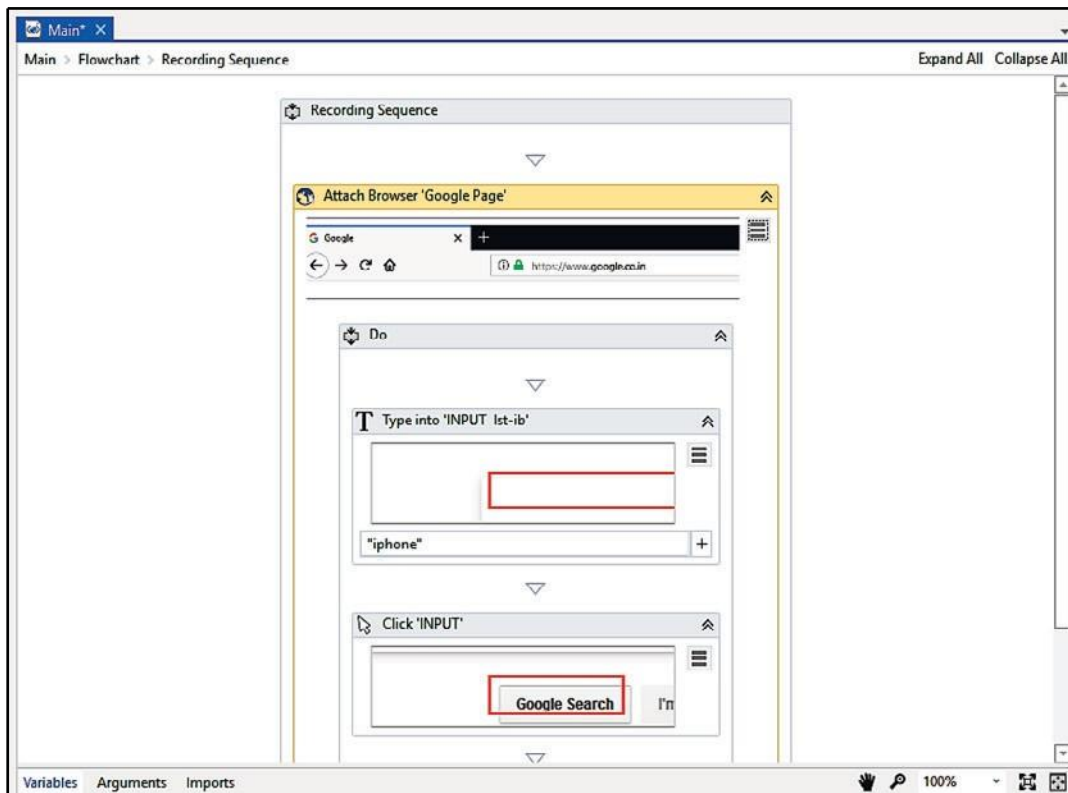
Web recording

Web Recording can be done by using the Web recorder. For recording web actions, the UiPath extension for that browser should be installed. Otherwise, you will not be able to automate tasks or actions using Web recording.

You just have to click on the Setup icon and then click on Setup Extensions. Now, choose your browser and click on it. The UiPath extension will be added to your specified browser.

Web Recording is similar to Desktop Recording. You just have to record the actions and save it.

Create a **Blank** project. Drag and drop a **Flowchart** activity. Now, click on the **Recording** icon and choose **Web** recording. You can record your actions on the web on your own and then save it. In our case, we have opened a web page using Google Chrome and logged in to <http://www.google.com>. Then, we started the recording by clicking on the Record button of the web recorder. Next, we typed some text in the search bar of Google and performed the Click activity. Then, we pressed the *Esc* key to exit the recording and clicked on the **Save and Exit** button. Now, a recording sequence is generated in our Designer panel. Connect this sequence to the **Start** node. Hit the **Run** button to see the result. In the following screenshot, you can see the sequence generated by the Web recorder:



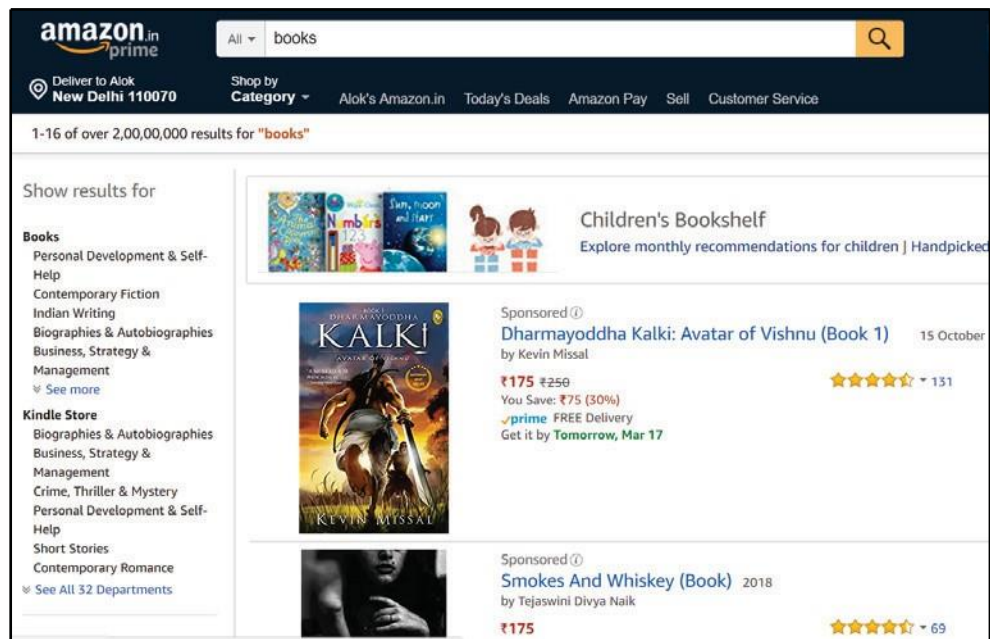
Before running the UiPath workflow, make sure you are on the Google homepage.

We have seen Web recording and it is very easy. There is also another option to extract information from websites.

We can easily extract information from websites using data scraping.

Suppose we want to extract data from Amazon's website. Say we want to search for books on Amazon and extract the search results. Extracting data from websites becomes very easy with data scraping:

1. Create a blank project and give it a meaningful name. Click **Create**.
2. Log on to Amazon's website and search for books. A detailed list of books is listed on your screen:



3. Drag and drop a **Flowchart** activity on the Designer panel. Now, click on the **Data Scraping** icon. A window will pop up.
4. Click on the **Next** button.
5. You have to indicate the first book's entities. Entities can be name, price, author, and so on. It is your choice.
6. Let's, indicate the book's name. After that, it will ask for the next book's entities. Indicate the second book entity as well. Click on **Next**.

7. This means you have to indicate the second book's entities: however, the entities will be the same. If you choose name as the first book's entity then you have to be specific and choose name as the second book's entity. You should not choose name as the first book's entity and then choose price as the second book's entity.
8. Again, a window will pop up asking you to configure the columns. You can also extract the URL. If you want to do this, check the **Extract URL** checkbox.
9. You can specify the column name as well. Click on the **Next** button.
10. As you can see, all the book names are extracted to a window. If you want to extract more columns or more entities, then click on **Extract Correlated Data** and you have to again indicate another entity of the book to extract more columns, as we have done previously. After that, all the data will be extracted and will be added to this table. Here, we have one column but if you extract more entities, then more columns will be added to this table:

The screenshot shows a window titled "Extract Wizard" with a close button in the top right corner. Inside the window, there is a section labeled "Preview Data". Below this label is a table with one column, "Column1", containing a list of book titles. At the bottom of the window, there is a button labeled "Edit Data Definition", a label "Maximum number of results (0 for all)", and a text box containing the number "100". At the very bottom of the window, there are four buttons: "Help", "Cancel", "< Back", and "Extract Correlated Data", followed by a "Finish" button.

Column1
General Knowledge 2018
This Is Not Your Story
The Power of your Subconscious Mind
Think and Grow Rich
The Monk Who Sold His Ferrari
I Do What I Do
Three Thousand Stitches: Ordinary People, Extraordinary Lives
The Alchemist
A Wasted Hour
The Boy with a Broken Heart
The Theory of Everything
General Knowledge for All (English)
Wings of Fire: An Autobiography of Abdul Kalam
As a Man Thinketh
[Sponsored]Discover India - A Travel Map

Edit Data Definition Maximum number of results (0 for all) 100

Help Cancel < Back Extract Correlated Data Finish

9. Click on the **Finish** button. If the results of your query span multiple pages, it will ask you to indicate the page navigation link on the website (Next button of the website that we used to navigate to another/next page). If the results of your query span multiple pages, click on the **Yes** button and indicate the link, otherwise click on the **No** button.
10. We have clicked on the **No** button. A data scraping sequence is generated in our **Flowchart**.

It will also generate a data table. You can retrieve the information from the data table.

Citrix

When dealing with the Remote Desktop connection, methods such as Basic Recording and Desktop Recording cannot be used. In an RDP environment, images will be sent from one desktop to another, and will be mapped by analyzing the position of the pointer of the mouse button. Hence, basic and desktop recording cannot be used, as these recording techniques fail to interact with the images. In a Citrix environment, we have the **Click Text** and **Click Image** activities, using which we can work with images with ease.

You can clearly see the activities that are listed in a Citrix Recording:

1. Click Image
2. Click Text
3. Type
4. Select & Copy
5. Screen Scraping
6. Element
7. Text
8. Image

All these activities are used extensively in a Citrix environment.

You can use these activities as you have used Basic Recording or Desktop Recording: the only difference is that you have to indicate a point on the screen, or you have to indicate an anchor element as you have used in previous sections.

Screen Scraping

Screen Scraping is a method of extracting data from documents, websites, and PDFs. It is a very powerful method for extracting text. We can extract text using the Screen Scraper wizard. The Screen Scraper wizard has three scraping methods:

- Full Text
- Native
- OCR
-

We shall elaborate on each of these methods one by one. One should have a clear understanding of these methods in order to know when to use which method. There will be situations when we have to choose the best method for our needs:

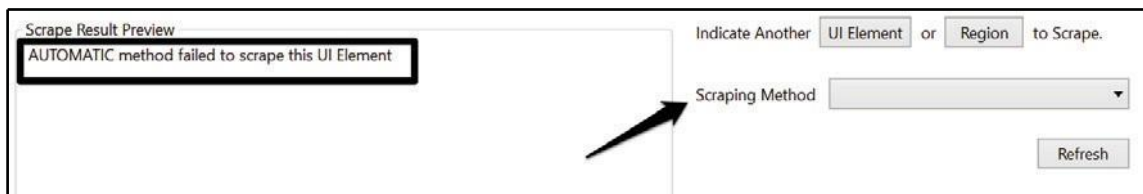
- **Full text:** The Full text activity is used to extract information from various types of documents and websites. It has a 100% accuracy rate. It is the fastest method among all three methods. It even works in the background. It is also capable of extracting hidden text. However, it is not suitable for Citrix environments.
- **Native:** This is similar to the Full text method but has some differences. It has a slower speed than the Full text method. It has a 100% accuracy rate, like the Full text method. It does not work in the background. It has an advantage over the Full text method in that it is also capable of extracting the text's position. It cannot extract hidden text. It also does not work with a Citrix environment.
- **OCR:** This method is used when the previous two methods fail to extract information. It uses the two OCR engines: Microsoft OCR and Google OCR. It has also a scale property: you can choose the scale level as per your need. Changing the scale property will give the best results:

Capability Method	Speed	Accuracy	Background Execution	Extract text position	Extract hidden text	Support for Citrix
Full Text	10/10	100%	Yes	No	Yes	No
Native	8/10	100%	No	Yes	No	No
OCR	3/10	98%	No	Yes	No	Yes

Let us consider an example of extracting text from the UiPath website's main page:

1. Create a **Blank** project and give it a meaningful name.
2. Log on to the UiPath website by logging in to <https://www.uipath.com/> in your browser.
3. Drag and drop a **Flowchart** activity on the Designer panel. Click on the **Screen Scraping** icon and locate the area from which you want to extract the information. Just choose an area on the UiPath website. A window will pop up stating that the **AUTOMATIC** method failed to scrape this UI Element.

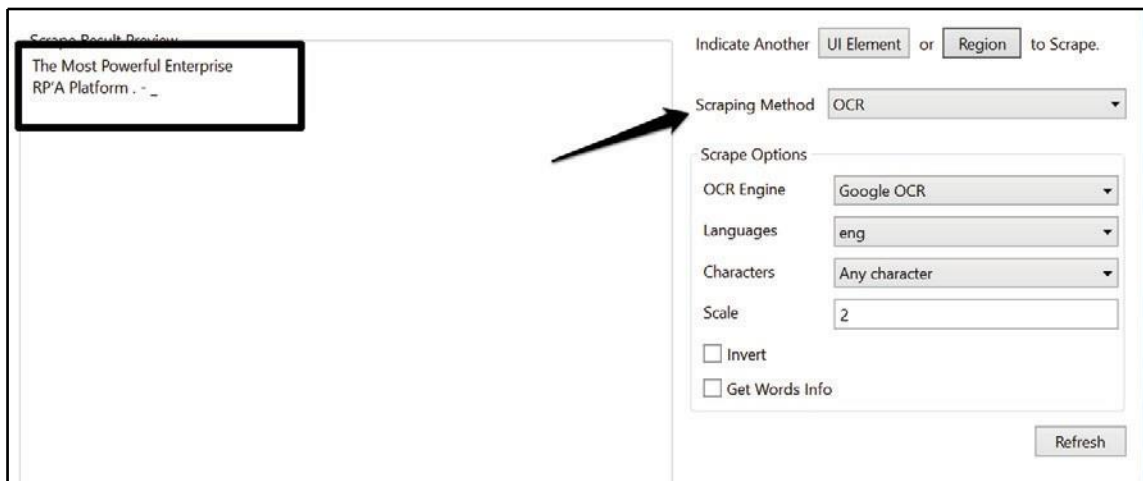
By default, the **Screen Scraper Wizard** chooses the best scraping method to extract data, but it failed to do so in our case:



4. Try choosing another method. We shall choose the Full text method. This too will fail. Next, choose the Native method. This will also fail, as you can see in the following screenshot:



5. This time, choose the OCR Scraping method. You can clearly see the extracted text:



When to use OCR

There are some scenarios where normal activities such as **Get Text** and **Click Text** activities fail to extract the text or perform an action. This is when OCR comes in, giving us the flexibility to perform actions when existing activities fail to do their job.

OCR stands for Optical Character Recognition. It is a text recognition technique that transforms printed documents that are scanned into electronic formats.

OCR is used mainly for images, scanned documents, PDFs, and so on, to extract information or perform actions. Extracting information or data from images, scanned documents, or PDFs is a very tedious job. Normal activities are not recommended for extracting these types of inputs. OCR uses a different method and approach to extract the information.

There are two OCRs available in UiPath Studio:

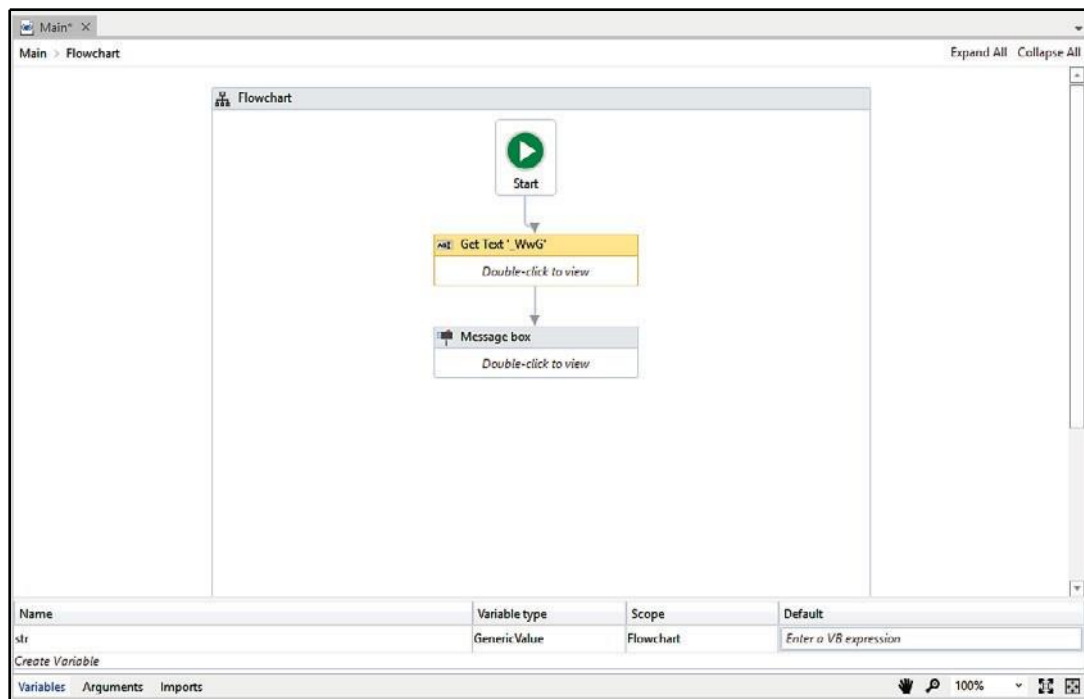
1. Microsoft OCR
2. Google OCR

Microsoft's OCR is known as MODI, and Google's OCR is called Tesseract. OCR is not limited to only these two types of OCR. You are free to use another type of OCR.

Let's see an example of when we should use OCR.

Consider we are going to extract some text from a Word document by using the **Get Text** activity:

1. Create a **Blank** project and give it a meaningful name.
2. Drag and drop a **Flowchart** activity on the Designer panel. Also, drag and drop a **Get Text** activity inside the Designer panel. Now right-click on the **Get Text** activity and choose **Set as Start Node**.
3. Double-click on the **Get Text** activity. Click on **Indicate on screen**. Now indicate the text from which you want to extract information. You have to supply the output value for receiving the text from the **Get text** activity. Create a **GenericValue** type of variable and specify the variable name `str`.
4. Drag and drop a **Message box** activity. Connect it to the **Get Text** activity. Double-click on the **Message box** activity and specify the variable's name (`str`) that you created earlier:



Hit the **Run** button to see the result. You can clearly see in this example that using the **Get Text** activity does not extract, text properly.

This is where OCR enters the picture. In the next section, we will see the extraction of text using OCR.

Types of OCR available

There are two OCRs available in our UiPath Studio:

1. Microsoft OCR
2. Google OCR

However, we are free to import other OCR engines into our project.

Both the Microsoft and Google OCR engines have their own advantages and disadvantages. The advantages of Google OCR include the following:

- Multiple language support can be added in Google OCR
- Suitable for extracting the text from a small area
- It has full support for color inversion It
- can filter only allowed characters

The advantages of Microsoft OCR include the following:

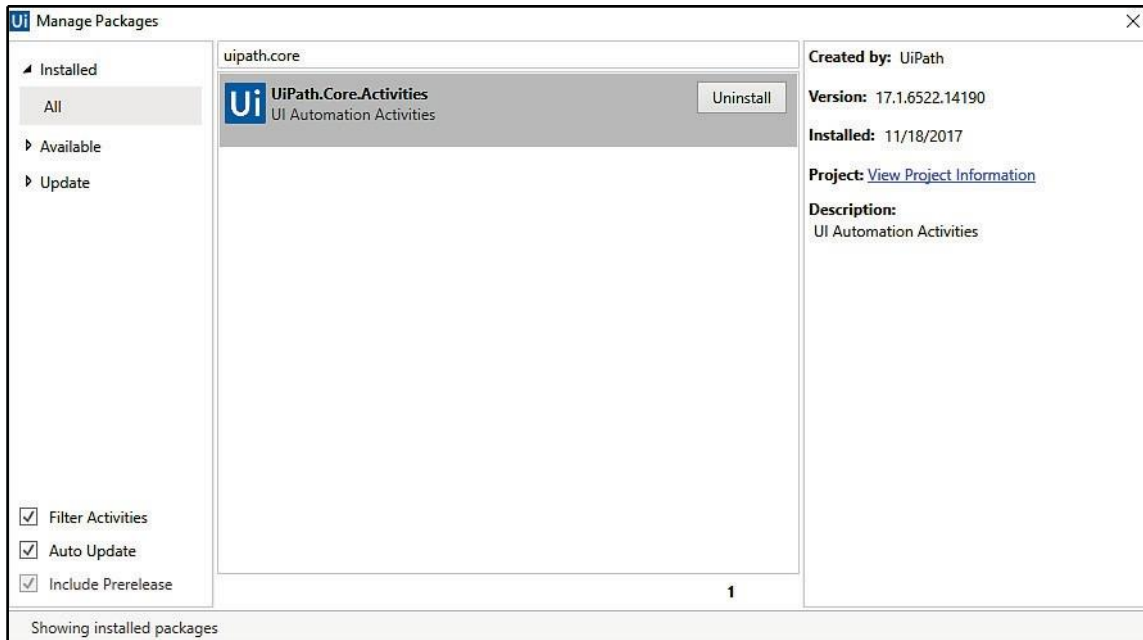
- Multiple languages are supported by default
- It is suitable for extracting text from a large area

OCR is not 100% accurate. It is useful for extracting text that other methods cannot successfully do. It works with all applications, including Citrix.

Microsoft and Google's OCRs are not the optimum for every situation. Sometimes, we have to look for more advanced OCRs to recognize more sophisticated text, such as handwritten documents and so on.

There is another OCR available in UiPath Studio, known as the Abbyy OCR Engine. You can find this OCR engine in the **Activities** panel by searching for `OCR`.

If you cannot find this OCR listed in the **Activities** panel, you need to install the `UiPath.Core.Activities` package:

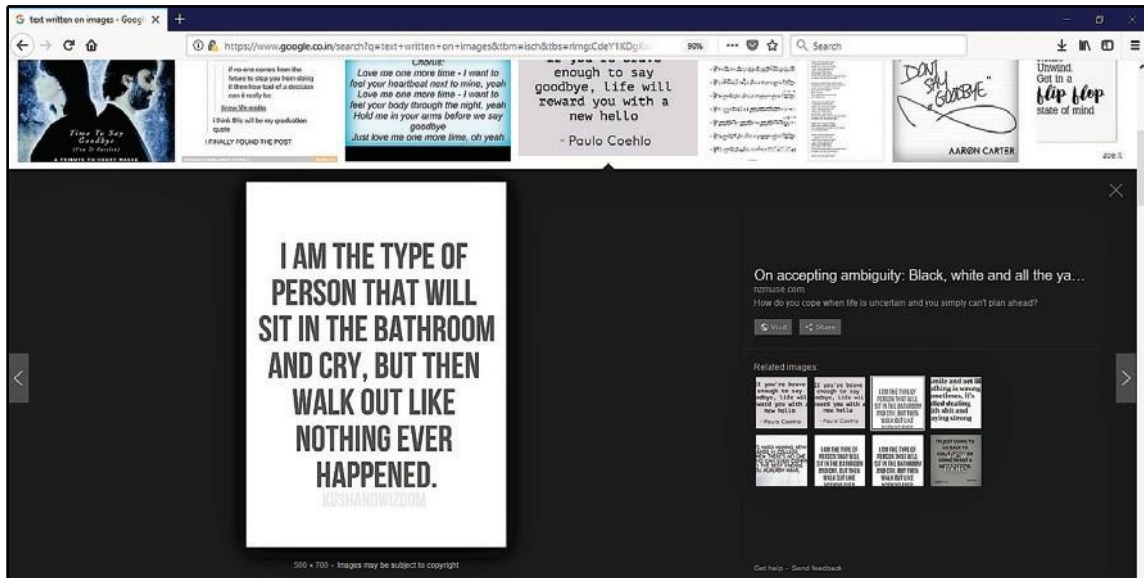


In the previous screenshot, the package has already been installed, which is why there is an **Uninstall** button on the right-hand side of `UiPath.Core.Activities`.

How to use OCR

In this section, we are going to see how we can use OCR. Suppose we have an image and we have to extract the text in it. In such a scenario, OCR becomes very handy.

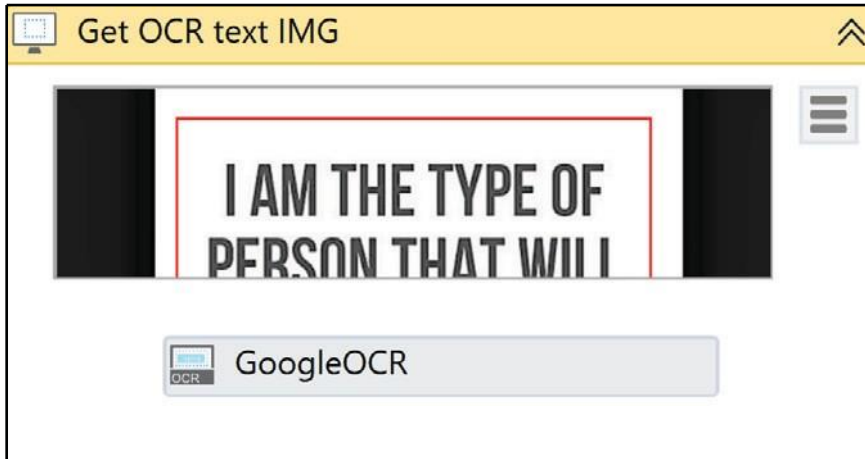
In the following example, we are going to use a random Google image with some text in it, as follows:



To extract text from the previous image, perform the following steps:

1. Open UiPath Studio and click on a **Blank** project. Give it a meaningful name. On the Designer panel, drag and drop a **Flowchart** activity.
2. Next, drag and drop a **Get OCR Text** activity from the **Activities** panel and set it as the start node. Double-click on it and click on the **Indicate on screen** option. Choose the specific area from which you want to extract the text from the image. In our case, we are using an image that we have searched for on Google.

- Now, click on the **Text** property of the **Get OCR Text** activity. A window will pop up as shown in the following screenshot. Right-click inside the window and choose **Create Variable**. Give it a meaningful name, press *Enter*, and click on the **OK** button. A variable will be created with that name:



- Drag and drop the **Message box** activity. Connect it to the Get OCR with text activity. Double-click on the **Message box** activity and specify the variable name that you have created earlier in the expression box. (In our case, it is the `result` variable).

Press *F5* to see the result.



Try to change the scale property of the OCR engine. It will give you better results in some cases.

Avoiding typical failure points

There are many scenarios where the normal implementation would fail. We are going to discuss these failure points and will see how to tackle them.

In this section, we will work with the following entities to tackle failure points:

- Selectors
- Scope of the variable
- Delay
- Element Exists Try
- / Catch
- toString method

Selectors

Sometimes, it is tedious to deal with selectors while working with them. This is because a selector has attributes, title, and class properties. When we select a UI element using the selector, it stores all these properties. Different instances of an application may have different properties of a UI element.

The problem with selectors is when you select a UI element, it captures its properties. These properties will differ when we select the UI element of a different instance of an application with the selector. Hence, the property will differ and the selector will fail to recognize the same UI element of another instance of the application.

We can easily fix this problem by using wildcard characters or by attaching it to a live element. Two wildcard characters are available with UiPath:

1. The question mark symbol, `?`, which replaces one character
2. The asterisk symbol, that is, `*`, which replaces a number of characters

We have to simply replace the variables (the name that changes continuously) with wildcard characters.

We can also use the **Attach to live element** option from the selector property window and indicate the element again. It automatically detects the variable properties and fixes them for us.

Scope of the variable

Sometimes we create a variable inside a Sequence or Do activity. In doing so, the scope of the variable is limited to only that activity.

When we try to access a variable from outside its scope, it cannot be accessed. We have to change the scope of the variable.

Delay activity

In some situations, we have to wait for a particular action. For example, when opening the Outlook application, it needs to connect to the server (for synchronization). When it is opened, it takes some time (the UI element is not stable at this stage).

In the meantime, the robot's activity is waiting for the UI element to be stable so that it can perform the action. After waiting for some time, if the UI element is not stable, it results in an error because the activity cannot find the UI element. Thus, we have to add a Delay activity to ensure that the UI element is stable for action. Specify the time for the delay in the expression text box of the Delay activity. This activity will delay the process for the specified period of time.

Element Exists

This activity is used to ensure that the required Element Exists. It is used to ensure that the element we are looking for exists in this context.

This is a good way of checking whether the activity exists or not.

Try/Catch

This is an exception handling mechanism used to tackle exceptions.

Put all suspicious activities inside the `Try` block. If an error occurs, it can be detected by the `Catch` block.

toString

Sometimes, we forget to use the `toString` method with variables and we end up with an error.

For example, when outputting an integer variable inside the **Message box**, we have to apply the `toString` method.

Summary

In this chapter, we learned about the various types of controls available in UiPath, including types of mouse and keyboard activities. We also revisited the Recording feature of UiPath Studio, this time learning more advanced features. UiExplorer is another topic we forayed into, which is basically used to customize the selector. Furthermore, we learned about OCR, its types, and its uses. Finally, we dealt with the extraction of data using Screen Scraping.

6 Tame that Application with Plugins and Extensions

So far, you have learned how to record steps to automate and also learn about control flow, and the use of variables and data tables. The most important part was understanding and mastering controls. Unless you are able to identify controls of an application properly, it is not possible to successfully automate a process. Now, in this chapter, we will step up to learn how to use external plugins and extensions. UiPath has many plugins and extensions to ease UI automation, apart from basic extraction and interaction with the desktop screen. These plugins allow us to directly interact with those applications or ease UI automation.

Some of the important sections that are going to be covered in this chapter are:

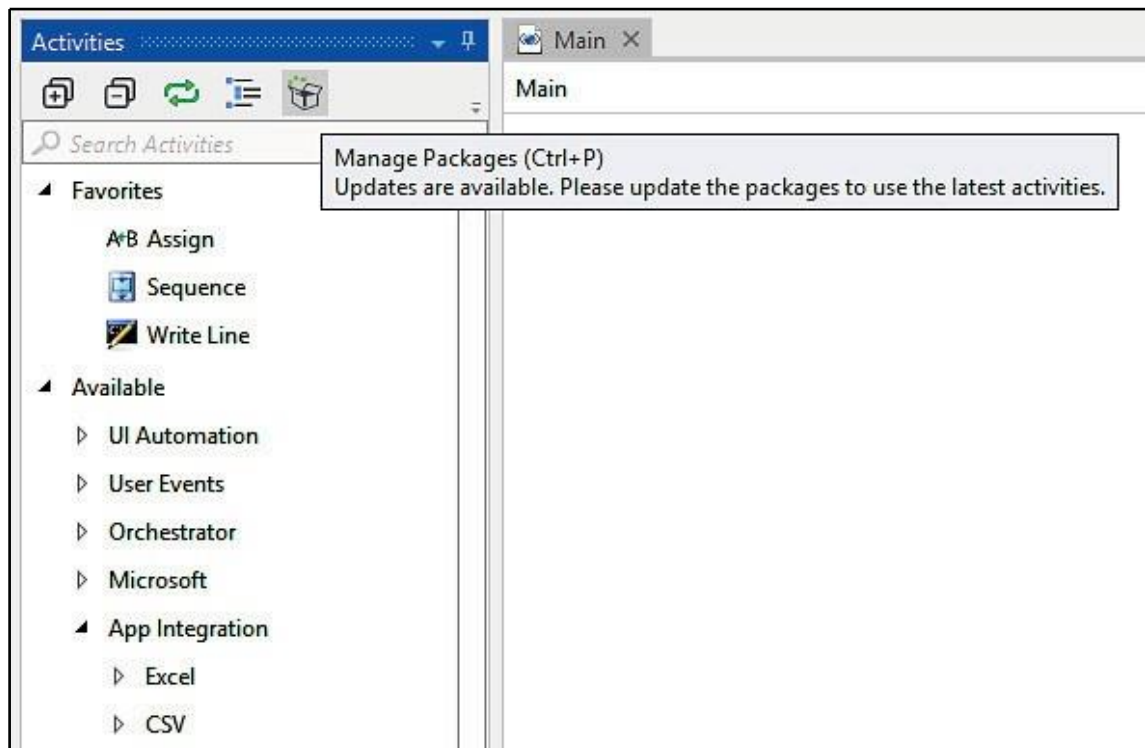
- Terminal plugin
- SAP Automation
- Java plugin
- Citrix Automation
- Mail plugin
- PDF plugin Web
- integration
- Excel and Word plugins
- Credentials management
- Extensions: Java, Chrome, Firefox, and Silverlight

Terminal plugin

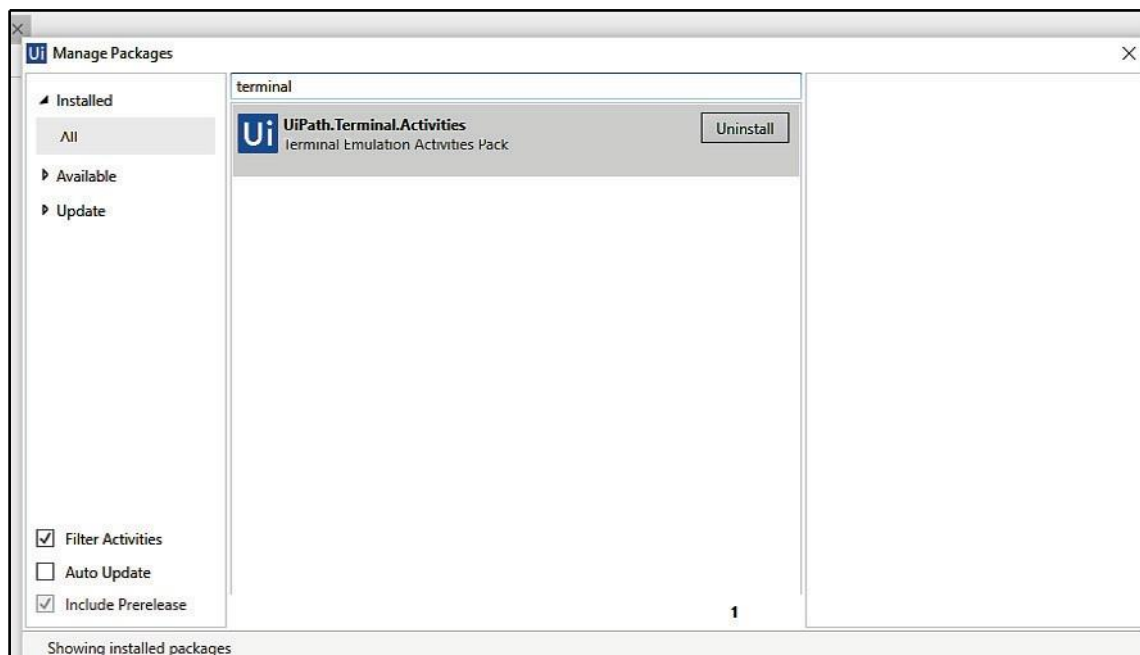
The Terminal plugin is used to execute commands in textual format (generally a black window). It works faster than the **Graphical User Interface (GUI)** methodology. It also has a broader scope in terms of authority and permissions.

In UiPath Studio, there is a NuGet Package called `UiPath.Terminal.Activities`. Terminal activity is pre-installed in UiPath Studio. In case it is not installed, we have to install it manually. To check whether the Terminal activity is installed or not, simply search for `Terminal` in the **Activities** panel. It will list all the terminal activities. If no activities are listed in the **Activities** panel, you have to install the `UiPath.Terminal.Activities` package.

To install Terminal activity NuGet Package, click on the **Manage Package** icon, as shown in the following screenshot:



The **Manage Packages** window will appear. Search for `terminal` in the search bar. As shown in the following screenshot, there is an **Uninstall** button next to **UiPath.Terminal.Activities**. This is because the **Terminal** activity is already installed in UiPath Studio. If it is not installed, an **Install** button will appear next to the **UiPath.Terminal.Activities**:



SAP automation

Whether in software or in the form of mechanical Robots, automation is everywhere. Businesses have not only the latest information but the most accurate too. In today's market, if businesses cannot be kept up to date then they will suffer.

SAP automation with UiPath is integrated with the latest techniques that are more suitable for organizations. Today, SAP automation is becoming the buzzword in the IT industry.

These days, error-prone databases and unfilled/partially filled documents are mostly discarded.

With SAP automation from UiPath Studio, these can be easily avoided. It automates tasks easily and makes organizations more productive.

Some of the advantages of UiPath's SAP Automation are listed as follows:

1. It is compatible with all the SAP automation techniques that are best suited to the organization.
2. It is accurate (99.999 % of the time) and delivers the best results. It avoids typical errors that humans/employees would be likely to make.
3. Human employees demand higher salary and consideration has to be given to (**full-time employees**) FTE too. SAP automation has a modular approach to FTE.
4. It is scalable. Just set it up with hundreds of Robots and you do not have to monitor them. There is no need to constantly watch them. The Robots can work independently.

SAP Automation requires some steps with which you should be careful while automating. There are some situations when the Click activity fails to recognize the UI element. Click Text activity and Click Image activity are used when Click activity does not work in some situations while automating SAP.

There are some scenarios when we have to extract tabular data from SAP. Iterating through the table is not easy. In such case use selectors to recognize the table cell. You can implement a loop to iterate through each table cell. Now, how will you know when your loop should stop iterating the table cell? You can put your code inside a Try Catch activity and when an exception occurs (exception occurs when the loop encounters an empty table cell or when the end of the table has been reached), it is caught by the Catch block. In this way, you will be able to iterate through all the table cells.

While interacting with checkboxes and radio buttons, make use of Get attribute activity to check whether they are checked or not.

When dealing with elusive UI elements, for example, a small button to the right of some text, you have to think in terms of human actions. How would a human react to such steps?

How SAP Automation affects data entry jobs

Data entry is a complex task. Employees have to work in a smart manner during the entire process, constantly checking for errors. There are some tasks that employees/humans do well, such as scanning a form for some information and extracting it. They can categorize the documents properly. Tasks like these are considered to be difficult for a system/computer. Of course, employees/humans can make errors that no software ever would. UiPath takes the best of both worlds—the benefits of automation and the benefits of mimicking humans, making software Robots that can be trained to scan forms, copy data, or notice a key being pressed. This also significantly reduces errors in the programs (in comparison to human employees undertaking the same tasks).

In addition to these processes, UiPath uses a method to ignore unrelated information on a website, SAP software, or any other application, only giving priority to the important ones.

It means that no matter how much your SAP application is difficult to handle, UiPath handles every action with ease. It can scale with any application on any platform.

Examples where SAP automation is used commonly

Some examples in which SAP automation is used are:

1. Filling in a form from any application.
2. Copying and pasting data between SAP and other applications.
3. Comparing data fields on the screen.
4. Updating the status of an entity in a system.
5. Scraping data from any application/websites.

UiPath is simple to use. In fact, there is no need to know a programming language or any scripting language. UiPath's Robots can be trained by the visual programming interface. You can define a complex workflow for your existing application and train your Robot. Once trained, Robots can run independently at a lower cost. It is estimated that a software Robot's work efficiency is equal to three employees. It saves a lot of time and money.

UiPath Studio comes with built-in libraries and activities so that the Robot can be trained and processes can be automated. It means that it can copy and paste the entities from one application to the other so that employees have more time to do complex logical work. It increases productivity and efficiency.

Java plugin

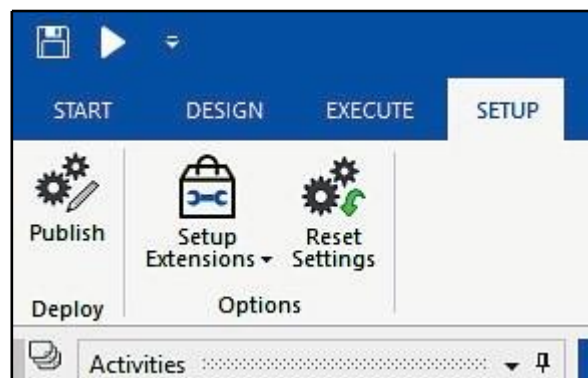
The Java plugin software is a component of the **Java Runtime Environment (JRE)**. The JRE allows applets (software programs written in the Java programming language) to run in various browsers.

Why are we using the Java plugin with UiPath Studio?

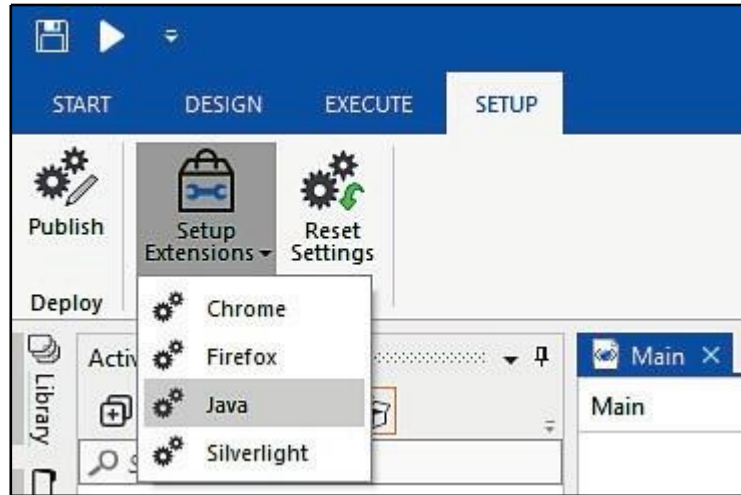
Suppose we have to automate a Java application. We cannot use pre-installed activities with the Java application as it will not recognize them properly. Hence, in order to use activities with the Java application, we have to install the Java plugin.

Follow these steps to install the Java plugin inside UiPath Studio:

1. Click on the **SETUP** wizard:



2. Next, click on **Setup Extensions** and choose **Java**:



A confirmation window will pop up stating that the Java plugin has been installed successfully.

To check whether the Java plugin is working properly, open UiExplorer, click on any Java application, and select an element. If the entire window is selected instead of that element, then your Java plugin has not been installed successfully. On the other hand, if the element is selected properly, then your Java plugin has been installed successfully.

Citrix automation

We have previously dealt with common automations—automating desktop applications or web applications. It is easy when we deal with these applications having graphical user interfaces. UiPath identifies the elements that we have clicked on and recognizes them.

Thus, the next time the Robot executes a process, it successfully identifies the same element. We have already seen these types of action.

But what if we have a remote desktop connection and we need to automate an application using this remote desktop connection? It will be a tedious job.

Can we automate an application running on another machine while we are accessing it remotely with the activities that we have used with simple GUI? The answer is no.

Let's investigate why this is so. Suppose we have to automate a desktop application so that the Robot does all the necessary actions on that application. We can simply use the click, double-click, and other activities to automate it. We cannot, however, use these activities when establishing a connection with another system remotely. Why do normal activities such as click and double click not work with a remote desktop connection? The problem with a remote desktop connection is that it sends the images of a system to another system. Recording activities such as click or double click may not accurately capture the position of that control inside the remote system.

You will get a better understanding with this example. Consider that machine A has a screen whose resolution is 1366×768, while another machine B has a screen whose resolution is 1024×768 resolution. Suppose we are connecting machine A to machine B using a remote desktop connection.

Now machine B with a resolution of 1024×768 is being accessed by machine A. What actually happens is that the frontend image of the machine B window is being sent to machine A. Hence, we cannot click on an image.

Since both machines have different resolutions, sending the coordinates of an element to another machine is error-prone or difficult during the remote desktop connection.

We have pointed out two problems when automating through the remote desktop connection:

1. We cannot click on an image
2. Sending the coordinates of an element to another machine is error-prone

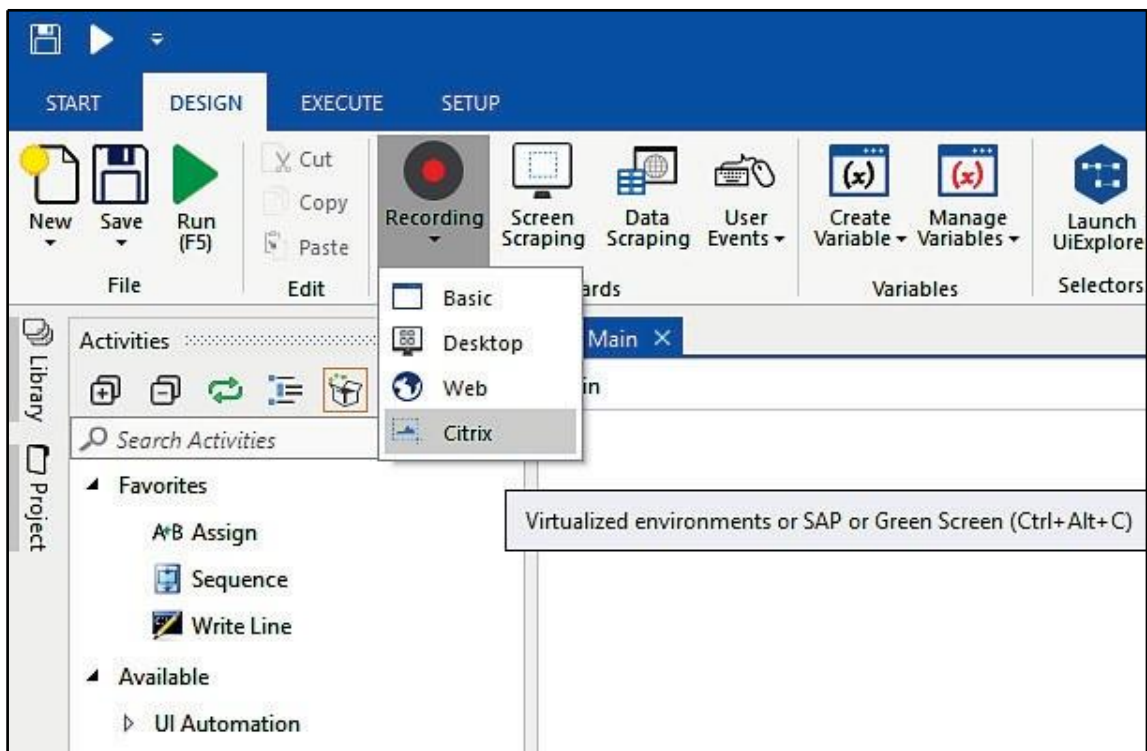
You will be wondering how we can overcome these two problems. Well, UiPath Studio comes with an environment called Citrix. Using the Citrix environment, it becomes very easy for us to automate an application when accessing it remotely.

It has a lot of options so that clicking on an image or sending the coordinates of an element become easy.

Some activities that Citrix supports are:

1. Click Image
2. Click text
3. Type
4. Send hotkey
5. Select and Copy
6. Screen Scraping and Scrape Data
7. Copy text

To automate using the Citrix environment, we have to select the Citrix Recording mode. In UiPath Studio, click on **Recording** and select **Citrix**:



Now, you can use a variety of controls and activities that help in automate applications remotely.

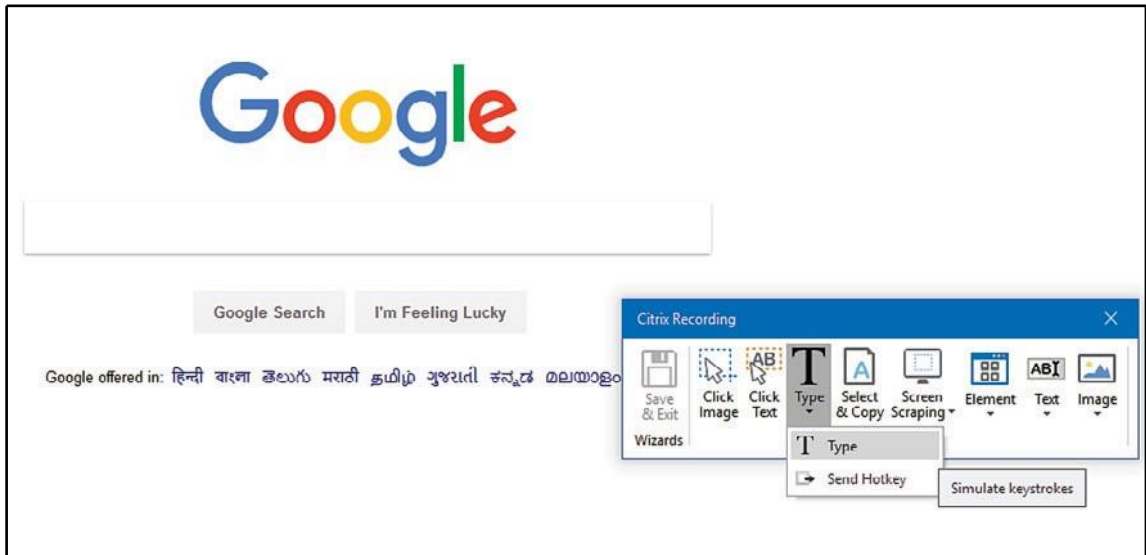
How does the Citrix environment work?

Citrix gives us the flexibility to work in **Remote Desktop Protocol (RDP)**. It captures the image and its position along with its relative elements so that they can easily be identified on another machine. No matter what the resolution of the screen, it can recognize elements easily. Have a look:

(We are not going to use RDP here. This is an example just to demonstrate the activity of Citrix).

Suppose we have to type into the Google search bar and click on the **Search** button.

Click on the **Recording** icon at the top of UiPath Studio. Select the **Citrix** option. A window will pop up. Now navigate to **Google** and click on the **Type** activity from the pop-up menu:



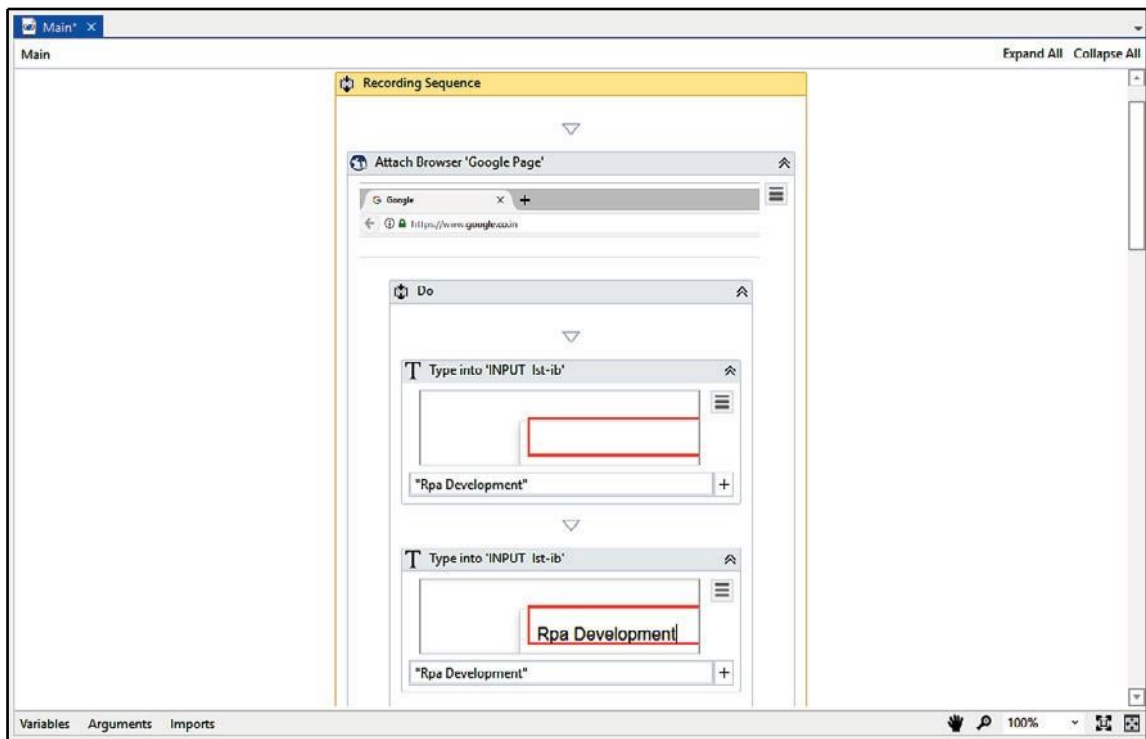
A pop-up window will be shown. Just type the text that you want to search for. Also, check the **Empty Field** option. Press **Enter**:



It will again show you a pop-up window. Now select **Click Image** from the Citrix Recording activity. You have to select the whole search area of the Google search bar. This time, it will ask you to indicate a point on the screen. Just point to the element that you have previously selected (in our case, it was the search bar area):



Click **OK**. Press **Save & Exit** and we are done. You can clearly see that UiPath generated the sequence shown in the following screenshot:



Press *F5* to see the result.

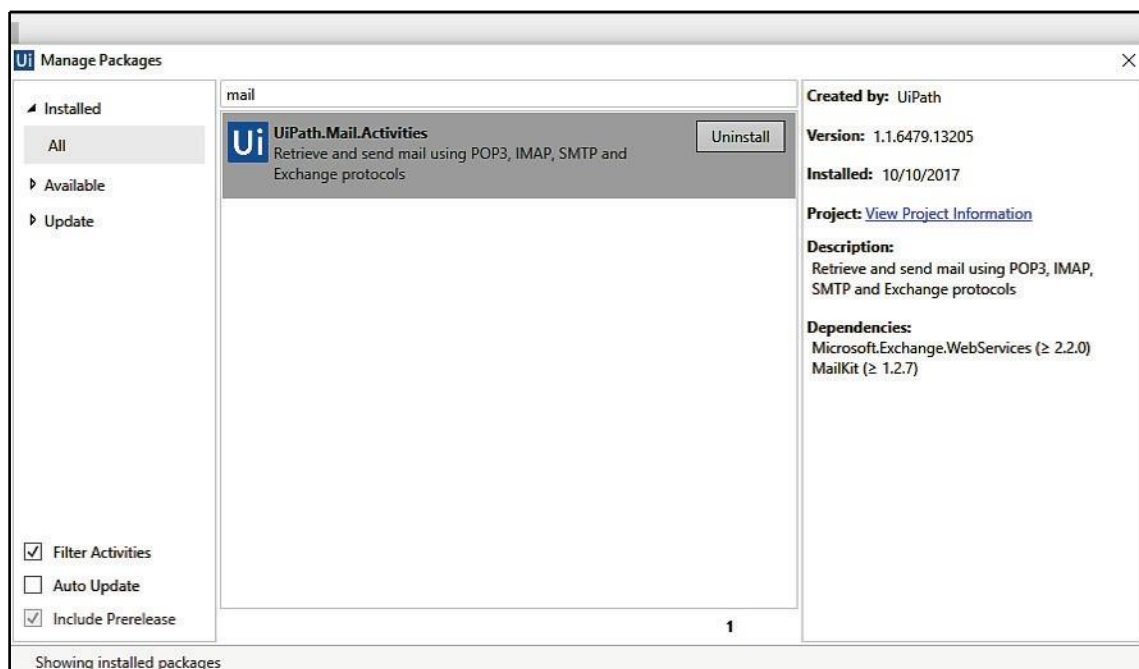


We have not used the **Open browser** activity. So, you have to log on to Google.com before executing this program. If you do not want this, then drag and drop the **Open Browser** activity before Recording Sequence.

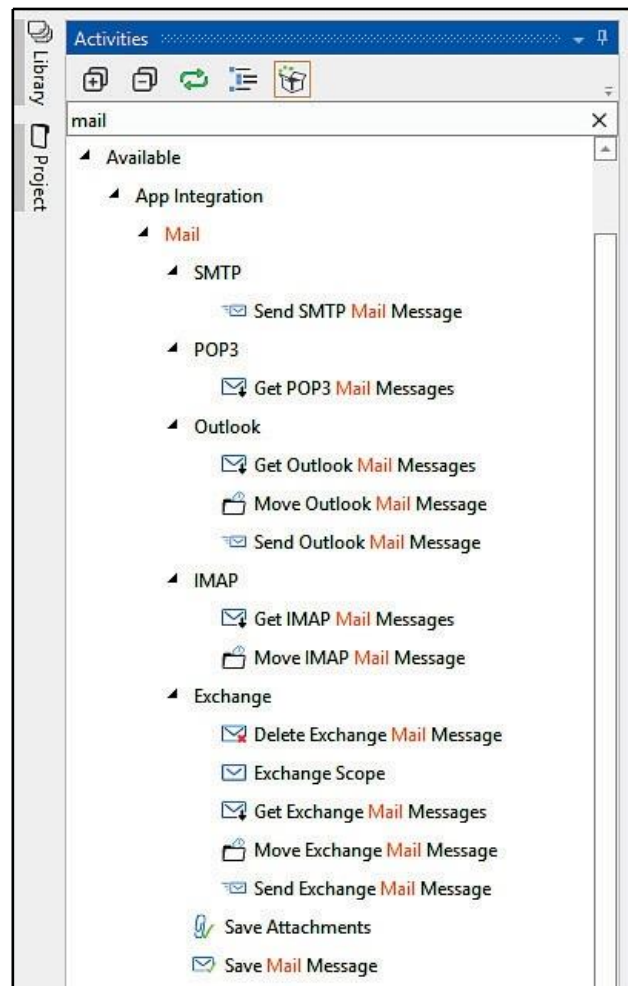
Mail plugin

To use any Mail activities, you have to install the Mail package. To check whether the Mail package is installed or not, simply search for `Mail` in the **Activities** panel. It will list all the mail activities. If the activities are not listed in the **Activities** panel then you have to install the `UiPath.Mail.Activities` package.

To install the Mail NuGet package, click on the **Manage Package** icon at the top of the **Activity** panel. The **Manage Packages** window will appear. Search for `mail` in the search bar. As shown in the following screenshot, there is an **Uninstall** button next to **UiPath.Mail.Activities**. This is because the Mail activity is already installed in UiPath Studio. If it is not installed, an **Install** button will appear next to the **UiPath.Mail.Activities**:



There are various mail activities that are used when working with UiPath Studio:



Some frequently used Mail activities are:

1. **SMTP:** It is used to send mail messages:
 - **Send SMTP Mail messages** activity: This activity is used to send a mail to another mail.
2. **POP3:** Although this is not the preferred choice, it is still used to receive mail messages:

- **Get POP3 Mail Messages** activity: This activity is used to receive mail messages.

3. **IMAP**: It is used to receive mail messages. It is a better option than POP3:
 - **Get IMAP Mail Messages** activity: This activity is also used to receive the mail messages. It gives us the flexibility to manipulate mail messages and can be accessed remotely.



Once you get familiar with these, you can easily try the rest of the activities.

PDF plugin

PDF stands for **Portable Document Format** and it is used to keep the document platform independent. Why do we use PDF?

Suppose we have a Microsoft Word 2007 application installed in our system and we make a .doc file. This doc file can be opened on any system. Consider a system with Microsoft Word 2017 installed on it. If we view that doc file in this application, the format of the application is not going to be the same. This is because both Microsoft applications have different sets of architecture and specifications. Their format is not the same.

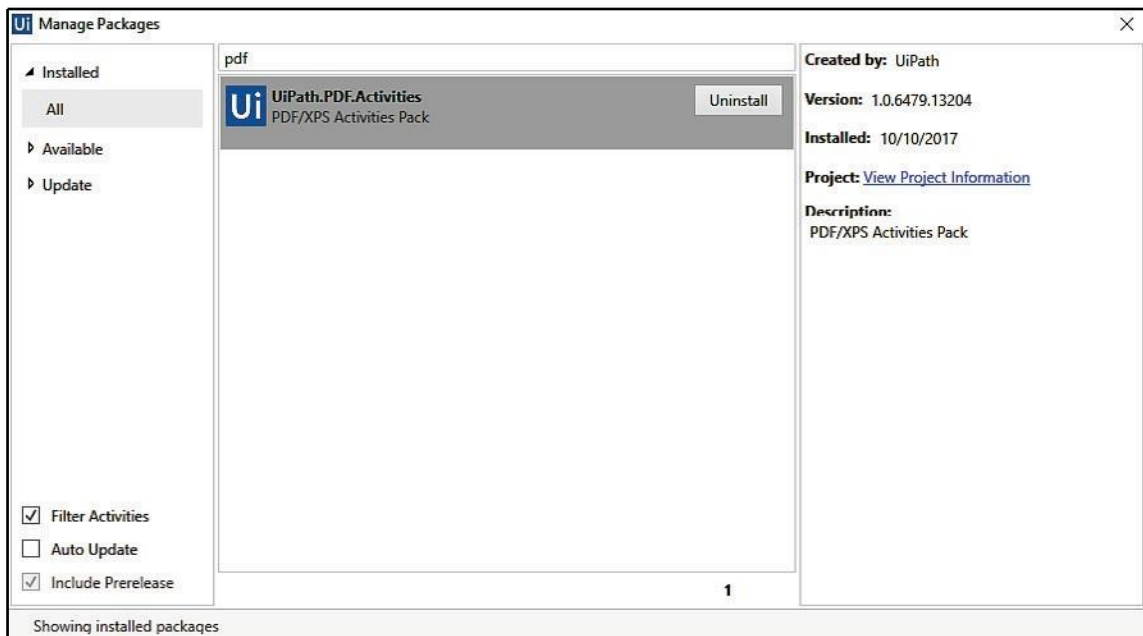
Here, PDF comes into action. It remains the same across all systems. That is why all confidential documents are sent or received using PDF. Also, if you do not want to change the behavior of the document across different platforms, you should convert the document into the PDF format.

To use any PDF activities, you have to install the PDF NuGet package. To check whether the PDF package is installed or not simply search for PDF activities in the **Activity** panel. It will list all PDF activities. If PDF activities are listed in the **Activities** panel, you have to install the `UiPath.PDF.Activities` package.

To install the PDF NuGet package, click on the **Manage Package** icon at the top of the **Activities** panel.

The **Manage Packages** window will appear. Search for `PDF` in the search bar. As shown in the following screenshot, there is an **Uninstall** button next to **UiPath.PDF.Activities**. This is because PDF activity is already installed in UiPath Studio. If it is not installed, an

Install button will appear next to **UiPath.PDF.Activities**:



In our case, the PDF activity is already installed. That is why there is an **Uninstall** button.

Some frequently used PDF activities are:

1. **Read PDF Text:** It is used to read the text written on any PDF document. However, the **Read PDF Text** activity cannot guarantee extraction of the entire text.

OR

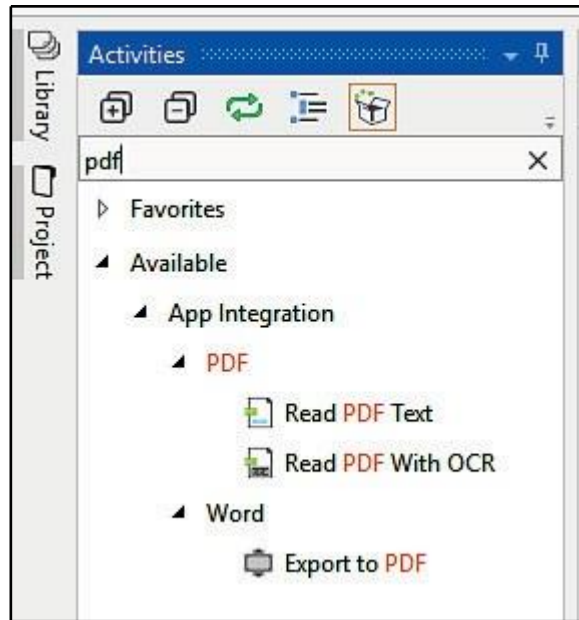
We can extract all the fields from the PDF file by using the Screen

Scraping activity. Start scraping the PDF file by clicking on the Screen Scraping icon in the menu and simply locating the area in which we have to extract the data. If the text extraction fails, then change the extraction type to OCR with a scale of 3 and above. Choose either Google or Microsoft OCR.

2. **Read PDF with OCR:** It is used to read the image part of the PDF file. Suppose there is an image in the PDF file and there is some text written on it. The Read PDF text activity will fail to read that text. This is where OCR is used. There might be scenarios in which some text is written on a colorful background. Such cases can easily be tackled by

using the **Read Text With OCR** activity.

If both **Read PDF Text** and **Read PDF with OCR** methods fail to extract the text, we still have the Screen Scraping method to extract the data from PDF (sometimes, we have to indicate the related element in order to recognize the text):



Select the proper method according to your needs and whatever is best suited to your project.

Web integration

Creating a web project, such as a company's website, B2B portal, or e-commerce website, requires the involvement of a wide range of technologies such as database design, networking, design, UX, user accessibility, SEO, and project management. These technologies also require HTML, CSS, JavaScript, JQuery, AJAX, system analysis and design, testing, operation, and a number of other methodologies. All of these activities can be classified under web integration.

Web integration, therefore, involves a wide process of connecting all these technologies and components that are essential for completing the web project.

The following are the methodologies that are used extensively for integrating with the web:

- **Application Programming Interface (API)**
 - **Extensible Markup Language (XML)**
 - **Simple Object Access Protocol (SOAP)**
 - **JavaScript Object Notation (JSON)**
 - **Representational State Transfer (REST)**
1. **API:** API integration is used quite frequently. It is impossible to find a modern web application or website that is not exposed to an API. API integration allows a software or web application to interact with other software or web applications in real time.

UiPath Studio works both ways: It can automatically extract data from one application and pass it to a web service. Also, it can retrieve data from a web service and input it to another application.

You have already explored and implemented how UiPath automates user interfaces. API is the easiest way to connect two applications or systems over the internet.

2. **XML:** XML is a markup language like **Hyper Text Markup Language (HTML)**. XML was designed to store and transport data and also to be self-descriptive. We can say that it extends the functionality of HTML. It is a software-independent and hardware-independent technique for storing and transporting data, for example:

```
<Message><To>John</To>

<From>Ava</From>

<Subject>Reminder</Subject>

< Message Body>Do not forget to meet me this
weekend!</Message body></Message>
```

You can make any Parent-Node structure in XML.

3. **SOAP:** SOAP is an XML-based messaging protocol for exchanging information between computers. You can say that SOAP is an application of XML.

The following are the advantages of SOAP:

- SOAP is a communication protocol designed to communicate over the internet.
 - SOAP can extend HTTP requests.
 - SOAP can be used for broadcasting a message.
 - SOAP is platform-independent.
 - SOAP is language-independent.
 - SOAP is the XML way of defining what information is sent and how.
 - SOAP enables client applications to easily connect to remote services and invoke remote methods. SOAP can also be used in a variety of messaging systems.
4. **JSON:** JSON is a method of lightweight data-interchange. It is self-describing and easy to understand. The most important part of JSON is that it is language independent.

When exchanging data between a browser and a server, the data can only be text. JSON is text-based. We can convert any JavaScript object into JSON, and send JSON to the server. Not only that; most languages have their methods for converting their objects into JSON and vice-versa.

We can also convert any JSON received from the server into JavaScript objects. In this way, we can work with the data as JavaScript objects, without any parsing.

5. **REST:** REST relies on a stateless, client-server, cacheable communication protocol. It is *an architectural style* for designing networked applications. The idea is that, rather than using complex techniques such as SOAP to connect between computers, a simple HTTP is used to make calls between machines.

The World Wide Web itself is based on HTTP and can be viewed as a REST-based architecture. RESTful applications use HTTP requests to post, read, and delete data. REST is lightweight. It is simple and fully-featured. That said, there is basically nothing you can do in web services that cannot be done with a REST architecture.

Excel and Word plugins

The most important plugins are Microsoft office plugins. In this section, we will cover Excel and Word plugins.

In most projects, one of these two plugins is used.

Excel plugin

Excel is an application program developed by Microsoft. It is a part of the Microsoft Office suite. Excel is capable of creating and manipulating files that are saved with .xls or .xlsx extensions. General uses of Excel include (an individual cell) cell-based calculation. For example, with an Excel spreadsheet, you can create a table, use formulae to compute every row and column, make your own monthly expense list, and so on.

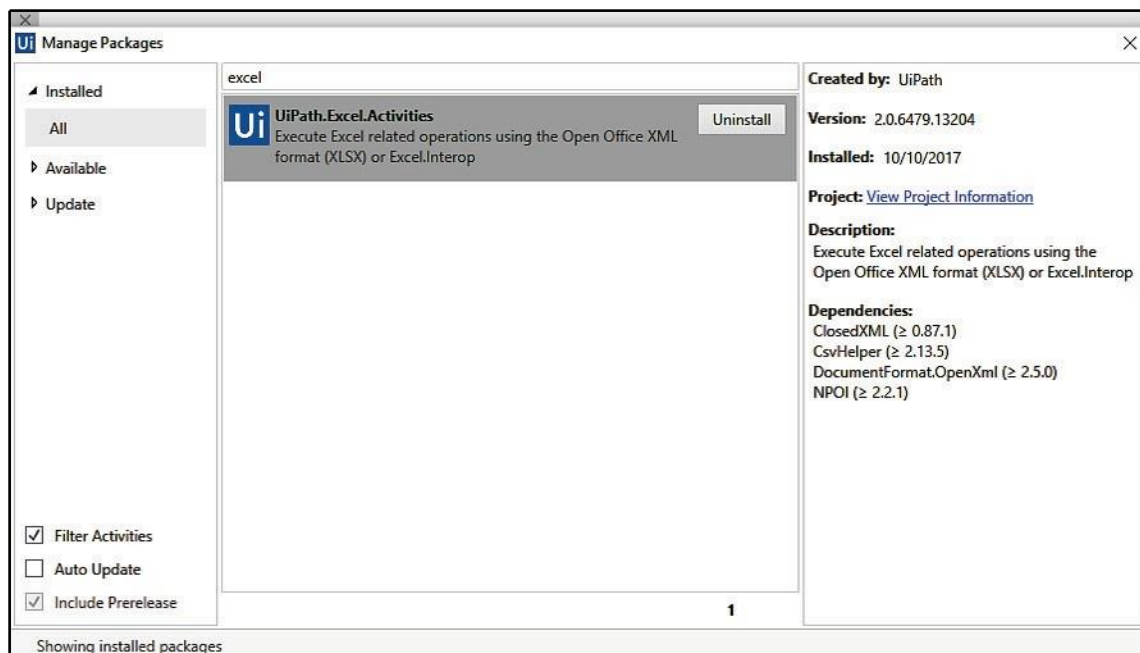
Unlike a word processor such as Microsoft Word, Excel documents consist of rows and columns. Each column consists of a cell in which we can store a value. The value can either be text, string, or number.

In UiPath Studio, there is a NuGet Package called `UiPath.Excel.Activities`. The Excel activity is pre-installed in UiPath Studio. In case it is not installed, it can be installed manually.

To check if the Excel activity is installed or not, simply search for `Excel` activities on the **Activities** panel. It will list all Excel activities. If Excel activities are not listed in the **Activities** panel, then you need to install the **UiPath.Excel.Activities** package.

To install the Excel NuGet package, click on the **Manage Package** icon located at the top of the **Activities** panel.

A **Manage Packages** window will appear. Search for `Excel` in the search bar. As shown in the following screenshot, there is an **Uninstall** button next to **UiPath.Excel.Activities**. This is because Excel activities are already installed in UiPath Studio. If it is not installed, an **Install** button will appear next to **UiPath.Excel.Activities**:



Word plugin

Microsoft Word is often called simply Word or MS Word. Microsoft Word is a widely used commercial word processor designed by Microsoft. Microsoft Word is a component of the Microsoft Office suite.

Microsoft Word is available for both Windows and Mac operating systems.

The following are the Microsoft Word features:

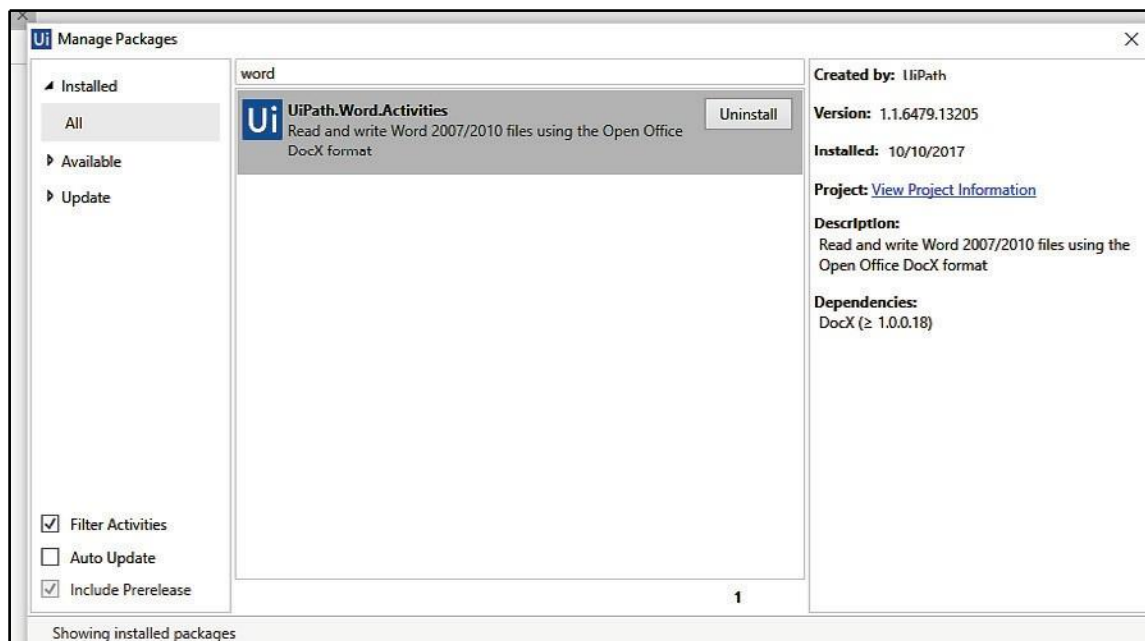
1. It makes it possible for everything displayed on the screen to appear in the same way when printed or moved to another program.
2. Microsoft Word has a built-in dictionary for spell checking.
3. Text-level features such as bold, underline, italic, and strike.

4. Page-level features such as paragraphing and justification.
5. Microsoft Word is compatible with many other programs, the most common being the other members of the Office suite.

In UiPath Studio, there is a NuGet package called **UiPath.Word.Activities**. Word activity is pre-installed in UiPath Studio. In case it is not installed, we have to install it manually.

To check whether the Word activity is installed or not, simply search for `Word` in the **Activities** panel and press **Enter**. It will list all the Word activities. If Word activities are not listed in the **Activities** panel then you have to install the **UiPath.Word.Activities** package.

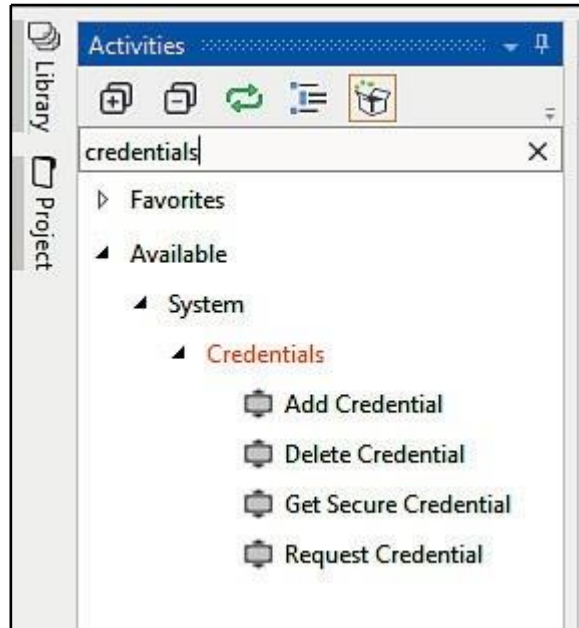
To install the Word NuGet package, click on the **Manage Package** icon. The **Manage Packages** window will appear. Search for `Word` in the search bar. As shown in the following screenshot, there is an **Uninstall** button next to **UiPath.Word.Activities**. This is because Word activities are already installed in UiPath Studio. If it is not installed, there will be an **Install** button next to **UiPath.Word.Activities**:



Credential management

In the Windows operating system, you can view your network login credentials, that is, the username and password, by using Credential Manager.

With the help of UiPath Studio, you can automate the process of creating, manipulating, and deleting credentials using some credential activities:



The following are credential activities:

1. **Add Credential activity:** You can add a credential activity for use in your project. You just simply have to specify the username and the password.
2. **Get Credential activity:** This activity is used to store the username and password for future use. It enables us to further check whether credentials are valid or not.
3. **Request Credential activity:** This activity is used to display a dialog box to a user asking them about their credentials. It then stores the username and password in string variables that can be further used to log in to the applications.
4. **Delete credential activity:** This activity is used to delete the credentials that we have stored.

Extensions – Java, Chrome, Firefox, and Silverlight

Extensions are small software programs that can be used to modify and extend the functionality of any browser. You can build your own extension by using web technologies such as HTML, JavaScript, and CSS. Extensions have little or no user interface.

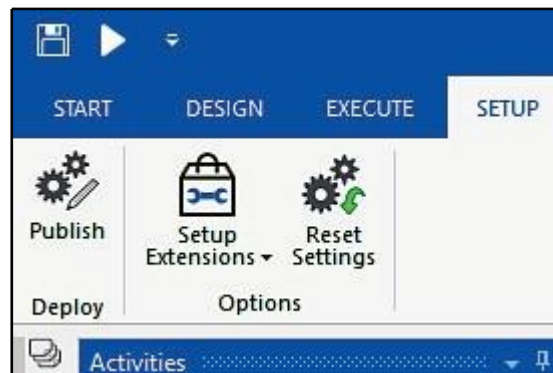
Java extensions are useful when you have to automate a Java application. Without this extension, UiPath Studio does not correctly identify the UI element of a Java application.

Chrome and Firefox extensions are used when you are working with the Chrome/Firefox browsers. By this, we mean that while automating with UiPath Studio, if you are interacting with a browser, you first have to install the extension for that browser.

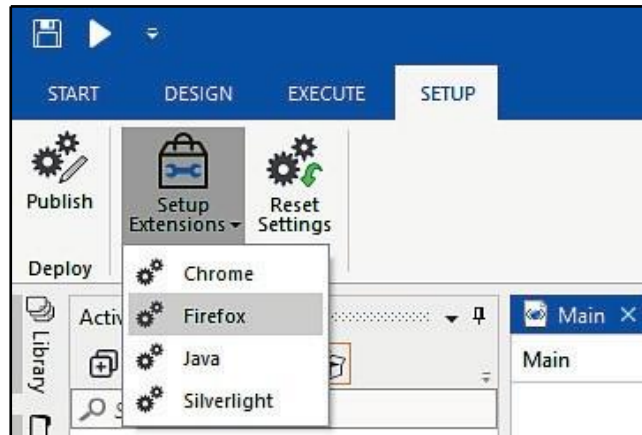
Similarly, the Silverlight extension has to be installed if you want to integrate your application using Microsoft's Silverlight.

Almost all extensions have a similar installation method. Once you get familiar with the method, you can install the remaining of the extensions on your own.

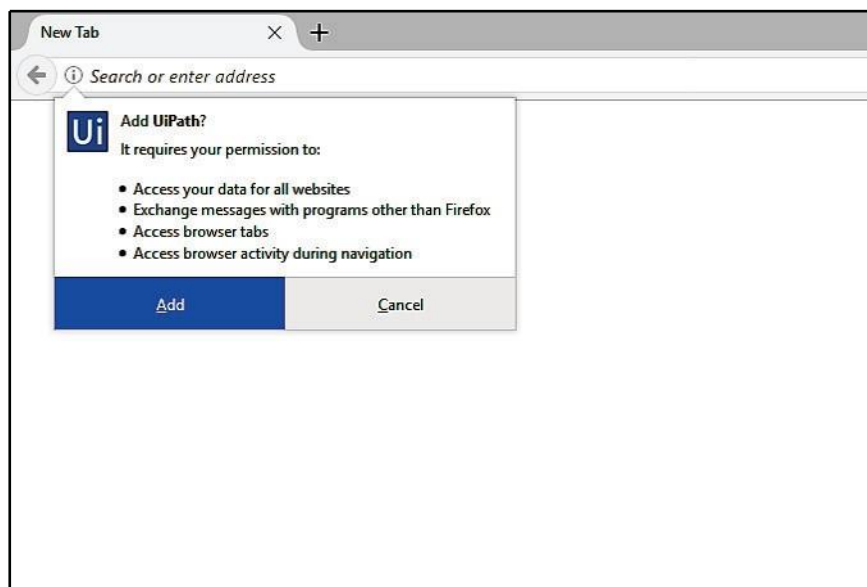
In UiPath Studio, you can find all the extensions in the **Setup Extensions** menu. Click on the **SETUP** tab at the top of the UiPath Studio window:



Now click on the **Set Up Extensions** icon and select the extension that you want to select:



In this session, we are going to select the Firefox extension from the drop-down list. Automatically, the Firefox browser is opened and asks you to add UiPath. Just click on the **Add** button:



Your extension has been successfully installed. A dialog box will pop up confirming this.

If you want to install other extensions, you just have to select the desired extension from the drop-down list.

Summary

In this chapter, you have learned about the role of plugins and how they can increase the scope for automation. Over time, many new plugins will be introduced, the inclusion of those plugins and extensions will be on a similar line; however, the internal workings of those plugins may vary. You have also learned that the Terminal plugin is different from the PDF plugin. In this chapter, Java, Chrome, Firefox, and Silverlight extensions were also covered. You also learned about Mail, Web, and SAP integrations. Credential management was a nice addition to this chapter.

In the next chapter, we will focus on assistant bots and event triggers.

7 Handling User Events and Assistant Bots

In UiPath, there are two types of Robot that are used for automating any process. One is the back office Robot, which works in the background. It works independently, which means it does not require inputs from users or any user interaction. The other one is the **Front Office Robot**, which is also known as an **Assistant Robot**.

This chapter deals with front office bots. Here, we will learn the different ways in which events in the automation process can be triggered—by a simple press of a key, click of the mouse, and so on. To make things clearer, we will take examples of monitoring various events.

We will cover the following topics in this chapter:

- What are assistant bots?
- Monitoring system event triggers
- Monitoring image and element triggers
- Launching assistant bots on a keyboard event

What are assistant bots?

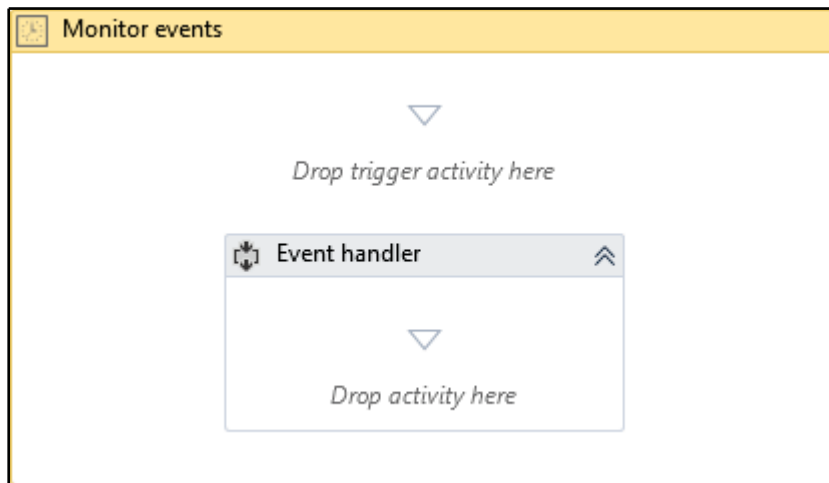
Assistant Robots are front office Robots that require some user interaction. In this case, the automation will run only when a certain event or user action is triggered.

Trigger events are basically commands to tell the Robot to start its automation process.

For example, say I want some text to be typed into the Notepad application. In particular, I want the Robot to type into the Notepad once I click on the text area (clicking being the trigger activity in this case) in the Notepad application.

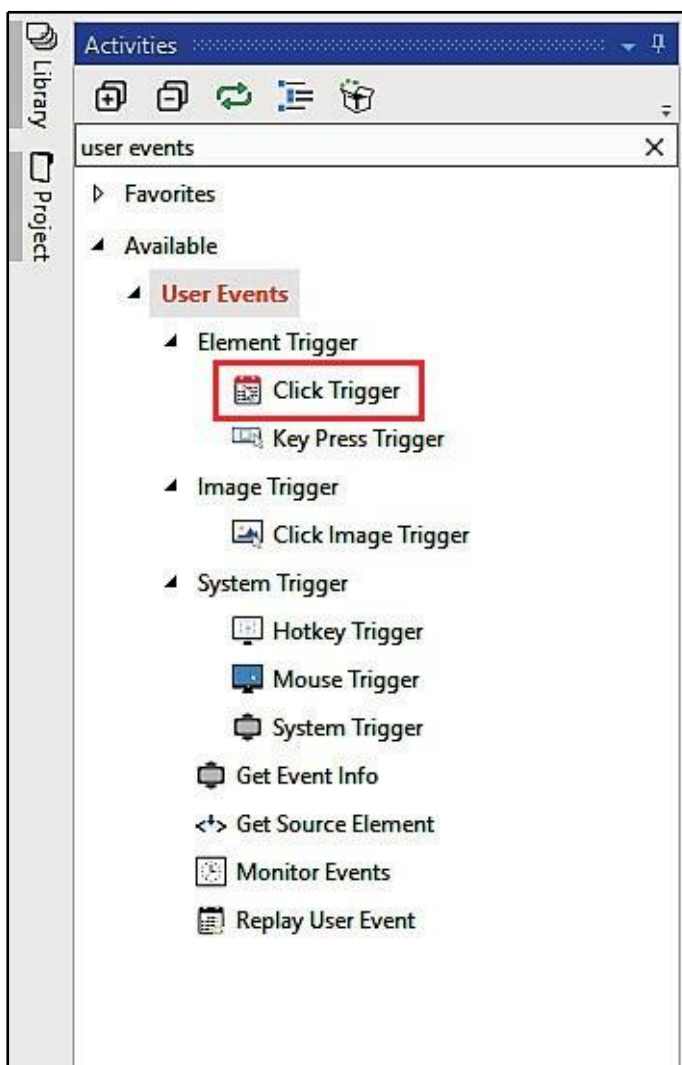
Let us look at the following steps to understand more:

1. **Drag and drop the Monitor events activity:** Here, we drag and drop a **Monitor events** activity from the **Activities** panel inside which the trigger events will work; otherwise it will show you an error. The **Monitor events** activity looks like this:

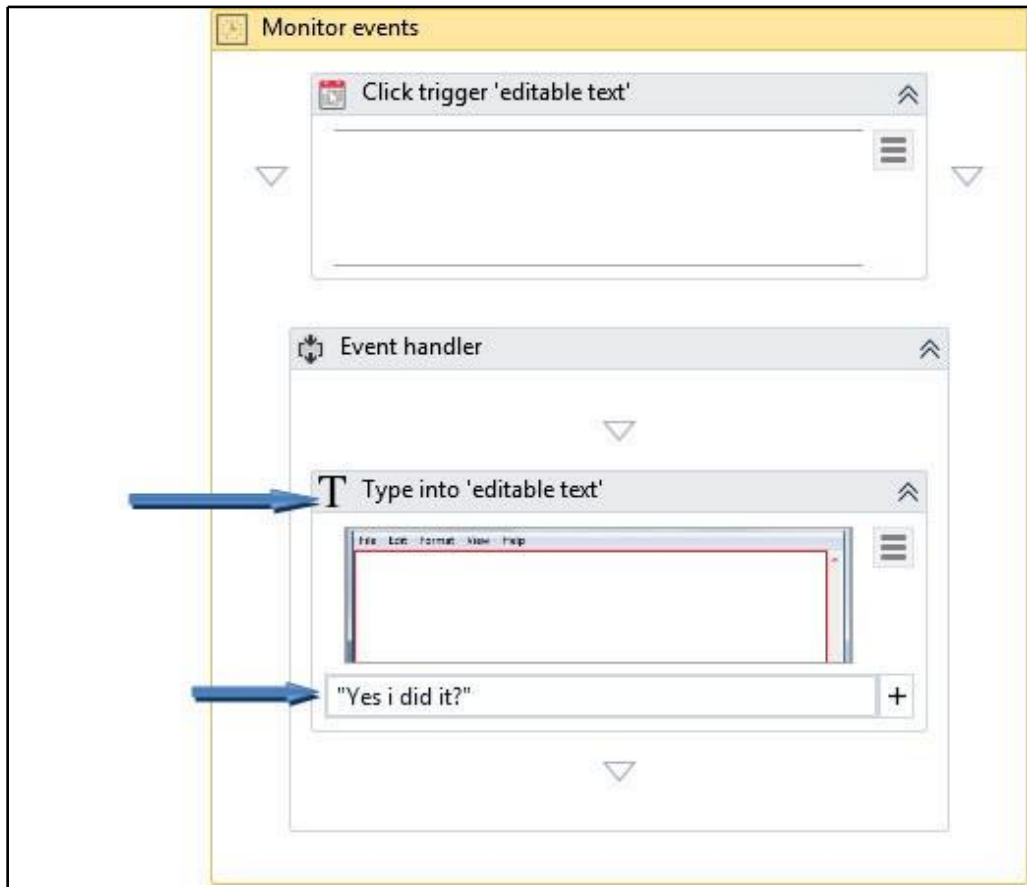


2. **Drag and drop the trigger activity of choice:** In the drop trigger area, drag and drop the trigger activity that you want.

There are a lot of trigger activities shown in the **Activities** panel. In this case, we will choose the **Click Trigger** activity:



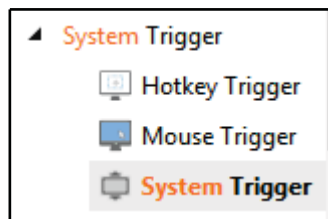
3. **Create workflow inside the Monitor events activity:** Now inside the **Event handler** space in the **Monitor events** activity, we have to create the workflow or the set of tasks we are required to do once the trigger activity works. In this case, we are using the **Type into** activity. Indicate the blank area of a Notepad window:



This was an overview of how assistant bots work.

Monitoring system event triggers

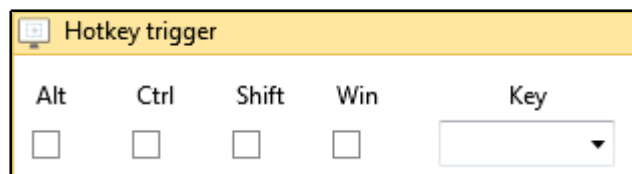
There are three system trigger events—**Hotkey Trigger**, **Mouse Trigger**, and **System Trigger**:



Though all three triggers are used for triggering activities, they are used differently as explained in the following section.

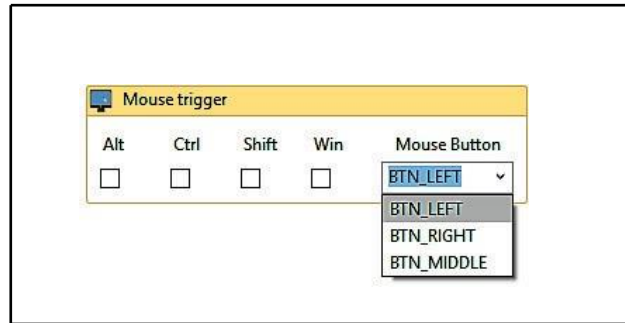
Hotkey trigger

Hotkey trigger works for shortcut keys. Suppose we want a certain workflow to work once the user presses the *Alt + F4* keys or any other shortcut key. In such a case, we will use the **Hotkey trigger**:



Mouse trigger

This is used when we want to trigger events on performing a mouse action (left-click, right-click, or middle-click) as shown in the following screenshot:



As shown in the screenshot, we can select the type of click with which we want to trigger events. We can also use other special keys with mouse actions as shown.

System trigger

This is the last type of system trigger activity. A system trigger is used to trigger events on mouse actions, keyboard actions, or both, all of which we can select from the **Properties** panel. We can also select the action to be performed, that is, forwarding the event or blocking the event as shown in the following screenshot:

The screenshot shows the 'Properties' window for a 'SystemTrigger' activity in UiPath. The window is titled 'Properties' and has a search icon. Below the title bar, the activity name 'UiPath.Core.Activities.SystemTrigger' is displayed. The properties are organized into sections: 'Common', 'Input', 'Misc', and 'Options'. The 'Common' section contains the 'DisplayName' property, which is set to 'System trigger'. The 'Input' section contains the 'TriggerInput' property, which is set to 'Keyboard, Mouse'. The 'Misc' section contains the 'Private' property, which is an unchecked checkbox. The 'Options' section contains the 'EventMode' property, which is set to 'EVENT_FORWARD'.

Properties	
UiPath.Core.Activities.SystemTrigger	
Common	
DisplayName	System trigger
Input	
TriggerInput	Keyboard, Mouse
Misc	
Private	☐
Options	
EventMode	EVENT_FORWARD

Monitoring image and element triggers

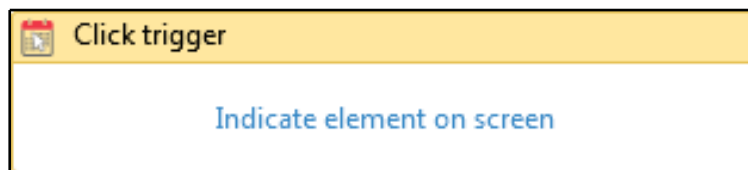
With an image trigger, the events will occur once the user has clicked on a certain image that is indicated in the **Click Image Trigger** activity.

By clicking on **Indicate element on screen**, we have to select an image that will trigger the event when clicked.

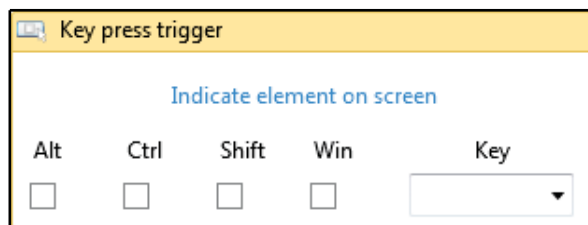
In the **Element Trigger**, there are two activities that come into play. These are **Click Trigger** and **Key Press Trigger** as shown in the following screenshot:



- The **Click trigger** activity is used to trigger events when a user simply clicks on a UI element:



- The **Key press trigger** activity is used when we need to trigger events by pressing a certain key or by selecting the image on the screen to trigger events:

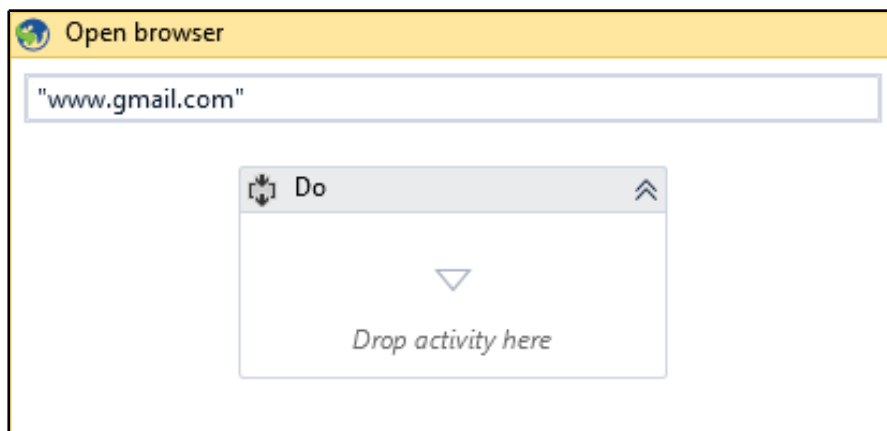


The image shows a configuration window titled "Key press trigger". Inside the window, there is a blue text prompt "Indicate element on screen". Below this prompt, there are five checkboxes labeled "Alt", "Ctrl", "Shift", "Win", and "Key". Each checkbox is currently unchecked. To the right of the "Key" checkbox is a dropdown menu with a downward arrow.

An example of monitoring email

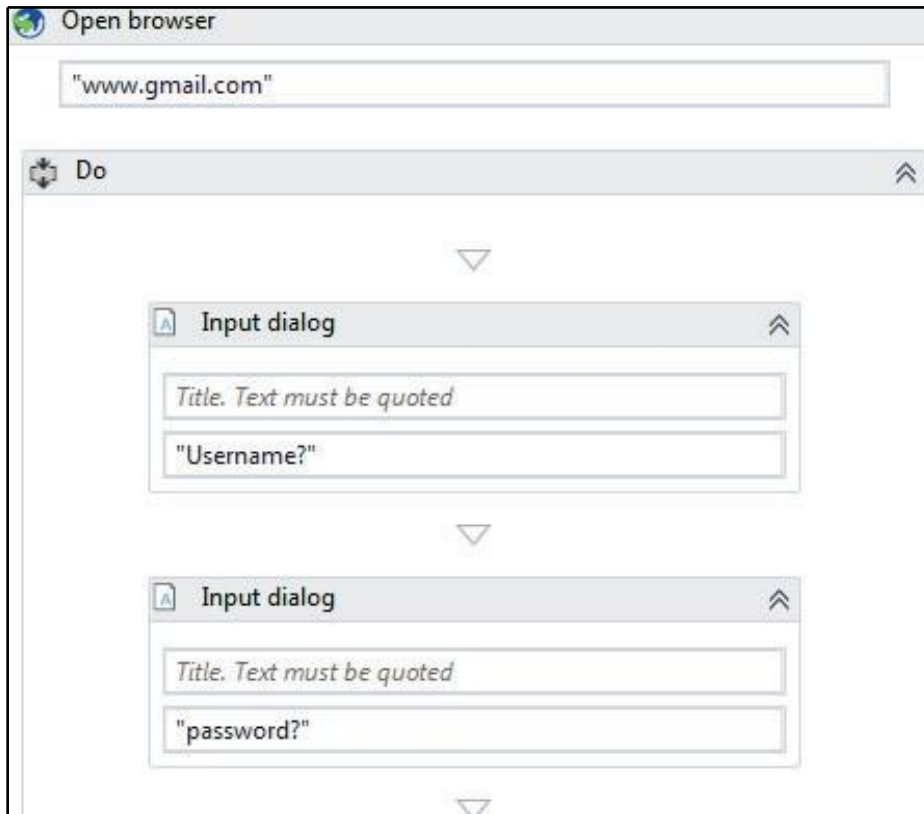
To make things clearer, we will monitor a send email event through Gmail. The steps are listed as follows:

1. **Open the browser and browse to `www.gmail.com`:** To do this, drag and drop the **Open browser** activity. In the required field for the address, enter `www.gmail.com`:

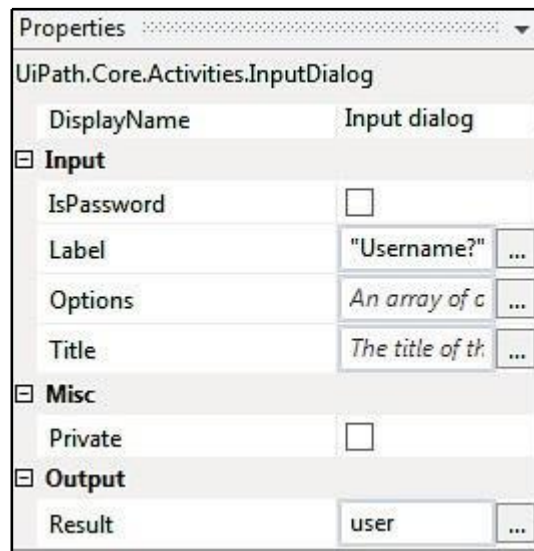


The image shows a configuration window titled "Open browser". At the top, there is a text input field containing the address "www.gmail.com". Below this field, there is a smaller window titled "Do" with a downward arrow and the text "Drop activity here", indicating where to drop another activity.

2. **Getting username and password:** After typing in the address, we have to ask the user for a username and password. For this, we will use the **Input dialog** activity as shown in the following screenshot. We have dragged and dropped two **Input dialog** activities to ask the user for a username and password respectively. Until the user types in each dialog and presses okay, the Robot will not work:

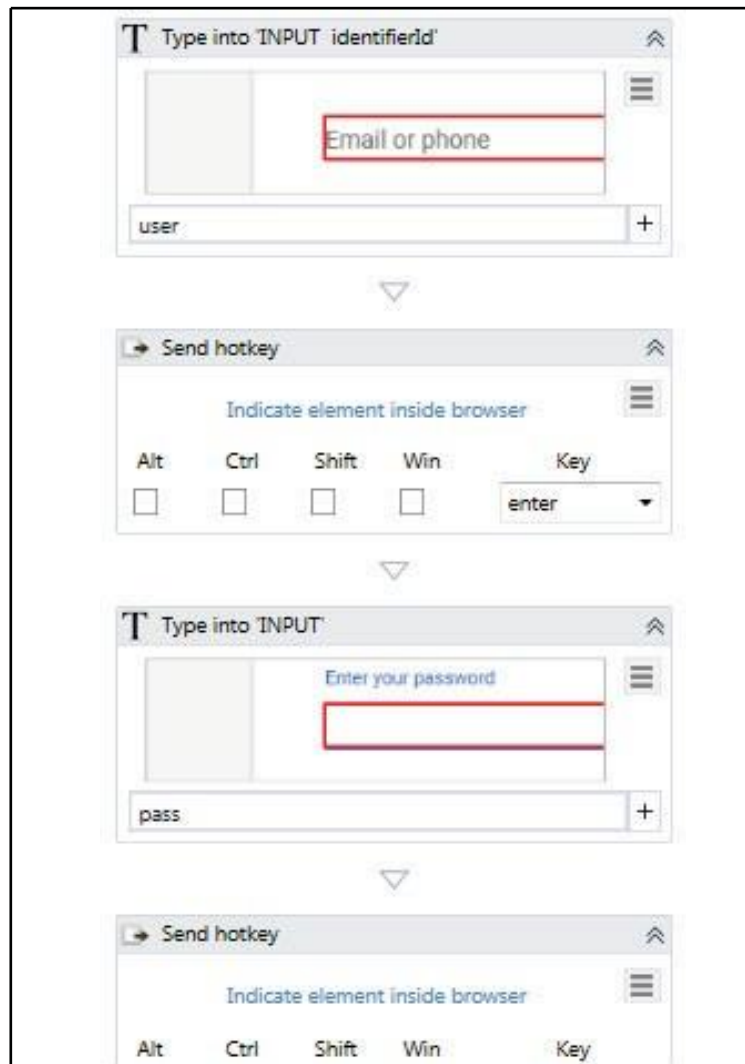


Once the user types in the **username** and **password**, we save these details into two variables: `user` and `pass`. You can convert their values into a variable by going to the **Input dialog** property in the **Properties** panel. Just right-click on the empty text box of the **Result** property and choose **Create Variable**. We have named it **user** as shown in the following screenshot:

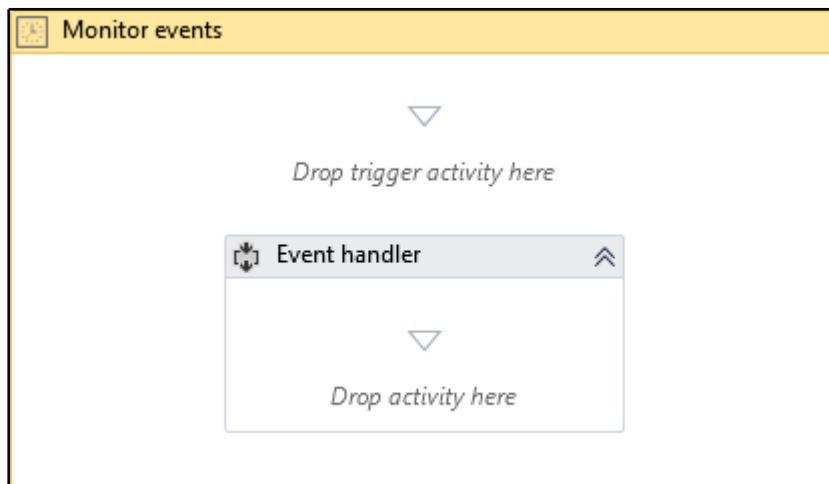


3. **Entering a username and password:** We shall use the **Type into** activity to enter a username and password by indicating the respective fields for typing in the username and password.

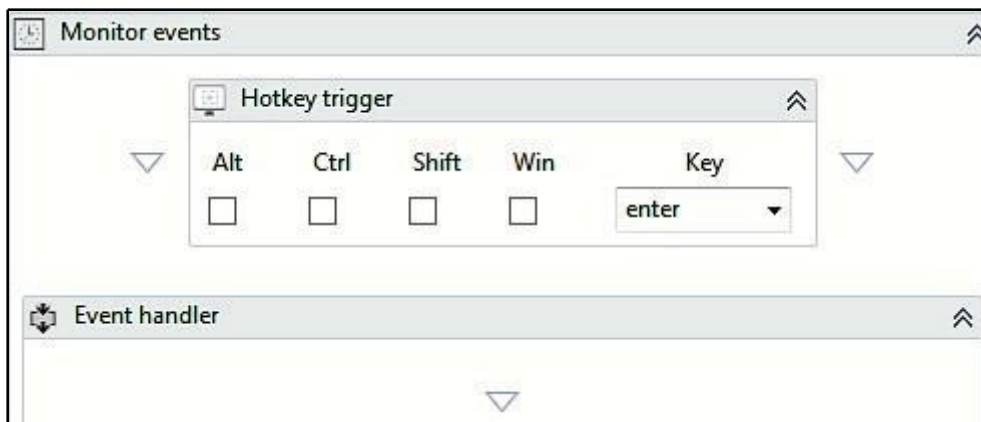
Once the user enters the username and password, he needs to login which he can either do by clicking on the login button or by pressing the *Enter* key on the keyboard. We will use the **Send hotkey** activity to send the *Enter* key (as shown in the following screenshot). By doing so, the login button is clicked:



4. **Trigger the send email event with a Hotkey trigger:** Our next step is to trigger the send mail event. Here, pressing the *Enter* key will be the trigger. On pressing it, the Robot performs the rest of the send email task. For this, we will use the Hotkey trigger activity. We first have to drag and drop the **Monitor events** activity as trigger activities only work under it:



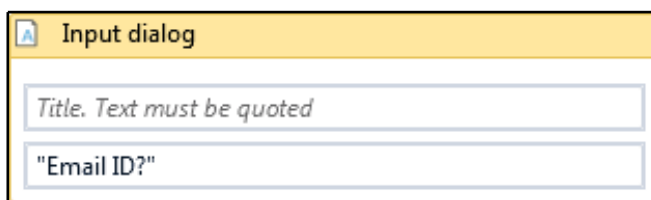
Since we are using the **Hotkey trigger**, we have dropped the **Hotkey trigger** activity in that area:



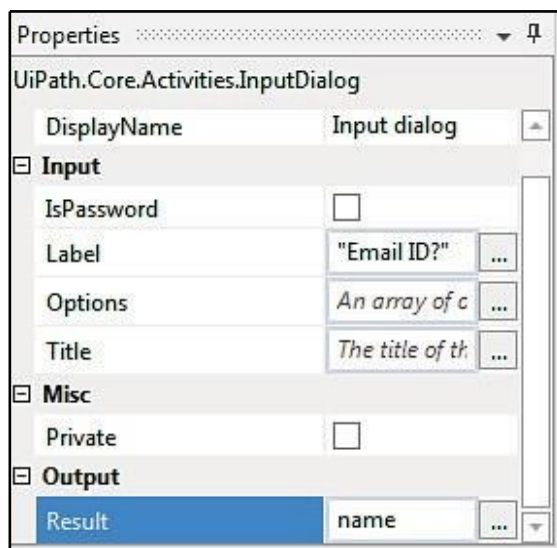
In the area for the **Event handler**, we need to give the sequence of steps for sending the mail, which will involve several steps. For this, we have created a workflow showing all the steps to be followed to send an email. This ranges from clicking on Compose mail to clicking on the Send button as explained in the following steps.

5. **Ask the user for the email ID of the recipient, the subject of the email, and its body:**
Our next step is to ask the user for details. We will use three Input dialogs, one for the email ID, one for the subject, and one for the content.

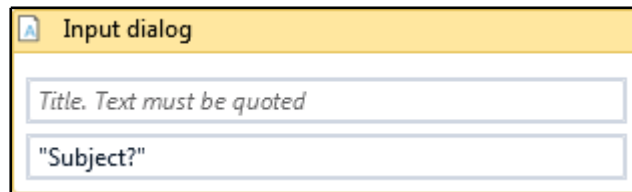
As shown in the screenshot, we have used an Input dialog to obtain the recipient's email ID:



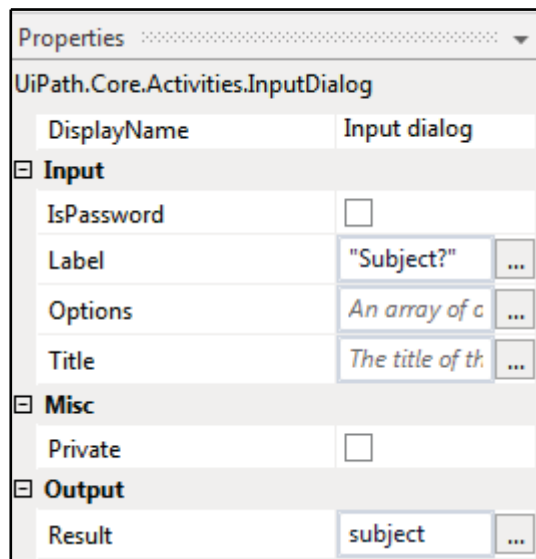
We now save the user input email ID inside a variable called `name` (you can easily create a variable by pressing `Ctrl + K` inside the **Output** box in **Properties**):



In the second **Input dialog**, we will ask the user to input the subject for the email:

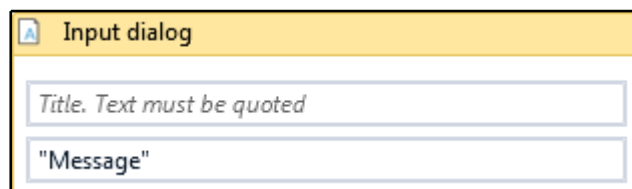


The output, that is, the response entered by the user, is saved as a new variable called `Subject?` as shown in the following screenshot:



Properties	
UiPath.Core.Activities.InputDialog	
DisplayName	Input dialog
Input	
IsPassword	☐
Label	"Subject?"
Options	An array of c
Title	The title of th
Misc	
Private	☐
Output	
Result	subject

In the third input dialog, the user has to input the message/mail he or she wants to send:



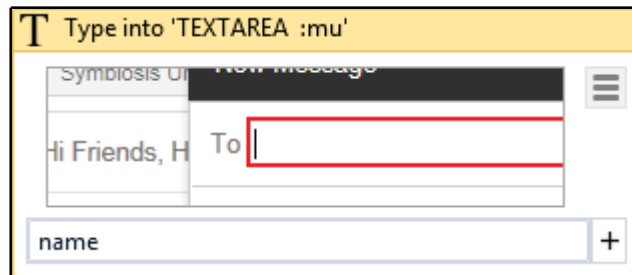
We shall store the user output as a variable called `message`:

UiPath.Core.Activities.InputDialog	
DisplayName	Input dialog
Input	
IsPassword	☐
Label	"Message" ...
Options	An array of c ...
Title	The title of th ...
Misc	
Private	☐
Output	
Result	message ...

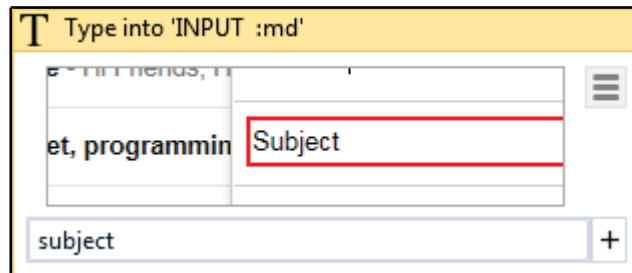
6. **Type in the details:** Now that we have all the details that are required for sending the mail, our next step will be to type into the required fields for sending the email. We will use the **Type Into** activity for this step:



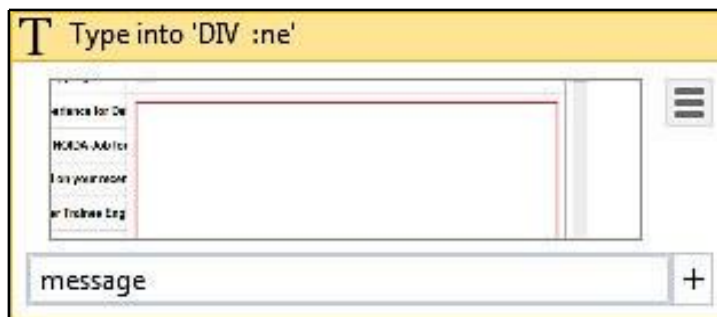
Drag and drop the **Type into** activity. Then, double-click on it and indicate the area where you want to type the email ID. Since we have saved the email ID as a variable, `name`, we enter this in the field provided, as shown in the following screenshot:



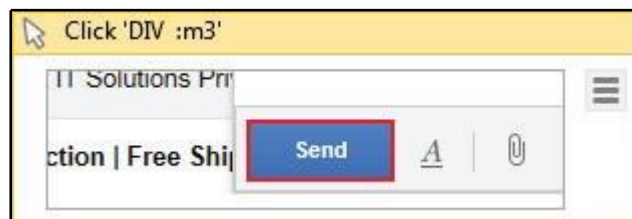
Our next requirement will be to indicate the area where we want to type the subject of the mail. Since we have saved the subject as a variable, `Subject`, we enter this in the field provided as shown in the following screenshot:



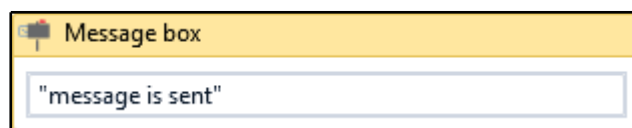
Now you are required to indicate the area where you want to type the message/mail as indicated in the screenshot. Since we have saved the content of the mail to be sent as a variable, `message`, we enter this in the field provided as shown in the following screenshot:



7. **Click on Send and confirm if successfully sent:** Our final step is to click on the **Send** button so that the mail is sent and the process is completed. In order to click on the **Send** button, we will use the **Click** activity and indicate the **Send** button. Doing so enables the Robot to easily recognize where to click:



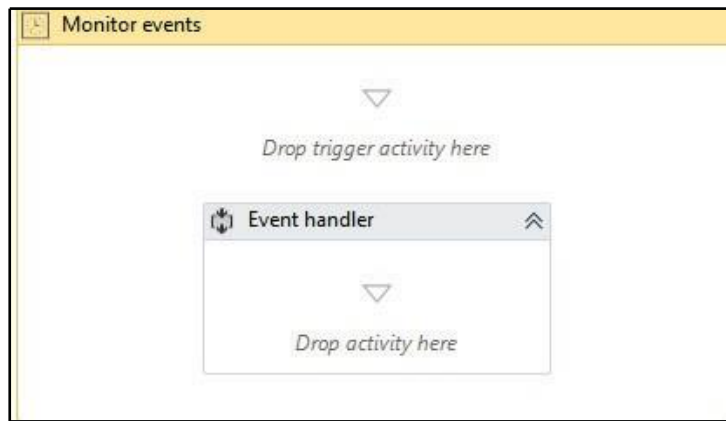
If you want, the Robot can also give a notification once the mail is sent. For this notification we will use the **Message box** activity, which will display the message, **message is sent**, as shown in the following screenshot. When the message is displayed, and after the user has pressed **OK**, the whole workflow will terminate since all of the steps have been executed:



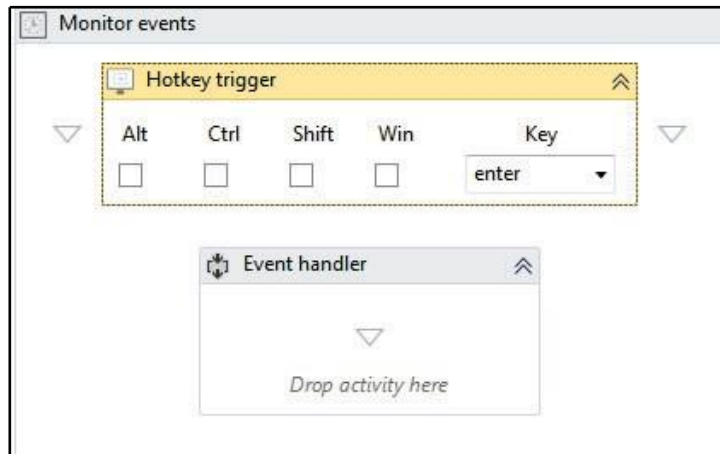
Example of monitoring a copying event and blocking it

Let us take an example of monitoring a copying event and blocking it. In this example, we have an Excel file from which we want the data to be copied as soon as the user presses the *Enter* key:

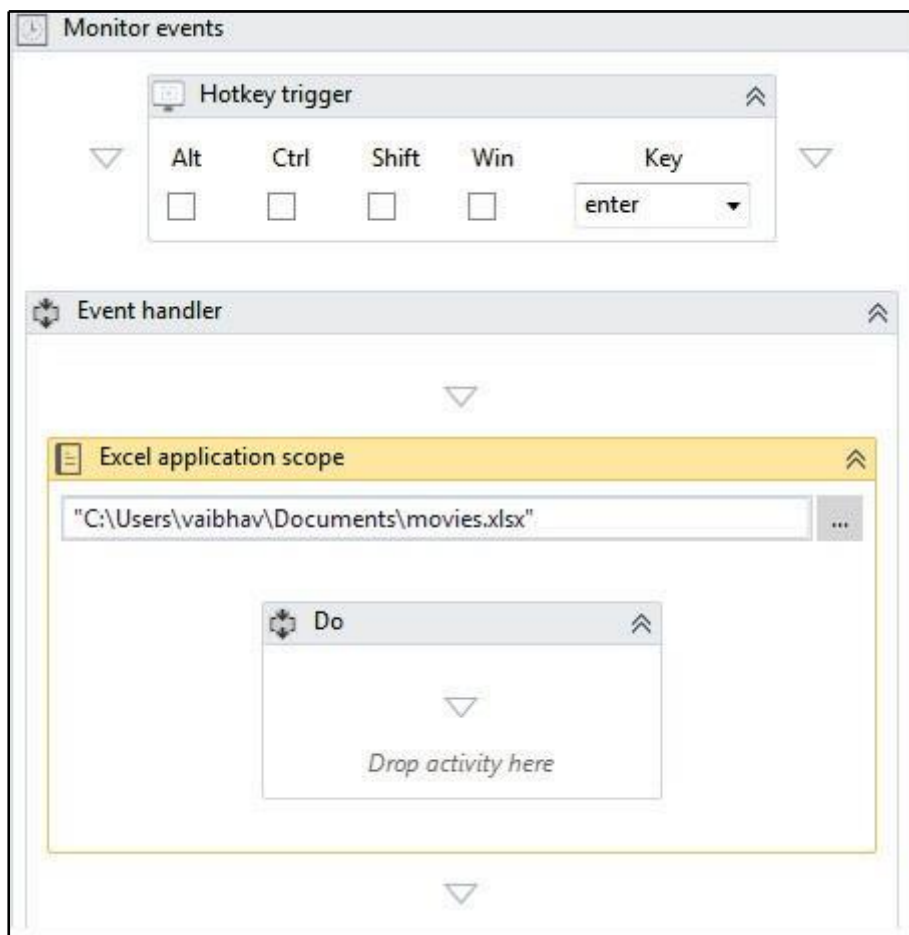
1. **Drag and drop the Monitor events activity and the drop trigger activity into it:** Drag and drop the **Monitor events** activity. Double-click on it:



Drag and drop the **Hotkey trigger** activity and select the *Enter* key from the drop-down list, as shown in the screenshot:

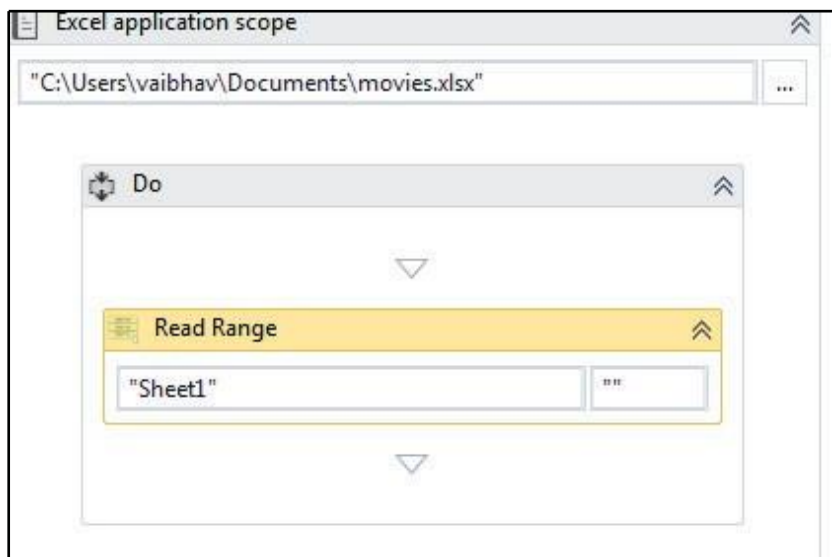


2. **Drag and drop an Excel application scope inside the Event handler portion:** We are required to drop an activity under **Event Handler**. In our case, the activity is copying data from Excel and pasting it. When we drag and drop the **Excel application scope** activity inside the **Event handler** and double-click on it, we see that first we have to browse to the Excel file from which we want to copy the information:

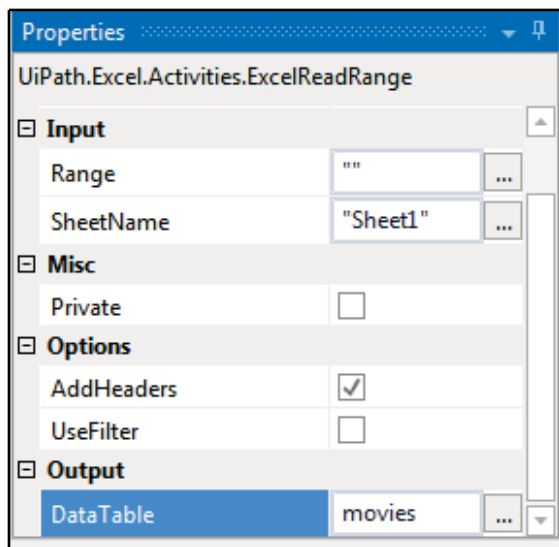


As shown in the screenshot, we have selected an Excel file whose name is `movies`; now we want to copy this file's content.

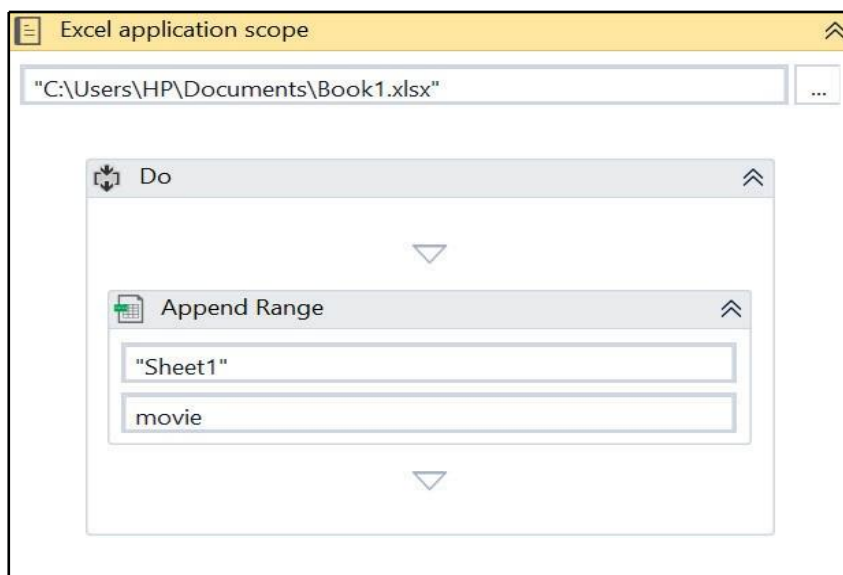
3. **Use the Read Range activity, extract the data and paste it into a new Excel file:** Now, inside the **Do** activity, drag and drop the **Read Range** activity to read all the data from this Excel file. We will keep this extracted data in a variable named `movies`, as shown in the screenshot:



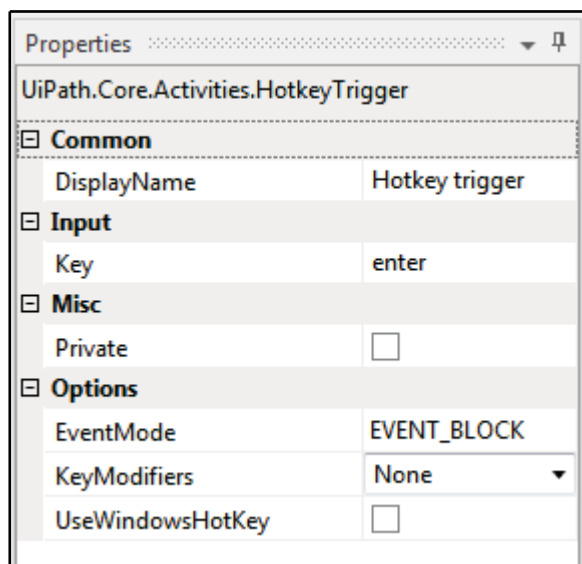
We have read the data from the Excel file. Next, we want to keep it in a variable. For this, just click on the **Read Range** activity and go to the **Properties** panel. Then create a variable by pressing *Ctrl* + *K* and name it `movies`:



4. **Append to another Excel file:** Now, since we have all the data saved, we can just drag and drop another Excel application scope. Then we will indicate the file that we want to append this data to. In the **Do** activity, just drag and drop the **Append Range** activity. Select the input as the variable we declared earlier, that is, `movies` as shown in the screenshot:



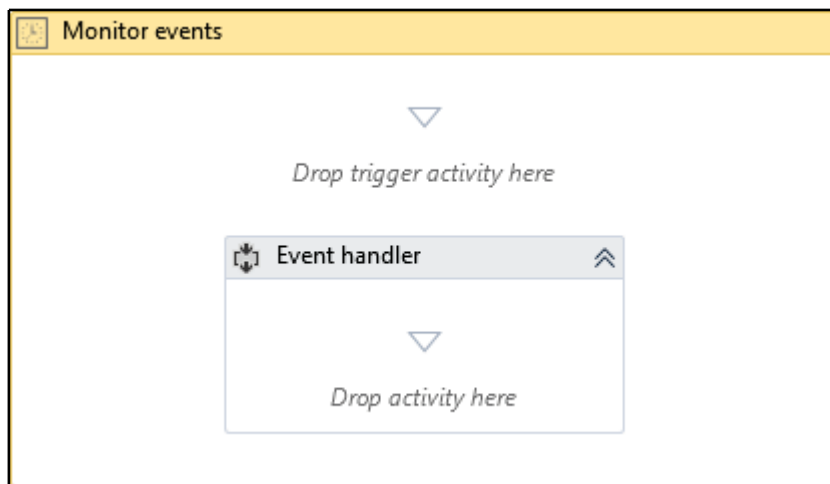
5. **Block the triggered event:** Now, in order to block triggered events you can select the `EVENT_BLOCK` event as the event type from the properties of the trigger in the **Properties** panel as shown in the following screenshot:



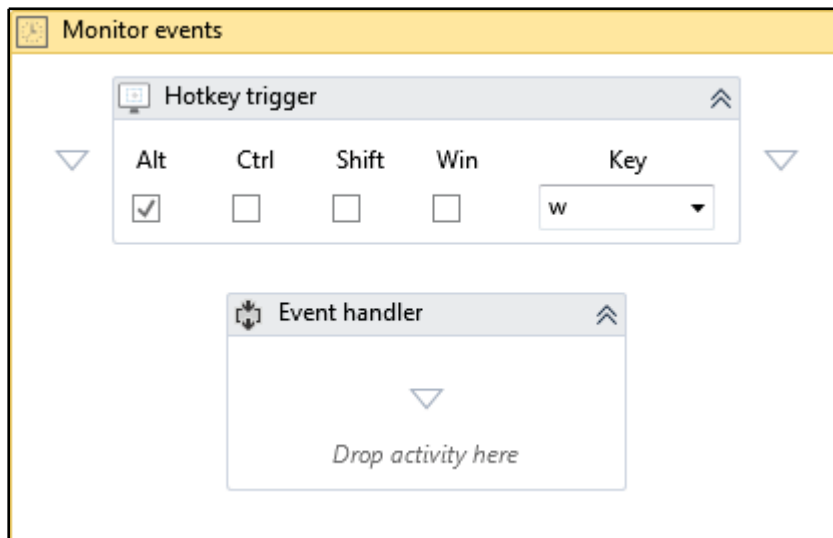
Launching an assistant bot on a keyboard event

Let us say we want our assistant bot to start automating only when we trigger an event. For example, the user wants his Robot to open and start typing in the Notepad window when he presses *Alt + W*. This can be achieved using the Hotkey trigger. Also, inside the Event handler, just create or record the sequence of steps to be followed. The detailed procedure has been explained in the following sections:

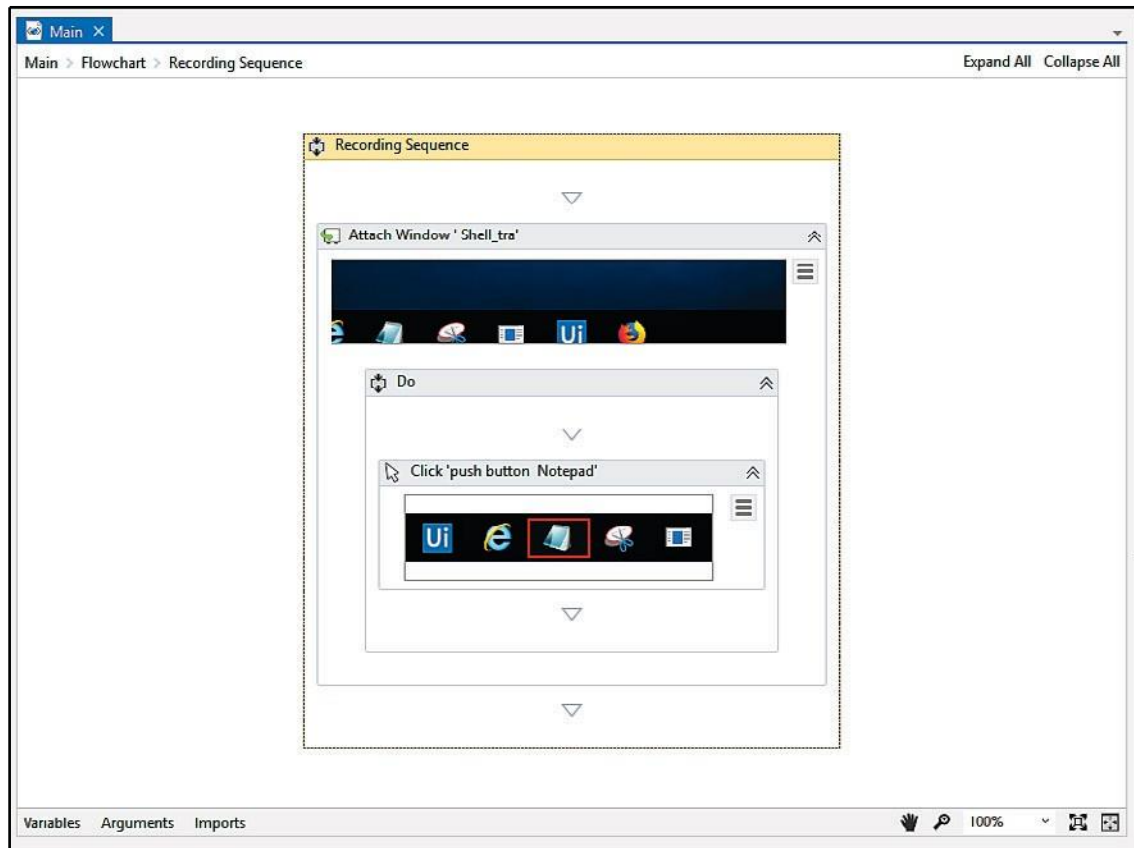
1. **Drag and drop the Monitor events activity:** In this step, we will just drag and drop the **Monitor events** activity into the workflow. When we double-click on it, it will look like this:



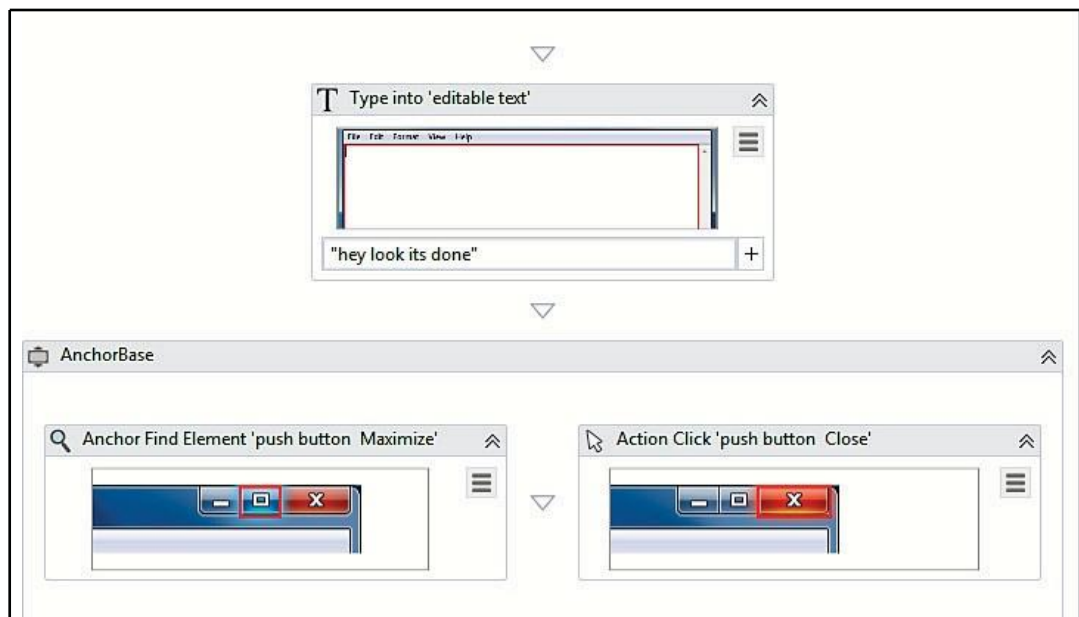
2. **Drag the Hotkey trigger activity:** In the next step, we will use the **Hotkey trigger** activity for the user to start the automation process. Assign *Alt + W* to the hotkey so that, when the user presses this hotkey, the event will be executed:



3. **Open Notepad and type into it:** Our final step is to record the sequence of the steps to be performed. In this case, this is to open Notepad and then type into it. For that just use the help of the **Desktop** recorder. First, we double-click on the Notepad application in the window as shown in the screenshot. Select the **ClickType** as **CLICK_DOUBLE** from the **Properties** panel:



After that, we record the typing action and close the Notepad window. Then click on **Do not Save** because you do not want to save your file. The sequence is shown in the following screenshot:



We have also indicated the anchor to recognize the correct button to be clicked (in this case, the close window button's anchor is the maximize button). This makes it easier for the Robot to find the UI element.

Now, on pressing *Alt + W* the Robot will start executing the sequence.

Summary

In this chapter, we learnt about the assistant bot's utility. We also covered all the monitoring events that can be used to trigger actions and also saw examples of them. Once your automation program is made, there may still be problems that you are likely to face while executing it. To handle such scenarios, we will learn about Exception Handling in the following chapter.

8

Exception Handling, Debugging, and Logging

Sometimes, automation program may fail to execute. To deal with such cases, we use exception handling activities. In this chapter, we start with the various types of exception handling method available in UiPath, the exceptions that you may encounter, and how to deal with them. We will also learn about logging. An important topic covered in this chapter is debugging to check whether workflows are working properly or not and to rectify any errors.

Before we begin, let us look at the topics that will be covered in this chapter:

- Exception handling
- Common exceptions and ways to handle them
- Logging and taking screenshots
- Debugging techniques
- Collecting crash dumps
- Error reporting

Exception handling

Exception handling is a way to handle exceptions for a process that the program or the procedure has failed to execute.

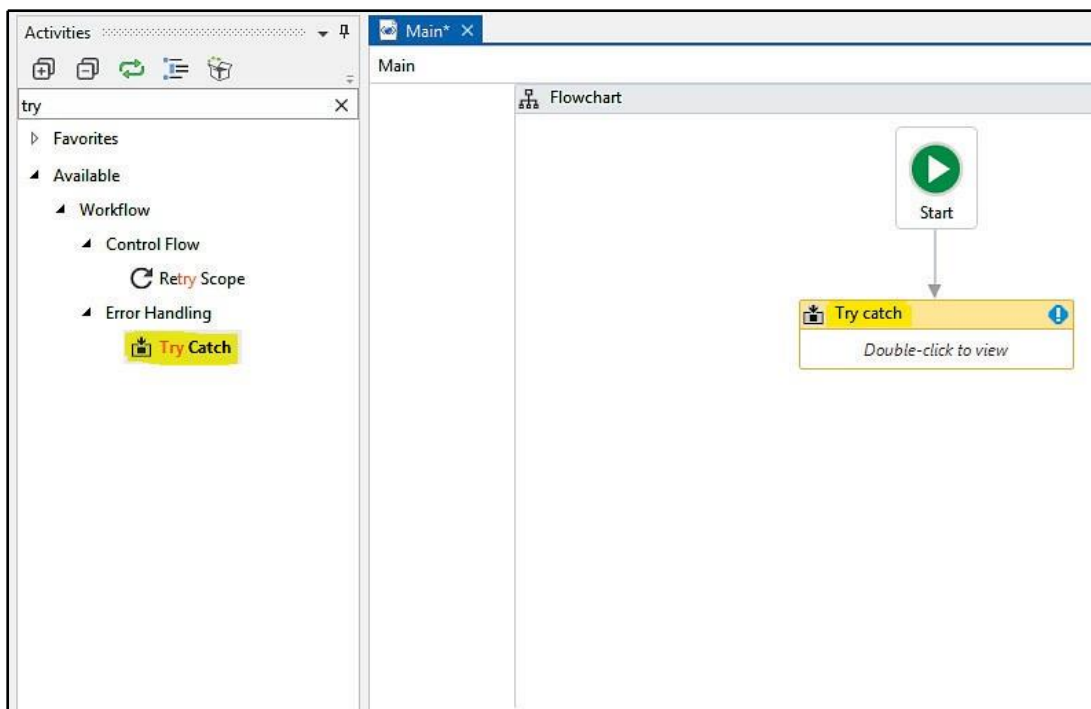
For handling exceptions in a program, the best practice considered is to use the **Try catch** activity.

The **Try catch** activity can be found in the Activities panel. By dragging and dropping the **Try catch** activity into the workspace, we can handle exceptions. For handling errors in the **Try catch** block, we can divide the whole process into four parts just to make it simpler:

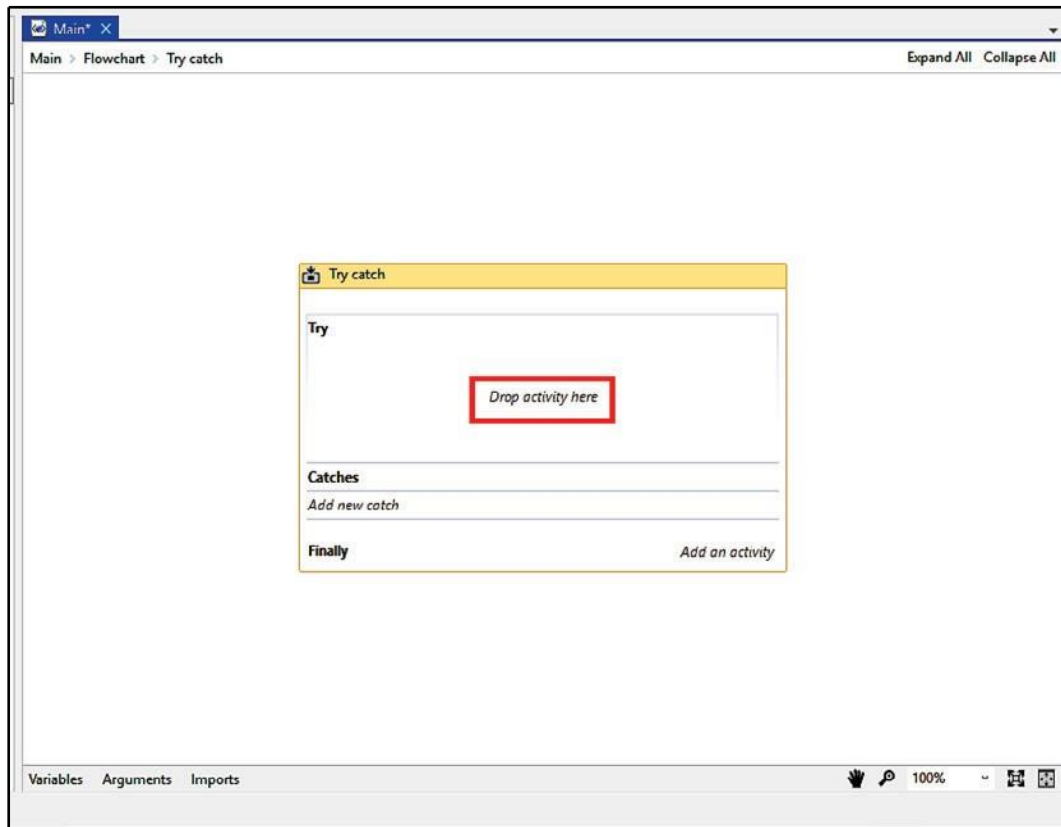
- Drag and drop the **Try catch** activity Try block
- Catch block
- Finally block

Let's build a Try catch block to handle exceptions, in following steps:

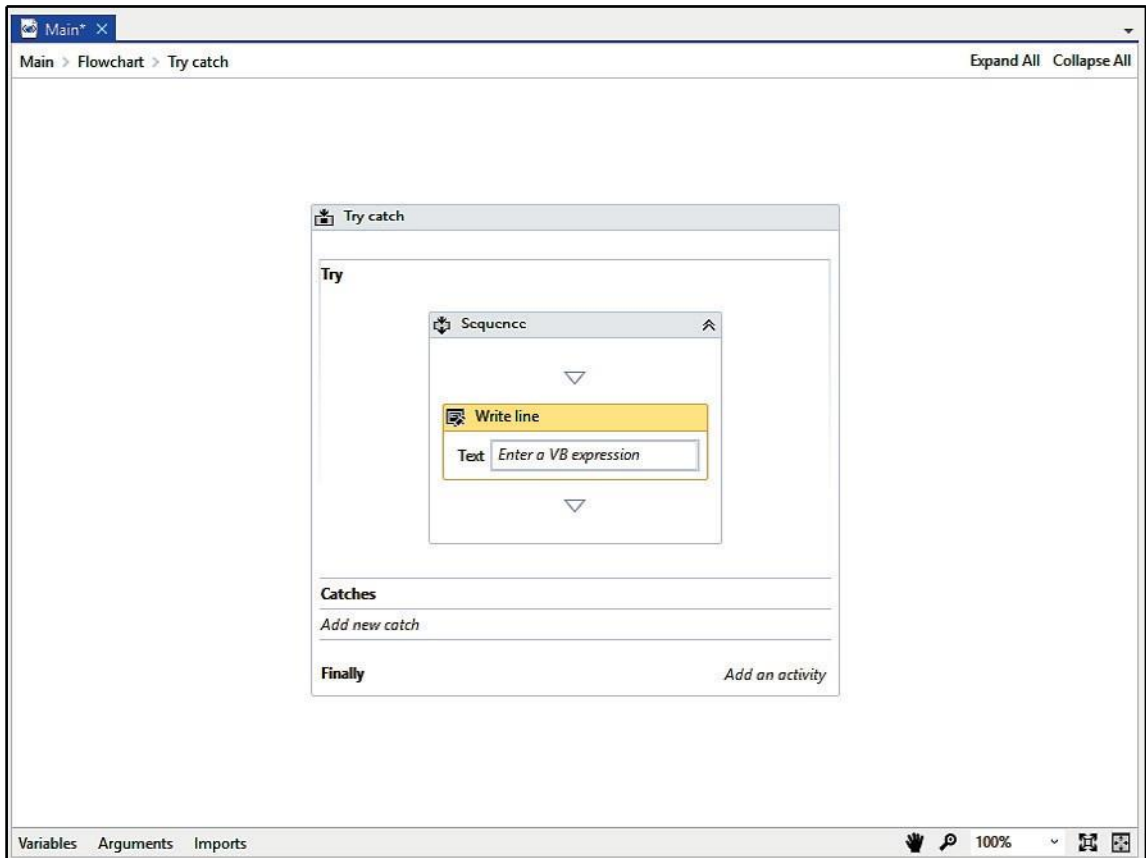
1. Drag and drop the **Try catch** activity: Create a blank project. Drag and drop the **Flowchart** activity into the Designer panel. Search for the **Try catch** activity in the **Activities** panel and drag it into the **Flowchart**. Set it as the **Start** node:



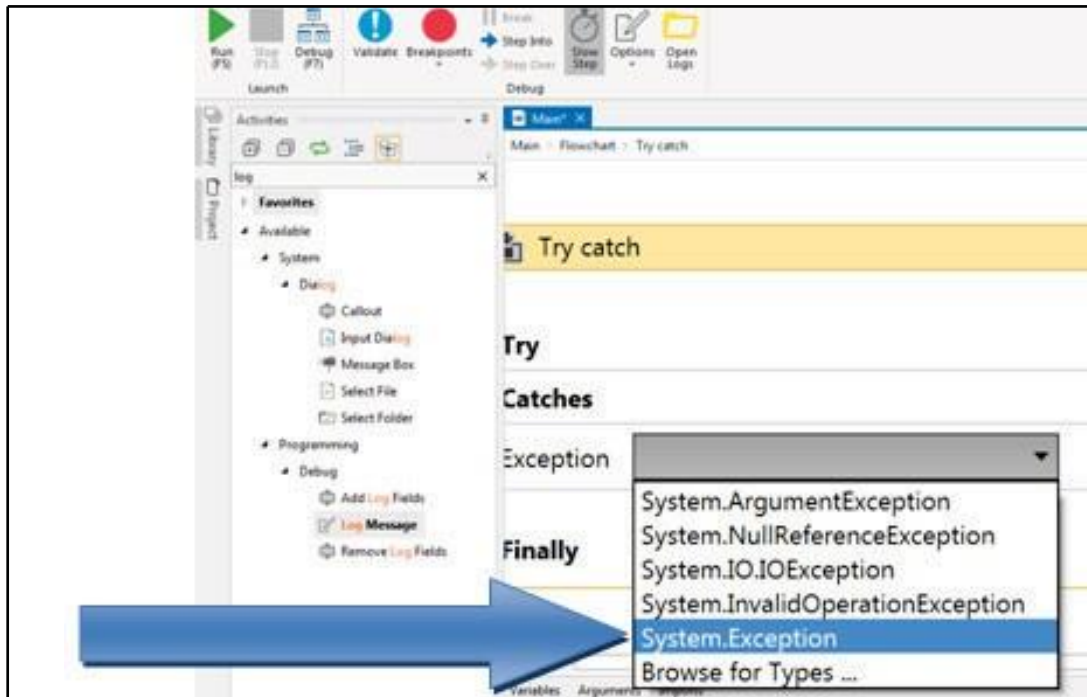
2. **Try:** When we double-click on the **Try catch** activity, dragged and dropped inside the workspace, space for the **Try** activity appears, as shown in the following screenshot:



Inside the **Try** block, we have to drop the activity we want to perform. Drop a **Write line** activity to test the working of **Try Catch** block, as shown in the following screenshot:

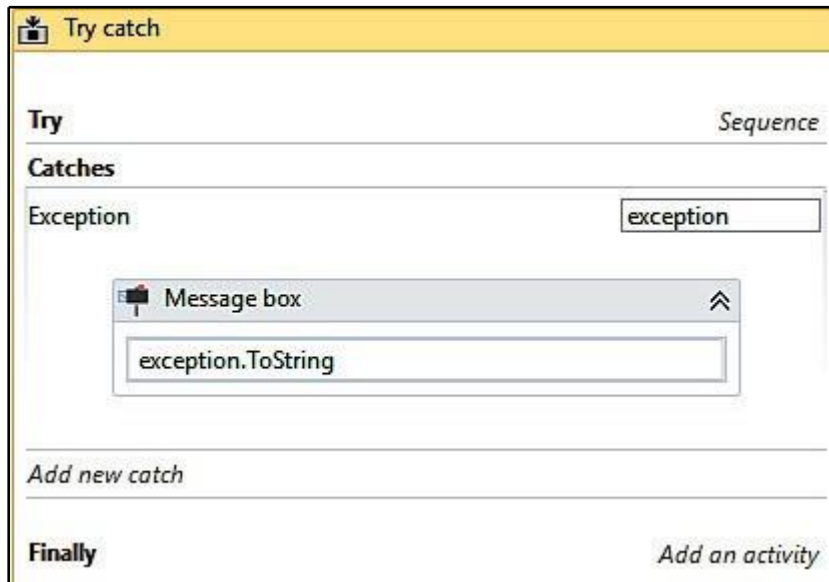


3. **Catches:** Inside the **Catches** activity, first we have to click on **Add new Catch** and then click on **Add Exception** option, from which we have to select the type of exception. In most cases, **System.Exception** is preferred. The following screenshot shows the types of exception. There are many more exceptions which can be viewed by clicking on the **Browse for Types** option:



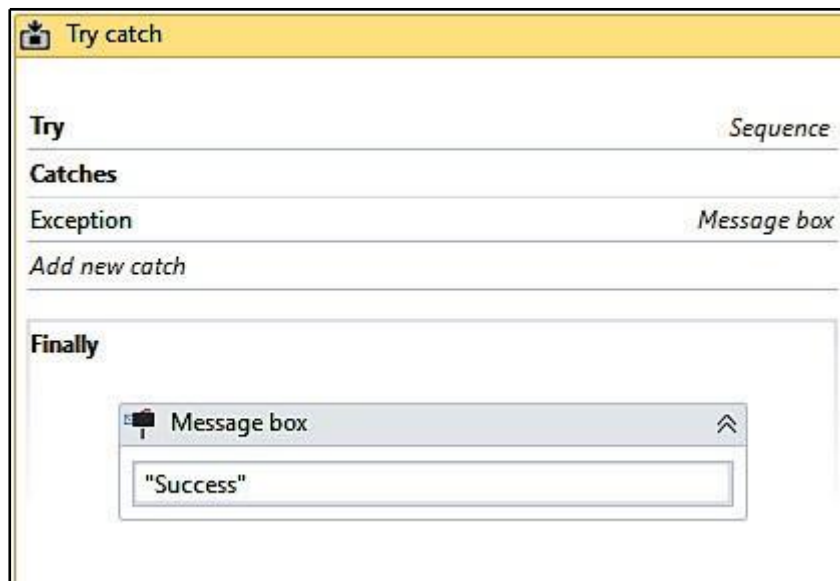
Exceptions that can be viewed when clicked on the **Browse for Types** option

Say the execution fails: for example, the **Click** activity is unable to be executed because of the unavailability of a UI element. In such a case, we can use the **Catches** block in order to either view the error that has occurred or for an alternative method to be used if that particular error occurs. As shown in the following screenshot, we will drop the activity in the **Catches** block. To print a message, we use a **Message box**:



When we click on **Add new catch**, we are asked to select the type of exception. We have selected **System.Exception**. Now inside the exception block, we have dropped a **Message box** activity. Entering `exception.ToString` will display the error that occurred during execution.

4. **Finally:** When we have defined the exception for our sequence, the **Finally** block will always work, regardless of whether the execution was successful or not. Suppose we want to display a message to the user notifying that the process is complete. To make sure that the whole **Try catch** activity is executed, we will just drop a **Message box** activity in the area provided in the **Finally** block, as shown in the following screenshot:

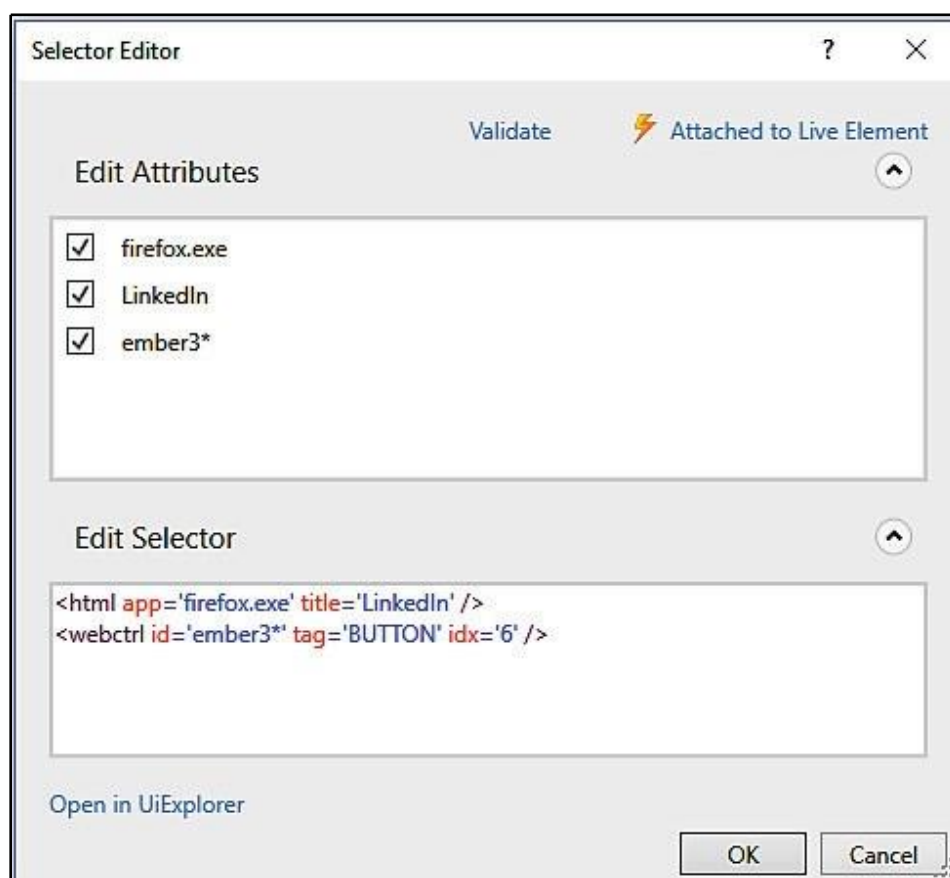


Common exceptions and ways to handle them

Implementing exception handling enables the robot to work in every possible situation and tackle any exception that may arise. There are some common exceptions that we usually face while working on UiPath.

Unavailability of UI element

When working on UiPath, especially on the web, we may encounter this type of error. This is because the UI element was not found due to the dynamic behavior of the web page. To handle this exception, we have to make changes in the selector attributes or we have to add new attributes to the selector so that the UI element can be easily found. For example, if we have a variable which is dynamically changing, we can use a wildcard so that it can be easily found by the robot. As shown in the following screenshot, we can edit the dynamic attribute of the selector using wildcard characters (in our case, *). Another way is to attach it to the live element:



Handling runtime exceptions

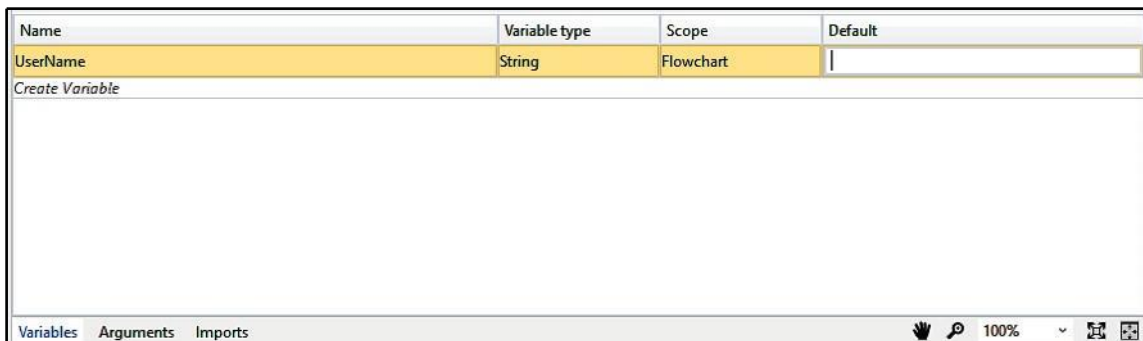
We may encounter runtime errors while working in UiPath. To rectify these errors, one of the best practices is to use the Try catch activity, which can be used to handle exceptions at runtime. By keeping an alternative inside the catch block, we can also overcome the error which we encountered before.

So placing your sequence or workflow inside the Try catch activity will help you handle runtime exceptions.

Orbit reference not set to the instant of an object

This type of error usually occurs when the default value required for some variable is not provided. In that case, we are required to give a default value to the required variable, as shown in the following screenshot.

In the empty area indicated in the following screenshot, just type the default value of the variable in order to overcome this error:



Index was outside the bounds of an array. Index out of the range

This error occurs when we try to iterate array elements by an index which is out of range. This happens when we are not aware of the size of the array and we just randomly type the index to access the element.

To resolve this, we must check the size of the indexes of the array or the collective list.

Image not found in the provided timeout

This type of exception is thrown because the image was not found. This may be due to a change of environment, such as resolution or theme settings. In this case, using some a selector attribute or indicating an anchor will work well:



As shown in the preceding screenshot, when we cannot identify the image properly, **Indicate Anchor** will help us indicate the UI element nearby so that the recorder can identify the correct image.

Click Generic error - cannot use UI CONTROL API on this UI node please use UI Hardware ELEMENTS method

This type of error occurs when the environment in which we are trying to use the

Click activity does not support Simulate or Send message activity (used by us to click the UI element).

Sometimes, either **SimulateClick** or the **SendWindowMessages** may be checked. In both cases, when an exception is thrown we just have to uncheck the appropriate box.

Logging and taking screenshots

UiPath has a multi-process architecture that offers to execute each workflow separately in the executor. Executors are managed by UI robots. So, if any executor stops working, then the entire process will not be affected.

Client logging

Client logs basically enable a server to record connections. These logs can be used by content providers in various scenarios, such as to generate billing, to trace media server usage, or to deliver suitable quality content depending on the speed of the client's server.

For client logging in UiPath, we have an NLog configuration file which makes it easy and flexible to integrate with databases, servers, or any other NLog targets. Logging can be configured with this `NLog.config` file. UiPath Studio, Robot, and workflow execution generate log messages on the client side:

- Messages which are produced by the workflow execution are logged with the execution logging source. Messages produced by UiPath Studio are logged as Studio Source and those produced by UiPath Robot are logged as Robot logging Source.
- We can also access these logs from UiPath Studio.

We can access the stored logs by clicking on **Open Logs** in the **EXECUTE** option. By default, these Logs are saved in `%Local App %\UiPath\Logs`:

- The automatic logging mechanism for all errors generated, including values of variables and arguments, may be enabled in the `UiRobot.exe` config file, which is present in `C:\Users\Username\AppData\Local\UiPath\app-17.1.6435`, by setting the `Log` parameter from 0 to 1 inside the `<Switches>` section.
- We have two activities that can be used for logging and these are the `Log message` and **WriteLine** activities.

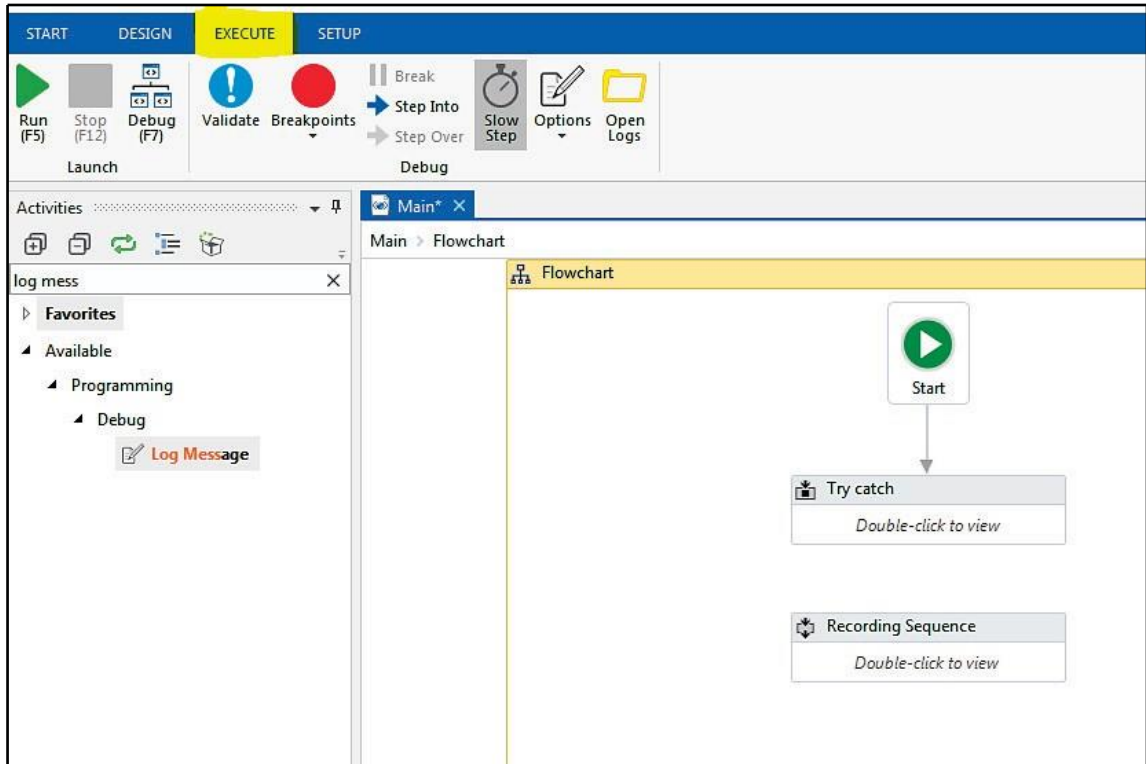
Server logging

If you have configured the UiPath server, then all logs generated by the execution are also sent to the server.

You can take a screenshot anytime by pressing `Ctrl + PrtScrn`.

Debugging techniques

There are various techniques provided by UiPath Studio for debugging in order to check whether the workflow is running successfully or to find out errors in order to rectify them. At the top of the UiPath window, we can see various available methods of debugging inside the **EXECUTE** block, as shown in the following screenshot:

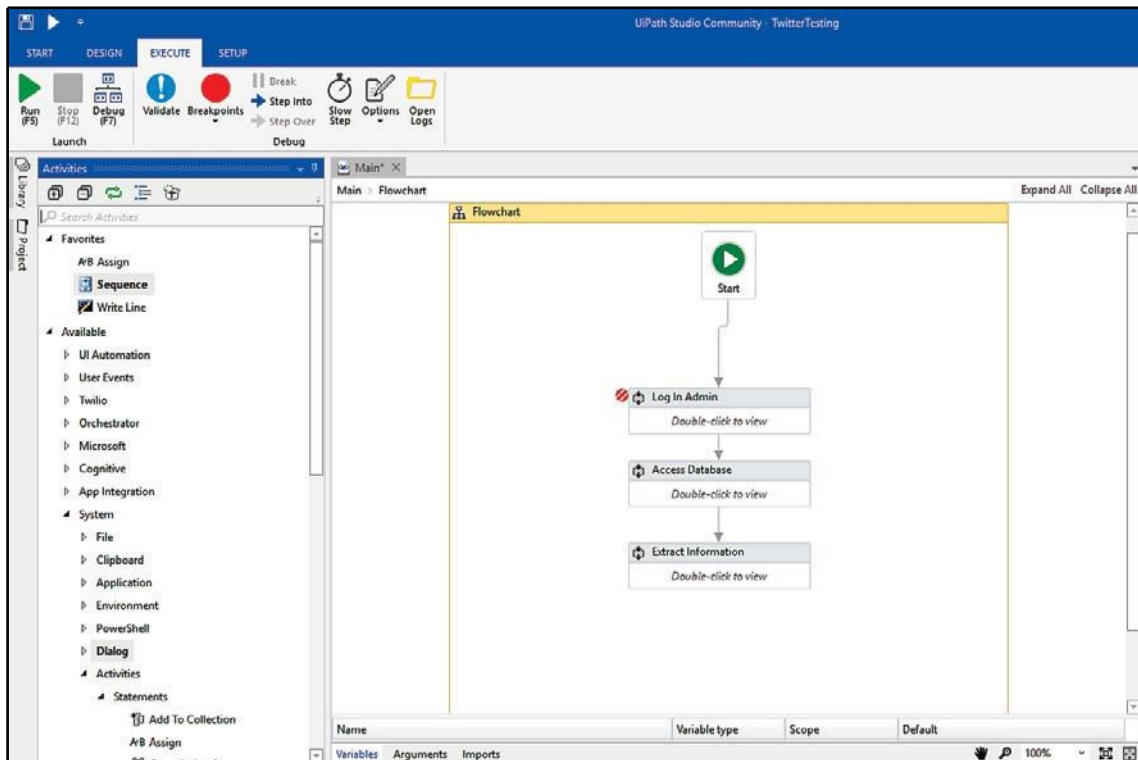


As shown in the preceding screenshot, there are various techniques for debugging. They are:

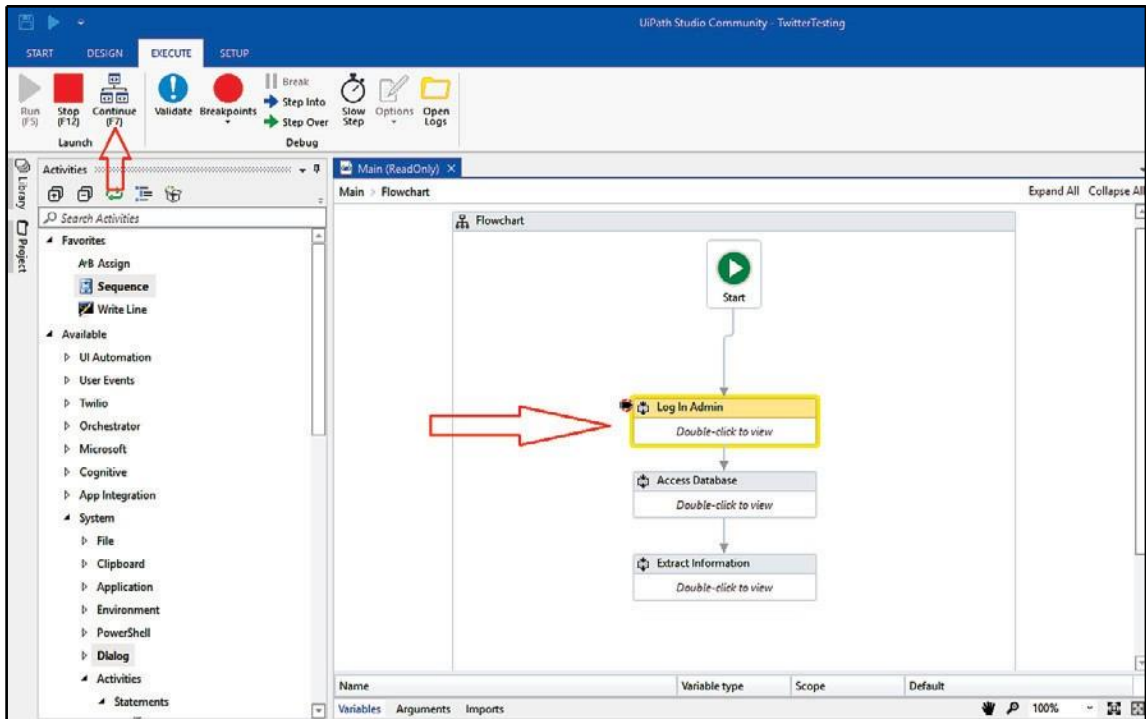
- Setting breakpoints
- Slow step
- Highlighting
- Break

Setting breakpoints

While debugging a workflow, we can set breakpoints in between if we want to run the program up to a specific location. This is useful when we have to stop before an activity ends completely. In such a case, we should use a breakpoint on the previous activity, as shown in the following screenshot:



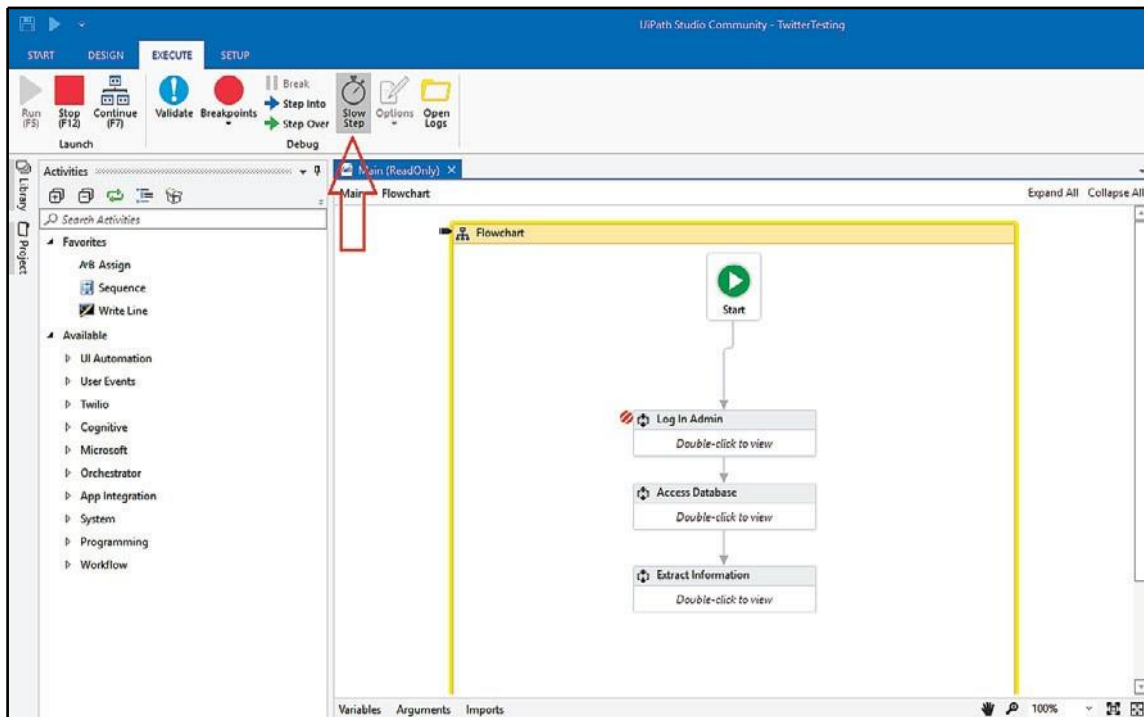
The highlighted region indicates the breakpoint since the execution stops just after the breakpoint. In order to continue any further, we have to click on the **Continue** button on the top corner indicated by the arrow:



When we click on **Step into**, the relevant part will start to execute; after we click on **Step over**, execution will jump to the next part, and so on.

Slow step

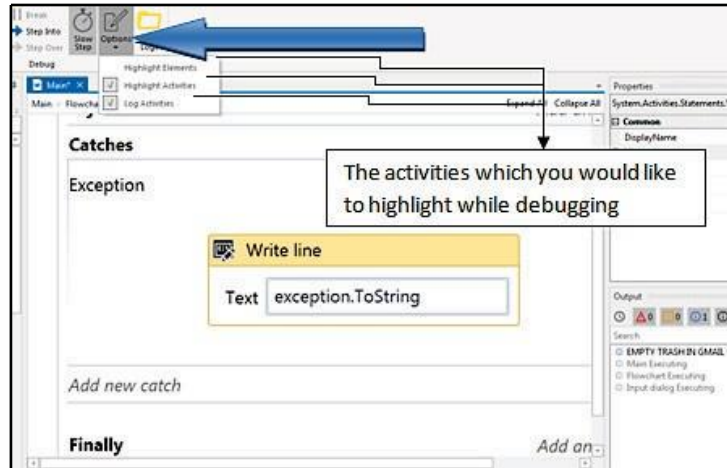
This is an activity in the **EXECUTE** block through which we can reduce the execution speed of a particular process or activity. This way, we can identify each and every process and keep an eye on where to find the error. In the Output panel, all activities or steps can be viewed. The following screenshot shows how to use the **Slow step** activity:



As indicated by the arrow in the preceding screenshot, when we click on **Slow step** the execution time for this particular step increases.

Highlighting

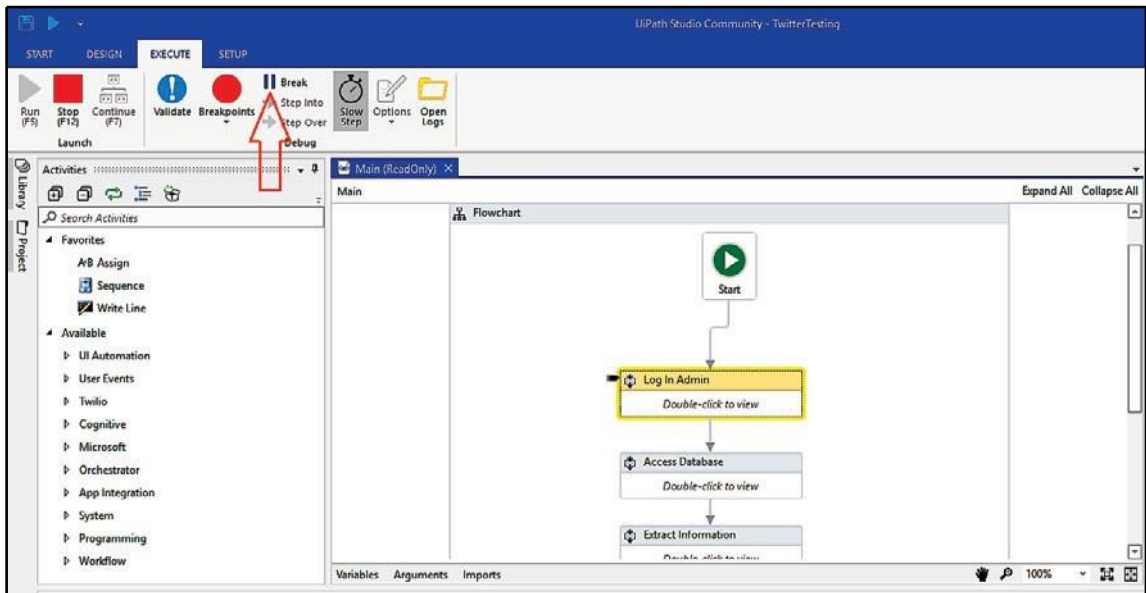
Highlighting is used to highlight the steps we have taken during automation and to identify each and every step in the workflow. It is very useful while debugging and its panel can be found in the **Options** menu of the **Execute** section in the Ribbon:



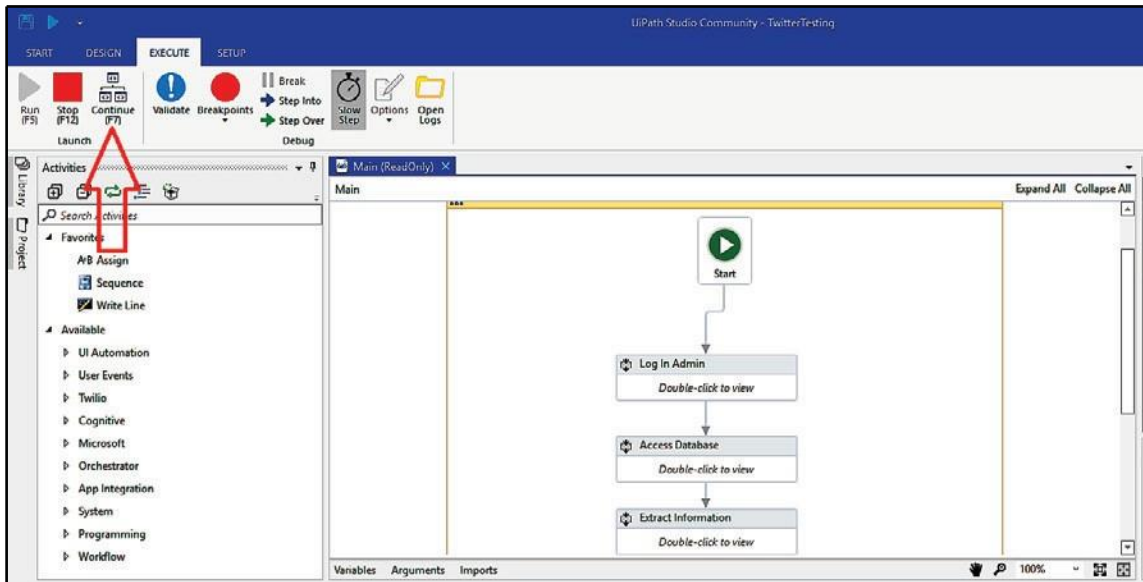
Various activities for highlight.

Break

The Break activity is used to break a process at a certain point. Suppose we have a sequence performing seven activities together and we want to break the execution at a certain activity. For this, we can use the **Break** activity, as shown in the following screenshot:



While debugging, an option for **Break** is available (indicated by the arrow in the preceding screenshot). We can break at any point we want to. If we want to continue any further, we just have to click on **Continue**, as indicated by the arrow in the following screenshot:



Or we can stop the execution at that point by clicking on the **Stop** option.

Collecting crash dumps

Collecting crash dumps basically refers to collecting information when your UiPath Studio crashes. We can enable and disable crash dumps. These dumps provide us with information regarding the UiPath crash.

Memory dumps are of two types—**full dumps** and **minidumps**. Full dumps provide us with complete information about the encountered crash while minidumps provide us with just the main information regarding the crash.

When a crash is encountered, we first have to identify the process which has crashed. Usually, a dialog will appear on the screen indicating the nature of the crash and the application involved. A UiPath process could crash, such as `UiStudio.exe`,

`Uiexplorer.exe`, or `Uilauncher.exe`, or the target application you want to automate may crash.

Enabling crash dumps

The following are the steps to enable crash dumps:

- 1 To enable crash dumps, we first have to download the `EnableFullDump.erg` file for full dumps from <https://cdn2.hubspot.net/hubfs/416323/QuickAnswers/EnableFullDump.reg?t=1513326308120> or the `EnableMiniDump.erg` file from <https://cdn2.hubspot.net/hubfs/416323/QuickAnswers/EnableMinDump.reg?t=1513326308120>
- 2 Double-click the file and click Yes. Administrator rights are needed to access the registry settings
- 3 The dumps folder is `%TEMP%` whose complete path is like `C:\\users 2;username\\AppData\\Local\\TEMP`
- 4 When the application crashes, you will find the `.dmp` file in the `TEMP` folder. For example, if `UiExplorer` crashes then a file such as `UiExplorer.exe.7429.dmp` will be found in the `TEMP` folder

Disabling crash dumps

To disable crash dumps, perform the following steps:

- 1 Download the `DisableDump.reg` file from <https://cdn2.hubspot.net/hubfs/416323/QuickAnswers/DisableDump.reg?t=1513326308120>.
- 2 Double-click the file and click Yes to disable crash dumps, Administration rights are needed for this action.

Error reporting

A user may encounter an error in UiPath and want to report it. As mentioned before, there are two types of customer for UiPath:

- Enterprise Edition customers
- Community Edition customers

Enterprise Edition customers

If you are an Enterprise customer, then you can report the error to the UiPath community in a very simple way:

1. Just go to this link: <https://www.UiPath.com/contact-technical-and-activations>.
2. You will be re-directed to a page where you have to fill out a simple form containing some basic details and then upload the file in which the error is encountered, as shown in the following screenshot.
3. After uploading, just click on the **Submit** button. UiPath will respond to you with the proper solution:

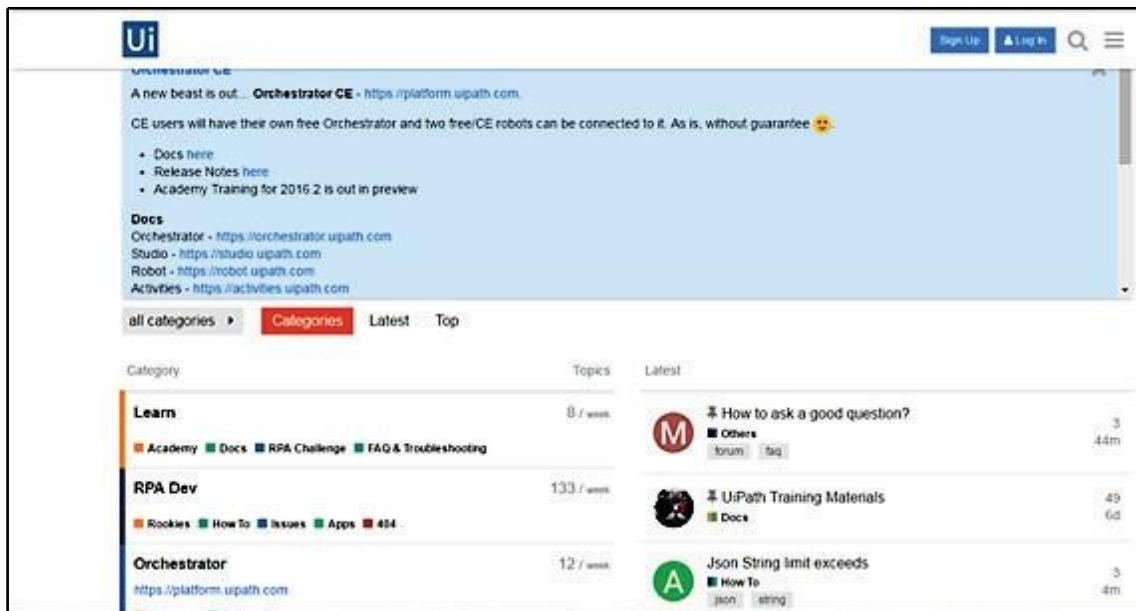
The screenshot shows a web form for reporting an issue. On the left, three boxes labeled 'name', 'Email', and 'Choose file' have blue arrows pointing to their respective fields on the right. The 'name' field is labeled 'Name*'. The 'Email' field is labeled 'Email*'. Below the email field is a question: 'Do you want to report an incident or submit a Service Request?'. This is followed by explanatory text: 'Incidents may refer to an error or malfunction of a UiPath component, leading to an unplanned interruption or degradation of a service. Service requests may refer to a request for information or advice, software activation and licensing.*'. Below this is a dropdown menu with the text '- Please Select -'. A red bullet point below the dropdown says 'Please select an option from the dropdown.'. Under the heading 'Attachment', there is text: 'Please add an archive with the relevant workflows, logs, error message screenshot etc.'. Below this is a 'Choose File' button and the text 'No file chosen'. At the bottom, there is a checkbox labeled 'I'm not a robot' and a CAPTCHA image.

Details that need to be filled while uploading the encountered error file

Community Edition users

Since the Community Edition is free, UiPath does not provide support to Community Edition users. However, all solutions to errors encountered can be found by going to the UiPath forum. All types of errors and their solutions are properly discussed in the forum. You can also go to the resources page and find the solution to your problem.

For this, visit <https://forum.UiPath.com/>:



Resource page

Summary

In this chapter, you have learned how to use exception handling techniques and log error screenshots, and other useful information to be used for debugging or reporting. You have also learned how to debug code.

You have learned how to make an automation project, as well as how to use exception handling. Your learning, however, does not end here. In the next chapter, we will refine your learning by teaching you best practices for managing and maintaining your code.

3. Remanufacturing

1 Introduction

Remanufacture returns a used product to like-new condition; it is a process of recapturing the value added to the material when a product was first manufactured. Remanufacture results in reduced energy and material use, and production cost

reductions. In the context of drivers such as the Landfill Directive, the revenue that remanufacture generates from 'waste' coupled with environmental advantages place the process as potentially a major contributor to Sustainable Development (SD) and movements towards a Factor 4 society.

Remanufacturing is an industry worth £5 billion per year in the UK and has been identified as a potential contributor to sustainable development (OHL, 2004). However the actual practice of remanufacture is limited, especially considering that the value of the UK manufacturing industry in 2003 was defined by the Economic & Social Research Council (ESRC) as £447 billion (ESRC, 2003). There are several strategies which may increase the practice of remanufacture; one of these strategies is Design for Remanufacture, which comprises of both product strategy/business model design and detailed product design.

The process of remanufacture was first brought to the fore at an industrial level by tank remanufacturing in World War I; for a time after World War II, it was profitable for all car manufacturers in the UK to engage in remanufacture. However remanufacture is still generally a niche practice today. Caterpillar, Xerox and Flextronics (who remanufacture a small number of Xerox products in Europe and a majority in the USA (Cosgrove, 2007b)) are leading global remanufacturers with a high profile, but the majority of remanufacture is carried out by much smaller organisations.

The 2004 Oakdene Hollins Ltd (OHL) report "Remanufacturing in the UK: a significant contributor to sustainable development?" provides a detailed evaluation of the state of the UK remanufacturing industry and identified future opportunities for remanufacture, highlighting that remanufacture can lead to a reduction in carbon emissions. This report investigates the links between design and remanufacture and builds on elements of the OHL report and contributions to elements of analysis by OHL. Nabil Nasr and Michael Thurston's 2006 paper "Remanufacturing: A Key Enabler to Sustainable Product Systems" brings Design for Remanufacture to the discussion and suggests the full potential of remanufacture to contribute to sustainable systems through the closed-loop economy; whereas Erik Sundin's work in "Product and Process Design for Successful Remanufacturing" brings detailed design to the fore.

This report investigates Design for Remanufacture in terms of both detailed product design and the business context in which Design for Remanufacture may operate.

2 Research Objectives

2.1 Key Study Objectives

- To understand the link between design and remanufacture

- To understand how Design for Remanufacture can lead to increased innovation and Sustainable Development (SD)
- To identify proactive strategies to further Design for Remanufacture

2.2 Research Methodology and Problems Encountered

This study into design and remanufacture was carried out using a combination of secondary and primary research. Initial, broad research into remanufacture became focussed on Design for Remanufacture as research avenues were identified.

Desk based research was planned to develop a research foundation of best international practice upon which informed questions could be framed. Information was found to be disparately spread and not always freely available; budget constraints required the purchase of documentation to be carefully considered.

Primary research was carried out in the form of a structured questionnaire, and interviews with national and international experts. The limited number of experts in remanufacturing and their time constraints placed some pressure on the study. The organisation of an invite-only, expert remanufacturing workshop considered these time pressures; the workshop successfully brought together practicing remanufacturers, researchers and industry groups to exchange experience and views on remanufacture.

There is a very limited amount of information available concerning Design for Remanufacture and even less detailing specific strategies for detailed product design. Due to the lack of information and case studies, the possible strategies of Design for Remanufacture presented in this report are an amalgamation of eco-design, and Design for Multiple Life strategies and may be at times conceptual.

Research

This report has benefited from the generous contributions of time and effort by numerous individuals and organisations. In particular, we would like to thank the following individuals and organisations for their help and support:

David Parker, Oakdene Hollins
Ltd Andy Cosgrove, Xerox

Tim Barker, Caterpillar
John Morrell,
Consultant

Andrew King, Bristol University

3 Definition of Remanufacture

There are numerous elements of 're': refurbishment, reconditioning, repair etc which are easily and commonly confused with 'remanufacture' due to similarities in the process and final product.

Over the past decade, literature and research has sought to define 'remanufacture'. The persistent appearance of these definitions (also included in this report) demonstrates a lack of appreciation of remanufacturing as a concept and a process.

Uncertainty over the definition of remanufacture creates confusion, which restricts progress towards the increase of remanufacture. Logically, it would be hard to attribute a successful business model to remanufacture if the audience understands 'remanufacture' to mean 'recycling'.

'Remanufacturing is not a widely understood concept.'

Ron Giuntini, The Remanufacturing

An understanding of remanufacturing has not yet been disseminated effectively into the manufacturing and associated industries. In some areas, work is in progress to increase awareness however this is still limited to only a few organisations such as Envirolink Northwest. Envirolink Northwest exist to promote, strengthen and support the environmental technologies and services sector in England's Northwest, taking a role that includes promoting remanufacture.

The misunderstanding of definitions is compounded by preconceptions resulting from a limited knowledge of certain historical practices loosely related to remanufacturing. The practice of poorly re-treading car tyres, which originated in the first part of the 20th Century, compared with modern practices, is a classic example of how an industry has to fight against preconceptions.

3.1 Remanufacture

When determining if a product is or is not remanufactured, it is essential to consider the process utilised. Remanufacturing is a process of recapturing the value added to the material when a product was first manufactured.

Used components and spare parts can be considered remanufactured if they are brought to at least Original Equipment Manufacturer performance specification from the customer's perspective and given a warranty equal to that of an equivalent new product (Ijomah, 2002). A product may be remanufactured with or without brand or product identity.

In terms of understanding the drivers and design for remanufacture, it is useful to note that remanufacture may be divided into two categories:

- Spare parts e.g. Automotive clutch, starter motor
- Products e.g. CR Scanner, Vending Machine

This distinction is necessary to understand that drivers and design issues for remanufactured spare parts and products may be different. This in turn has implications for the design of business models. The spare part industry, which consists of 2/3 of all remanufacturing activity (Steinhilper, 2006), is primarily driven by the market; this is particularly prevalent in the automotive sector. In contrast, legislation is a significant driver for products. These aspects will be further explored later in the report.

The remanufacturing process consists of 8 steps (including the essential collection of core discussed further in Section 4.2). A basic sequence is given here, but depending on the product remanufactured, the order of the steps may change (Sundin, 2004):

- Collection of core
- Inspection and identification of faults
- Disassembly of whole product
- Cleaning of all parts (and storage)
- Reconditioning of parts (and replacement with new parts where required)
- Reassembly of product
- Testing to verify the product functions as a new product



Core

‘Core’ is the term commonly used to describe the component or product that will be retained through the remanufacturing process. ‘Hulk’ is a synonym for ‘core’, used notably by Xerox.

3.1.2 Remanufacture Synonyms

- Re-manufacture
- Second-life production (‘3rd life’ etc up to ‘nth life’ may also be used)
- Repetitive manufacture

Two other synonyms which are sometimes used to mean ‘remanufacturing’ are:

- Asset Recovery
- Asset Regeneration



International Remanufacture Synonyms

International languages should also be considered when understanding terms related to remanufacture. Translation of terms can potentially add another level of confusion,

where words which are similar to English words that are *not* ‘remanufacturing’ may be used to describe remanufacture. For example:

- Inverse manufacture (Japan)

- Produktrecycling (Germany)
- Rénovation (France)

Industry-Specific Remanufacture Synonyms

There are also a number of industry specific synonyms for remanufacture:

Rebuilt = Remanufactured

- 'Rebuilt' is synonymous with remanufacturing when used in connection with motor vehicle parts and systems but not the entire vehicle.

Recharged = Remanufactured

- 'Recharged' is used in connection with the remanufacture of imaging products e.g. laser toner cartridges.

Retread / Remoulded = Remanufactured

- 'Retread' or 'remoulding' is used in the tyre industry

Rewound = Remanufactured

- 'Rewinding' is synonymous with remanufacturing in the sector of electrical ("rotating electrics") equipment.

Overhaul = Remanufactured

- 'Overhaul' is synonymous with remanufacturing, particularly in the aerospace industry. The term "rotables" is also in common use for items on scheduled overhaul.

Terminology adds to the confusion about the extent to which remanufacture is practised. The Remanufacturing Institute (TRI) in the USA (2006) explains that some organisations are now using the concept of remanufacturing in their environmental literature but not the terminology. For example:

- The American Society of Mechanical Engineer's position paper on "Designing for the Environment" includes the concept of remanufacturing.
- The American Automobile Manufacturers Association include remanufacturing in the life cycle of a car in their "Key Facts about America's Car Companies: Environment"

In contrast Siemens AG Medical Solutions is an example where they use the term 'refurbished' alongside practise that may appear to adopt elements of remanufacture. However Siemens AG Medical Solutions do not remanufacture and 'refurbished' does not mean 'remanufactured' (Plumeyer, 2006); this demonstrates once more the importance of verifying terminology in every case.

3.2 Remanufacture is not...

There are many terms which may be confused with remanufacture. Remanufacture is not:

3.2.1 **Recondition / Refurbish / Restore**

Reconditioning restores a product functionally to as-new or almost as-new condition but may not come with a warranty that matches a new product. Reconditioning may return a product to like-new quality but the process may not disassemble and clean all of a product's components; this means it is not 'remanufacture'.

These terms may be applied to architectural interiors; predominantly for rental properties where these terms are used frequently on sales boards. These terms may also apply to antique or classic goods. Consumer products may also be reconditioned

or refurbished; some are sold as 'reconditioned' with an 'as new' warranty. The closeness of this practice to remanufacture is not clear, which leads to further confusion.

3.2.2 Re-used / Used

This term is generally applied to a product that has been used previously. The product will retain the problems

it acquired during its previous life as it will not have been repaired.

3.2.3 Repair

Repair makes a broken product operational again. An analysis of the root cause of the problem is generally not performed in the repair process which means the product may not perform like a new product. Typically, a warranty on a repair will only apply to the specific repair and not the whole item.



3.2.4 Recycled

Recycling returns a product to raw material form, which can be used as raw material for a future manufacturing process. The term 'recycling' is generally applied to consumable goods e.g. newspapers, glass bottles and aluminium cans but can also be applied to durable goods such as an engine. Recycling in this sense destroys the value added to the raw material by forming it in the original manufacturing process.

The term 'recycled' is used by some legitimate remanufacturers to describe a product that may meet the minimum remanufacturing requirements. However, a 'recycled' product may also be removed from a redundant product and resold with little or no work performed on it e.g. in the automotive sector as part of an end-of-life reclaim and reuse operation. In this instance, 'recycled' would not be considered remanufactured.

'Demanufacturing' describes a disassembly process and is often associated with recycling. Automobiles for example need to be disassembled to ensure segregation of materials such as steel, aluminium, various plastics, etc. Demanufacturing is a step of that could in theory be applied to any 're' but is not a process in its own right.

3.3 Remanufacturing in Context of ‘re’

The importance of remanufacturing’s role in the context of all the ‘re’s is explained by Nasr and Thurston of the National Center for Remanufacturing and Resource Recovery:

‘Component reuse will typically result in lower overall material and energy use than component remanufacturing. However, only components that retain their value and conformance can be reused without compromising the durability or reliability of the final product.’

Nabil Nasr and Michael Thurston, 2006

Remanufacture allows components to retain their worth, thereby placing this process as a sophisticated element of reuse and repair strategies.

‘Remanufacturing is typically a more efficient means of material recirculation than recycling. Remanufacturing retains more of the energy associated with the original conversion of raw materials to finished product.’

Nabil Nasr and Michael Thurston, 2006

The importance of Design for Remanufacture in promoting remanufacture over recycling is that with old assemblies or equipment not Designed for Remanufacture, it is seldom possible to do more than recover the materials, and even this process of recycling may be difficult and costly (Graedel and Allenby, 1998, cited in Sundin 2004).

The chart below (Steinhilper, 1998)¹, compares remanufacturing with repair. The chart has been slightly updated to take into account new understanding as a result of this research:

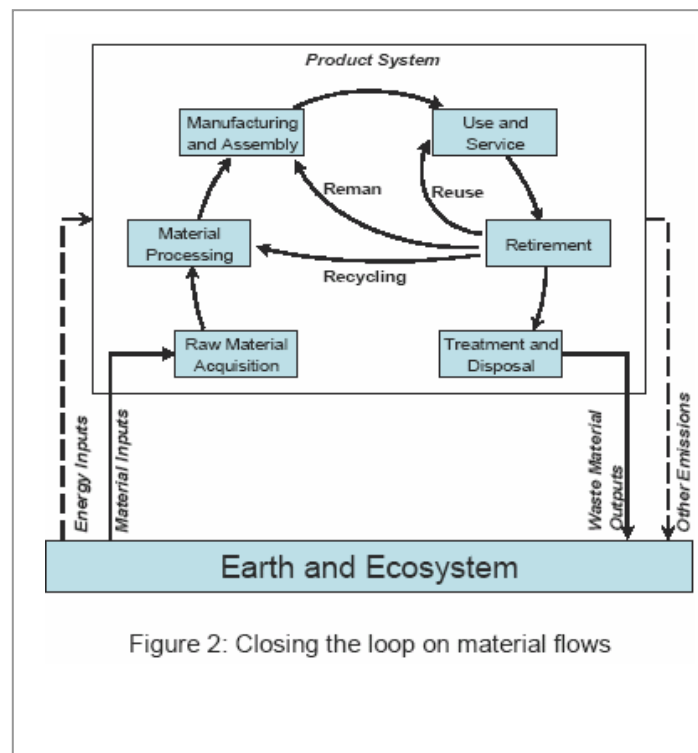
REMANUFACTURE	REPAIR
Applicability	
Used products	Defective products
Defective products*	
Process	
Complete disassembly	Failure detection
Cleaning of all parts	Disassembly of some parts
Remediation of parts to as new state / Replenishment of new parts / Upgrading of parts*	Restoration or replacement of defective part
Product reassembly	Reassembly of parts
Characteristics	
Industrialised process	Mechanic's work
Overall restoration to like new condition	Individual repair of defect
Customer receives anonymous product	Customer keeps his/her own product
Like-new or lifetime warranty	Warranty covering repair work only

Upgrading/Upcycling to state-of-art technology	Product retains earlier standard

The difference between remanufacture and other elements of 're' are therefore distinct in terms of its process and positioning within the material flows loop.

Nasr and Thurston's diagram below shows how remanufacturing sits within a spectrum of reuse. It is important to understand this to realise that depending on the particular product and its lifecycle position, all elements of 're' should be considered to identify the most appropriate end-of-life strategy.

¹ Entries denoted by and asterisk (*) are entries added as a result of consultation with Morley and Parker of Oakdene Hollins Ltd (2007).



However remanufacture cannot be applied in isolation, to operationalise the ‘circular economy’ you need all the “re’s” (Charter, 2006).

4 Opportunities for Sustainable Prosperity in South East England and UK

4.1 State of Art: Where is the South East now?

The UK generates approximately 400 million tonnes of waste every year and this is growing annually by 3%. Industrial and commercial waste accounts for around 20% of total waste and over half of this goes to landfill. Remanufacturing offers an alternative to landfill or other disposal options which eliminate the value added to materials by the manufacturing process. The remanufacturing industry is estimated to be worth £5

billion a year in the UK, which places it on a par with the recycling industry (OHL, 2004).

The majority of both remanufacture and product design in the UK occurs in the South East and London. The South East, an area with a high level of consumption, can utilise its good representation of these sectors to take the lead in the new approach in Sustainable Consumption and Production (SCP) (SEEDA, 2006). However most design engineers and product designers have little knowledge of eco-design, let alone Design for Remanufacture.

4.2 Financial Opportunities: Remanufacturing and Employment

Remanufacture is reputed to lead to increased profit and creates a demand for skilled employment. Remanufacture therefore has the potential to help the South East improve its economic prosperity through raising productivity and economic activity

across the region, and through releasing the untapped economic potential of all parts of the region.

Remanufacture can increase prosperity flows by bringing resources into reuse and increasing productivity. This relates to the South East's 6 drivers of prosperity given in the SEEDA Regional Economic Strategy (RES):

4.2.1 Employment

Remanufacture employs up to 50,000 men and women in the UK (OHL, 2004). Remanufacture demands skilled labour, however 3rd party remanufacturers, face economic challenges. Reduced costs through an optimisation of the remanufacture process via activities which may include Design for Remanufacture may help the preservation and creation of jobs in remanufacture. many small businesses face challenges such as costs.

4.2.2 Enterprise

Most remanufacturers are small 3rd party, engineering-oriented companies with only a few employees (OHL, 2004). These businesses have to react to opportunities in product and component remanufacture that are controlled by OEMs' original designs. Promotion of products that have been Designed for Remanufacture can increase the level of business start-up activity by increasing the opportunities for remanufacture.

Proof of the business case for remanufacture in the form of successful business models and products is required to encourage investment. This is likely to require collaboration between enterprising individuals, businesses and industrial sectors.

The UK Government's Business Resource Efficiency & Waste Programme (BREW) began in April 2005, to ensure that increases in Landfill Tax will not only incentivise businesses to reduce the amount of waste they send to landfill, but will also assist them in developing ways to achieve this.

4.2.3 Innovation and Creativity

Design for Remanufacture requires both innovation and creativity to optimise the processes of remanufacture. Promoting the optimisation of remanufacture can lead to increased research and development (R&D) and will drive the market for R&D, innovation and creative skills, and bring the South East into line with Europe's leading regions.

4.2.4 Skills

Remanufacturing presents difficulties to automation because the process is frequently concerned with batch productions of a range of components or products. This variety would mean that if machinery were to be used in disassembly etc, it would have to be frequently reset and reconfigured. This is likely to make automation prohibitively expensive for the majority of remanufacturers. Therefore remanufacturing often necessitates human intelligence and evaluation. The skilled work of the remanufacturer is reputed to lead to greater job satisfaction, particularly those involved in more creative remanufacturing (OHL, 2004), which would apply to Design for Remanufacture.

'[I used to mend and service machines] my job was far more interesting then.'

Ray Gould, Southern Appliances, 2006

Skill levels in the South East's workforce are relatively strong compared to most of the UK and Europe, however there are also

significant variations within the region, and parts of the Coastal South East are marked by much lower workforce skills than the regional, and in some cases national, averages (SEEDA, 2006). The promotion of remanufacture would create markets for skilled employees.

4.3 Sustainability Opportunities: Remanufacturing and the Environment

Remanufacture has inherent positive environmental consequences. These relate to the four priorities for action within Prosperity and Sustainability (SEEDA, 2006):

4.3.1 Sustainable Consumption and Production

Remanufacture views 'waste' as a resource. Promotion of remanufacture can therefore benefit both economy and environment. Design for Remanufacture can optimise the process of remanufacture, increase the practise of remanufacture and therefore increase the significant economic development opportunities for the South East region.

4.3.2 Climate Change and Energy

Remanufacturing typically uses 85% less energy than manufacturing (Steinhilper, 2006). Studies performed at the Fraunhofer Institute in Stuttgart, Germany, estimate the energy savings by remanufacturing worldwide equals the electricity generated by 5 nuclear power plants. This corresponds to 10,744,000 barrels of crude oil or a fleet of 233 oil tankers a year. By requiring significantly less energy than manufacture, remanufacture can address the major concerns over continuing security of energy supplies in the face of a prospective 'energy gap' in the South East.

A rough assessment of resource impact suggests remanufacturing is also saving in excess of 800,000 tonnes CO₂ emissions per annum (OHL, 2004).

4.3.3 Natural Resource Protection and Environmental Enhancement

Through reuse of resources, remanufacture leads to a reduction in landfill, pollution and natural resource use, which can preserve the quality of the natural and built environment.

Landfill Reduction

Products that are remanufactured are kept out of the waste stream for longer thereby preserving landfill space. If remanufacture is taken to its fullest extent, where everything that is manufactured (including construction waste, packaging waste etc) is remanufactured the practise of landfill could in theory be eliminated altogether, resulting in a closed-loop society. This is clearly a long-term view, which is likely to require all products to be fully Designed for Reuse.

Pollution Reduction

Air pollution is reduced by the remanufacture of products that would have had to be re-smelted or otherwise reprocessed in manufacture. The Chartered Institute of Environmental Health (CIEH) states that industrial processes such as smelting and metals processing require oil and solvent-based fuel mixes to be burnt, releasing fine particles measuring 2.5μ or less (PM2.5s), which have been linked to a number of diseases because of their ability to enter deep into the lung. In the last 10 years experts say that PM2.5s have risen 'astronomically' in the UK (CIEH, 2005).

Raw Materials Savings

The Fraunhofer Institute determined that raw materials saved by remanufacturing worldwide in a year would fill 155,000 railroad cars forming a train 1,100 miles long (TRI, 2006).

Quality of Life

Remanufacture can also increase quality of life by contributing to environmental preservation and driving the market for skilled labour; remanufacture can thereby preserve the competitive offer of the South East and promote the economic advantages of locating in the region.

4.3.4 'Factor 4' Society

Remanufacture's cumulative benefits of energy and material savings, landfill and pollution reduction place remanufacture as a major contributor to SD. The promotion of remanufacture is therefore a significant factor in progress towards a closed-loop, 'Factor 4' society.

5 Parameters for Successful Remanufacture

In theory, any product that can be manufactured can be remanufactured but the design of some products' business model and detailed product design means that some products can be remanufactured more profitably than others.

'Any product that can be manufactured can also be remanufactured. However, some products are remanufactured more often than others.'

Don Ciarletti, 2006

5.1 Product Parameters

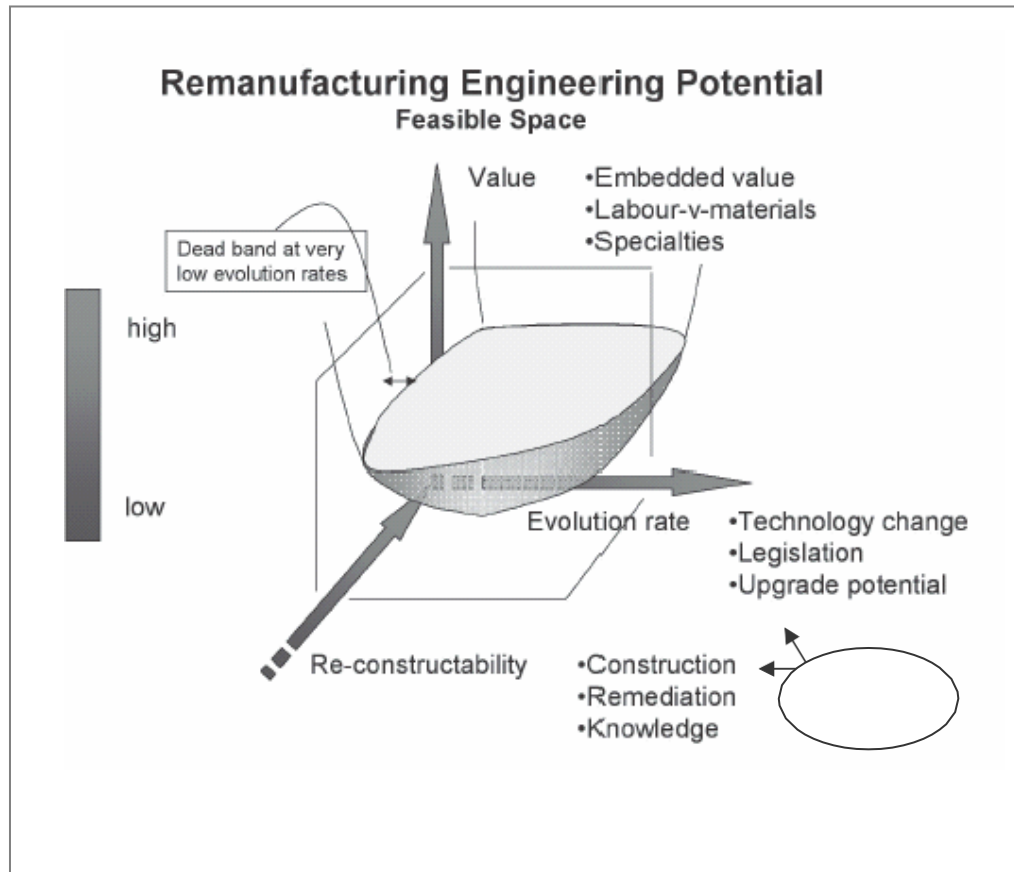
The criteria required for a product to be successfully remanufactured and sold are described by William Hauser and Robert Lund (date unknown) based on 25 years of research into the remanufacturing industry:

- Technology exists to restore product. Technology must be able to extract a component without damage (Nasr and Thurston, 2006).

- Product is made up of standard interchangeable parts.
- Cost of core is low relative to savings in product cost achieved through core reuse.
- Product technology is stable over more than one life cycle.
- Sufficient market demand to sustain enterprise.
- An evaluation of disposal options and environmental impact of legislation is also necessary to determine a product's suitability for remanufacture (Nasr and Thurston, 2006).

'If product design permits, and there is an adequate process for return of used products (reverse logistics chain), there can be a strong business model in product remanufacturing.'

These concepts are visually demonstrated by OHL's 'Remanufacturing Engineering Potential' model, showing a range where the parameters meet to make remanufacturing feasible.



The model shows that remanufacture is unlikely to be successful when there is low value in the core, there is a high evolution rate and no return channels. Within this model, design is particularly relevant to 'upgrade potential' and 'construction' (Morley, 2006).

An overview of which product characteristics and the remanufacturing process they relate to is given below. After establishing which stage of remanufacturing is an issue for a particular product, this chart may be used as a guide to indicate what product characteristic could be improved by design.

Product Property \ Remanufacturing Step	Inspection	Cleaning	Disassembly	Storage	Reprocess	Reassembly	Testing
Ease of Identification	x		x	x			x
Ease of Verification	x						
Ease of Access	x	x	x		x		x
Ease of Handling			x	x	x	x	
Ease of Separation			x		x		
Ease of Securing						x	
Ease of Alignment						x	
Ease of Stacking				x			
Wear Resistance		x	x		x	x	

Figure 24. The RemPro-matrix showing the relationship between the preferable properties and the generic remanufacturing process steps.

Of the above, Sundin (2004) found that the following four properties were most frequently found to be important for products' and components' remanufacture:

- Ease of access
- Ease of identification
- Wear resistance
- Ease of handling

The importance of 'ease of access' on numerous steps of remanufacture makes the effective application of Design for Disassembly, imperative in Design for Remanufacture.

5.2 Core Consideration

Remanufacture cannot occur without core, because without core there would be nothing to remanufacture! Core is generally collected through a specifically designed business model however the remanufacturing process relies so fundamentally on the collection of core that any steps that may help the collection of core should be utilised. If the detailed product design can aid core collection, collection should be considered in the product design process. Detailed product design for remanufacture therefore

extends the conventionally understood remanufacturing process to consider core collection.

5.3 Currently Remanufactured Products

The following is a short list of some of products that match these criteria and currently remanufactured:

- Aerospace
- Bakery Equipment
- Compressors
- Data Communication Equipment
- Gaming Machines
- High end electronics and electricals
- Industrial machinery e.g. machines and tooling
- Motor Vehicle Parts (notably engine parts and tyres)
- Laser Toner Cartridges
- Musical Instruments
- Office Furniture
- Photocopiers
- Refrigeration
- Robots
- Vending Machines

Source: OHL, 2004; TRI, 2006

Remanufacturing is least prevalent in areas producing consumer goods subject to fashion or status-related purchasing decisions. It is most prevalent and holding up, in sectors of very high value or technological content such as aerospace, military and power turbines. Remanufacturing also thrives in sectors which have embraced the concept of Product Service Systems (PSS) because in these instances there are shared motives for product longevity, durability and performance. (OHL, 2004).

6 Remanufacture and Product Design in the UK

The ESRC states that to remain competitive, the UK's manufacturing sector needs to successfully move from competing on relatively low cost to competing on unique value and innovation (ESRC, 2006). Design offers manufacturing a service, which can deliver both value and innovation.

6.1 UK Remanufacturing Industry

As previously mentioned, in 2004, remanufacturing in the UK was worth an estimated £5 billion per year and employed up to 50,000 people (OHL, 2004). In contrast, the UK manufacturing industry is far bigger and employs far more people; for example in 2003 UK manufacturing was worth £447 billion and employed approximately 3,500,000 people (ESRC, 2006). This suggests there is a very significant proportion of manufactured products, which could be remanufactured if the business model and products were designed to facilitate remanufacture; but they are not remanufactured. Therefore the opportunity for increasing remanufacture in the UK is potentially high.

The majority of remanufacturers in the UK are small, engineering-oriented companies with only a few employees (OHL, 2004). As an indication of remanufacture in the

UK, a brief survey of the Yellow Pages (2006) identified 166 companies listed under 'remanufacture'. Due to the nature of advertising, the Yellow Pages is unlikely to include all remanufacturers and of the 166 listed, many are retailers of remanufactured items such as ink cartridges and not remanufacturers themselves. However the listing may be taken as indicative of the concentration of remanufacturers in the South East of England:

Location	Companies listed under 'remanufacture'
South East England (excl. London)	30
London	41
South West England	20
Wales	8
Midlands	26
North East England	3
North West England	18
Scotland	12
Northern Ireland	8
Total UK	166

Source: www.yell.com, 2006

Anecdotal evidence (Steinhilper, 2006) suggests that remanufacture may be up to twice as profitable as manufacture, however it is hard to confirm this due to the difficulties in accurate data collection (Parker, 2007). This may be explained by the way in which data can be realistically

collected. Remanufacture is carried out mostly by small companies however data is more readily available from larger remanufacturers. Small companies may have a less predictable turnover

'We can make the same sort of margin on either product [remanufactured or not remanufactured].'

than large companies; for example, interviews with some small remanufacturers carried out during this study, identified that small remanufacturers can face financial difficulties. Larger remanufacturers may also benefit from economies of scale and purchasing power and so their profits may be higher than profits at smaller companies. Therefore the data from larger companies can easily skew research results, especially if larger companies form an unrepresentative proportion of the overall data.

6.2 UK Design Industry

Design for Remanufacture can optimise the process of remanufacture and logically, its practise is controlled by OEMs who initiate the design and manufacture of products. In practise design and remanufacture are linked by only a few OEMs, such as Xerox and Caterpillar.

A profile of the UK design industry puts the practitioners of detailed product design, who, if educated in Design for Remanufacture could potentially put it into practise (Design Council, 2005):

- 14,841 product & industrial design businesses in the UK, approximately 5,500 of whom are in-house teams.
- 65% (3,600) of in-house designers work in the private sector, 30% in the public sector with only a small number in charity, community or voluntary sectors.
- 20% of product and industrial design businesses are based in the South East and a further 21% are based in London.
- 79% of companies in product and industrial design have 4 or less employees; the second largest group comprises 8% of Product & Industrial design companies with 50-249 employees.
- 17% of design businesses say their main competition comes from outside the UK and 79% of product and industrial design companies report that overseas competition has increased
- 40% of design consultancies use informal mentoring to develop their staff.

To what extent Design for Remanufacture is practised in the UK is unknown but it is likely to be limited to a small number of Original Equipment Manufacturers (OEMs) who engage in the remanufacture of their own products and therefore have an incentive to increase the efficiency of remanufacture. Although consultancies may in theory be able to offer Design for Remanufacture (if they have the capability) to give clients added value, time constraints and low levels of awareness make this highly unlikely.

Awareness of eco-design, let alone Design for Remanufacture in the design industry is low. This situation is perpetuated by the lack of applied courses in eco-design and sustainable design, which results in professional designers and design students not receiving training in these areas. This lack of awareness is one of the reasons why the practise of Design for Remanufacture is limited.

6.3 Remanufacture and Product Design

The imperative for connecting design and remanufacture is established by Nasr and Thurston (2006) who state that the full societal benefits of remanufacturing (reduced energy and material consumption and reduced wastes) cannot be achieved unless Design for Remanufacture becomes an integral part of the product development process.

‘Xerox has been able to maximise the end-of-life potential of products and components by

incorporating reuse considerations into its design process.'

Xerox, date unknown a

Remanufacture begins with core and all core is designed prior to manufacture. Components' form, material, fastenings etc can all affect the ease with which a product undergoes the various stages of remanufacture e.g. disassembly or cleaning. Therefore Design for Remanufacture has a clear part to play in improving the efficiency of remanufacture.

In combination with other strategies, the different thought processes required for Design for Remanufacture (and more specifically Design for Upgrade) may be able to feed into innovation and new business models for UK industry. Xerox is an example of how Design for Remanufacture can benefit a business. Xerox began remanufacturing operations in the early 1990s and is now a world leader in this field.

The profitable link between design and remanufacture is however poorly understood or irrelevant to the majority of remanufacturers because as small independent remanufacturers they have little or no control over the design process. To put OEMs' involvement in remanufacturing in context, in the USA, OEMs account for less than 5% of total remanufacturing activity (Guide, 2000, cited in Sundin, 2004). Proactive ways of stimulating industry in the UK by encouraging Design for Remanufacture will be discussed later in this report.

7 Design for Remanufacture

Consideration of design and remanufacture brings to the fore three questions regarding Design for Remanufacture:

- What is 'design' in the context of remanufacture?
- What can detailed product design do for remanufacture?
- How can the practice of Design for Remanufacture be enabled?

The actual process of detailed product Design for Remanufacture will be discussed in the section 'What is Design for Remanufacture?'.

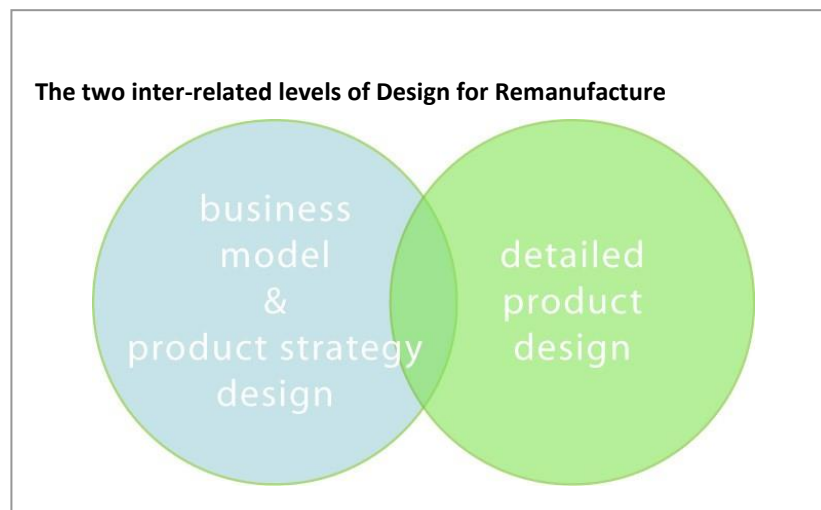
7.1 Design in the context of remanufacture

Remanufacture is affected by the physical characteristics specified during the design phase, whether the product has been Designed for Remanufacture or not. Product Design for Remanufacture is enabled by business models which recognise the benefits of remanufacture.

We can therefore consider Design for Remanufacture on two levels:

- Product strategy, including sales, marketing, service support, reverse logistics/core collection.
- Detailed product design and engineering, including core collection and functional

design.



Some aspects of remanufacture may span both levels. For example, core collection mechanisms may require the design of the business model to facilitate collection e.g. PSS as used by Xerox, or financial incentives for the core's return as used by Caterpillar; but the detailed product design may also communicate information including collection methods to all stakeholders, using graphics for example.

Generally the discussion of Design for Remanufacture technologies focuses on the detailed product design issues (Nasr and Thurston 2006). This report investigates both these aspects of Design for Remanufacture; this section will primarily explore detailed product and engineering; further sections explore the drivers and challenges that relate to the business model design for remanufacture.

A consistent understanding of the job roles related to detailed design and remanufacture is also useful to demonstrate that different roles may be able to feed different skills into the different aspects of Design for Remanufacture:

Product Design

'Product Design can be defined as the idea generation, concept development, testing and manufacturing or implementation of a physical object or service...Product Designers conceptualise and evaluate ideas, making them tangible through products in a more systematic approach [than Industrial Design]. The role of a product designer encompasses many characteristics of the marketing manager, product management, industrial designer and design engineer.' (Wikipedia, 2006g)

Engineer

‘An engineer is someone who is trained or professionally engaged in a branch of engineering. Engineers use creativity, technology, and scientific knowledge to solve practical problems. People who work as engineers normally have an academic degree (or equivalent work experience) in one of the engineering disciplines.’ (Wikipedia, 2006h)

Design Engineer

‘A Design Engineer is a general term that covers multiple engineering disciplines - Electrical, Mechanical and Civil Engineering are the basic three. The Design Engineer is responsible for the design and development of new products, equipment or facilities.

The Design Engineer is distinguished from the designer / drafter by virtue of the fact that a Design Engineer sets the direction of the design effort. The Designer/Drafter is usually directed by the Design Engineer.' (Wikipedia, 2006f)

7.2 What Can Detailed Product Design do for Remanufacture?

The most effective way to boost remanufacture is through the product and process design approach (Amezquita and Bras, 1996, cited in Sundin, 2004). Design for Remanufacture can identify and prevent inefficiencies in remanufacture.

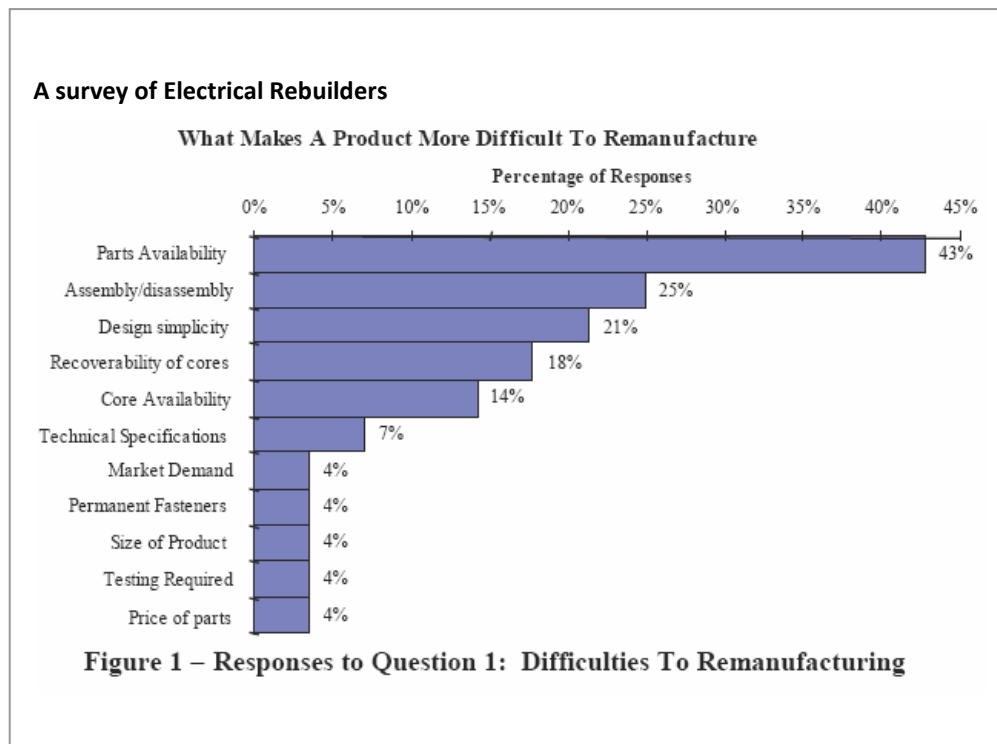
In contrast, it is important to note that it is also possible to design products to ensure they cannot be remanufactured. Some OEMs who want to prevent 3rd party remanufacturers may actively prevent remanufacture of their products using assembly techniques such as ultrasonic welding, which may make it impossible to disassemble a product without destroying the very components that are sought.

'GEMedicalSystems wanted to test out the profitability of a voluntary take- back and remanufacture of their used equipment. To test it, a manager was asked to run a pilot in an old plant for a year, after which period he had to report if it was profitable. The reply was: "Yes, we can break even, but if the products were designed for remanufacture, then we could have made a lot of money.'

Walter Stahel, The Product-Life Institute, speaking at the Remanufacturing Stakeholder event

7.2.1 Remanufacturing's Design-Related Needs

Remanufacture can be twice as profitable as manufacture (Steinhilper, 2006) but profit generation remains a priority for remanufacture as shown by a survey of remanufacturing executives. This survey revealed that 60% cite the increased pressure to reduce lead times continuously as the biggest threat to their remanufacturing activities (Guide, 2000, cited in Sundin 2004).



Design for Remanufacture can optimise remanufacture, making profit margins greater. A survey of American automotive remanufacturers showed the main issues with regard to Design for Remanufacture can be grouped as concerning (Hammond et. al, 1998):

- Complexity
- Fastening methods
- Means of assembly and disassembly
- Increased part fragility

Design for Disassembly relates strongly to all the above, by allowing ease of disassembly which results in faster disassembly times and greater recovery of intact parts. Design for Disassembly is therefore an important component of Design for Remanufacture. In instances and for products where remanufacture is suitable, Design for Remanufacture may improve the efficiency of remanufacture by:

- Reducing disassembly and reassembly times and thereby also reducing inspection/evaluation time and costs
- Specifying materials and forms appropriate for repetitive remanufacture
- Building mechanisms into the product or component to ensure the return of cores.

7.2.2 Difficulties in Ranking Design and Remanufacturing Process Issues

It is difficult to obtain a clear ranking of design and remanufacturing process issues and this may hinder the development of methods for assessing a product's remanufacturability (Hammond et. al, 1998).

The target of any design effort might change depending on the product type, infrastructure and where the product sits within the spectrum. Hammond et. al (1998) say that, a large source for the variations appears to originate from the different perspectives about remanufacturing. This is a result of differences in the scale of businesses and the type of customer primarily serviced.

7.2.3 Design for Remanufacture in Context

Although Design for Remanufacture may contribute to improving the efficiency of remanufacture, it is important to note that Design for Remanufacture is not always the best option, either environmentally or economically. For example, from an environmental perspective it may be better in some circumstances to promote other strategies such as reuse, whereas fast-moving consumer goods may best designed for recycling.

In some cases, other strategies concerning the business model may have a greater impact in stimulating the remanufacture of a particular product than detailed product design. A few of these options may concern:

- Reverse logistics
- Collection of core parts
- Reducing the rejection of remanufactured products

7.3 Enabling Design for Remanufacture

Research in compiling this report has revealed that the pressures of time to market and cost outweigh other aspects such as eco-design which may be considered non-essential (aside from compliance with legislation). It is reasonable to assume that Design for Remanufacture is therefore not a priority for the vast majority of companies in the normal business model.

Business model design that recognises the value of improved efficiency in remanufacture (and therefore Design for Remanufacture) is required in situations where Design for Remanufacture is

appropriate. This requires examples of successful remanufacture and the role of Design for Remanufacture in achieving it.

The strategies useful in implementing sustainability into environmental management systems (EMSs) and

'During the design process, the end-of life option (reuse, remanufacture, recycle, or treat and dispose) for each module, component, etc. should be evaluated... The end-of-life assessment should be an iterative (or synchronous) process with product and manufacturing system design.'

Nabil Nassar and Michael Thurston, 2006

business model design for sustainability are well-documented in other publications and as such, are not explored fully in this report.

8 What is Design for Remanufacture?

Design for Remanufacture is a combination of design processes whereby an item is designed to facilitate remanufacture. Design for Remanufacture is guided by an assessment of product or component value over time. This value may vary

depending on the market and material demand and supply, legislation and technological improvements.

As part of a business model, Design for Remanufacture can reduce the constraining issues and preserve the facilitating properties for remanufacture. Design for Remanufacture therefore can play a part in optimising the process of remanufacture. Design for Remanufacture requires two stages:

- Evaluation of the product and its relation to the remanufacturing process to discern what Design for 'X' methods are required. This may be done using the criteria given in Section 4 'Parameters for Successful Remanufacture'.
- Application of the selected design techniques to optimise the process of remanufacture.

Design for Remanufacture incorporates these characteristics into particular design steps or Design for X strategies, which are described in this section. These steps are related and frequently overlap. They are considered as separate here for the sake of clarity. The Table below links the Design for X strategies with their corresponding remanufacturing steps (including core collection).

DESIGN STRATEGY	REMANUFACTURING PROCESS						
	Core Collection	Inspection	Disassembly	Cleaning & storage	Remediation	Reassembly	Testing
Design for Core Collection	√	√					
Eco-Design		√	√	√		√	
Design for Disassembly		√	√	√	√	√	
Design for Multiple Lifecycles				√	√		
Design for					√		

Upgrade							
Design for Evaluation n		√					√

Depending on the product, Design for Remanufacture may concentrate on a particular problematic stage of the remanufacturing process e.g. disassembly which may make remanufacture of a particular product uneconomical. However taken to the full implication of the phrase, Design for Remanufacture can optimise the entire process and system of remanufacture.

8.1 Design for Core Collection

Core return relies on both the business model and detailed product design. The business models of Xerox and Caterpillar for core collection will be described in later sections.

Detailed product design can also visually communicate end-of-life processes. For example, this may include labels positioned on packaging, the product's exterior or inside the product. The positioning of the label will depend on the intended viewer, how they may view it and how they will use the information. Labels may also incorporate radio frequency identification (RFID) to allow a vast array of information to be held.

Graphical communication, packaging or even the form of a product may also be able to communicate collection. The basic concept of using graphics to simply communicate product information is demonstrated by Perkins guide for identifying engine type:

An example of visual communication of product information at Perkins

Engine Number Location Guide

(Left Side View) (Right Side View)

(Left Side View) (Right Side View)

(Right Side View)

Engine Type	Position
Eagle/2000 Series	R
3008 Series	T
3012 Series	S
2300/2800 Series	W
4006 Series	U
4008 Series	U
4012 Series	V or X
4016 Series	V or X

Year of Manufacture Code

This code indicates the calendar year of manufacture. The letters I, O, Q, R and Z will not be used.

A 1974	M 1985	B 1996
B 1975	N 1986	C 1997
C 1976	P 1987	D 1998
D 1977	S 1988	E 1/1/99-31/3/99
E 1978	T 1989	F 1/4/99-31/12/99
F 1979	U 1990	G 2000
G 1980	V 1991	H 2001
H 1981	W 1992	J 2002
J 1982	X 1993	K 2003
K 1983	Y 1994	L 2004
L 1984	A 1995	M 2005

Country of Manufacture Code

This code indicates the country in which the basic engine was manufactured.

A Argentina	HM Indonesia	P Poland
B Brazil	HK Iraq	S India
C Australia	HU Uruguay	SA South Africa
D Germany	J Japan	T Turkey
E Spain	K Korea	U UK
F France	L Italy	V Pakistan
G Greece	M Mexico	W Iran
H Group	MX Mexico	X Peru
HC China	N USA	Y Yugoslavia

© Perkins Engines Co Ltd Peterborough PE1 5NA UK
Parts Publication Number PPe27/EY5

8.2 Eco-Design

The aim of integrating environmental considerations into product design and development is to reduce products' adverse environmental impacts throughout a product's lifecycle (BSI, 2002) and is detailed by ISO TR 14062. Specifically, Design for Remanufacture suggests designing with a product's end-of-life in mind, however there are lifecycle considerations that may also impact on the process of remanufacture. Examples of possible eco-design approaches that may impact on Design for Remanufacture are (BSI, 2002) shown in the following sections 7.2.1-

7.2.6. Eco-design's backbone is lifecycle thinking and management, however examples of specific eco-design strategies which can support lifecycle thinking are shown in sections 7.3-7.6.

8.2.1 Improvement of Materials' Efficiency

This technique asks whether environmental impact can be reduced by minimal use of materials, use of low impact materials, use of low impact materials, use of renewable materials, and/or use of recovered materials. The specification of minimal materials may reduce time and complexity of remanufacture.

8.2.2 Improvement of Energy Efficiency

This technique considers total energy use throughout the product's lifecycle, and questions whether it is possible to reduce energy in use, use low impact energy sources, use energy from renewable sources. Design for Remanufacture may specify materials which require low energy in manufacture and may also reduce the energy costs of remanufacture. Business models Designed for Remanufacture may specify the use of renewable energies in remanufacture.

8.2.3 Reduced Land Use

This aspect of eco-design may be particularly relevant when land-consuming infrastructure is utilised in the product system. Both detailed product design and business models may investigate reducing the physical footprint of a remanufacturing facility to reduce costs and increase production efficiencies. Reduced land-use may result in smaller remanufacturing facilities and therefore reduced costs for the remanufacturing operation.

8.2.4 Design for Cleaner Production and Use

This aspect promotes the use of cleaner production techniques, avoiding the use of hazardous consumables and auxiliary materials and using an overall systems perspective to avoid decisions based on a single environmental criterion. The avoidance of hazardous chemicals in production may eliminate the need for such chemicals in remanufacture. The avoidance of potentially hazardous substances in the product may protect human health, safety and environmental aspects during remanufacture.

8.2.5 Design for Optimising Functionality

Considering opportunities for multiple functions, modularity, automated control and optimization may promote remanufacture by the use of common and/or modular parts.

Design for Durability is another aspect of Eco-design and will be considered in the following sections.

8.2.6 Design for Reuse

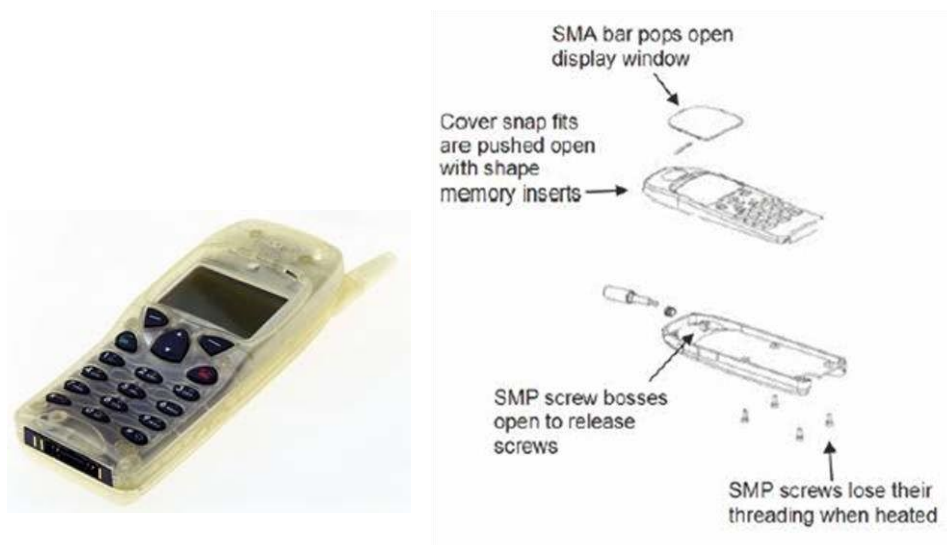
Design for Reuse, Recovery and Recycling may be the broad eco-design approach that most clearly relates to a product's end-of-life and may therefore increase the speed and reduce the costs of remanufacture is Design for Reuse.

Design for Disassembly is likely to be an essential component of any efforts to recover intact components and materials; it is therefore an important aspect of Design for Remanufacture.

8.3 Design for Disassembly

Design for Disassembly is an important enabling component of end-of-life procedures. The elimination of hazardous materials is an aspect of eco-design that facilitates Design for Disassembly and remanufacture by reducing the safeguards required. Aside from those specified in legislation, various 'black', 'grey' and 'white' lists of materials are freely available on the internet from organisations such as Volvo.

Active Disassembly using Shape Memory Alloys and Shape Memory Polymers

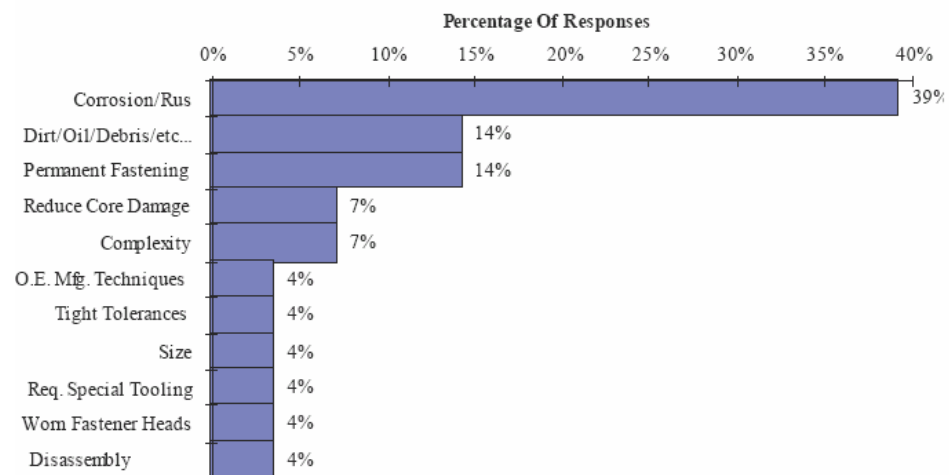


The impact of Design for Disassembly on assembly and reassembly times is contentious with some sources citing it as slowing assembly times and others saying it reduces time required for assembly. Essentially in terms of remanufacture, Design for Disassembly enables the removal of parts without damage and can therefore reduce the remediation process and requirements for new, replacement parts.

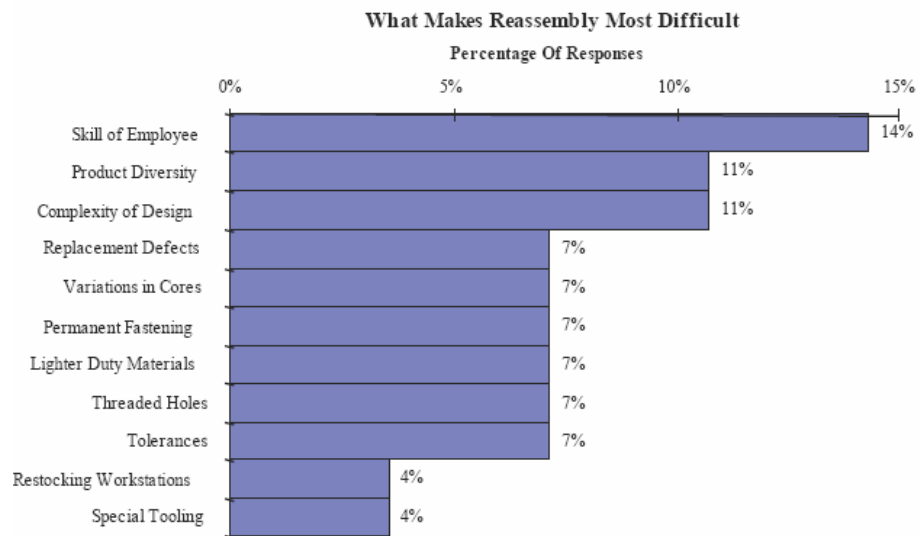
Research into disassembly is being done by a few organisations in the UK such as Active Disassembly Research Ltd. The Nokia Research Centre is also investigating disassembly mechanisms and has developed a prototype phone with a built in "active disassembly" mechanism, designed to aid the recycling of materials at the end of the phone's life (Kurk and McNamara, date unknown).

In the Automotive industry, Hammond et. al (1998) identify that corrosion/rust is the biggest hindrance to disassembly. From a detailed design perspective, the problem of corrosion could be helped by, for example, better isolation of parts from the

elements, the specification of non-corrosive materials or the use of different fastening mechanisms to name.

A survey of Electrical Rebuilders**What Makes Disassembly Most Difficult****Figure 3 – Responses to Question 3: Difficulties To Disassembly**

The lack of skills is the biggest hindrance for reassembly in the Electrical Rebuilders section of the automotive industry (Hammond et. al, 1998). This could suggest employees need greater training, but also makes the case for simpler product designs leading to more intuitive reassembly procedures.

A survey of Electrical Rebuilders**Figure 7 – Responses to Question 7: Difficulties to Reassembly**

8.4 Design for Multiple Lifecycles

Design for Multiple Lifecycles includes consideration of:

- Cleaning
- Product reliability
- Durability
- Remediation

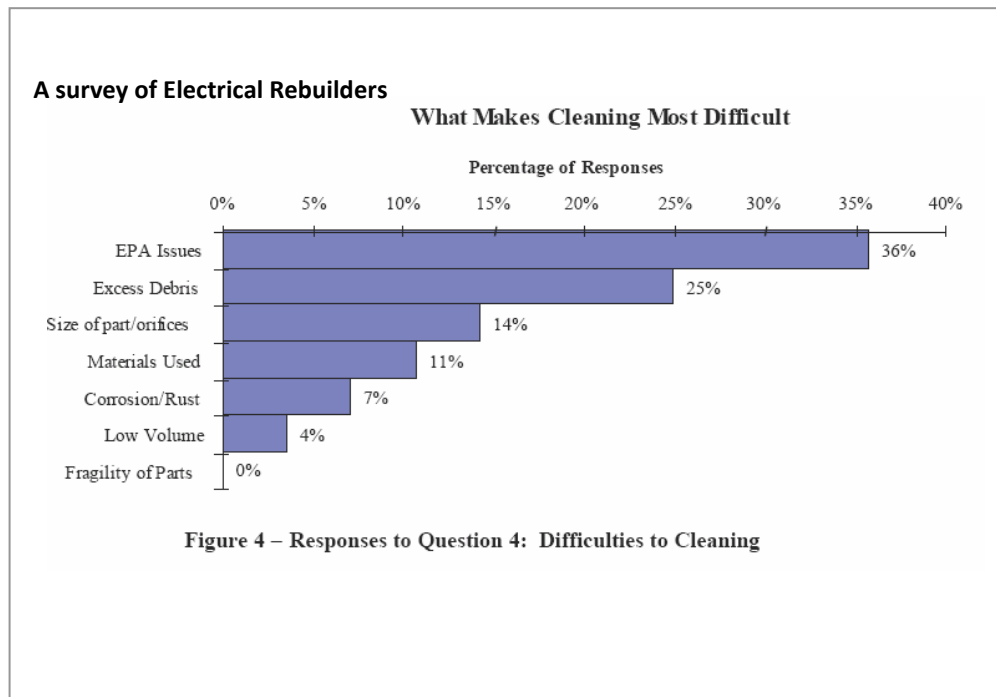
However Design for Multiple Lifecycles is not necessarily required for all products or components. Nasr and Thurston (2006) argue that some components may be designated, by design, for single or multiple reuse, for single or multiple remanufacturing, for recycling, or for disposal. Depending on the product sector, multiple life spans may infer product upgrade. In this report, Design for Upgrade will be considered separately from Design for Multiple Lifecycles.



8.4.1 Design for Cleaning

The component's physical form should enable appropriate cleaning. Such physical structures may mean that materials do not accumulate on the part during its lifecycle, resulting in better performance and reliability. In some product applications this could be actually be negative in terms of a particular component's functionality; in these cases the designer must make a judgement.

The graph below suggests the priority areas that Design for Remanufacture may tackle regarding cleaning in the automotive industry.



8.4.2 Design for Reliability & Durability

Design for Reliability includes specification of materials fit for purpose, and also communication to the user, or a person acting for the manufacturer, when testing or check-ups are required. This communication may be done by a system of manual notification e.g. by a letter that is automatically sent out

‘Environmentally, there is a need to decrease the duration of remanufacture and the frequency a product is remanufactured.’

Don Giuntini, 2006

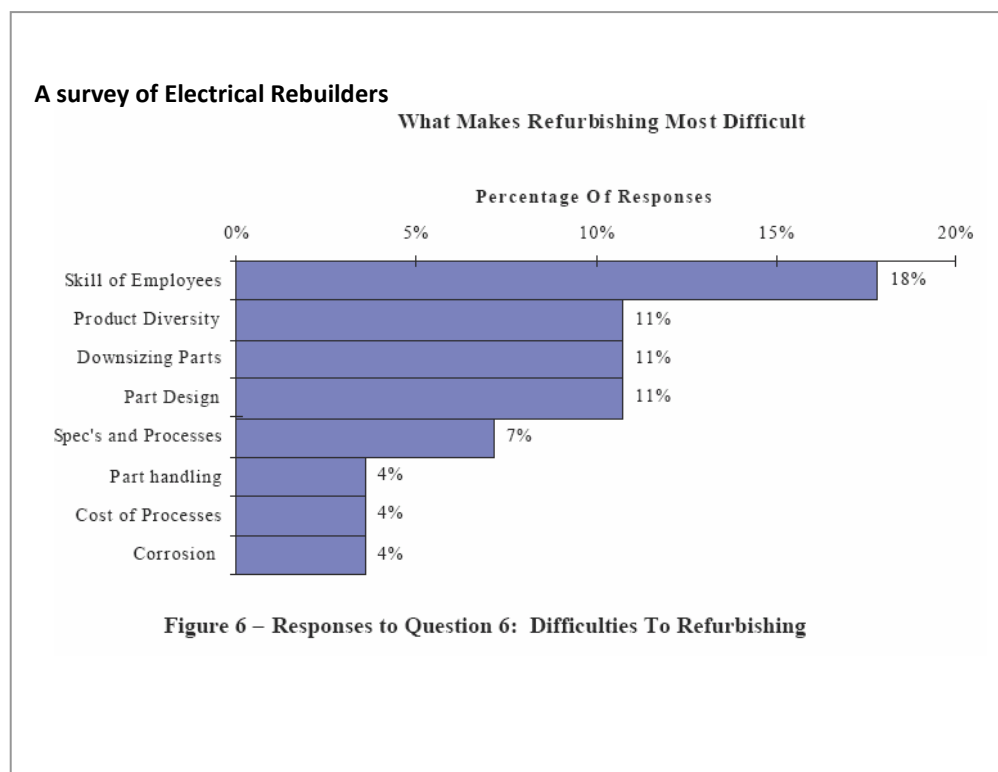
according to the product’s average life span. A more product-personal and technologically sophisticated method may be available, depending on the product and its usage e.g. automotive mechatronics where the product may record its life and history and report back to an engineer that it may need replacing.

Even though remanufacture generally uses considerably less energy than manufacture, if a product or component is more reliable it will require remanufacture less often and therefore less energy will be used; giving an environmental saving.

Design for Durability requires the specification of materials appropriate to intended usage over the lifetime of the product however these materials may not themselves have to last the full lifetime of the product, but in this case they should be selected to enable remediation.

8.4.3 Design for Remediation

Design for Remediation will include the specification of component materials and consideration of other materials likely to be encountered in the lifespan of the product e.g. oils. Where applicable, cleaning substances for those materials should also be identified and specified simultaneous to the design process.



8.5 Design for Upgrade

Design for Upgrade is applicable in cases where components or products have a market life that brings them into the realms of technological change. As such, Design for Upgrade may be considered a component of Design for Multiple Lifecycles. The increasing rates of technological/aesthetic change in the consumer markets, make design for upgrade particularly relevant to remanufacturing in this sector. Upgrading product function to meet customer requirements can prolong the functional life of products (Sundin, 2004).

Although Design for Upgrade is very closely related to Design for Multiple Lifecycles, for clarity, it is useful to consider Design for Upgrade separately. Design for upgrade comprises:

- Platform Design
- Design for Optimum Life



8.5.1 Platform Design

'Platform' or 'Modular' design is used to cluster components with similar technical and market life. In platform design, essential components can be grouped in platforms and those that are likely to become faulty or defunct can also be grouped; creating platforms upon which the next generation can be based.

In order to preserve the useful aspects of components, platform design may actually require more components rather than less, which would allow defunct aspects to be removed whilst retaining the useful aspects.

8.5.2 Design for Optimum Life

Design for Optimum Lifespan requires communication to the customer of when the optimum lifespan has been reached. 'Optimum' may be used in the context of environmental impact, functionality and/or profit and may apply to one or more products, however the three areas are linked and can to an extent run in parallel.

In contrast, Harrison (2006) states 'Ideally all the redundancies should happen at the same time.' This would ensure that the full value had been achieved from the product over

'Other components that may incur variable levels of degradation during product use cannot necessarily have a predetermined end-of-life strategy.'

These components must first undergo a condition assessment process to determine the residual life and value.

For these components, the end-of-life decision criteria should be spelled out during product development.'

the lifespan, and enable all of the components to be remanufactured at the same time.

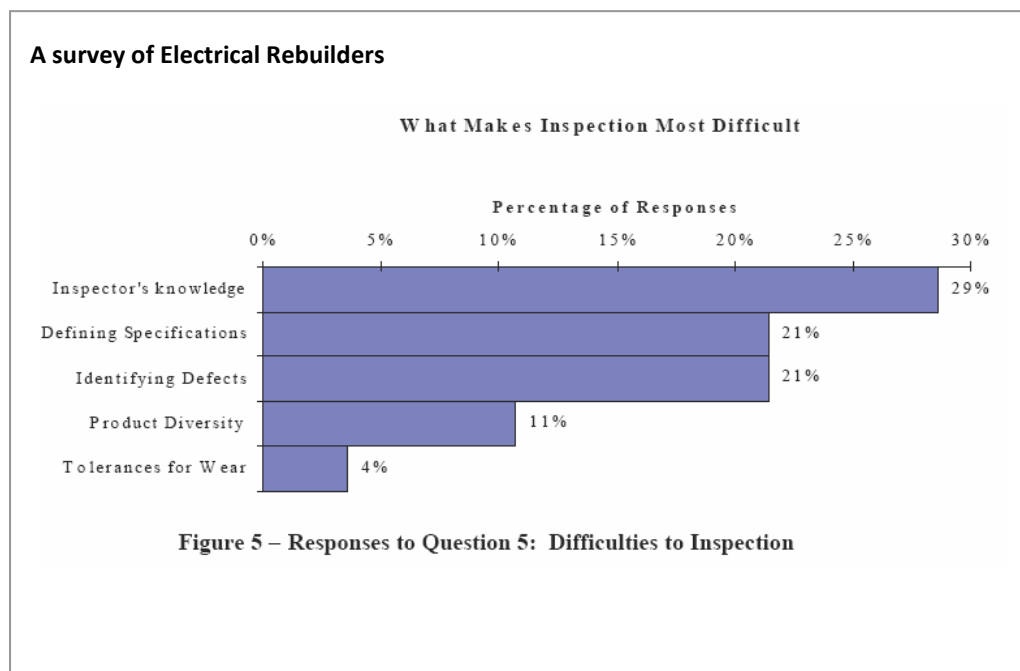
Examples of how Optimum Life may be communicated to the user may be:

- **Functional Obsolescence.** A single key redundancy could happen to bring the product in for remanufacture. In this case the other components could just be reused.

- Aesthetic obsolescence. In consumer markets, it may even be possible to use aesthetic obsolescence as a means of communicating lifespan for remanufacture.
- RFID may be used to inform the customer or supplier that replacement / upgrade is required.

8.6 Design for Evaluation

The inspection of a product to establish its current status and history is necessary in order to apply appropriate methods for remediation.



In Hammond, et. al, (1998), the inspector's knowledge of the inspected parts and required quality was a big issue. Other aspects such as the ability to define specifications could be resolved by legislation such as the Freedom of Information Act, or detailed design activities.

Detailed product design for evaluation may provide information regarding embedded condition or usage monitoring. This could be done through:

- Embedded data recording, or RFID which may be updated during product servicing etc.
- Sacrificial parts. The application of sacrificial aspects to a component, which would wear during use and thereby give an indication of the components' treatment over time.
- Traditional data recording of the products' multiple life history. Xerox for example keep accessible data of all their products previous lives.

8.6.1 Six Sigma

Six Sigma techniques allow remanufacturers to guarantee the quality of their remanufactured goods and are oft cited by remanufacturers as being applied to their

remanufacturing operation. Six Sigma focuses on the processes themselves, ensuring that they are carried out better and faster and consequently, at lower cost.

9 Overview of the leading UK organisations and practitioners

Remanufacturing is practiced by OEMs, sub-contractors and independent remanufacturers to some extent in each industrial sector.

OEMs are starting to engage in remanufacturing, often through 3rd parties. Five years ago most OEMs were not interested in remanufacturing, and may have described remanufacturers as 'pirates' but now OEMs are seeing the value (Giuntini, 2006).

'the product developer/manufacturer can get the most benefit from sustainable design practices by maintaining product responsibility throughout the product life cycle.'

Michael Thompson, 2006

However as previously mentioned, OEMs in the USA, account for less than 5% of total remanufacturing activity (Guide, 2000, cited in Sundin, 2004). It should also be noted that many OEMs (such as the B2C automotive industry) will not publicise their involvement in remanufacture to customers; this poses difficulties in understanding the real practice of remanufacture.

OEMs engaged in remanufacture are likely to encourage the use of 'genuine' parts remanufactured by the OEM themselves e.g. Perkins Engines.

9.1 Xerox

Xerox's comprehensive process for taking back 'end of life' photocopiers, printers, scanners from customers was established in the early 1990s, creating a remanufacture and parts reuse programme that forms the foundation of waste free initiatives.

Xerox take advantage of remanufacture through the design of their business model, making them global leaders in this field. Xerox's Production Systems Group (PSG) work to strict corporate targets e.g. in 2003 95% of all waste produced was recycled, which equates to 175,000 tonnes and increases year on year.

'The very nature of [Xerox's] business is to reuse existing components.'

Xerox, date unknown a

Remanufactured output now accounts for 25% of Xerox's output despite double the labour cost compared to new equipment. Globally, it is estimated to have saved \$200m and recovered over 25,000 tonnes of materials from landfill. Xerox have some design functions in Welwyn Garden City, UK, but design for the global market.

9.1.1 Remanufacture

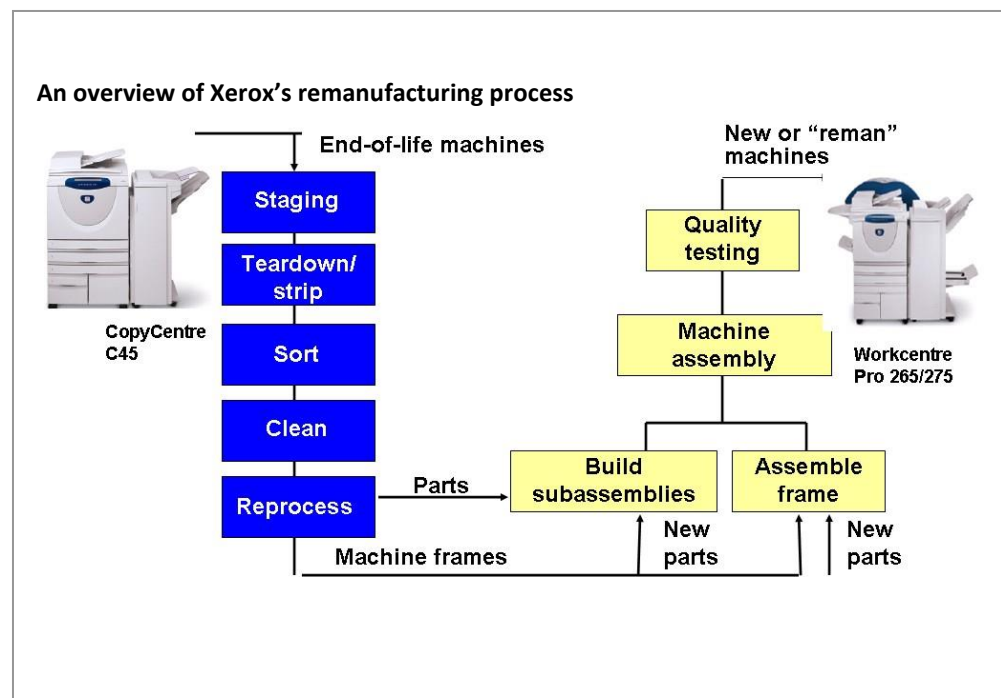
At the Xerox Dundalk, Ireland site, remanufacture and 'new' production is not split making it difficult to ascertain the exact profitability of remanufacture. However it is safe to assume that remanufacturing at Xerox is a profitable business, where Xerox are able to generate up to 7 revenue streams from 2nd, 3rd, 4th, 5th, 6th and 7th life products. It is also notable to say that consumables are more profitable than original products (Cosgrove at CfSD, 2006).



Xerox use 3rd party remanufacturers where it is more efficient to do so

depending on location and Xerox's business priorities in that area. Xerox predominantly out-source to Flextronics. Flextronics only remanufacture one product in Europe but remanufacture the majority of Xerox products in the USA (Cosgrove, 2007b).

The remanufacturing process starts with the business model which leases machines to customers rather than selling them. Xerox takes old machines back from customers after 5 years or more, depending on the number of copies the machine has taken.



Once a product reaches the Xerox remanufacturing plant it undergoes the remanufacturing process (Xerox, date unknown a)

The products to be remanufactured are taken to the CO₂ spray booth where they are blasted using the principle of dry ice and compressed air.



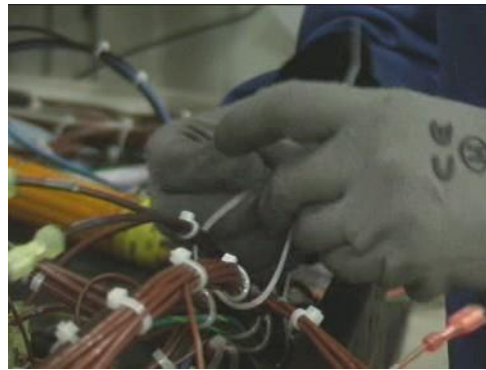
The cleaned carcasses are then taken to the spray booth where they are re-sprayed as new.



Other smaller parts are put through an ultrasonic cleaning process. They are rinsed in several tanks containing a solvent while also being bombarded with ultrasonic.



These are stripped down into up to 8,000 individual parts. Up to 80% are reused and those which are not are recycled.



Skilled and trained people take these parts and build up the new carcasses from scratch.



The final stage of the process is securing the panels onto the carcass.



Xerox's drive to remanufacture is driven by marketing and asset recovery. Collection of used equipment remains a key question for Xerox.

9.1.2 Design

Xerox's Design for Remanufacture process starts with strategic intent. Lessons can be learnt from their approach. The company or organisation must evaluate the feasibility of remanufacture and identify it as a strategic enabler. This applies to many elements:

- Cost
- Pricing
- Customer requirement
- Environmental benefit
- Product diversification
- Market positioning
- Asset management, etc.

'Xerox products are designed to reduce environmental impacts in all phases of the life cycle. We design our products for remanufacturing and design imaging supply items for return, reuse and recycling. These early decisions mean that once the product's supply item or the product itself has reached its end-of-life, we maximize the assets and minimize the waste.'

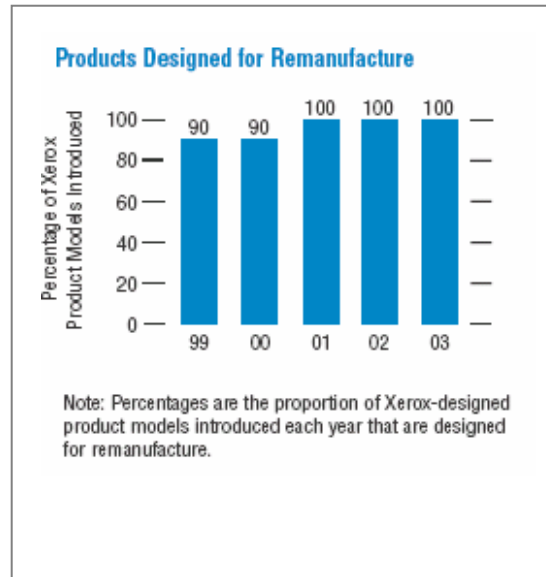
Jack Azar, Xerox VP for Environment,
Health & Safety, First Sustainability Report 2004

Once the business model is defined, the design teams can design the products with the full life cycle requirements understood.

Machines at Xerox are designed for disassembly and contain fewer parts because of this. Parts are also designed for durability over multiple product lifecycles. As a result each new generation of Xerox products offers increasing functionality while conserving energy and materials, and requiring fewer hazardous substances throughout the product lifecycle. (Xerox, date unknown)

Since the early 1990's the strategy within Xerox has been to remanufacture certain product platforms. The designers use a common platform approach to ensure that the main engine of the

machine and peripheral equipment is common across many different models. This ensures the minimum amount of unique parts and obsolescence. The production systems will have a life cycle of about 15 years but during this time the equipment may be remanufactured many times, this will include remanufacture to the same as specification or in many cases capacity/software upgrades to a better than specification. (Cosgrove at CfSD, 2006).



9.2 Perkins Engines

Perkins Engines make and remanufacture diesel engines and transmissions for armoured fighting vehicles, mobile diesel generator sets, naval and marine propulsion engines and generator sets,

construction equipment.

Perkins Engines was formed by Frank Perkins in 1932 in Peterborough, UK. Perkins Engines works in the industrial plant sector and is located in Shrewsbury, Shropshire, UK. The company was sold to Massey-Ferguson in 1959 and later became part of LucasVarity. In 1998

'Perkins engines have earned a reputation for ease of service along with low costs of maintenance and repair.'

Their rugged and simple design enables local repairs to be made with a large global network of distributors.'

Caterpillar Inc. purchased the Perkins Engines Company Ltd from LucasVarity Plc for

\$1.35 billion. Perkins and Caterpillar engines are in service in both combat and logistics vehicles with over fifty armies around the world (www.army-technology.com, 2006).

By buying an existing plant in the UK, Caterpillar were able to enter the European market. Having a facility closer to its European customers offers the following benefits to Caterpillar:

- Significantly improves Caterpillar's response time
- Allows more customised products to meet regional customer needs
- Reduced transportation costs

Operations in Shrewsbury are beginning with the remanufacture of Caterpillar and Perkins diesel engines and components from all over Europe, Africa and the Middle East.

9.2.1 Remanufacture

In January 2004, the Caterpillar Perkins plant in Shrewsbury switched from manufacturing new diesel engines to remanufacturing. To give the scope of the impact of Perkins Engines, worldwide one in five tractors is powered by a Perkins engine. A multi-million pound factory re-fit was carried out including substantial investment in new machinery.

9.2.2 Design

The business model of Perkins Powerparts has been designed so that all products carry a 12-months warranty from date of purchase, which Perkins' website describes as 'unequivocal'. The warranty will indemnify the purchaser for the following costs associated with making the repair.

Detailed design is carried out by Caterpillar in the USA. The geographic separation of detailed design and remanufacture could potentially pose communication difficulties.

Packaging design emphasising Perkins' genuine parts



9.3 Milliken Carpets

Milliken are the first major carpet manufacturer to offer a true remanufacture capability on a carpet system. This has required the development of a unique modular carpet product. Milliken sell their product globally.

9.3.1 Remanufacture

Milliken's remanufacturing operation is driven by the market and legislation. Remanufacturing at Milliken was developed initially as the result of the vision of the technical director of Milliken Carpet in the USA, in anticipation of social and business environment trends concerning reuse and recycling, and the need to reduce resources. Observation of other industries being subject to increasing legislative constraint on their activities led Milliken to consciously adapt its manufacturing technology and range to enable its own products to be recycled.

'Life-cycle costing of the remanufactured product [Milliken's] shows clear benefits over newly manufactured items. Even better, the margin on the remanufactured product, despite the lower market price, is still very attractive.'

www.sustainablecarpets.com, 2006

However, remanufactured products currently account for only a small percentage of Milliken's sales.

9.3.2 Design

Milliken's remanufactured carpet range has been designed so that unwanted carpet tiles can be recovered, cleaned, retextured, and restyled in preparation for reuse, either within the same company or at another location. The cost is about half the cost of new carpet (Blum-Evitts, 2000). Milliken have also developed a glue-free carpet installation system called TractionBack™ which is available on all modular carpet products manufactured by Milliken. This is a high-friction coating applied to the backing of the modular carpet tiles. The coating allows the tile to be installed without using wet adhesive or "peel and stick" adhesives. The absence of adhesive reduces contamination and makes reprocessing easier. Milliken's remanufacturing process allows the carpet to stay intact, eliminating carpet waste.

9.4 InfoTeam

9.4.1 Remanufacture

InfoTeam International Services became the main assembly and repair centre for Sony UK and Sony Entertainment Europe in 1996. It has succeeded in retaining this prestigious contract, an achievement in the competitive outsourcing market. InfoTeam are currently remanufacturing a Sony games console. Sony have an interest to remanufacture the units and sell them to secondary markets because every game sold is additional profit (Leppard, 2006).

9.4.2 Design

InfoTeam are involved in the remanufacture of others' products and as such are not involved in the design of those products.

10 International Remanufacturing Activities

International differences in the practice of remanufacturing are driven by numerous factors including geography and culture. The historically independent countries and languages which make up Europe mean that OEM products are not spread evenly.

10.1 Key International Remanufacturing Practice

10.1.1 US Military

The US military is the biggest remanufacturer in the world (Giuntini, 2006).

Remanufacture

An example of the US military's eagerness to take advantage of the benefits of remanufacture was demonstrated in June 2005, when BAE Systems was awarded a series of delivery orders and contract

modifications worth \$1.127 billion from the U.S. Army Tank-automotive and Armaments Command (TACOM). The contract was to remanufacture and upgrade more than 500 Bradley Combat System vehicles.

'The military don't care if the product is 'new' or not.'

Ron Giuntini, 2006

Design

The extent to which the US military engage in feedback between design and remanufacture is unknown.

10.1.2 Caterpillar

Caterpillar's global remanufacturing business is currently one of the largest in the world in volume terms, recycling more than 50,000 tonnes of products (over 2.2 million end-of-life units) each year. Through their large scale activities, Caterpillar have helped change the business of remanufacture (Giuntini 2006).

'President Bush Supports Remanufacturing...

In cooperation with relevant international organizations such as the OECD, we will seek through this initiative to...Reduce barriers to the international flow of goods and materials for recycling and remanufacturing, recycled and remanufactured products, and cleaner, more efficient technologies, consistent with existing environmental and trade obligations and frameworks'

Remanufacture

Caterpillar first entered remanufacturing in 1972 due to a favour Caterpillar reluctantly did for Ford Motor Company: to lower its own costs, Ford's truck-making subsidiary wanted a source of rebuilt engines, which generally sell for half

the price of new ones. Caterpillar's first remanufacturing plant was quickly overworked and Caterpillar soon realised the business benefits.

Caterpillar Remanufacturing Services is one of Caterpillar's fastest growing divisions - annual revenue is over \$1 billion and is reputedly growing at 20% a year.

In 2005, Caterpillar spent \$1.5 billion on purchasing remanufacturing facilities around the world so that Caterpillar could remanufacture in those markets. Caterpillar now has remanufacturing facilities across the globe, including Shrewsbury, UK and Nuevo Laredo, Mexico; Shanghai, China was the 14th facility and opened in early 2006. Caterpillar has been operating in China for almost 80 years. In 2005, its remanufacturing division generated over \$1 billion in revenue.

Design

This revenue is in part preserved by Caterpillar branding. Any part which is remanufactured by Caterpillar is given a designed “reman” logo. This branding distinguishes Caterpillar remanufactured parts and those of 3rd parties, thereby preserving trust and creating ongoing demand for Caterpillar-branded remanufactured products.

By engaging in remanufacture, Caterpillar can increase the possibility of achieving feedback from remanufacture and design of the original products.

10.1.3 Flextronics

Flextronics offer full-spectrum design, manufacturing, remanufacturing and logistics services vertically integrated with world-class component capabilities to help OEMs create market leading products. Flextronics remanufacture only a very few of Xerox’s products in Europe, but in the USA they remanufacture a large proportion of Xerox’s products.

Remanufacture

Flextronics’ Reverse Logistics expertise lies in:

- Consumer and midrange products
- High volume, with low to medium complexity
- Printers
- PDAs
- Mobile phones
- Consumer medical devices
- Notebooks
- PCs
- Set top boxes
- Highly complex infrastructure products

Examples of the services Flextronics provide in the area of remanufacturing include:

- Component warranty management
- Part harvesting
- Failure analysis
- Spares planning

- Remanufacturing
- A to B replenishment

Speed of service delivery is critical in the After Market sector. To this end, Flextronics provides same day turn around on repair of products such as notebook PCs. Throughout all of Flextronics Logistics' operations, they are committed to continuous improvement driven by a Six Sigma platform.

Design

By offering integrated reverse logistics solutions Flextronics can provide feedback loops into design, engineering, assembly, distribution and customer service.

10.1.4 Hanover

Hanover is a supplier of new, used, remanufactured, OEM and aftermarket parts and equipment for maintenance, repair and overhaul of equipment for natural gas compression. More research into Hanover's remanufacturing activities is required.

'[Hanover] is probably the most intensive remanufacturer in the world'

Ben Giuntini, 2006

10.1.5 Siemens AG Refurbished Systems – a comparative example

Siemens Medical Systems refurbish (not remanufacture) some product lines. This example is given to demonstrate that the importance of customer assurance of quality applies to all elements of 're' and is not unique to remanufacture.

'During comprehensive refurbishment of used systems, only original parts are used. We apply the same strict quality standards as for new systems'

Hans-Peter Seubert, Manager, New

Since 1998, Siemens Medical Systems has offered to take old devices back from its customers for the process of refurbishment. The 'refurbished'

Siemens product is likely to be sold with a reduced warranty and would not be considered to meet the quality standards of a new Siemens product.

In 2000, the Refurbished Systems product group was established to market used systems. In the past year, Siemens say nearly 800 refurbished systems with the "Proven Excellence" quality seal have been sold worldwide.

The Siemens Medical Solutions Refurbished Systems division offers



refurbished medical equipment from several different modalities. Through the Siemens “Proven Excellence” programme, all systems undergo an in-depth refurbishment process to ensure the high quality which they claim only the original manufacturer can offer. These refurbished systems are offered to customers world-wide.

10.2 Key International Research Expertise

Design for Remanufacture where practised at all, is likely to be limited to a small number of OEMs who engage in the remanufacture of their own products and therefore have an incentive to increase the efficiency of remanufacture. Information on international Design for Remanufacture activities is therefore difficult to access.

There is research occurring as to how the remanufacturing process may be optimised for specific products e.g. by Rolf Steinhilper at The European Research Center for Remanufacturing, there appears to be little work being carried out on Design for Remanufacture.

As an example of international activities, a brief overview of some international research expertise is given here:

10.2.1 UK

Organisation: Oakdene Hollins Ltd

Aim: Supported by Defra, OHL aim to act as a focal or coordination point for the understanding of remanufacturing issues. OHL aim to target research going forward and offer operational support to initiatives that have a direct impact on remanufacturing practices.

Sector: Non-sector specific

Leader: David Parker

10.2.2 Germany

Organisation: European Research Center for Remanufacturing, Bayreuth,

Germany. Sector: Car Mechatronics

Leader: Professor Rolf Steinhilper

10.2.3 USA

Organisation: US National Center for Remanufacturing, Rochester, N.Y.,

USA. Sector: Office & Industrial Electronics

Leader: Professor Nabil Nasr

10.2.4 China

Organisation: National Key Laboratory for Remanufacturing, Beijing,

China Sector: Mechanical & Surface Treatment

Leader: Professor Zhu Sheng

10.2.5 South Africa

Organisation: National c/o Life Cycle Engineering University, Pretoria, South

Africa Sector: New markets for remanufactured products

Leader: Professor Alan Brent

10.2.6 Korea

Organisation: Korea Institute of Industrial Technology (KITECH) & Korea National Cleaner Production Center (KNCPC)

Sector: Automotive, electronics, etc

Leader: Unknown

10.2.7 International

Organisation: Fraunhofer - Institute for Experimental Software Engineering (IESE)

Sector: Processes, methods, and techniques for developing software-based systems according to engineering-style principles

Leader: Unknown

11 Drivers of Remanufacture

An understanding of the drivers of remanufacture gives insight into how business models may be Designed for Remanufacture. The two key drivers of the remanufacturing industry are market demand and legislation; of these two drivers, market demand is the major driver.

Drivers vary depending on whether the remanufactured item is a spare part or product, and their market (B2G, B2B or B2C). The impact of some drivers is unclear, particularly legislation such as the Waste Electrical and Electronic Equipment (WEEE) Directive, which may actually pose barriers as well as theoretically drive remanufacture. Distinct challenges will be discussed in the following section, whereas those that may be both a driver and a barrier are discussed here.

11.1 Market demand

Market demand may vary depending on the product type or market being considered. Design for Remanufacture (particularly business model design) must take the market into consideration.

11.1.1 Price

As in other industries, the more established the market, the higher the price for remanufactured products. Although remanufactured spare parts could be (and sometimes are) sold at 30-70% of the

cost of new, remanufacture is shown to work best in markets when there is no discount for remanufactured items compared to new. In these cases the

market has accepted the remanufacturer's offer that its goods are as good as new. The supply of core is so vital to remanufacturers

that often in these markets, remanufacturers (such as Caterpillar) initially pay for cores in order to secure a good flow of high quality cores (Morley, 2006).

'Remanufacturing is generally less successful when remanufactured products are sold at a large discount to new.'

Nick Morley, 2007

The generally reduced production costs of remanufacture mean that remanufacture is a profitable industry, although as previously discussed the evidence for this is predominantly anecdotal. For example, Steinhilper (2006) recalls a story about a company who began remanufacturing vending machines and selling them at the same price as new, and doubled their profit margin.

11.1.2 Purchase Considerations

Procurement in B2G and B2B transactions is likely to involve a greater consideration of cost than B2C issues. Quality will

also be considered by these markets in particular, in terms of fitness for purpose. Although a remanufactured product would by default be as fit for purpose as a new product, the time and thought required by purchasers and the potential obstacles posed by others, may make the simplicity of 'business as usual' i.e. buying a known 'new' product, the easiest option.

Buying 'new' is particularly prevalent in B2C markets where misconceptions and concerns over the quality of '2nd hand' products are diffuse. This makes B2C markets especially

resistant to remanufactured products. Design for Remanufacture, may be able to contribute to changing this situation, particularly through branding.

Where the purchase is *forced* rather than *chosen* such as in the case of a spare part required to repair a broken down car, cost is a more significant deciding factor, because the expense was not planned. In this case the customer is likely to be happy to buy a remanufactured spare part, which costs less than a new part.

However, when a product is considered valuable by the owner e.g. a vehicle which is new or only a few years old, customers are more prepared to spend extra on purchasing a 'new' spare part (Steinhilper, 2006). In contrast, to spare parts, products are generally purchased consciously after a consideration of needs/wants and costs.

Market	Basic Purchase considerations				
	Functionality	Cost	Environment	Development time	Emotive factors
B2G	✓	✓	✓		
B2B	✓	✓		✓	
B2C	✓				✓

11.1.3 Continued supply

OEMs may be obliged to supply spare parts for products for a certain length

'If [Morvend] sell something they [generally] sell new. If a customer wants a machine that isn't manufactured anymore, then remanufactured products are useful in that way.'

of time after production ceases. By designing a business model which provides these spare parts through remanufacture, OEMs are able to cease production of small numbers of new spare parts for old product lines. This means remanufacture can offer a further cost saving to the OEM.

11.2 Legislation

Legislation can be both a driver and a barrier for remanufacture. The impact of legislation on remanufacture needs to be understood to design successful business models for remanufacture. Within this, it is useful to remember that the majority of existing legislation applies to products but does not apply to spare parts.

Some legislation such as the Sales of Goods Act (SoGA) appears to constrain remanufacture, whereas it is still unclear whether other, new legislation such as the WEEE Directive will promote or hinder remanufacture. This section will describe legislation which may drive and constrain remanufacture, whereas legislation which only appears to constrain remanufacture will be described in the following section 11 'Challenges to Increased Remanufacture'.

Legislation which may have an impact (whether positive or negative) on remanufacture is:

- Landfill Directive
- Waste Electrical and Electronic Equipment (WEEE) Directive
- Restriction of Hazardous Substances (RoHS) Directive
- End of Life Vehicle (ELV) Directive
- Energy-Using-Products (EuP) Directive
- Freedom of Information Act (FoIA)

11.2.1 Landfill Directive

With increasing Landfill taxes, the advantage for producers to rely on landfill is decreasing with time. Landfill is neither an environmentally or commercially sustainable option and so producers are exploring alternative end-of-life scenarios. Remanufacture is one of many alternatives to landfill and is increasingly being seen as an effective means of generating revenue.

11.2.2 Waste Electrical and Electronic Equipment (WEEE) Directive

The WEEE Directive aims to reduce landfill and support more sustainable development by providing the impetus to boost recycling. The UK finally passed WEEE in early 2007 and it will require all products placed on

'WEEE is stopping remanufacture.'

Ray Gould, Southern Appliances, 2006

the market after 1 April 2007 to be marked with a crossed out wheelie bin symbol (Clements, 2007). The UK was one of the last two countries in Europe to implement.

The Directive mostly affects electronic and electrical manufacturers and importers. When the Directive eventually comes into force in the UK, OEMs will be required to take responsibility for treating and recycling their new products when the products are no longer wanted by their owner. The Directive applies to new products but producers will also

'We are involved in remanufacture but we see no advantage to remanufacture with regard to these Directives [WEEE]'

Dr Kieren Mayers,
Environmental Programmes Manager,

be made collectively responsible for goods already on the market.

The implications of WEEE for remanufacturing are still unclear. Some see it as a driver for remanufacture while some

believe its emphasis on recycling actually prevents remanufacture (Gould, 2006; Parker, 2007). One reason that re-use is not encouraged is that levels of re-use are difficult to record. The Government is exploring ways of recording 'reuse' and making

'[WEEE] will force a lot of equipment to be scrapped and not reused in the short to medium term especially if provenance is unknown.'

it count towards recovery targets but they are having difficulties in establishing an effective methodology (Harding, 2006).

Recharger Magazine (2002) stated that one of the most important parts of the WEEE Directive is Article 4 which says that Member States are required to ensure that producers do not prevent WEEE from being re-used '...unless such specific design features or manufacturing processes present overriding advantages, for example, with regard to the protection of the environment and/or safety

'[establishing methods for recording reuse and making it count towards recovery targets] is problematic at the moment.'

Adrian Harding, Policy Advisor
(Producer Responsibility),
Environment Agency

requirements.' Re-use in this case can relate to remanufacture and this shows that WEEE does not close the door firmly against OEMs preventing remanufacture.

Adding to current confusion on the impact of WEEE on remanufacture, last minute revisions to WEEE include three new exemptions, two of which are potentially relevant to remanufacture; these concern:

'I think the WEEE Directive could provide the tipping point – not necessarily from the outset, but perhaps over time.'

Guy Leppard, InfoTeam, 2006

- Repair/refurbishment
- Storage

Under the terms of WEEE, buildings such as retail shops which may be used to store products e.g. a washing machine returned for remanufacture, have to be registered to store 'hazardous waste' (Gould, 2006). This adds to the complications of remanufacturing.

If a product has been put on the UK market before, and is remanufactured² and put on the market again, it has no obligation under WEEE. However, if a product is sold in another country and then imported into the UK and sold as a 'refurbished' product, it will be considered a new product and so WEEE will apply (Harding, 2006).

Further information is planned for release in very early 2007 when the DTI will publish guidance to WEEE; there will be reference to remanufacturing and whether or not it incurs producer responsibility. The Environment Agency will also be publishing information in early 2007 (Harding, 2006).

²The term 'refurbished' was used by Harding (2006); in the context it was understood by the Author to have been used as a general term for re-use strategies, including 'remanufacture'.

Revisions to WEEE will be completed by 2008. These revisions potentially pose opportunities for the correction of aspects of WEEE that may be counter-intuitive to the Directive's aims to reduce waste.

11.2.3 Restriction of Hazardous Substances (RoHS) Directive

The Restriction of Hazardous Substances (RoHS) legislation removes six hazardous materials from electronics products and aims to reduce environmental impacts of waste and improve recyclability. The Directive restricts the use of a range of potentially hazardous materials in products manufactured for sale in the UK market.

By restricting dangerous substances, RoHS aids eco-design and specifically Design for Disassembly making the work of the remanufacturer safer; RoHS can therefore be seen as a facilitator of remanufacture. However markets for remanufactured products in developing countries may be negatively affected by RoHS.

In the short to medium term, RoHS will force a lot of equipment to be scrapped and not reused, especially if its provenance is unknown (Morley and Parker, 2007). The identification of provenance and components' history may be useful in preventing scrapping of potentially

'Well-meaning schemes that pass on the industrialised world's mobile phones and computers to the developing world could constitute toxic dumping by the back door, a meeting organised by the United Nations heard this week.'

Steve Freeman, 2006

remanufacturable parts. Detailed product design may be able to contribute here through Design for

Evaluation strategies as previously discussed.

11.2.4 End of Life Vehicle (ELV) Directive

The ELV Directive (2000/53/EC) is concerned with cars, vans and certain three-wheeled vehicles. The Directive was due to be transposed into national law in all Member States by 21 April 2002. The UK and most other Member States missed this deadline.

'With the current pressures on OEM's to meet environmental targets were optimistic that this [ELV Directive] will force their hand to remanufacture/reuse a greater percentage of end of life vehicle components.'

Kevin Haynes, OEM Contracts Manager,

Around two million vehicles reach the end of their life in the UK each year.

Currently, between 74-80% of the

weight of a typical ELV is re-used or recycled. European Union Member States must ensure:

- Producers limit the use of certain hazardous substances in the manufacture of new vehicles and automotive components and promote the recyclability of their vehicles
- ELVs are subject to de-pollution prior to dismantling, recycling or disposal
- Treatment facilities operate to higher environmental standards and have permits if they want to deal with undepolluted ELVs
- Certain recovery and recycling targets are met by 1 January 2006 and 1 January 2015

- Producers pay 'all or a significant part' of the costs of treating negative or nil value ELVs at treatment facilities by 2007

The ELV Directive will require vehicle manufacturers to take responsibility of vehicles and therefore potentially bring remanufacture into consideration. However the Directive suffers similar issues to WEEE - no credit is offered for remanufacture. The DTI have limited interest in promoting the remanufacture of vehicles as they see a recovery route for the largest metal fraction; business models must ensure that economics can drive recovery routes for the rest (Morley and Parker, 2007).

11.2.5 Energy using Products (EuP) Directive

The EuP Directive provides a coherent framework for continuous improvement of the environmental performance of energy-using products, which avoids the adoption of uncoordinated measures which could lead to an overall negative environmental impact (Papdoyannakis, 2006). Under EuP, products will be designed, manufactured and operated with a focus on ease of end-of-life management (Craw-Ivanco, 2006).

EuP therefore is a driver of eco-design, and end-of-life strategies which include but are not limited to, remanufacture.

11.2.6 Freedom of Information Act (FoIA)

The FoIA seeks to give individuals the right of access to information held by governments and corporate bodies. In the USA FoIA allows remanufacturers access to OEMs' design specifications allowing 3rd party remanufacturers to remanufacture to original specifications.

12 Challenges to Increased Design for Remanufacture

Aside from the normal business pressures of time and money, there are numerous challenges to remanufacture. These challenges can be roughly grouped by issue:

- Markets
- Legislation
- Design
- Business

- Knowledge

12.1 Market Demand

12.1.1 Establishing Markets

Establishing markets is key to driving remanufacture. Customer belief in remanufactured products needs to be established in order for markets to strengthen. Customer belief in a product (and therefore market strength) means that 'as new' prices can be charged, enabling remanufacturers to use their larger profit margin as leverage for core return.

12.1.2 Trade Barriers

Trade barriers mean that remanufacturers may have to procure a remanufacturing plant in a particular country in order to sell remanufactured products in that country. To give an example of this, in 2005, Caterpillar spent \$1.5 billion to have access to remanufacturers in different countries around the world.

12.1.3 The Open market

The open market provides challenges to remanufacture by introducing more competition into the market and complicating distribution/collection systems. For example, BT used to remanufacture/refurbish all of their products. This was possible because 15 years ago (when the market was closed e.g. when there was a monopoly) and only BT phone's could be used on the system. BT therefore had control over the products, delivery and collection and could refurbish the phones. Since the market opened up, this controlled system has been compromised (Chiodo, 2007).

12.1.4 Concept of 'New' and the Concept of 'Waste'

Establishing markets for products will also require a challenge to the desirability of the 'new'. There is a temptation to purchase new products rather than used (Steinhilper, 2006). This is present in all markets to an extent but particularly in B2C markets. The elements of 're' need to be marketed to demonstrate their positive characteristics and eliminate

'If [commercial customers] lease they are quite happy to [lease] a new machine. Vending machines do not cost a lot of money in the grand scheme of things so it doesn't really matter.'

Apex Marketing, 2006

misconceptions and fundamental assumptions of poor quality and low status. Financial factors also affect the decision to buy new or used e.g. a new product may cost the same or less than a used product.

Legislation can describe used products as waste and so the storage of used products can be determined to be the storage of hazardous waste (Gould, 2006). This definition of waste therefore acts as a disincentive to organisations engaging in the reuse of products. A redefinition of what constitutes 'waste' is required to enable further remanufacture (Statham, 2006). As discussed previously, legislation which defines waste (notably e-waste from developed countries) as containing hazardous substances has the potential to challenge some markets for remanufactured products. Design may be utilised to remove hazardous substances from products if they are discovered to be hazardous or unnecessary.

12.1.5 Lifecycle costs

Getting customers to consider lifecycle costs is a key challenge for all movements towards sustainability and remanufacturing is no exception. The price of new products needs to reflect the true environmental cost.

The method of payment for services such as repair also needs to be addressed. This is particularly poignant in certain sectors of the B2C market such as white goods; for a customer to repair a washing machine, they must pay cash upfront, whereas if they were to buy new, they may be able to purchase it on credit (Gould, 2006). This is a big challenge for repair and demonstrates that the payment and reward for remanufacture should be immediate.

12.2 Legislation

As described, the impact of legislation on remanufacture is varied; in particular the impact of WEEE and RoHS on remanufacture is uncertain. The Trade Descriptions Act (TDA), the FoIA and the SoGA are also hampering remanufacture to varying degrees. The legislative framework needs to be changed to facilitate remanufacture rather than hinder it. This primarily requires an understanding of how to measure reuse.

12.2.1 Waste Electrical and Electronic Equipment (WEEE) Directive

As mentioned previously, the implications of WEEE for remanufacturing are unclear. Some see it as a driver for remanufacture whilst others believe its concentration on recycling prevents remanufacture. Design of the business model for remanufacture could be used to prevent OEM objections to remanufacture by making remanufacture a more desirable option than 'business as usual'.

12.2.2 Restriction of Hazardous Substances (RoHS) Directive

RoHS may restrict markets for remanufactured products that contain substances that are now termed 'hazardous'. Design for Remanufacture, and in particular Design for Upgrade may be used to remove hazardous aspects and take future legislation into account.

12.2.3 The Trade Descriptions Act (TDA)

The TDA 1968 is an Act of the Parliament that prevents manufacturers, retailers or service industry providers misleading consumers as to what they are purchasing. The Act makes it an offence if a trader:

- Applies a false trade description to any goods; or
- Supplies or offers to supply any goods to which a false trade description is applied; or
- Makes certain kinds of false statement about the provision of any services, accommodation or facilities.

If a product has been remanufactured, this must be communicated to the customer in some way e.g. in the contract of sale. It is possible that potential remanufacturers, particularly in B2C markets, may be reluctant to advertise their remanufacturing activities because of perceived customer perceptions. In this way TDA may discourage remanufacture.

12.2.4 Freedom of Information Act (FoIA)

In contrast to the USA, in Europe there is a lack of access to OEMs' design specifications and this requires independent remanufacturers to reverse engineer products and parts to discover the specification data, adding to the development of time and investment costs of remanufacture.

This situation affects different industry sectors in different ways. Although it is possible to acquire data from the USA where the FoIA is more diffuse, the automotive industry is particularly challenged by the barriers of withheld information (OHL, 2004). It is difficult to determine incentives for OEMs to communicate product specification data to independent remanufacturers but if remanufacture by 3rd parties is to be encouraged, incentives for OEMs to reveal specifications must be established.

12.2.5 Sales of Goods Act (SoGA)

SoGA relates to remanufacture because it affects the retailers relationship with warranties. SoGA declares that when goods are bought, they must conform to contract. This means they must be as described, fit for purpose and of satisfactory quality (i.e. not inherently faulty at the time of sale). Goods are of satisfactory quality if they reach the standard that a reasonable person would regard as satisfactory, taking into account the price and any description. Aspects of quality include fitness for purpose, freedom from minor defects, appearance and finish, durability and safety. SoGA therefore has implications for warranties, and incentives to manufacturers to design quality goods.

SoGA applies predominantly to B2C transactions because it is assumed that consumers are reliant on the retailer's knowledge whereas businesses are assumed to have the ability to make its own evaluation that an item "conforms to contract".

'The rules, regulations and warranties associated with second hand goods mean that the simplest way [for retail] is to not get involved.'

Ray Gould, 2006

SoGA states that it is the retailer, not

the manufacturer, who is responsible if goods go wrong. When a consumer purchases a product, the warranty is held by the retailer not the consumer. Therefore when the consumer claims for a malfunctioning part or product, that claim is against the retailer; the retailer then has to sue the manufacturer for the amount. This means that there is no direct connection or responsibility between the manufacturer and the end consumer and therefore little incentive for the manufacturers to produce reliable products.

12.3 Design Skills and Education

12.3.1 Connecting Design and Remanufacture

There is currently little connection between product design, engineering and remanufacture. This disconnect is symptomatic of historical difficulties design has faced in making its value appreciated by business.

'They ought to be giving feedback between design and remanufacture.'

'It is absolutely essential for the European Commission to press the case for environmentally friendly design – certain OEM's spin their environment credentials while their actions prove the opposite is true'

Alan Castro, Lasercare Anglia in UKCRA,

David Harrison, 2006

In order for an effective link between design and remanufacture to occur, the potential of design to aid remanufacture needs to be better understood by all stakeholders and making both the business, as well as the environmental case is essential.

'As more companies begin to adopt sustainable product strategies, technologies and design practices that aid in sustainable product design (such as Design for Remanufacturing) will become more important.'

Nabil Nasr and Michael Thurston, 2006

OEMs control both the design and production side and as such hold the

key to connecting design and remanufacture. Despite the majority of design businesses being in-house teams, and in an ideal position to Design for Remanufacture, remanufacture is still not in the remit of OEMs who generally focus on first-life production.

12.3.2 Increase number of remanufacturable products

Not all products are remanufacturable as explained in Section 4. Consumer goods most fit the model of ‘not designed for manufacture’.

Design for Remanufacture needs to connect with non-remanufacturable products on both business and detailed product

design levels to change the products’ framework; this may enable increased remanufacture in the future.

‘The role of financial considerations would have to be applied before environmental lifespan optimisation could be applied to real-world situations.’

Connecting design with remanufacture requires recognition from remanufacturers that the remanufacturing process’ efficiency requires improvement.

12.3.3 Increase Process Efficiency

Remanufacturers need to ensure their remanufacturing process is as efficient as possible. The variety of products for remanufacture ensures that there are few rules of thumb. Typical process efficiency activities are carried out at the University of Bayreuth by Professor Steinhilper and his team. These include identifying the most appropriate cleaning materials and factory layout.

Remanufacturing with loss of original product identity encompasses some unique challenges in inventory management and disassembly sequence development. Some of the open questions relate to the commonality of parts in products of different generations, the uncertainty in the supply of used products, and their relationship with production planning (Wikipedia, 2006).

12.3.4 Education

Design for Remanufacture is sometimes defined as eco-design even by people in the remanufacturing industry. In a holistic sense this understanding might be correct if you consider the net benefits of remanufacturing activities. This confusion of terminology may show merely a relaxed use of phrases but it may also demonstrate

a misunderstanding of the specific concepts of Design for Remanufacture within the industry itself.

Design for Remanufacture needs to be built into education in both product design and engineering fields to counter this. Institutions which are currently providing education in the areas of eco-design include the University College for the Creative Arts,

'Design engineers are easy to persuade [to remanufacture] you just change the parameters!'

Ben Crompton, 2006

Kingston University, Cranfield University and Loughborough University. In addition, the knowledge of Design for Remanufacture should reach such a level that it becomes inherent to designers' practice so as to not impinge on timescales or cost.

12.4 Business

The design of business models is key to establishing markets for remanufactured products. This fundamentally requires building the business case. The case needs to be made for the supply and procurement of remanufactured products. Government has a role to lead this drive through Green or Sustainable Public Procurement initiatives which can be used to push and pull both suppliers and markets as clarified by the Simms Report (2006).

12.4.1 Business Case

The business case for remanufacturing is not understood evenly across industry sectors. Although the business case is established in certain industries (e.g. automotive, aircraft) there are opportunities for development in other areas e.g. furniture, construction, consumer electronics.

'The business model is key [to linking design and remanufacture]. It has to be part of the business model. In most cases remanufacture is seen as a threat to new product development.'

Ron Giuntini, 2006

Remanufacturing in business needs to be viewed at an organisational level, whereby the business model is designed to accommodate and encourage remanufacture. This will require appropriate buy-in from executives and a business model to match, which accepts responsibility for the full lifecycle of the product. Remanufacturing therefore needs to be seen as a zone of opportunity for profit and an integral part of sustaining profit in the future.

'Remanufacturing is an excellent example of how environmental responsibility is not only the right thing to do, it also makes good business sense.'

Mike Baunton, Caterpillar Vice President with responsibility for Perkins in The

Entrepreneurs need to be encouraged to take advantage of the business opportunities offered by remanufacturing. Although they operate in reconditioning rather than remanufacturing, Maxitech is an excellent example of a successful IT reuse business. Maxitech provide solutions for redundant IT equipment and the training of unemployed people

'Increasing remanufacturing will entail a mind-shift from people not just thinking "how can I dispose of my goods?" but "how can I maximise my return?"'

Derek Croft, Business Development Manager

e.g. social enterprise. The success of Maxitech is recognised by numerous awards given to the Managing Director, Peter Paduh e.g. winner of London Business Awards' Young Business Person of the Year 2005; Maxitech was also one of the regional winners in HSBC's 'Start-up Stars 2006'.

12.4.2 OEM Engagement in Remanufacture

OEMs hold and connect many parts of the remanufacturing puzzle: design, production, business models and marketing. As such OEMs are an essential facilitator to remanufacture but are also a potential barrier.

There are numerous reasons why OEMs may choose to not engage with remanufacture. From that perspective remanufacture may be seen as a competitor to 1st life manufacturing activities. Understandably OEMs who do engage in remanufacture, will aim to convince customers to purchase their remanufactured parts rather than those from a 3rd party remanufacturer.

'the product developer/manufacture can get the most benefit from sustainable design practices by maintaining product responsibility throughout the product life cycle.'

'The Risks of Using Non Genuine Parts:

Non-genuine parts are designed using reverse engineering (copying an original component). Manufacturers of non-genuine parts do not have access to Perkins technical data and drawings so these parts are designed and manufactured in isolation from the rest of the engine.

In most cases the material specification is rationalised, so there is a real risk that these parts will not fit correctly. This can result in rapid wear, compromised performance, poor reliability, higher oil and fuel consumption and even component damage or engine failure.

Suppliers of non-genuine parts will only provide a small range of the high- volume part numbers. They will not support the complete Perkins engine range. Non-genuine parts will not be covered by the Powerpart warranty.'

www.perkins.com, 2006

Some OEMs go further to actively provide challenges to remanufacture. OEMs may introduce new technology to their products to actively prevent their remanufacture by 3rd parties e.g. in the cartridge industry. In addition OEMs may try to steer legislation to prevent remanufacture, or block remanufacture by discrediting it as low quality or promoting the environmental benefits of recycling.

However the situation of OEMs blocking remanufacture may be changing as some start to realise the business benefits of remanufacturing.

12.4.3 Management of Core

The availability of core is a major challenge to remanufacturing. The management of core needs to be built into the business system. This requires not only building core collection methods into the business model but also providing incentives to ensure that those collections are carried out effectively.

Xerox for example, provide incentives to their national managers to collect core by linking the success of core collection to annual bonuses. By retaining ownership of copiers etc through PSSs, Xerox are

able to track their core, or 'hubs' as they term it. In contrast, Caterpillar ensures that core is returned by only giving the discount of remanufacture when the core is returned by the customer. This business model acts as an incentive to the customer at point of purchase, and the prospect of a discount on a future purchase

'Our main obstacle is the availability of core. To date all our customers find it difficult to control/manage their core. Dealers typically dispose of the worn core on receipt of new.'

Kevin Haynes, ATP Group, 2006

gives the customer reason to return, but most importantly ensures a flow of core back to Caterpillar.

Where 3rd party retailers or dealers become involved, the complexity of core collection is increased. To encourage core collection the real value of the core needs to be maintained and communicated at every point of its life. This means providing incentives to all along the chain to pass the core onto the appropriate body.

12.5 Knowledge

The availability of knowledge about remanufacturing applies to both detailed product design and business model design, and is relevant to both in terms of promoting remanufacture.

'The key to increasing [the practice of remanufacture] is to improve awareness of what re-manufacturing is.'

Jackie Seddon, Envirolink Northwest,

12.5.1 Definitions

Understanding of the definitions of remanufacture and other elements of 're' is confused, as discussed earlier. The misunderstanding results in misconceptions.

A further misunderstanding is that most people confuse the categories of 'product' and 'spare parts' by mistake (Steinhilper, 2006). The consequence of this for business model design is that the specific drivers, markets and challenges for each can be confused and therefore opportunities missed.

12.5.2 OEM perceptions

Poor understanding of remanufacturing and knowledge of out-of-date historical references to 'remanufactured' products (the classic example being the dangerous retreading of tyres which began in the 1930s) generate embedded perceptions. These perceptions and a general lack of awareness mean OEMs are generally unaware of the

'Manufacturers need to be researched to see their views on remanufacture. Some are for it, others are not, why?'

'Commercial issues are the reason and some manufacturers see it as a threat to their new sales.'

commercial advantages of remanufacturing; they will therefore not design their businesses to take advantage of remanufacture opportunities.

The whole OEM enterprise needs to be made aware of remanufacturing through training and publicity. If remanufacturing is not built into the business model it will always been seen as an add-on. This means that internal stakeholders/customers need to be involved or else they will act as internal barrier (King, 2006b).

However, the real perceptions of both OEMs and customers about remanufacturing are not well understood.

12.5.3 Skills increase

Experience and know-how is a prerequisite to remanufacturing however the skills are not more complicated than manufacturing (Freiberger, 2006). Knowledge required:

- Old core sorting /identification

- Old core analysing
- Technical know-how in setting up machines and test equipment
- Production planning

The boundaries of this know-how are changing over time. For example, with the increasing complexity of products such as cars, ‘mechatronics’ (the combination of electronics and mechanics) is a significant industry sector.

12.5.4 Recruitment

There is a lack of skilled personnel for remanufacturing. Workers in remanufacturing require experience and know-how of the processes. Although aspects or interpretations of ‘sustainability’ are increasingly taught as part of the education syllabus, specific aspects such as remanufacturing rely on the expertise and enthusiasm of the tutor. The number of tutors in the UK with specific interests in remanufacturing is limited and in-depth training of remanufacturing is therefore not guaranteed in engineering or design courses.

Tutors with an established interest in remanufacture, such as Rolf Steinhilper at the University of Bayreuth become a focus for know-how when students leave for industry. This allows tutors to maintain a good network in the industry.

However, remanufacturers tend to be small companies rather than big name companies. Therein lies a disincentive for many new engineers and designers. This reiterates the need for OEMs to get involved in remanufacture and thereby drive the development of skills.

12.5.5 Understanding Best Practice – Internally & Externally

Best practice needs to be understood within organisations. There is a lack of dissemination of information between departments related to remanufacture and also between

businesses operating within the same parent company. Creating links between these and incentivising people to act on these links to further remanufacture is a key challenge. The lack of knowledge and exchange amongst the international sphere reduces capacity for knowledge transfer of best practice. Identification of international markets and suppliers

are also compromised.

‘I think most remanufacturers are small companies and engineers do not want to work for small companies’

Stefan Freiburger, University of

‘The research community has been developing design for remanufacturing technologies for quite some time, however these tools need to be more widely applied in industry in order to be refined and proven.’

The national and international remanufacturing picture is also unclear and the number of International networks is limited. Facts and figures can become dated quickly given the rate at which new business spring up and close. International networks are also limited.

13 Potential Proactive Approaches

The numerous opportunities for sustainable prosperity that remanufacture brings can be increased and amplified by proactive promotion of Design for Remanufacture in the South East to move remanufacturing into mainstream business practice and achieve the aims of SEEDA's Regional Economic Strategy (RES).

In this section, a selection of concept approaches is discussed. It is unclear at this stage which of these methodologies would be most effective.

13.1 International Definitions

Exposure of businesses to the global remanufacturing market is hampered by the persistent confusion of terminology. A resolution of this issue would facilitate interchanges of trade, investment, people and knowledge.

International definitions of remanufacture could be introduced to enable greater understanding and lessen confusion. These definitions may be made ISO standards.

'It would be helpful for us to have a worldwide definition of the terms refurbishment/remanufacturing and so on.'

Martin Plumeyer, Siemens AG Medical

There are BSI standards which relate to remanufacture, but only one BS AU 257:2002 - Code of practice for remanufacture of spark and

compression ignition engines, that refers to the term 'remanufacture' specifically.

Other quality management standards such as BS EN ISO9001:2000 - Quality management systems, ensure quality standards in processing. This can be applied to remanufactured machines to ensure they undergo rigorous performance and safety tests before being returned to service.

13.2 Design for Remanufacture - Product case studies

Further examples of successful remanufactured products and business models are required, particularly in industry sectors where remanufacture is scarce. These may include such industries as marine, leisure and furniture.

Product case studies would benefit from a combination of skills and collaborating to innovate across regional, national and continental borders. Product case studies may seek to develop and provide guidance of the Design for Remanufacture process and its specific application to different product areas, thereby raising skill levels to match improvements elsewhere in remanufacture in the global economy.

Due to the traditional barriers to remanufacture and therefore Design for Remanufacture, these case studies may provide entrepreneurial opportunities for design-entrepreneurs in all sectors that seek out new opportunities and new markets, rather than relying on existing products, processes and supply chains.

13.3 Education

Currently there is no known education on Design for Remanufacture in the South-East, making knowledge and application zero or relative zero. With education,

designers and particularly design engineers, can gain the skills required to practise Design for Remanufacture with fluency so that the process does not negatively impact on development time or costs.

Bristol University and Bath University hold skills in remanufacture but these are rare examples in the UK. Skill levels in remanufacture and Design for Remanufacture need to be raised; building levels of collaboration with other EU regions with complementary expertise in remanufacture would help this process.

Building the demand and capacity for remanufacture may be achieved through promoting Modules on Design for Remanufacture in education. This may be complemented through collaborations in the form of work experience to innovate across regional, national and continental borders, thereby raising skill levels to match improvements elsewhere in the global economy.

13.4 Research

An anonymous study is currently being carried out by academics working in remanufacturing around the world. The study is being run by Andrew King at Bristol University and uses a web version of the Delphi Technique to establish the best research questions and then gather existing evidence/findings and ways forward.

'What is (I judge) particularly needed is a way of bringing researchers outside of the engineering/business world into a research network. Surely policy studies, sociology, marketing, legal, politics, economics all need to be involved too.'

Researchers from outside of the engineering/business world, such as sociology, marketing, legal, politics and economics need to be brought together (King, 2006b) to understand real perceptions and expectations. Notably this is necessary to discover what OEMs and customers really think about "second hand", "new" and a product's unknown history, and what the root causes of deeply held perceptions may be.

13.5 Dissemination

There is very little research into remanufacture or Design for Remanufacture and so there are correspondingly few articles in the manufacturing or design press. Exposing many more of the region's businesses to the global market through dissemination which offers

interchanges of trade, investment, people and knowledge would promote awareness and raise skill



levels in remanufacture and Design for Remanufacture.

Examples of the current very low dissemination levels are:

- Recharger magazine for the printing industry
www.rechargermag.com
- Trade fairs e.g. Remax 2005, World Expo 2006, annual events for the

recycling of imaging products e.g. remanufacture of ink cartridges; held since 1994.

These methods are specific to niche remanufacturing sectors e.g. remanufactured ink cartridges and as such are unlikely to lead to cross-fertilisation. There is also a very limited number of broader instances of dissemination of remanufacture:

- Conferences e.g. Nabil Nasr and Michael Thurston's keynote presentation at 'IEEE 2006'.
- 'Remanufacturing: The Ultimate Form of Recycling' by Rolf Steinhilper's 1998
- Reports e.g. Oakdene Hollins Ltd's 2004 report 'Remanufacturing in the UK: a significant contributor to sustainable development?'
- Organisations e.g. The Remanufacturing Institute. A not-for-profit organisation in the USA.

The dissemination of targeted and marketed evidence is required, including:

- Case studies and information
- What has worked and why
- What has not worked and why New

modes of dissemination could include:

Exhibition

A specific remanufacturing conference or exhibition to showcase and award remanufactured products across different industries. The exhibition might comprise of national and/or international activities. An exhibition could be specific to remanufacturing; or a 'plug-in' to a manufacturing exhibition - in this case the interface between remanufacture and manufacture must be considered to avoid conflict.

Mentoring Club

A 3rd party mentoring club offering specific and cross-sector advice and advising companies of opportunities in remanufacture and relevant weaknesses in their current operations.

Workshops

Workshops could explain technical processes.

Publications

A book could be written and distributed cross-sector, demonstrating:

- General remanufacturing information/statistics contrasted with sector information.
- Lists of potential suppliers for remanufactured products.
- How to remanufacture. Including case studies of business models designed for remanufacture, and Design for Remanufacture processes.

“Reman” Retailer

A shop retailing solely remanufactured products with a reputation for reliable, guaranteed remanufactured products could potentially provide a single, trusted point of contact for procurers.

“The Product Re-Life Centre”

OHL is developing a concept for the promotion of product re-life as a prototype attempt to coordinate research, policy, innovation and support. OHL are leading a project funded by BREW which will examine remanufacturing and how it could contribute to the sustainable production and consumption economy. The work is currently planned to comprises these key activities:

- Understanding the real and perceived positives and negatives of re-using products.
- Determining under what circumstances re-using a product is a better environmental option than recycling it and buying new.
- Compiling a series of case studies that describe the environmental and business impact of remanufacturing, and leading practitioners.
- Proposing and testing a range of policies and actions that might increase the up-take of remanufactured products, both in the UK and elsewhere.

14 The Future of Remanufacture

The information available on remanufacture and other reuse strategies has increased in the last few years and government sponsored activities would suggest that it is now being more seriously considered for its environmental and commercial benefits. It is however difficult to state whether the remanufacturing industry is in growth or decline, but it can be seen that remanufacture is still marginal compared to the manufacturing sector.

14.1 Markets

Remanufacturing is growing in South East Asia (SE Asia). In Thailand for example, Western ‘waste’ is being collected for reprocessing. In simple terms, these practices increasingly direct material flows away from the West and increase the relative manufacturing strength and capability of South East Asia.

However these instances give businesses in the South East the opportunity to build trading links with these emerging economic powerhouses, through the development of trade, investment and collaboration in education and innovation.

14.2 Legislation

EuP, RoHS and the ELV Directives look set to promote end-of-life options such as remanufacture. WEEE may help emphasise the business case for remanufacture but the implications of the Directive for remanufacture are still not fully understood.

14.3 Design for Remanufacture

Research into remanufacturing is limited but design is one area which has, relatively speaking, received attention recently, however the reality of this interest is research being carried out globally by a handful of people. Recent examples include this study, Erik Sundin's work on product and process design for remanufacturing, Nabil Nasr and Michael Thurston's research at The Rochester Institute, and Andrew King's work at Bristol University. Levels of Design for Remanufacture education in engineering and design courses at university level are very low and it is unclear if they will increase; although it is reasonable to presume that the increasing emphasis on sustainable design, will on some level raise awareness of Design for Remanufacture.

14.4 Original Equipment Manufacturers

OEMs are increasingly becoming involved in remanufacture, notably through out-sourcing through organisations such as InfoTeam. Although the engagement of OEMs in remanufacture has the potential to form positive links between design and remanufacture, out-sourcing has the strong potential to distance those links between design and remanufacture. In order to establish efficiencies in successful remanufacture OEMs and any sub-contractors must ensure feedback mechanisms are maintained.

15 Conclusion

Remanufacturing is a process of recapturing the value added to material when a product is first manufactured. By reducing energy use, carbon emissions and creating skilled employment, remanufacture offers a significant contribution in the move toward a sustainable society. Design for Remanufacture optimises the remanufacturing process and business model and can therefore increase competitiveness and innovation.

Remanufacturing is not an existing science however it is also not rocket science. Despite this the proportion of remanufacture in the UK compared to overall manufacturing activities is small, with just 166 companies listed under 'remanufacture' in the Yellow Pages. The proportion of OEMs engaging in remanufacture is small and it is only OEMs who are able make the connection between design and remanufacture.

15.1 Linking Design and Remanufacture

There is currently little Design for Remanufacture being practised. Those who remanufacture are commonly small organisations who have no control over the design phase. In general terms, those OEMs who control design (either in-house or through a design consultancy) do not engage in remanufacture. Therefore the optimisation of remanufacture through Design for Remanufacture seldom occurs, with the exception of some notable OEMs such as Xerox and Caterpillar.

15.1.1 OEM involvement

OEMs hold the key to creating markets for remanufactured products and linking design with remanufacture. The evidence base for remanufacture needs to be strengthened across sector and this will require a better understanding of

remanufacturing in the UK and internationally. This information needs to be disseminated both horizontally and vertically.

Dissemination of this information and further examples of remanufacture can be used to promote the business case in which producers are directly linked to the end user by the producers' continued engagement with the product. One method of enforcing this would be to change retail law to link the producer and customer directly by warranty.

Warranties and business models that lead to successful remanufacture need to be promoted and encouraged. Design for Remanufacture needs to be used in this respect to increase the number of products that can be manufactured.

15.1.2 Understanding Design for Remanufacture

Design for Remanufacture concerns both the design of business models and detailed product design. Detailed product Design for Remanufacture consists of an interrelated group of design strategies, which pragmatically build upon the foundation of eco-design strategies to capture the commercial opportunities of multiple lives and upgrading. However Design for Remanufacture is a grey area and in order to proactively promote it, Design for Remanufacture needs to be clearly defined and proved.

For the evidence base to be built, more product designers, engineers and design engineers need to gain the skills required to evaluate a product's relationship to remanufacture and understand how to effectively apply Design for Remanufacture strategies. In this, functionality is certain to take precedent over remanufacturability in the vast majority of cases and this suggests that the frame of reference for design should be changed to take other aspects e.g. environmental performance, into account. Whilst this is unlikely in the current social paradigm, alternative methods to Design for Remanufacture without compromising functionality need to be identified.

Design for Remanufacture suffers the same barriers as eco-design in general, including apathy, suspicion and ignorance of the potential for generating ongoing profit. The value of design in the remanufacturing process needs to be better appreciated and acted upon by all stakeholders in remanufacture, but notably OEMs who have the ability to control both design and remanufacture.

Design for Remanufacture also encompasses the design of the business model. Business models must take retail laws into account to ensure that the products can be stored and processed cost-effectively with minimal hassle to stakeholders – notably the retailer. This will require consideration on the business model side.

The detailed design process will require engagement by product designers, engineers or design engineers. The way that these professions connect with Design for Remanufacture and how they can interact further to combine their skills requires further exploration and testing by industry.

15.2 Design for Remanufacture – the benefits

Design for Remanufacture of both product strategy and detailed product design can lead to reduced carbon emissions, landfill and secure, skilled employment. The very concept of remanufacture implies closed-loop systems, which could, in the distant future lead to the replacement of virgin manufacture with ongoing remanufacture.

The implications of a closed-loop remanufacturing system for business, society and sustainable development need to be explored.

Remanufacture is reported to be up to twice as profitable compared to manufacture, however the profit varies according to product and sector and the figures remain unconfirmed. Design for Remanufacture can optimise the remanufacture and broaden the range of products that can be economically remanufactured.

However Design for Remanufacture is not widely practised because of numerous barriers and so the efficiencies resulting in reduced CO₂ emissions and increased profit through innovation and new product development remain largely unexploited.

15.3 Design and Remanufacture – the barriers

The linking of design and remanufacture faces significant hurdles alongside the traditional barriers to sustainability. These additional barriers can be termed as:

- Definitions and understanding
- Preconceptions
- OEM engagement

15.3.1 Definitions & Understanding

There is common misunderstanding of the definition of remanufacture and confusion with other elements of 're' (reconditioning, repair, recycling). Confusion and misunderstanding devalues appreciation of the evidence base for remanufacturing, dilutes the message and therefore the potential for increased remanufacture. Definitions require addressing through the official publication of international remanufacturing standards and process, by an organisation such as ISO and/or national standards bodies e.g. in the UK BSI.

15.3.2 Preconceptions

Customers are believed to have preconceptions about remanufacture, but aside from the historical references of poor quality e.g. retread tyres from the 1930s, the root causes of these perceptions in both OEMs and in society as a whole are unclear. Therefore there is a possibility that current understanding of others' opinions may be based on assumption rather than fact. OEMs' and customers' perceptions and the root causes of them need further research to allow more effective design of business models.

15.3.3 Market Development

The market for remanufactured products (as opposed to spare parts) needs to be developed through the business model design and detailed Design for Remanufacture. This will also require a confrontation with the widely held temptation for the 'new'. As is the case in other spheres of sustainability, this demands a realignment of pricing to reflect lifecycle costing. How the lure of the 'new' can be extinguished requires investigation, particularly in terms of perceptions and root causes.

15.4 Implications for SEEDA

SEEDA's Regional Economic Strategy (RES) (see www.seeda.co.uk/res) lays down a framework for sustainable prosperity. Remanufacture can offer prosperous business

opportunities to entrepreneurs and OEMs alike in conjunction with reductions in energy use and CO₂ emissions.

Design for Remanufacture optimises the process of remanufacture making remanufacture more profitable. Proactive strategies to further Design for Remanufacture are likely to involve resolving understanding and awareness of remanufacture and generate trust in remanufacture.

The publication of an official definition such as an ISO standard could resolve confusion over definitions. These definitions may usefully extend to the actual design process for Design for Remanufacture to aid designers and engineers, whilst making remanufacturers aware of what could be done to improve their process.

Fundamental to encouraging remanufacture in the UK is to engage OEMs with both business models that are Designed for Remanufacture and that promote detailed product Design for Remanufacture. Increasing OEMs' engagement in remanufacture and then optimising remanufacture through Design for Remanufacture, may be achieved by generating a trustworthy evidence base for remanufacture. An evidence base could showcase products Designed for Remanufacture and compare them with equivalent products not Designed for Remanufacture. At first, this could feed into Design for Remanufacture standards and definitions; it could then be used to demonstrate the Design for Remanufacture process and techniques through example, ensuring that designers can acquire knowledge in this field. Importantly, an evidence base of products Designed for Remanufacture, could publish reliable

financial reports of remanufacture to show the economics of remanufacture versus manufacture.

These strategies may provide OEMs with a greater and accurate understanding of remanufacture, sufficient to incentivise some to engage in not only remanufacture, but Design for Remanufacture.

SEEDA's promotion of remanufacture through profit optimisation via Design for Remanufacture and the proactive strategies listed in this report will allow SEEDA to meet the aims of its RES: innovation, sustainable economic growth and Sustainable Development and contribute to the transformation of the South East into a Factor 4 region.

4. Research Overview

User-Centered HRI: HRI Research Methodology for Designers

M. Kim, K. Oh, J. Choi, J. Jung, and Y. Kim

Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea

Abstract. This chapter introduces the field of user-centered HRI, which differs from the existing technology-driven approach adopted by HRI researchers in emphasizing the technological improvement of robots. It proposes a basic framework for user-centered HRI research, which comprises the three elements of “aesthetic”, “operational”, and “social” contextuability. This framework is made for robot product designers

seeking to incorporate user perspectives and needs; it is intended to allow easy identification and efficient study of issues in user-centered HRI design. The case studies introduced in this chapter, all of which are based on the aforementioned framework, will facilitate understanding of user-centered HRI, and create new research opportunities for designers, non-experts, and robot engineers.

Keywords: Human-Robot Interaction, Robot Design, User-Centered HRI.

1 Introduction

Thanks to advancements in robotics technology, the groundbreaking results of academic research, and the growth of the robotics market brought on by rising demand, robotics is moving beyond the realm of engineering and becoming an active field of research in the humanities and social sciences. Robot design, in particular, is gaining increasing importance as a means of presenting robots, commonly regarded as technocentric hardware, to new users as software products that combine communication media with services suited for everyday life.

However, the most urgent cause for vitalizing research into robot design lies in the dominant approach taken by existing researchers: because previous work on robotics aimed primarily at technological achievement, the focus tended to be almost exclusively on enhancing the skills and capabilities of robots from a developer's point of view, rather than on improving user-friendliness. For this reason, there are few products that have succeeded in winning market popularity, despite that the cognitive and decision-making capabilities of robots have reached an astonishing level. To redress this problem, active effort has begun to be expended on the study of the social attributes of robots (e.g. Fong et al., 2003; Breazeal, 2004; Dautenhan, 1999). Therefore, this chapter focuses on aspects that were lacking in the approaches taken by previous technocentric HRI studies: user-centered HRI is defined to examine the requirements of robot users as well as changes in user behavior caused by robots, and a framework for HRI research

based on the findings of this examination is proposed. In addition, case studies are employed to facilitate an understanding of the framework, and future possibilities for research on user-centered HRI are explored.

2 User-Centered HRI

Even when two robots are based on the same technology, they can be experienced in completely different ways by users depending on the type of interaction system they employ. For this reason, interaction design—which, together with character design and appearance design, is one of the three elements of robot design (Oh et al., 2005)—tends to be weighted above the other two elements, and interaction-related factors such as modality, autonomy, and interactivity figure prominently among the five properties of social robots mentioned by Bartneck and Forlizzi (2004). Therefore, robot designers must not only be proficient in their traditional task of appearance design, but also develop a solid grasp of the more fundamental issue of HRI.

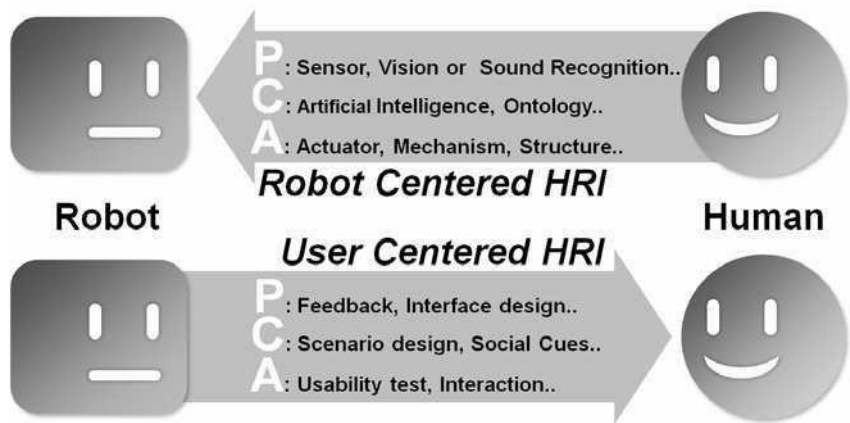


Fig. 1. Dual approaches to HRI research: robot-centered HRI and user-centered HRI

Research on human-robot interaction, or HRI, can be divided largely into two approaches: ‘robot-centered HRI’ and ‘user-centered HRI’ (Oh and Kim, 2007) (Figure 1). The majority of existing studies on HRI concentrate on how robots could be made to effectively perceive, cognize, and act in their given environment in order to interact well with humans. This falls under the former category of robot-centered interaction research, a perspective

typically adopted by robot engineers. R. A. Brooks, C. Breazeal, and others at MIT are some of the most representative proponents of robot-centered HRI research, which focuses on robot control and implementation technologies. Unlike ICRA (International Conference on Robotics and Automation) and IROS (International Conference on Intelligent Robots and Systems), both established traditional academic conferences on

robotics, RO-MAN deals only with specialized research topics relating to HRI. However, even the topics presented at this workshop, aside from rare exceptions such as “philosophical issues of robot/human coexistence,” “cognitive psychology in robot/human interactions,” and “intuitive instruction of robot assistants,” focus primarily on how to control robots in order to enhance their recognition and perception of humans (Kim and Kim, 2005).

The second perspective, as mentioned above, is user-centered interaction research. This approach prioritizes how robot users perceive and respond to robots, and how they adjust their behavior as a result of such response, over how robots cognize and accommodate humans. In other words, user-centered HRI research can be defined as the study of interaction aimed at enabling robots to better assist human users in accomplishing their desired objectives and to provide a higher quality of user experience. Today, user-centered HRI research is becoming the province of robot designers on the grounds that designers acquire and develop specialized knowledge about how users perceive and regard products from an overall point of view (Lobach, 1997), and that they prioritize the needs and experiences of the user over those of the robot. It is also an area of research that robot engineers should explore more actively. Examples of user-centered HRI research include the study of emotional bonds or feelings that users develop in the course of interacting regularly or closely with robots, and of changes in the value of life or the ecology of other products (Forlizzi, 2007a) brought on by the infiltration of robots into everyday existence.

3 HRI Research Framework for Designers

Because the objective of designers researching user-centered HRI diverges from that of robot engineers, the very definition of the problem needs to be differentiated. This means that a differentiated outline and framework for research tailored to designers must first be developed. Thus, this chapter examines the respective research elements covered by robot engineering and robot design, and integrates the two to propose a set of research elements distinctive to HRI design,

i.e. the research field of user-centered HRI.

3.1 Research Elements in Robotics Engineering: P, C, A

Research elements in robotics engineering can be divided largely into three categories based on the graduate curriculum of the Robotics Institute at Carnegie Mellon University: 1) sensor technology, which allows a robot to perceive its external environment; 2) AI technology, which enables a robot to cerebrate using the wide-ranging information and knowledge databases thus perceived, and to autonomously control its various components; and 3) actuator technology, which endows a robot with mobility and the capacity for action. Also conducted is comprehensive research on the mathematical correlation of these three elements.

When these research elements are mapped onto the cognitive process humans carry out in response to their environment (Norman, 1988; Card et al., 1983), sensor technology corresponds to '**perception**', AI technology to '**cognition**', and

actuator technology to **‘action.’** When an intelligent robot equipped with such a system communicates with human beings, the model for this communication can be schematized as shown in Figure 2 (Kim and Kim, 2005).

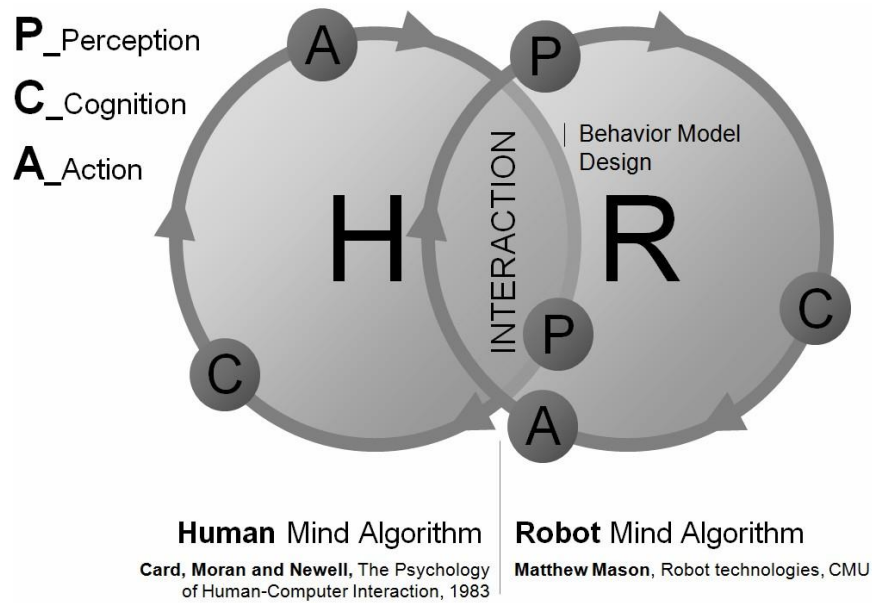


Fig. 2. P.C.A model for HRI

In the above model, the human user and the robot, each possessing the ability for P (perception), C (cognition), and A (action), follow the PCA cycle as they alternate between independent and interdependent action. Human-robot interaction thus occurs when the two PCA cycles overlap, resulting in communication. The zone in which this interaction event occurs can be termed the **‘human-robot interface.’**

3.2 Research Elements in Design

In order to link the research elements of robotics engineering—i.e. P, C, and A— with the research elements of product design, it is necessary to examine the background characteristics distinctive to the academic discipline of product design. According to studies by the pioneers of design, the objects of research can be divided into “the determinate” and “the indeterminate”; whereas the study of science falls under the former

heading, design can be defined as the study of the indeterminate through invention and creation (Buchanan, 1995; Swann, 2002; Cross, 2001; Simon, 1969).

Rather than concentrating on the changing objects or media of design, design researchers must focus on the mental process, or flow of thought, involved in how we change or create objects and media. Richard Buchanan (1995) saw this mental

process as a rhetorical one, and proposed the sequence known as the “4 orders of design,” consisting of signs and images, physical objects, actions and services, and ideas and systems (Figure 3). Buchanan’s attempt reflects the understanding that design research must regard its objects as more than mere objects, by contemplating their fundamental nature and characteristics.

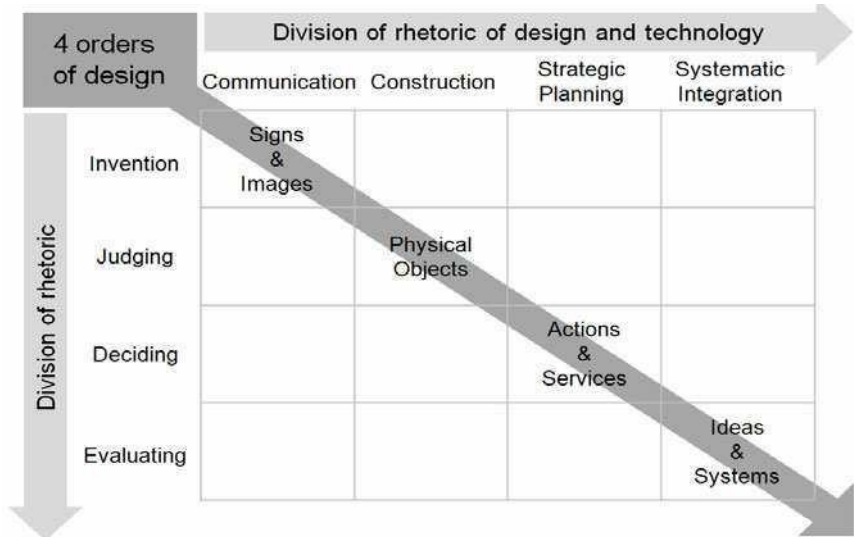


Fig. 3. Matrix of abilities and disciplines in design (Modified from Buchanan, 2001)

3.3 Research Elements in HRI Design

Each of Buchanan’s four orders constitutes a specialized area of design (Figure 3). When restricted to the research elements of HRI design, “signs and images” and “physical objects” can together be mapped onto “aesthetic contextuability,” while “actions and services” and “ideas and systems” can be mapped onto a robot’s “operational contextuability” and “social contextuability,” respectively. In short, when HRI is recast in light of the research elements of design, perception can be linked to “aesthetic contextuability,” action to “operational contextuability,” and cognition to “social contextuability.”

First, “aesthetic contextuability” is intimately related to the information that is exchanged in the very first moments of interaction between human and robot. In terms of a user-centered approach, interaction research centering on aesthetic contextuability can be defined as the study of how

users experience and respond to the external aspects of a robot's makeup, such as shape, color, material, sounds and vibrations, size and proportions, etc. which are first seen, heard, and felt through the user's five senses. In the robot design process, aesthetic contextuability is dealt with partially during the appearance design stage, but it is not restricted to a single element of robot design. Form, which constitutes the core element of aesthetic contextuability, indirectly conveys impression and character

at the initiation of conversation between human and robot, and affects the determination of their method of communication. Therefore, HRI designers must not only understand form from the perspective of design, but also from that of cognitive psychology.

Some examples of such research include studies on how robots may become “human-like” as a result of certain external factors and on the “uncanny valley” hypothesis, as well as studies on formal elements suited to a robot’s specific character and role, or on what materials might best benefit a robot’s given use.

Second, “operational contextuability” involves aspects of handling and manipulation, patterns of movement, and usability during actual interaction between user and robot. Such questions as how the functions served by the robot are understood and regarded by the user, and how user-robot interaction fulfills the needs of the user both fall under the heading of operational contextuability research. Depending on what function a product employs to assist the user’s task, and on what method is used to implement a given function, the user can go beyond merely having his or her expectations satisfied to experiencing pleasure (Hassenzahl, 2002).

A user’s needs are not fixed; they change according to time, place, and situation. If one can ascertain what kinds of desires can be inspired and fulfilled by using and interacting with a robot, and apply this knowledge to robot design, it will be possible to transform robots into products that can broadly impact the public’s everyday attitudes and activities, just as past innovations such as automobiles and mobile phones were able to do.

HRI research on operational contextuability starts with observing and understanding how people use robots. Although robots are operated by a user who can manipulate its controls, they are distinct from other existing products in that they perform their functions autonomously on the basis of user-robot communication or identification of environmental factors. For this reason, close examination of the perception and attitudes of users regarding the implementation of a robot’s functions is imperative for the development of superior robots.

Third and last, “social contextuability” deals with such social and metaphysical values as intimacy, sense of connectedness, significance, worth, trust, and feeling, which a user attains as a result of interacting with a robot. Human beings do much more than simply perceive, feel, and use products—particularly when it comes to products like robots, which are

made to execute human functions, have conversation skills, and express emotion. Research on social contextuability thus covers modifications in role assignment caused by the placement of a robot in a human user's everyday environment, changes in living patterns as examined in Forlizzi's study (2007b), changes to product value including product attachment (Mugge et al., 2008), and so on.

Characteristics such as those mentioned above are not so much inherent in robots themselves, as produced through the intermingling of past knowledge, experience, and cultural habits during a user's interaction with a robot. Therefore, studying these social and metaphysical characteristics requires an understanding of the sociocultural context surrounding users and robots, as well as an anthropological and qualitative approach to research.

The three research elements of HRI design examined thus far are aligned with the three levels of design (visceral, behavioral, and reflective) mentioned by Norman (2004), and with the three ways (aesthetically, functionally, symbolically) in which people collect and use products, as mentioned by Forlizzi (2007a). This classification, however, remains basic at best; it would be impossible to divide all research on HRI design accurately into these three categories. Still, it is possible to position studies covering aspects of these research elements in varying proportions upon a continuous triangular plane whose three axes are constituted by the three elements (Figure 4).

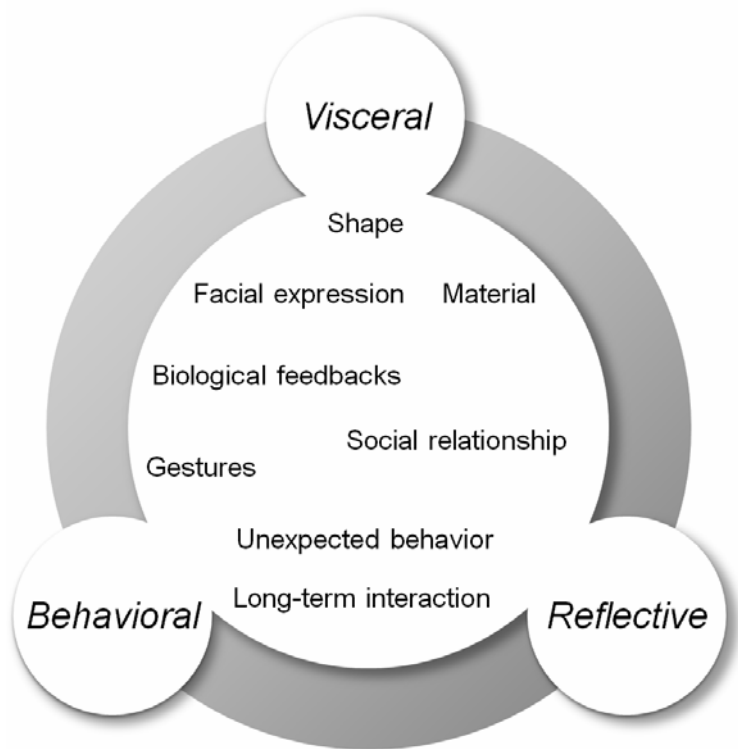


Fig. 4. The research elements of HRI design: Keywords in the inner circle imply main issues of following case studies (Chapter 4)

3.4 Research Framework for HRI Design

Because creating a robot requires enormous investments of time and money, prototyping methods have been proposed as a way to make robot

research easier and more efficient for general researchers including designers, as opposed to robot engineers (Bartneck, 2004).

User-centered HRI research, unlike robot engineering, does not seek to discover specific functions explaining how HRI affects the perception and behavior of robots

and users. Rather, it aims at analyzing user perception and its resultant responses, as well as the user requirements that ensue. This means that by clearly specifying the research objectives and effectively setting the independent variables for research, HRI studies can be carried out successfully even with prototypes of relatively low fidelity.

One way to study the three research elements covered in Section 3.3 is to employ the three kinds of prototypes mentioned by Yang (2004). First, to study aesthetic contextuability, it is necessary to manufacture a “look and feel” prototype. This prototype needs to recreate sensory stimuli closely approximating those provided by the actual robot, including visuals and sound.

Second, to study operational contextuability, a “function prototype” needs to be created. A function prototype is a prototype made to perform functions similar to those of the actual robot; it thus needs to be able to execute the functions intended by the designer and identify relevant user behavior. Type-To-Speech (TTS) and Wizard of Oz (WOz) are two techniques that can be used for this purpose (Gould et al., 1983). This type of prototype allows for research on task execution, such as human-robot collaboration and puzzle solving.

To study the third element of social contextuability, “role prototypes” are needed. This type of prototype is used to acquire an understanding of usage context, through the utilization of storyboarding or scenario techniques. Social contextuability is a relatively more difficult field to research than the above two, and can only be studied through insightful analysis based on long-time observation of human-robot interaction. However, when the goal and scope of research is restricted to a clear-cut, peripheral problem, using a role prototype to have a user virtually experience and imagine usage situations can allow a researcher to identify user responses or requirements in terms of social contextuability.

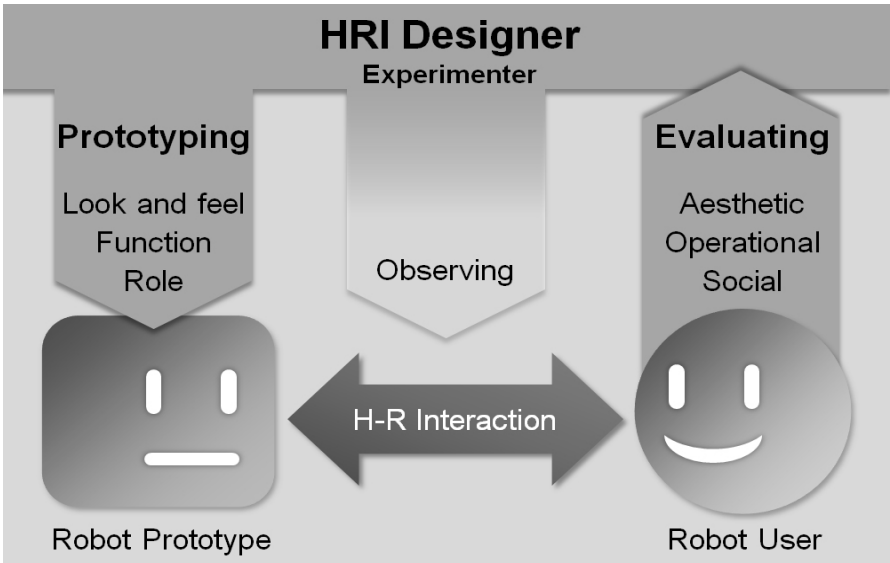


Fig. 5. Prototypes and elements for HRI research

These three prototypes and elements for HRI research can be schematized into an HRI research framework, as shown in Figure 5. This framework can not only facilitate the understanding of user-centered HRI, but also help novice researchers wishing to take a user-centered approach including designers, define HRI problems, and set research areas and methodologies.

4 User-Centered HRI Design Case Studies

This chapter is intended to help readers form a more concrete understanding of the proposed user-centered HRI design framework by introducing several research case studies that employ this framework. Case studies pertaining to the three categories of HRI design enumerated above—i.e. aesthetic contextuability, operational contextuability, and social contextuability—are presented below, with summaries of background, objectives, methodologies, and results.

4.1 Case Studies on Aesthetic Contextuability in HRI

4.1.1 Matching Proper Materials to the Role Images of Teaching Assistant Robots for Elementary School

The purpose of this case study was to extrapolate role images for a teaching assistant robot suited to elementary school students by focusing on one specific element of robot appearance: the materials used to manufacture a robot. This study presupposes that external materials constitute an important factor influencing the effective expression of a robot's role. The main topic of inquiry in this study, the materials forming a robot's exterior, falls under the heading of "aesthetic contextuability" in terms of user-centered HRI research; the significance of this study lies in its focus on "materials," an element of a robot's exterior that had heretofore been neglected by researchers. A robot's shape, color, and other basic elements of design have been addressed in previous studies as factors that can influence human-robot interaction (HRI). However, material—despite being an important element, along with shape and color, in the design development of robots—has been largely overlooked in HRI research thus far. Nonetheless, the materials that constitute a robot's exterior are closely connected to communication by touch as well as by sight; it is thus an

element that deserves increasing attention as robots continue to be incorporated into the everyday living spaces of human beings.

The main purpose of this case study was to match materials with specific groups of adjectives relating to a robot's given role. Using a robotic platform with a simple structure and function, a wide range of materials including metal, glossy plastic, matte plastic, wood, fabric, and rubber were applied to the platform exterior to develop robot prototypes for testing (Figure 6). A prototype endowed with such form and function, known as a "look and feel" prototype, is a representative variety of prototype used in HRI research involving aesthetic contextuability. The tasks to be carried out by the test subjects were set through

the robots' eye and head movement and through simple dialogue; while carrying out these tasks, the subjects were asked to evaluate the degree to which each robot's material accorded with its role image (adjective group) as a teaching assistant robot.



Fig. 6. Look and feel prototype used in the experiment

This case study was limited by its use of materials with colors unique to each material; further observation that separates touch-based cognitive responses to the material from visual cognitive responses to color is needed in future. Nonetheless, it clearly demonstrates that material can be an important factor in robot design for the effective expression of a robot's given role. Above all, this case study is valuable for its presentation of practical HRI research results that can be referenced by actual designers in the process of robot design development.

4.1.2 A Study on External Form Design Factors of Teaching Assistant Robots for Elementary School

The purpose of this case study was to identify the factors of a robot's physique that effectively express an image appropriate to a teaching assistant robot, and thereby propose guidelines for the external form design of teaching assistant robots. Like the study examined in 4.1.1, this study also has the ultimate goal of developing an optimal design for teaching assistant robots; instead of "materials," however, it focuses on a robot's "physique."

Prior to the experiment, images related to the role of teaching assistant robots were canvassed through textual research and a factor analysis, and eight factors of physique for teaching assistant robots were extracted from

among the diverse factors that compose the human physique. Using these factors as variables, various 3D external form samples were created for testing.

These 3D robot models were projected in life size against the inner wall of the laboratory (Figure 7); they were viewed by children—the main user group for teaching assistant robots—who then filled out a questionnaire gauging their response to each model's physique. Whereas the study examined in 4.1.1 created physical look and feel prototypes, this study utilized 3D models that exist only in

software form. However, it adopted an innovative new method for the use of “look and feel” prototypes: to maximize the subjects’ sense of the 3D models’ volume and presence, animation of each prototype in 360-degree rotation was created, and the models were presented in full life-size projection.

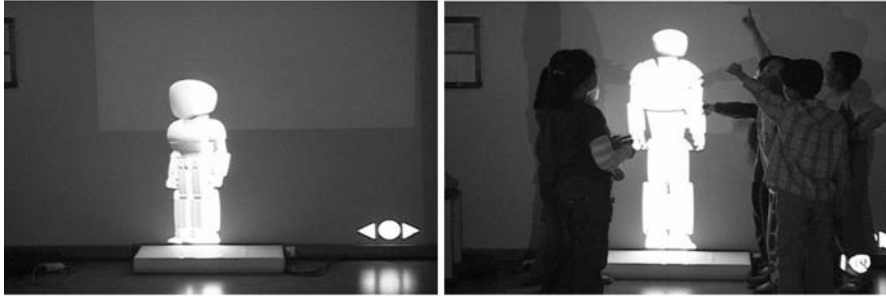


Fig. 7. 3D robot models presented in full life-size projection

The experiment results showed that the ratio of head length (vertical) to the body, height, ratio of chest circumference, and waist circumference were related to role images for teaching assistant robots. They further demonstrated that height and waist were particularly important, with waist circumference having a high correlation to all role images. To convey a gentle and kind image, for instance, the ratio of head length needed to be adjusted in relation to waist circumference.

Based on the results of analyzing the relationships among various factors of robot physique, this study proposed guidelines for designing the external form of teaching assistant robots. As such, this work does more than research the development of a particular type of robot, i.e. teaching assistant robots; rather, the methodology described in the study proposes new guidelines that can be used to research the development of new external forms for robots with different functions, uses, and user bases.

4.1.3 A Study on an Emotion Eliciting Algorithm and Facial Expression for Designing Intelligent Robots

Many humanoid robots that express emotion through facial expressions have been developed thus far. While it is easy to find HRI studies dealing with technical factors related to robot behavior, including facial expressions, it is relatively harder to find works, like the current case study,

that rigorously address the perspective of the user who must interact with and use robots face-to-face. This case study can be classified as user-centered HRI research that focuses on the user's cognitive response to—and assessment of—robots.

The process of changing from one facial expression to another takes time, and the length of this process is affected by the type of emotion being expressed as well as by the strength of the external stimuli driving the expression change. The degree of “naturalness” conveyed by the robot's facial expression changes according to the dynamic between these factors, which in turn affects the

“humanness” or “humanlike-ness” a human user feels when faced with a robot. This is the hypothesis informing the case study examined here (Figure 8). The study also falls under the heading of research on aesthetic contextuability, and, like the preceding examples, uses “look and feel” robot prototypes to test its experimental hypothesis.

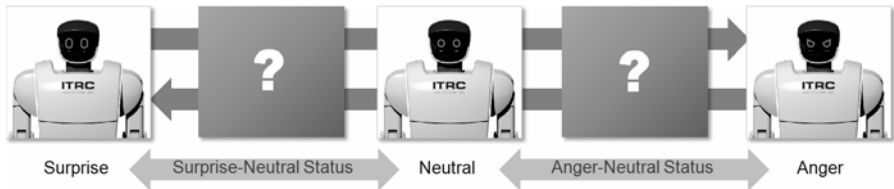


Fig. 8. The intermediary status of facial expression

As the technology for enabling robots to express emotion becomes increasingly sophisticated, the “humanness” of robots naturally increases as well. However, when the objective is to realize facial expressions under limited technological and budgetary conditions, enhancing the naturalness of the robot’s emotional expression can be one way of increasing its humanness. This particular case study focuses on external form—specifically, elements relating to a robot’s face and expressions—as a factor that increases humanness.

In this case study, four human expressions (anger, happiness, sadness, surprise) were selected for re-creation. A robot prototype capable of realizing these four expressions through alterations in the shape of its eyes and mouth was developed and used for experimentation. The main outcome of the study was the identification of the duration of facial expression change appropriate to each emotion, achieved by measuring the humanness of the robot as perceived by the subject in relation to the length of time required for the robot to change its facial expression. The relationship between the robot’s emotional characteristics and facial expressions illuminated in the study suggests the need for more in-depth HRI research on a broader range of physical elements for emotional expression besides facial expressions. Above all, it supports the argument that the definition and classification of emotional models suited to robots, and of the corresponding tools and methods of expression, are needed to enable smooth human-robot communication and enhance a robot’s capacity for task execution.

4.2 Case Studies on the Operational Contextuability of HRI

4.2.1 A Study on Humanoid Robot's Gesture Design

Among the many competitive advantages possessed by humanoid robots, the ability to gesture plays a particularly crucial role in enabling more intuitive and abundant communication during human-robot interaction. Gesture design also has a large impact on the determination of a robot's personality. However, there is as yet insufficient user-centered design research on what specific roles a robot's

gestures serve during actual interaction with a human being, how gestures reflect a robot's personality, and what kinds of gestures are appropriate to the nature of various tasks. In this context, the current case study had three primary objectives:

1) to identify the impact of gestures in HRI, 2) to design a robot's personality through gesture design, and 3) to propose personality designs appropriate to the nature of various HRI tasks.

To achieve these goals, several experiments were conducted using an actual humanoid robot platform as a function prototype. The robot used in the experiments was a wheel-type humanoid robot named Amiet. The size, speed, and frequency of its upper-body movements—i.e. head, arm, and waist gestures— could be controlled; speech generation was enabled using TTS (text to speech) technology. Although Amiet was not developed specifically for this study, it was conducive to effectively designing the experiments in question, since it was a general humanoid type suitable for research relating to robotic gesture design. However, for optimized control of the various elements, human-robot dialogue scenarios and the corresponding speech and gestures were newly prototyped using the WOZ technique (Figure 9).



Fig. 9. Experiments using the humanoid robot platform

The discoveries made through the current case study are as follows: First, robotic gestures were found to have a stronger impact than speech when it came to effectively communicating meaning to human users; it was also found that a lack of consistency between gesture and speech left a negative impression on the user. Second, a robot's personality could be appropriately expressed, and perceived by the user, by controlling the size, speed, and frequency of its gestures. Third, depending on the type of personality derived from gesture design, there were different HRI tasks that were found to be contextually appropriate.

In conclusion, this case study effectively carried out experiments from the perspective of user-centered design by utilizing a function prototype to enable a fundamental understanding and application of humanoid gesture design, and provided for more intuitive usability through such experimentation. It further confirmed the operational contextuability of implementing gesture-based HRI design that is advantageous to the user's tasks.

4.2.2 Design of the Anthropomorphic Sound Feedback for Service Robot Malfunction

With the advancement of research fields related to service robots, human-robot interaction is taking on a more complicated aspect. As a result, users sometimes fail to understand System or structure of the robot and thus they are unable to respond appropriately to malfunction of robot. If an error occurs in a robot equipped with a powerful actuator, the user could even be placed in unexpected danger. Therefore, there is a strong need to look into the feedback design for the effective communication of such error situations (Lee et al., 2010). The majority of existing studies on robot feedback design address usability during normal operation; research on feedback for malfunction remains insufficient as yet.

The current case study aimed at applying anthropomorphic elements to feedback design for the malfunction of service robots, and at ascertaining the effects of such application on usability. It also attempted to discover how usability provided by anthropomorphic feedback differs according to the level of seriousness for errors. The cleaning robot, which has recently come into the spotlight as a commercialized everyday robot, was selected as the medium for achieving the goals of the experiment; to effectively control the various experimental factors, a function prototype possessing the external appearance of existing cleaning robots was created. All other elements were omitted from the prototype, besides the speaker, wheel motor and the wireless transmitting module and thus it could be controlled remotely just for the experimental purpose. For the scenario in which the cleaning robot was working normally, the sound of a vacuum cleaner played to induce the test subject to perceive that the robot was in cleaning operation. For malfunction situations, both anthropomorphic and non-anthropomorphic sound feedback was prototyped for two different levels of seriousness. Non-anthropomorphic feedback was simply mechanical sound while the anthropomorphic feedback did not have any linguistic functions and thus it sounded like higher animal such as dogs, dolphins and etc. Subjects evaluated the usability of feedback in each situation provided by the prepared scenarios in terms of its effectiveness, efficiency, and satisfaction. Video analysis of each subject's actions was also carried out (Figure 10).



Fig. 10. Experiments using a function prototype of cleaning robots

The findings of this study can be summarized as follows: 1) Anthropomorphic sound feedback was more effective to communicate general errors than its

non-anthropomorphic counterpart. 2) When the level of seriousness was high, short and clear mechanical sound was more effective, since the situation demands urgency. 3) Therefore, it was clear that appropriate feedback design differed according to error type. As these findings illustrate, this study emphasized that response to error situations must be considered from the perspective of user-centered design, rather than through a technology-driven approach.

For credible incorporations with various robots, HRI design studies are responsible for developing the effective strategy to deal with a wide range of malfunctions of the robots. The conclusions derived from this case study provide inspiration regarding a new role for user-centered HRI design as a means of overcoming the negative aspects of HRI error that are bound to be experienced through the daily use of robots in the future.

4.2.3 Application of Unexpectedness to Behavioral Design for the Entertainment Robot

As products become increasingly intelligent, users have begun to expect pleasurable communication that goes beyond the effective execution of a product's given function. The most representative approach for enabling a product to provide pleasure to human users is interaction design for entertainment. Since robots can be seen as occupying the extreme end of the spectrum of intelligent products, they are naturally being required to provide entertainment to users. However, despite Sony's development of a variety of entertainment robots, including AIBO, QRIO, and Rolly, there are few instances of robot products that have been deemed successful. This is likely due to the fact that the design of robot behavior is severely limited as yet: the novelty effect of robot products quickly wears off as a result.

The current case study thus aimed at introducing unexpectedness to the behavioral design of entertainment robots. To this end, it specified three categories of unexpectedness and the methods used to express each type of unexpectedness, as well as the behavioral factors needed to generate a sense of fun. Based on these aspects, the study ascertained through experimentation how each type of expectation disconfirmation differently affects user perception.

Experiments were designed and carried out in two stages. In the first experiment, brief videos of robot behaviors expressing various types of unexpectedness were shown to the subjects, then they evaluated each behavior in terms of novelty, enjoyment, familiarity, performance,

reliability, and satisfaction. This method, used to evaluate human-robot interaction scenarios, can be categorized as a video-based function prototype supported by study of Woods (2006), which showed that both video and live demonstrations produced virtually identical result values. The second experiment examined how the user's impression of, and satisfaction with, the robot differed according to the behavioral factors of unexpectedness. For effective experimentation, Robosapien V2, a toy robot currently on the market, was selected as a physical function prototype. Robosapien is a humanoid type robot measuring approximately 55cm; the actuators for its leg and arm joints can be controlled manually. The experiment

consisted of direct interaction between subject and robot using the WOz technique in accordance with various specified scenarios; the subjects perceived and evaluated the robot's behavior (Figure 11).



Fig. 11. Experiments using a humanoid toy

Based on the results of the study, explanations of user experience by type of unexpected behavior were presented, and behavioral design guidelines by target user were proposed. Such user-centered design knowledge points to an active role for HRI design that goes beyond simply providing improved usability through robot behavior, to aim ultimately at giving pleasure to human beings.

4.3 Studies on the Social Contextuability of HRI

4.3.1 Interaction Design of Teaching Assistant Robots Based on Reinforcement Theory

This case study is aimed at examining whether reinforcement theory, a brand of behaviorist learning theory, can be effectively applied to teaching assistant robots. Interaction systems were designed variously by robot type on the basis of reinforcement theory, and student's interactions with each robot type were observed. The process of student-robot interaction was recorded on video and in observation logs; these were used for a post-experiment analysis. After the experiments were completed, the changes wrought in student behavior by each robot type and the impressions received by students from the robots were analyzed.

Educational robots, which were the object of this case study, have been shown to convey a friendlier image than other media in assisting children's learning, and to be particularly effective in motivating young children (Han et al., 2006; Kanda et al., 2004). For this reason, they are garnering attention as a useful educational medium. However, to introduce robots to the educational scene, one

must do more than simply use robots in education; it is necessary to apply various existing theories on education to the interaction design of robots, and to verify the effects by conducting studies involving children.

In the current case study, robots embodying three types of role models based on reinforcement theory—"Ching-chan-ee(rewarder)", providing positive reinforcement; "Um-bul-ee(punisher)", providing negative reinforcement, and "Sang-bul-ee(twofer)", providing both positive and negative reinforcement—were designed to examine changes in the interaction context of users. Test participants were required to interact with all three types of robots in a designated educational context, i.e. solving math problems, and were asked afterward to check their impression of and preferences for the various robots through interviews and questionnaires (Figure 12).



Fig. 12. Experiments for ascertaining the effects of role models for teaching assistant robots

This study demonstrated the need for teaching assistant robots in educational environments; it also proved that a student's impression of and preferences for teaching robots can vary according to the type of reinforcement used. Furthermore, regarding HRI research methodologies, it advocates the need not only for studies on user-robot interaction contexts in the field and in virtual usage environments, but also for those focusing on the relationship between user and robot.

4.3.2 Interaction Design of Teaching Assistant Robots Based on Reinforcement Theory

The purpose of this case study was to carry out a comparative analysis of the childcare-related behaviors of dual-income and single-income families, and to propose design guidelines for childcare robots tailored to each type

of household. Unlike the studies examined in the preceding sections, the current study did not observe user-robot relationships in a laboratory environment, but rather observed actual behaviors in the field by visiting dual-income and single-income families selected for the study and analyzing journals kept by the users themselves.

This study canvassed the needs that arise in the interactions between mother and child either between user and product through in-depth interviews, home tours, and childcare journals. In the interviews, basic information relating to childcare, including the parents' personal information, the average amount expended on childcare, and educational facilities used, were obtained. In addition, questions on everyday childcare in dual-income and single-income families were posed, with clear distinction between weekday and weekend activities. After the interview, home tours were undertaken to observe and photograph the products and environments used for childcare within the home. In addition, the mothers of the various families were asked to keep a childcare journal over a one-week period, in which they stated their satisfaction level for each activity recorded and the product used at the time, along with the corresponding reasons.

The experiment results showed that needs relating to time and schedules, communication, and playmates arose more frequently in dual-income families, while needs relating to emotional communication were more frequent in single-income families. Needs relating to monitoring occurred with similar frequency in both types of families. Based on the needs thus observed, design guidelines for childcare robots capable of fulfilling such needs were devised. This study contributes to robot design research by showing that childcare robots need to be endowed with differing characteristics according to usage environment, family makeup, living pattern, and espoused values.

4.3.3 Focus Group Interview for Designing a Growing Robot

This case study aimed at observing long-term interaction between humans and their non-human counterparts for the purpose of designing a growing robot. To analyze interactions between a human and a physically changing, growing counterpart, focus group interview with people who keep puppies, plants, or products emulating plants was conducted.

Although most HRI studies focus on immediate responses caused by immediate perception, the novelty effect stemming from immediate interaction between human and robot drastically decreases once curiosity toward a new robot has dissipated. Therefore, sustained long-term interaction is needed to supplement this novelty effect and strengthen the social bond between a human user and a social robot.

In this case study, focus group interviews were conducted on subjects who keep a variety of growing animals and objects (dogs, cats, Flip Flops, etc.), and the interviews content was geared to gauge how the subjects' sense of closeness to their pet, plant, or plant substitute changed through time. This change, along with raising information, stimuli, and condition of the pet or plant, was mapped onto a graph as shown below (Figure 13).

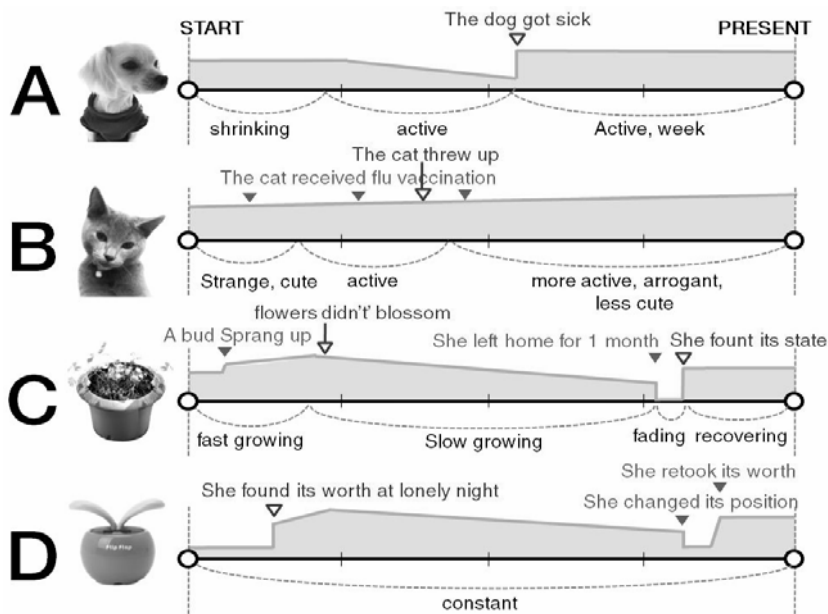


Fig. 13. Changes in the user's sense of closeness by object of relationship

The study arrived at the conclusion that people form special feelings for growing things or creatures, and see them as providing psychological stability, enjoyment, and recreational diversion. It also found that unexpected incidents and anticipated situations have a great impact on a user's actions and the sense of attachment he or she feels. Based on these results, this study can be said to have significance as basic research for the development of long-term interaction design between human users and growing robots.

5 Conclusion

Until now, robots have existed primarily as objects of experimentation within laboratory settings. However, robotics research is ultimately aimed at enriching the quality of life for human beings. Therefore, the need for realistic user-centered HRI design research is being raised as a means to create robots capable of fulfilling roles as helpers and partners for humans in everyday life. This chapter examined the definition, function, and methodology of user-centered design research, which supplements the

robot-centered technological development that heretofore formed the main method of research on human-robot interaction (HRI). User-centered HRI design research focuses on the user's senses, behaviors, and values in looking at design problems; this focus can be divided into the three elements of aesthetic contextuability, operational contextuability, and social contextuability. Accordingly, this type of research uses intuitive approaches that employ "look like," "function," and "role" prototypes, which users can actually

feel, communicate with, and judge. For each field of design addressed in user-centered HRI research, this chapter introduced the purpose, method, and results of representative case studies. Looking at recent trends in HRI research, it is possible to discern a transition from studies dealing with aspects of immediate sensory experience to those focusing on actions and behaviors; in the future, increasing emphasis will be placed on issues of value. None of the case studies introduced above are confined exclusively to the realm of aesthetic, operational, or social contextuability. This is because robots, by nature, perform interactions that are always accompanied by all three aspects. In the end, to design robots that can coexist naturally and usefully with humans in everyday life, a harmonious balance must be struck among the three elements of HRI research identified in this study.

The field of HRI research is so profound, extensive, and complicated that a human being engaging in such research may be compared to a creator making a creature, or robot, in his or her own image. For this reason, the boundaries between academic disciplines must be increasingly lowered and researchers in engineering, design, and the humanities must come together to ponder the many problems that remain.

References

- Bartneck, C., Forlizzi, J.: A design-centered framework for social Human-robot interaction. In: Proceedings of the 13th IEEE International Workshop on Robot and Human Interactive Communication, Piscataway, USA, pp. 31–33 (2004)
- Bartneck, C.: Rapid prototyping for interactive robots, In: Proceedings of the 8th Conference on Intelligent Autonomous Systems(IAS-8), [CD ROM], Amsterdam, Netherland (2004)
- Breazeal, C.: Social interactions in HRI: The robot view. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 34(2), 181–186 (2004)
- Buchanan, R.: Rhetoric, humanism, and design. In: Buchanan, R., Margolin, V. (eds.) Discovering design: Explorations in design studies, pp. 23–66. University of Chicago Press, Chicago (1995)
- Buchanan, R.: Design research and the new learning. Design Issues 17(4), 23–30 (2001) Card, S.K., Moran, T.P., Newell, A.: The Psychology of Human-Computer Interaction. CRC Press, Boca Raton (1983)
- Cross, N.: Design/Science/Research: Developing a discipline. In: Proceedings of the 5th Asian Design Conference, [CD ROM], Seoul, Korea (2001a)

Cross, N.: Designerly ways of knowing: Design discipline versus design science. *Design Issues* 17(3), 49–55 (2001b)

Dautenhahn, K.: Robots as social actors: AuRoRA and the case of autism. In: *Proceedings of the 3rd Conference on Cognitive Technology*, Los Alamitos, USA, pp. 359–374 (1999)

Fong, T., Nourbakhsh, I., Dautenhahn, K.: A Survey of socially interactive robots. *Robotics and Autonomous Systems* 42(3-4), 143–166 (2003)

Forlizzi, J.: How robotic products become social products: An ethnographic study of robotic products in the home. In: *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, New York, USA, pp. 129–136 (2007a)

- Forlizzi, J.: The product ecology: Understanding social product use and supporting design culture. *International Journal of Design* 2(1), 11–20 (2007b)
- Gould, J.D., Conti, J., Hovanyecz, T.: Composing letters with a simulated listening typewriter. *Communications of the ACM* 26(4), 295–308 (1983)
- Green, S.W., Jordan, W.P.: *Pleasure with Products: Beyond Usability*. CRC Press, Boca Raton (2002)
- Han, J., Kim, D.: Field trial on robots as teaching assistants and peer tutors for children. In: *Proceeding of the Asia Pacific International Symposium on Information Technology*, [CD ROM], Hangzhou, China (2006)
- Hassenzahl, M.: The effects of perceived hedonic quality on product appealingness. *International Journal of Human-Computer Interaction* 13(4), 479–497 (2002)
- Kanda, T., Hirano, T., Eaton, D., Ishiguro, H.: Interactive robots as social partners and peer tutors for children: A field trial. *Human-Computer Interaction* 19(1-2), 61–84 (2004)
- Kim, J., Kim, M.: A fundamental study design-centered HRI research framework. In: *Proceedings of 2005 Bi-annual Design Conference of Korea Society of Design Science*, Seoul, Korea, pp. 68–69 (2005)
- Lee, M., Kiesler, S., Forlizzi, J., Srinivasa, S., Rybski, P.: Gracefully mitigating breakdowns in robotic services. In: *Proceedings of Human Robot Interaction 2010*, [CD ROM], Osaka, Japan (2010)
- Lobach, B.: In: Lee, B. (ed.) *Industrial design*, Jo-hyung Kyou-youk, Seoul (2000)
- Mugge, R., Schoormans, J.P.L., Schifferstein, H.N.J.: Product attachment: Design strategies to stimulate the emotional bonding to products. In: Schifferstein, H.N.J., Hekkert, P. (eds.) *Product experience*, pp. 425–440. Elsevier, New York (2008)
- Norman, D.A.: *The psychology of everyday things*. Basic Books, New York (1998)
- Norman, D.A.: *Emotional design*. Basic Books, New York (2004)
- Oh, K., Kim, J., Kim, M.: Development of humanoid robot design process-focused on the concurrent engineering based humanoid robot design. In: *Proceedings of IASDR 2005 Conference*, [CD ROM], Yunlin, Taiwan (2005)
- Oh, K., Kim, M.: The HRI experiment framework for designers. In: *Proceedings of IASDR 2007 Conference*, [CD ROM], Kolon, Hong Kong (2007)
- Schon, D.A.: *The reflective practitioner*. Basic Books, New York (1983)
- Simon, H.A.: *The science of the artificial*. The MIT Press, Cambridge (1969)
- Swann, C.: Action research and the practice of design. *Design Issues* 18(1), 53–54 (2002)
- Wood, S.N., Walters, M.L., Koay, K.L., Dautenhahn, K.: Methodological issues in HRI: A comparison of live and video-based methods in robot to human approach direction

trials. In: Proceedings of Human Robot Interaction Conference 2006, [CD ROM], Utah, USA (2006)

Yang, M.C.: An examination of prototyping and design outcome. In: Proceedings of Design Engineering Technology Conferences and Computers in Information and Engineering Conference 2004, [CD ROM], Salt Lake City, USA (2004)

5. Methodology

Chapter 1

Introduction

Abstract End-of-life (EOL) treatment is a main stage in the life cycle of a product, and often aims to recover the remaining embodied value of the disposed products. Disassembly is a key activity in efficient EOL treatment. This chapter presents the general idea of the disassembly of products in view of EOL treatment. An introduction of the current research interest in this field, especially disassembly automation, is provided.

1.1 End-of-Life Product Treatment

Life Cycle Engineering (LCE) is an engineering discipline that focuses on a systematic approach to designing a product, considering its entire life cycle and incorporating the environmental aspects along with the economic, technical and social aspects during the product development [1]. The life cycle of products consists of four stages, namely the (a) *material stage*, (b) *manufacturing process*, (c) *usage*, and (d) *end-of-life (EOL)*, as shown in Fig. 1.1. The analysis of the entire life cycle is known as a “*cradle-to-grave*” analysis. In the beginning stages, the raw materials are produced by the material suppliers and then conveyed to the manufacturing process for production. After manufacturing, the products are distributed to the market and used by the consumer during the usage stage. Finally, the products are disposed of in the end-of-life stage.

EOL products disposed of in an inappropriate manner can result in environmental problems. EOL products that contain hazardous substances, such as electronic waste, require special treatment options. One can also attempt to recover the value remaining in the products; this recovery can take many forms, e.g. energy, materials, components, even entire products. These outcomes can be fed back into other stages of the product life cycle. This provides benefits not only in the environmental but also the economic aspect.

This book focuses on the EOL stage. Products arrive at the EOL stage in differing conditions according to the conditions experienced during the usage stage.

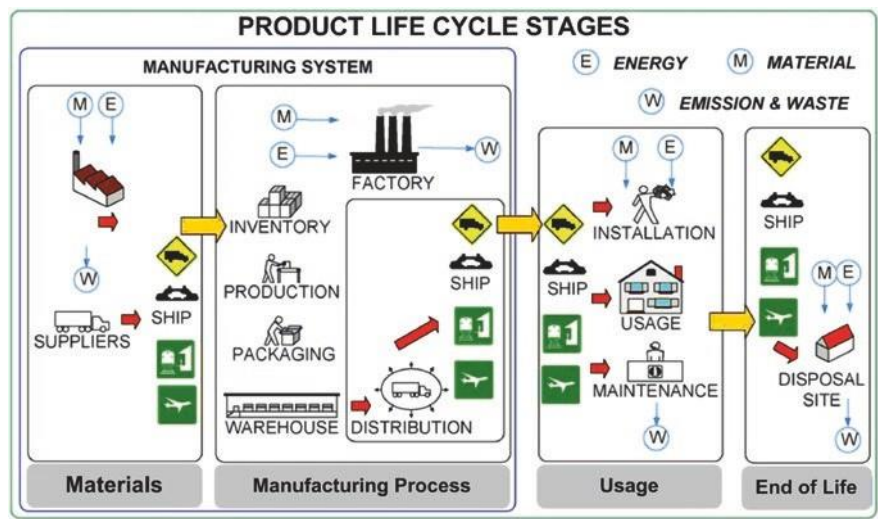


Fig. 1.1 Product life cycle stages [2]

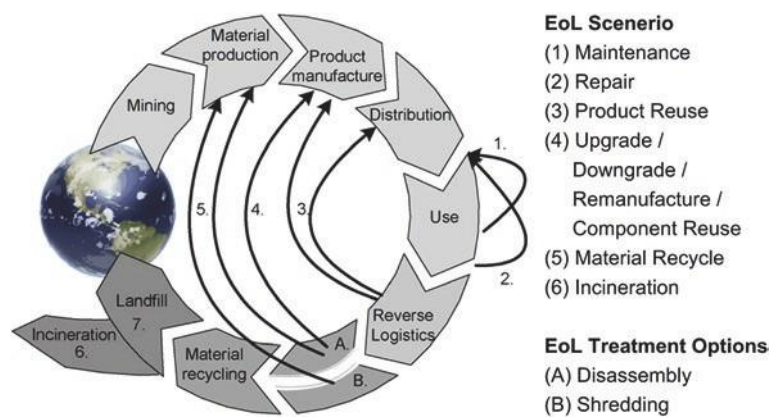


Fig. 1.2 Scenario of end-of-life products [3]

After the usage stage, the disposed EOL products are collected by the reverse logistics for appropriate treatment. Value recovery from EOL products in EOL treatment can be achieved by three major activities, namely (a) *reusing*,

(b) *remanufacturing*, and (c) *recycling*. The most appropriate process can be decided by considering the environmental and economic aspects, which can be analysed based on the Life Cycle Assessment (LCA) method.

From Fig. 1.2, EOL products that are in good condition can be immediately *reused* by distributing them back to the customers. These may also be sent back to the manufacturers for refurbishment. Non-functional EOL products can be

considered for *remanufacturing* or *recycling*. Before these treatment options can be carried out, the products¹ must be disintegrated into subassemblies,² components,³ or materials according to the requirements of each treatment option. Options for disintegration include *disassembly* and *shredding*.

Disassembly systematically separates the product into its components and/ or sub-assemblies [4]. The outcomes of disassembly can supply a wide range of treatment options, according to the desired purpose and downstream condition. Apart from EOL treatment, disassembly also serves the purpose of repair-maintenance if appropriate disassembly techniques are applied. A disassembly process can preserve the value of detached parts, providing neat and high quality outcomes suitable for refurbishment or reuse. However, disassembly processes are typically expensive due to the difficulty of the task and the demand for manual labour.

Shredding is a destructive process that breaks the products into small pieces or particles, eventually to be supplied to recycling processes. The outcome of shredding is a low-quality blend of material which requires a subsequent sorting process to separate the valuable material from the scraps. The material blend is sorted via their physical properties using a number of techniques e.g. magnetic, electrostatic, and eddy-current separation. Shredding is commonly implemented in industry due to the low operating cost. However, a major disadvantage is the loss of value of the parts and components. In addition, the shredding of components containing hazardous substances may also be problematic due to the contamination of the workplace and other materials in the process [5].

From Fig. 1.2, for *remanufacturing*, products are disintegrated into subassemblies, components, or parts, with minimal damage, in order to retain their functionality. The embodied value from manufacturing and the materials are hence recovered. On the contrary, for *recycling*, only the value of the materials is recovered. Disassembly can also be advantageous in obtaining a purer material fraction for recycling, with less sorting processes required.

In summary, the outcome of the disassembly process can be supplied to various EOL treatment processes, as summarised in Table 1.1. In comparison to the shredding process, disassembly tends to produce an outcome that is in better condition and better retains the embodied value

of the components. The high operating cost often exceeds the value recovered from the EOL products; hence, it becomes economically infeasible and is usually avoided in industry practice. For making disassembly economically feasible, much research has been conducted to develop the strategies and tools explained in the following sections.

¹ *Product* performs a particular functionality and consists of a number of discrete components and/or subassemblies connected together.

² *Subassembly* is a group of components connected together. It can be considered a *module* if the subassembly performs a particular functionality.

³ *Component* refers to a discrete part that cannot be further disassembled. (See detail in Sect. [2.1.2](#)).

Table 1.1 Destination of output of a disassembly facility (Adapted from [5])

Process outcome	Outcome of disintegration			Destination to EOL treatment process				
	No process	Disassembly	Shredding	Refurbish	Remanufacture	Reuse	Recycle	Incinerate/Landfill
								1

Unprocessed products	•			•			•	•
Modules or sub-assembly		•		•	•		•	•
Components		•				•	•	•
Damaged components		•					•	•
Waste			•					•

1.2 Disassembly of Products

The EOL treatment process has become more of concern due to the increasing number of products disposed of globally. In the field of LCE, much research focuses on products which have a high material-return rate, short life-cycle, and high volume of waste, e.g. Waste Electrical and Electronic Equipment (WEEE) [6] and EOL vehicles. In 2005, the amount of WEEE in EU-27 was around 8.3–9.1 million tons per year, consisting 40 % of large household appliances and 25 % of medium-sized household appliances. This figure is expected to grow 2.5–2.7 % each year [7]. Moreover, the amount of WEEE is 8 % of the whole municipal solid waste (MSW) worldwide [8]. In 2006, around 230 million cars were in use in Europe (EU-15) with

10.5 million tons of vehicle waste being disposed of every year. According to the European Directive 2000/53/EC, at least 80 % by mass of the EOL vehicles must be reused and recovered; 85 % of this must be recycled [9].

A number of investigations have been conducted in regard to the environmental and economic aspects of EOL disassembly processes [10–12]. The disassembly approach has also been compared with conventional waste treatment options, i.e. disposal and landfill [13]. These investigations concluded that disassembly greatly benefits the environment but it is not an economical process due the excess *direct* and *indirect* costs. The direct costs are the costs involved in the disintegration of the product, relating to the labour costs, disassembly tools, disassembly system, etc. On the other hand, the indirect costs relate to the activities beyond the disintegration process, e.g. stocks and logistics. The disassembly process may become economically feasible if a more optimal disassembly strategy with respect to the costs and benefits can be implemented.

1.2.1 Research Directions

Ongoing research focuses on developing a strategy for economically feasible disassembly. Gupta and McLean [14] provide an overview of the research direction

which can be categorised into four relevant areas: (a) *Design for Disassembly*, (b) *disassembly process planning*, (c) *design and implementation of disassembly systems*, and (d) *operations planning issues in the disassembly environment*.

Design for Disassembly (DfD) aims to resolve the difficulty in the disassembly process from the beginning, by designing the product to facilitate disassembly. This can bring a number of benefits, such as a reduction of the work needed in disassembly, quick and easy disassembly operations, simple product configuration, and easy handling. The guideline is presented in Boothroyd and Alting [15]. This area also includes *disassembly-embedded design* [16] and active disassembly [17]. *Disassembly process planning* deals with the development of rules, procedures, and software tools, used for formulating disassembly strategies and the disassembly system. Development of optimal disassembly sequence plans is one major theme in this area.

Design and implementation of disassembly systems deals with disassembly in the management level. This includes the methods used for establishing disassembly facilities and formulating economic and environmental evaluations of systems, as well as logistic networks for EOL treatment.

Operations planning issues in the disassembly environment deals with disassembly in the operational level. The operation level is affected by the logistics network and also concerns problems such as resource availability, collection and scheduling.

1.2.2 ***Automated Disassembly***

In addition to these research areas, automated disassembly can also be a key in achieving future economic feasibility for disassembly. Currently, the disassembly process is generally conducted by human labour, which results in high operating costs, especially in developed countries. A number of attempts have been made to automate the process. However, variations and uncertainties in product and process have restricted the adoption of automated disassembly in industry to only a few strategic instances, where a manufacturer has selected to remanufacture its own products. Developing a system that is flexible and robust enough to overcome these difficulties, particularly as an entity separate from the product manufacturer, is a challenging problem and the theme of this book.

The automatic disassembly system can be considered a robotic system that employs an artificial intelligence (AI) agent controlling the mechanical operation units and sensors. The system achieves flexibility by perceiving relevant information during the process and adapting the operation accordingly. The development of these systems involves a number of engineering disciplines in addition to the field of disassembly:

- Product analysis;
- Disassembly process planning;
- Mechanical and control systems;
- Vision and sensor systems; and,
- Intelligent planners.

This book aims to presents the background knowledge and a case-study imple- mentation according to these disciplines. The literature, regarding the existing research, theory and practice in the relevant areas, is also reviewed.

References

1. SMLCE (2013) What is Life Cycle Engineering, Sustainable Manufacturing & Life Cycle Engineering Research Group @ UNSW. <http://www.lceresearch.unsw.edu.au>
2. Kara S, Ibbotson S (2011) Embodied energy of manufacturing supply chains. CIRP J Manufact Sci Technol 4(3):317–323

3. Duflou JR, Seliger G, Kara S, Umeda Y, Ometto A, Willems B (2008) Efficiency and feasibility of product disassembly: a case-based study. *CIRP Ann Manuf Technol* 57(2):583–600
4. Kaebnick H, Ibbotson S, Kara S (2007) Cradle-to-cradle manufacturing. In: *Transitions: pathways towards sustainable urban development in Australia*, CSIRO Press, Australia, pp 521–536
5. Lambert AJD, Gupta M (2005) *Disassembly modeling for assembly, maintenance, reuse, and recycling*. CRC Press, Boca Raton
6. Parliament (2003) Directive 2002/96/EC of the European Parliament and of the council on waste electrical and electronic equipment (WEEE) of 27 Jan 2003
7. Jaco H, Federico M, Ruediger K, Claudia M, Clara D, Eniko A, Josef S, Ab S (2008) Final Report. Review of Directive 2002/96 on Waste Electrical and Electronic Equipment (WEEE). United Nations University
8. Babu BR, Parande AK, Basha CA (2007) Electrical and electronic waste: a global environmental problem. *Waste Manage Res* 25(4):307–318
9. Viganò F, Consonni S, Grosso M, Rigamonti L (2010) Material and energy recovery from Automotive Shredded Residues (ASR) via sequential gasification and combustion. *Waste Manage* 30(1):145–153
10. Chen KZ (2001) Development of integrated design for disassembly and recycling in concurrent engineering. *Integr Manuf Syst* 12(1):67–79
11. Gungor A, Gupta SM (1999) Issues in environmentally conscious manufacturing and product recovery: a survey. *Comput Ind Eng* 36:811–853
12. Li W, Zhang C, Wang HPB, Awoniyi SA (1995) Design for disassembly analysis for environmentally conscious design and manufacturing. In: *ASME International Mechanical Engineering Congress and Exposition*, pp 969–976
13. Ewers H-J, Schatz M, Fleischer G, Dose J (2001) Disassembly factories: economic and environmental options. In: *IEEE international symposium on assembly and task planning*, pp 447–452
14. Gupta M, McLean CR (1996) Disassembly of products. Paper presented at the 19th international conference on computers and industrial and engineering, Computer Industrial Engineering
15. Boothroyd G, Alting L (1992) Design for assembly and disassembly. *CIRP Ann Manuf Technol* 41(2):625–636
16. Masui K, Mizuhara K, Ishii K, Rose C (1999) Development of products embedded disassembly process based on end-of-life strategies. In: *The EcoDesign'99: 1st international symposium on environmentally conscious design and inverse manufacturing*, Tokyo, pp 570–575
17. Chiodo JD, Billett EH, Harrison DJ (1999) Active disassembly using Shape Memory Polymers for the mobile phone industry. In: *IEEE international symposium on electronics and the environment*, pp 151–156

Chapter 2

General Disassembly Process

Abstract The economic feasibility of the disassembly process is a main issue restricting its implementation in industry practice. Much research in the planning of disassembly processes and operations has been conducted in order to increase its economic feasibility. This chapter presents various aspects of the disassembly process including product representation, disassembly sequence planning (DSP), and dismantling techniques. This general knowledge is not limited to manual disassembly, but is also useful in automatic disassembly, which is presented in the following chapter.

2.1 Disassembly Process Planning (DPP)

The disassembly process is generally economically infeasible due to the difficulties in the process. Designing products according to Design for Disassembly (DfD) guidelines [1, 2] is expected to resolve this problem by making the disassembly process easier. However, few products nowadays are actually designed according to DfD. Therefore, the disassembly process remains difficult for the majority of products. Hence, this book focuses on

the means of improving the economic feasibility of disassembly apart from DfD.

Duflou et al. [3] summarise the factors that influence profitability of the disassembly process. Two major factors which are further explained in this book are the (a) *completeness of disassembly* and (b) *degree of autonomy* of the process. The desired completeness or depth of disassembly is a question addressed in disassembly process planning and disassembly sequencing, and is further explained in Sect. 2.2. The degree of autonomy can vary from complete manual disassembly to semi-automatic disassembly and fully-automatic disassembly. Since automation is a major theme of this book, an overview is explained in Chap. 3, with other details presented throughout the rest of the book.

This chapter gives an overview and literature review regarding disassembly process planning.

2.1.1 Difficulties in Disassembly

The disassembly process cannot be considered as the reverse of assembly. This is largely due to increased uncertainties: the disassembly process deals with unpredictable characteristics in both the quality and quantity of EOL products. This causes disassembly to be more difficult than assembly in the following aspects.

Uncertainties within models

Gungor and Gupta [4] summarise the physical uncertainties that can be found in EOL products manufactured under the same model. These uncertainties result from (a) *component defects*, (b) *upgrading or downgrading during usage* and (c) *damage during the disassembly operation*.

Defective main or connective components can result in difficulties during the removal operation which range from undesirable to dangerous. Examples include chemical leakage from batteries and broken fasteners that cannot be disestablished using common disassembly tools.

Upgrading and downgrading of the product during the usage stage can result in a change in product and component configuration. This situation is commonly found in devices containing exchangeable modules such as the personal computer (PC). Repairs and upgrades, e.g. involving the installation of random access memory (RAM) or graphics card, are common during the usage stage.

Damage in the disassembly process potentially occurs when the returned product is fragile. The disassembly process may need additional steps or a change in the disassembly sequence when certain parts are likely to break during the process.

Model-related variations

Products are manufactured into different models within the same product family. Different models contain variations in characteristics, including material, size and internal configuration. The same model may also be sold under different brands. Optimally, model information should be obtained from a product design database, taking the form of well-documented product specifications or a Computer-Aided Design (CAD)

model. Unfortunately, this information is usually unavailable by the time EOL products are returned. Therefore, the disassembly process needs to deal with incomplete product information, some of which is only revealed during the disassembly process. The challenge is to develop a disassembly plan that is general enough to deal with these uncertainties [5].

Difficulty of operations

Kroll et al. [6] define the term *disassemblability* to quantify the ease of product disassembly. A product is assessed for disassemblability according to the difficulty of the disassembly operation, by assessing it against five major criteria:

(a) *component accessibility*, (b) *precision in locating the component*, (c) *force required to perform tasks*, (d) *additional time*, and (e) *special problems* that cannot

be categorised in the other areas. Mok et al. [7] summarise the characteristics of an ease of product disassembly as follows:

- Minimal force exertion;
- Quick operation without excessive manual labour;
- Simple mechanism of disassembly;
- Minimal use of tool: ideal disassembly should be performed without tools;
- Minimal part repetition: parts easy to identify at each state of disassembly;
- Easy recognition of fasteners;
- Simple product structure; and,
- Avoidance in usage of toxic material.

Gupta and McLean [8] state that the development of optimal disassembly plans relies on four key phases: (a) *product analysis*, (b) *assembly analysis*, (c) *usage mode and effect analysis* and (d) *dismantling strategy*. Firstly, the product must be analysed and represented systematically. Options regarding the disassembly process can be generated from or represented using the product structure. The process can be considered at two levels, which are the *sequence plan* and the *operation*. The completeness of disassembly is considered a part of the sequence plan.

In summary, the uncertainties and variations found within returned products leads to uncertainties in the disassembly process. Uncertainties and variations of the disassembly process are summarised in Table 2.1.

Table 2.1 Summary of variations and uncertainties in the disassembly process

Category	Specific issues
Uncertainty in EOL condition	• Modification of product during usage phase
	• Condition of product
	• Condition of main component
	• Condition of connective component
Diversity of the supplied products	• Main product structure
	• Physical appearance of components
	• Quantity of components
	• Location of components
	• Variation in manufacture (quality)

Complexity in process planning and operations	• Disassembly sequence plan
	• Disassembly operation plan (considering prior actions)
	• Disassembly process parameters
	• Capabilities and limitations of detection technique
	• Precision of robot’s sensors—actuators
External factors	• Technology and design changes
	• Market driven factors

2.1.2 Representation of Product Structure

The structure of a product consists of (a) *components* and (b) *connections* [9]. A *component* is an element that keeps its extrinsic properties, i.e. functionality and material properties, after being detached from the product. A component cannot be further dismantled without using destructive disassembly methods. The *connection* or *liaison* is a relation that physically connects two components to restrict the motion between them. The task of disassembly is to disestablish these relations in order to separate the relevant components.

Fasteners

A fastener is a component or design element that serves the purpose of connecting other (main) components. Fasteners that are insignificant to the goal of disassembly may be modelled separately to the main components. Lambert and Gupta [9] considers such fasteners as *quasi-components*, which can be discrete components (e.g. screws, rivet, cable, etc.) or part of the main component (e.g. snap-fits). Connection-establishing elements, such as solder and weld joints, that do not form a component in themselves, can be considered *virtual components*.

Product structure

The structure of a product can be represented in many ways, of which two will be detailed here: the *connection diagram* and the *disassembly matrix*.

First, the *connection diagram* (liaison diagram) graphically represents the complete product structure using an undirected graph. The components are represented by nodes and the connections by arcs. According to the level of detail, the graph can be shown in three different forms: (a) *extended form* (b) *reduced form* and (c) *minimal form* (see Fig. 2.1).

In Fig. 2.1a, the product is a composition of three main components—A, B, and C. A and B are connected by mating. B and C are connected with a screw E which is considered a quasi-component. C and A are connected by a weld D which is a virtual component. The *extended form* shows full details of the

product with every component and fastener. All fasteners, including virtual connections, are modelled (see Fig. 2.1b). The *reduced form* represents the structure more concisely by hiding the virtual components and using dashed lines for quasi-components. In this case,

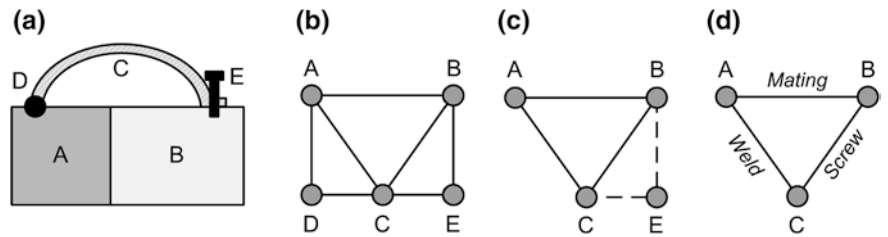


Fig. 2.1 Connection diagram [9]. **a** Product assembly. **b** Extended form. **c** Reduced form. **d** Minimal form

the connections associated with the virtual component, D–A and D–C, are removed. As a result, only connection A–C is retained representing the weld (see Fig. 2.1c). The *minimal form* shows the structure of the product in the most compact way by hiding both virtual and quasi-components. This form represents the product in the simplest way while preserving the information regarding the main components (see Fig. 2.1d).

Second, the product structure can be represented by a *disassembly matrix*, which a computing approach (e.g. Linear Programming (LP) or Integer Programming (IP)) can be used to solve the disassembly planning problem. The disassembly matrix is an $N \times N$ connectivity matrix where N is the number of the components. Each element of the matrix represents the existence of a connection between two corresponding components: “1” if a connection exists, and “0” if it does not. This information is completely represented by the lower left part of the matrix, since the matrix is symmetric and the elements on the diagonal axis are non-applicable. From this matrix, it is clear that the maximum number of connections is $\frac{1}{2}(N)(N - 1)$. The disassembly matrix of the example product in Fig. 2.1a is shown in Eq. (2.1).

	A	B	C	D	E
A					
B	1				
C	1	1			
D	1	0	1		
E	0	1	1	0	

Disassembly Matrix =

(2.1)

2.1.3 Disassembly Process Representation

The steps of the product disassembly process and their corresponding relationships can be schematically represented in many ways. Lambert and Gupta (2005) [9] summarise these approaches as follows:

Disassembly precedence graph

The disassembly precedence graph expresses sub-tasks of the disassembly process connected and constrained by precedence relationships. This can be represented in two forms: as a *component-oriented* or *task-oriented* graph (see Fig. 2.2). The arrows communicate the ordering in which tasks must be performed. This technique was originally used for assembly process representation and assembly line-balancing problems. Gungor and Gupta [10] introduce this to the disassembly process due to its simplicity. However, a major disadvantage is that a complete disassembly sequence cannot be expressed in one graph [11].

Disassembly tree

The disassembly tree expresses all possible choices for disassembly sequences, and is derived from a table containing all possible sequences sorted by level and

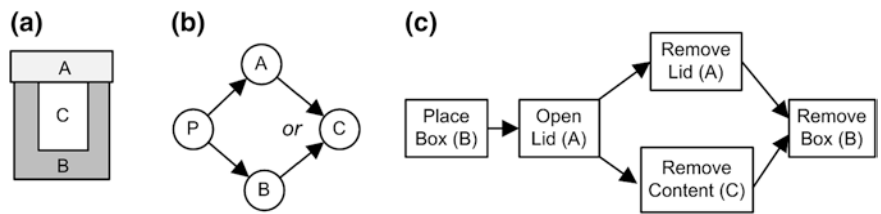


Fig. 2.2 Disassembly precedence [9]. a Assembly. b Component-oriented. c Task-oriented

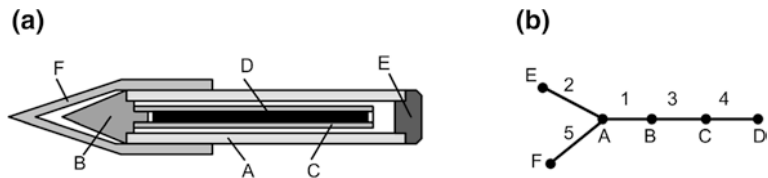


Fig. 2.3 Example product Bourjault's ballpoint [9]. a Assembly. b Connection diagram

operation type. A widely-used example is the *Bourjault tree* [12]. Two major drawbacks are the complexity arising in complex products and difficulty in rep- resenting parallel operations. Figure 2.4 shows a Bourjault tree representation of the disassembly process of a sample product, the Bourjault's ballpoint, which is shown in Fig. 2.3. This product will also be used to demonstrate the representation methods described in the following sections.

State diagram

The state diagram represents the disassembly sequence as an undirected graph, where each node represents a state of disassembly. This can be categorised into two approaches: (a) *connection-oriented* [13] and (b) *component-oriented* [14, 15] (see Fig. 2.5). All possible combinations of connections are represented by the nodes. Each edge represents the establishment or disestablishment of a connection. The major advantages are that the disassembly sequence of the complete product can be demonstrated in one diagram, and the diagram is compact even for complex products. However, state diagrams are unable to show how the

disestablishment of some connections cannot be done individually without affecting a combination of related connections.

Kara et al. [16] used a connection-oriented state diagram representation to develop a graphical representation method, the *disassembly-sequence diagram*, for representing the disassembly sequence to and from different stages of the process for selective disassembly. This diagram can be automatically generated from the liaison and precedence relations. An example is shown in Fig. 2.6.

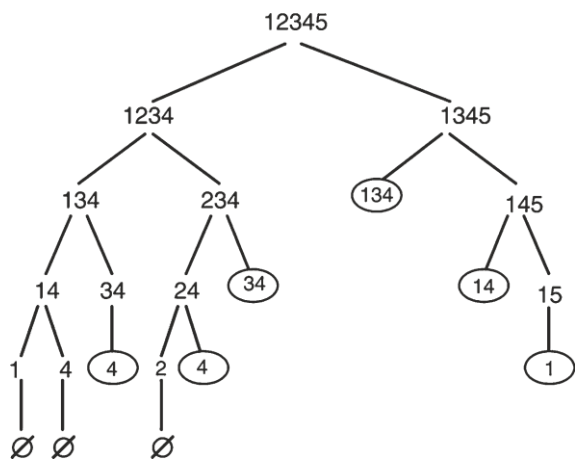


Fig. 2.4 Disassembly tree of the Bourjault's ballpoint [9]

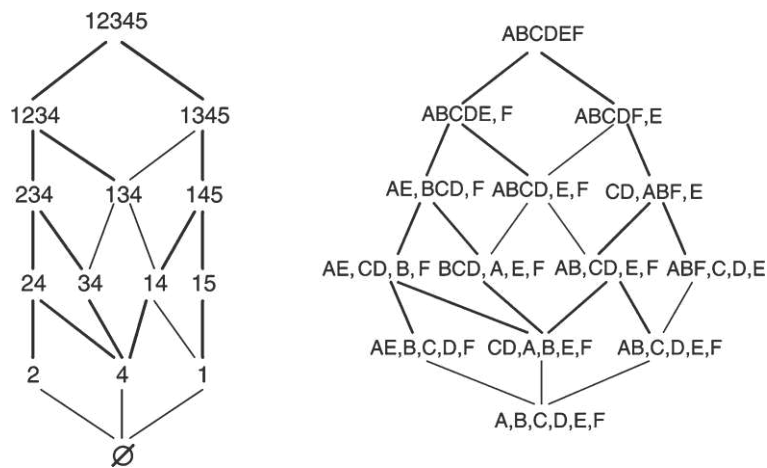


Fig. 2.5 State diagram of the Bourjault's ballpoint [9]

AND/OR graph (Hypergraph)

This graph represents disassembly sequences based on subassemblies. A process is represented by multiple-arcs (hyper-arcs) pointing from a parent to its child components (subassemblies) (see Fig. 2.7). This overcomes the drawback of the state diagram. However, a major drawback is the

complexity of the visual representation, which may become difficult to read when the number of components increases. Lambert [17] proposes a simplified version of this graph named the *concise AND/OR graph*. Further developments, aimed at representing the product model and its constraints more accurately, include the arborescence with hyper-graph [18], Petri net [19], and Hybrid graphs [20].

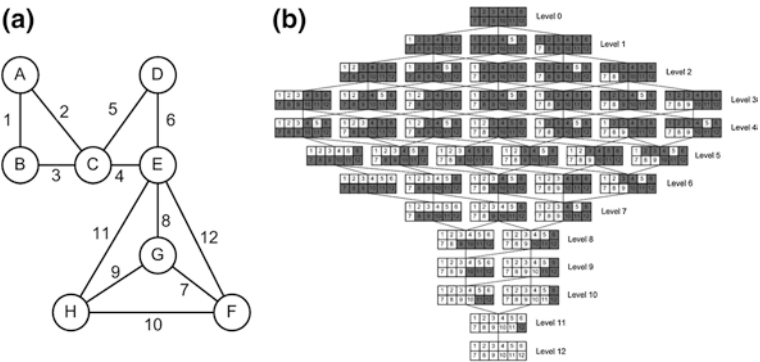
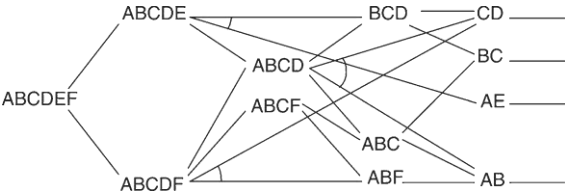


Fig. 2.6 Disassembly-sequence diagram [16]. **a** Liaison diagram. **b** Disassembly-sequence diagram

Fig. 2.7 AND/OR graph of the Bourjault's ballpoint [9]



2.1.4 Disassembly Sequence Planning (DSP)

A disassembly sequence is a procedure for the disestablishment of connections and detachment of parts in the disassembly operation. The initial state is defined as the complete product, and the final state, as a state where all desired components or subassemblies have been separated. The main purpose of disassembly sequence planning (DSP) is to find the optimal sequences of disassembling products with respect to certain factors, e.g. cost-effectiveness, material return, component recovery, and duration of operations. Theoretically, the number of possible sequences increases exponentially according to the number of

components. Therefore, finding the optimal solution is considered an NP-complete optimisation problem [4].

Lambert [5] summarises effective methodologies based on a product-oriented approach as follows. As adaptability is required for a flexible automatic disassembly system, the main theme of this book, emphasis is placed on the adaptive planners.

Mathematical programming (MP) method

The mathematical programming (MP) method aims to make the internal variables converge to their optimum value without considering the complete search space. The problem model is derived from a hypergraph (AND/OR graph). Costs are assigned to each action (arc) with respect to subassembly components (i.e. parent

and child) and stored in a transition matrix. This can then be effectively solved by mathematical solvers, e.g. using Linear Programming (LP), Mixed Integer Programming (MIP), or Dynamic Linear Programming (DLP). Petri nets are also used in case of a dynamic approach.

Heuristic methods

Gungor and Gupta [21] present a heuristic algorithm used to find near-optimal solutions to the disassembly sequencing problem. Near-optimal solutions are considered instead of optimal solutions, which are sometimes difficult to find due to the size of the search space. This method requires information of the precedence relationship among each of the components and the difficulty in performing each action. Efficiency is evaluated by the authors based on disassembly time. A case study regarding the DSP of a cell phone using the heuristic method and different search algorithms, e.g. greedy k-best and A*, is examined by Lambert and Gupta [22].

Artificial intelligence (AI) methods

Various techniques are used in artificial intelligence to generate and utilise constraints and reduce the size of the search space. Lambert [5] reviews typical AI techniques for disassembly sequence planning, including simulated annealing algorithms, genetic algorithms (GA), fuzzy sets, neural networks, multi-agent systems, and Bayesian networks. Other novel algorithms that have been efficiently applied to DSP include ant-colony optimisation [23], case-based reasoning [24] and rule-based sequence generation on clustering graphs [25].

Adaptive planner

An adaptive planner generates a disassembly sequence with respect to the uncertainties and unexpected circumstances encountered during the disassembly operation. Due to its particular relevance to automated disassembly, a number of publications relating to adaptive planners have been reviewed in this section. The literature handles the problem at two levels: the (a) *process planning level* and (b) *operation level*.

In the *process planning level*, Tang [26] proposes using a Fuzzy Petri net to model the dynamics of disassembly, including the uncertainties in product condition and human factors. The system is trained with data and feedback from the actual disassembly, and selects the appropriate disassembly plan based on past experience. Turowski et al. [27] presents an

implementation of a Fuzzy Coloured Petri Net for balancing a disassembly line. Grochowski and Tang [28] propose a learning approach using a Disassembly Petri Net (DPN) and Hybrid Bayesian network. Veerakamolmal and Gupta [29] propose using case-based reasoning (CBR) to generate disassembly plans for multiple products. The plan for a new product is adapted from existing plans by deriving it from a base case. Gao et al. [30] propose using a Fuzzy Reasoning Petri Net to adaptively generate the disassembly sequence according to the condition of the product observed at each state. Decisions are made based on the estimated value returned, hazard level, and disassembly cost.

In the *operation level*, Salomonski and Zussman [31] propose using a predictive model with DPN to adaptively generate the disassembly process plan according to real-time measurements conducted by a robot arm. Lee and Bailey-Van Kuren [32] address the uncertainties in the operation level by automatically recovering from a visually-detected error. In addition, Martinez et al. [18] propose a dynamic sequence generation method that generates an optimal disassembly plan during operations in response to unpredictable situations, e.g. failure to remove a corroded part, replacement of screws, etc. This system is modelled and controlled by a *multi-agent system* (MAS). ElSayed et al. [33] use GA to generate an optimal disassembly sequence according a supplied bill-of-materials (BOM) and components detected in real time. Relations defined in the original BOM must be preserved.

In conclusion, the existing adaptive planners deal with many types of uncertainty experienced during the disassembly process. The uncertainties relate to variations in the component conditions that deviate from the ideal case. The ability for an adaptive planner to handle these uncertainties stems from its ability to appropriately adapt existing knowledge into a new plan according to sensed information. Machine learning techniques are used to allow the system to improve its performance from past experience. However, the structure of the product, e.g. BOM and CAD model, generally needs to be supplied a priori. A methodology accounting for an uncertainty in the general product structure in real-time has not yet been proposed in any research. In addition, the learning process has only been implemented in the planning level. Hence, learning at the operation level, such as in optimising process parameters, should be further investigated.

2.2 Completeness of Disassembly

The completeness of disassembly can be categorised into two types: (a) *complete disassembly* and (b) *incomplete disassembly*. A *complete* or *full disassembly* is the process that separates every single component of the product. This is rarely done due to the technical constraints (particularly the

complexity and the uncertainties in the operation) and high labour costs. On the other hand, the *incomplete* or *selective disassembly* separates only the desired components or subassemblies, and terminates when the desired depth of disassembly is reached. Disassembly becomes more cost-efficient with a strategic choice of disassembly targets. Reasons for selective disassembly include recovering modules or components for use as spare parts, separating those that contain hazardous substances, and improving the quality and quantity of shredder residue[17].

Figure 2.8 illustrates the situation in maximising the profit in the disassembly process. The disassembly range refers to the completeness of disassembly. The cost of disassembly is due to operation time, varying according to the number and type of connections to be disestablished. This increases with the desired completeness of disassembly. Disassembly is economically feasible when the total profit from treating or recycling all products exceeds the cost of disassembly. The optimal strategy is the point at which the maximum profit can be obtained [34,35].

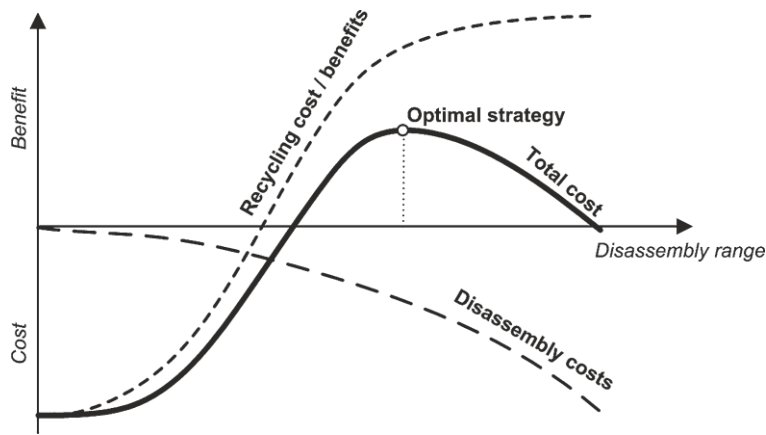


Fig. 2.8 Determination of optimal disassembly strategy [34]

The outcome of selective disassembly can be one of the three following types [9].

- **Homogeneous components:** parts that cannot be physically disassembled.
- **Complex components:** components comprised of a number of homogeneous components, joined together with fasteners, which can only be separated using destructive disassembly.
- **Modules:** sets of components that perform a self-contained function.

Modules

can be further disassembled via non-destructive or semi-destructive operations. However, maintaining their original condition and functionality can allow reuse of the entire module.

The researchers currently focus on developing a methodology to find optimal disassembly sequences in which the completeness of disassembly is taken into account. Kara et al. [36] propose the methodology of developing the optimal selective disassembly sequence which is the reverse of the methodology for assembly presented by Nevins and Whitney [37]. The disassembly sequences are generated from the product specifications, namely list of parts and subassembly, precedence rules, product representation model, and disassembly sequence diagram. Subsequently, the optimal sequences for removing the selected parts are obtained by removing invalid sequences according to liaison analysis. In regard to this concept, software that automatically generates and visualises

optimal sequences of selective disassembly from specified constraints is developed by Kara et al. [16, 38].

2.3 Disassembly Operations

2.3.1 *Types of Fasteners*

The disassembly operation is divided into two main tasks: *disestablishing fasteners* and *detaching main components*. The main component is detachable if the associated

Table 2.2 Disassembly methods according to fastener type

Discrete components	Not deformed	Bundling	Shear cut
		Spring	Deform/pull
		Screw, bolt, nut, washer	Unscrew, drill
	Reversibly deformed	Cotter pin, staple	Pull
	Irreversibly deformed	Rivet	Pry out, drill
		Adhesive: glue, seal	Peel, pry out, break
Parts of components	Reversible connection (semi-reversible)	Surface: mating	Remove
		Surface: press fit	Pull, pry out
		Snap fit	Deform, pry out, pull
	Irreversible connection	Surface: press fit	Pull, pry out
		Surface: mould	Break
		Seam fold	Deform
		Seal	Peel, pry out, break
Virtual components	Irreversible	Solder	Shear cut, break, melt
		Weld	Saw cut, break

connections are located and disestablished. Specific techniques are required for effectively disestablishing different types of fasteners. Lambert and Gupta (2005)

[9] categorise the fasteners commonly found in mechanical and electronic-electrical products into 13 types. Different types of fasteners require different disestablishment methods and display different levels of reversibility. A summary of fastener types and their respective disassembly methods is shown in Table 2.2.

2.3.2 *Dismantling Techniques*

Disassembly operations can be broadly categorised into three types: (a) *non-destructive*, (b) *semi-destructive*, and (c) *destructive* disassembly. The characteristics of each category are explained in detail as follows.

Non-destructive disassembly

All outputs of non-destructive disassembly remain undamaged. This is desired for maintenance, component reuse and remanufacture. All fasteners within the product must be reversible or semi-reversible. The dismantling of reversible fasteners (e.g. screws) is generally easier than that of semi-reversible fasteners (e.g. snap-fits). The operation cost is generally high, as high flexibility is required, particularly due to difficulties such as rust and partial damage. Even though a number of tools have been specially developed to facilitate non-destructive disassembly, e.g. for the disassembly of screws [39] and snap-fits [40], the non-destructive approach is still generally economically infeasible [3].

Semi-destructive disassembly

The semi-destructive approach aims to destroy only connective components, e.g. via breaking, folding or cutting, leaving main components with little or no damage. This increases the efficiency of the operation and has been found in many cases to be economically feasible. Many research works relating to automatic disassembly use semi-destructive techniques to overcome the uncertainties in the product condition and geometry. Examples of such techniques include the drilling out of screw heads during the disassembly of electric motors [41], creation of new surfaces allowing torque transmission for unscrewing [34] and cutting off of screw heads using a cut-off wheel [42].

Destructive disassembly

Destructive disassembly deals with the partial or complete destruction of obstructing components. Components or irreversible fasteners, e.g. welds, are destroyed using destructive tools such as a hammer, crowbar or grinder. These operations are fast, efficient and inherently flexible. As a result, destructive disassembly is economically feasible and commonly performed in industry practice. One common application of destructive disassembly is in the opening of a covering component to reach the more valuable components inside. Examples include the breaking of the separating line [34] and using plasma arc cutting to destroy the metal casing of consumer appliances [43].

In summary, semi-destructive and destructive disassembly allow techniques that are more capable of efficiently dealing with the uncertainties in product condition, therefore allowing more economically feasible operation. On the contrary, non-destructive disassembly tends to have high operation costs but may be unavoidable in maintenance or for component reuse. More detail regarding the operations and specially-developed tools can be found in Sect. 3.3.2.

2.4 Conclusions

Disassembly can be a key step in an efficient EOL treatment process, however, is usually economically infeasible due to high operating costs relating to the variation and uncertainties in the products and process, as summarised in Table 2.1. This chapter presents three major considerations which should be addressed to improve the economic feasibility of disassembly.

Firstly, the disassembly plan can be optimised with respect to a goal, which can be operating time or cost. A number of techniques regarding the representation of the product structure and disassembly process are described. With an appropriate representation, an optimal or near-optimal ordering of the disassembly operations can be found via various optimisation strategies described in Sect. 2.1.4. Particularly the adaptive planners are of interest, since they are able to respond to minor uncertainties like product damage. A significant amount of product knowledge, e.g. the product structure or a CAD model, are required before planning.

Secondly, regarding the completeness of disassembly, performing selective disassembly to a certain depth is more feasible than the full disassembly. The optimal disassembly depth should be determined during the planning phase.

Finally, the difficulty of the disassembly operations results from the type of fasteners used, and the product and fasteners' conditions. Different fasteners can be disestablished using different tools and techniques. Semi-destructive and destructive operations are generally preferable due to the shorter operating time and effective operation in spite of uncertainties.

In conclusion, the main source of difficulty in the disassembly process is the need to deal with a high level of uncertainties and variations. If the disassembly is not conducted by the product manufacturers, information regarding the product is generally at first incomplete. Even when the expected outcomes are known, poor product or fastener condition may require deviations to the usual plan. This causes higher operating time in manual disassembly, and is a primary factor hindering the industrial application of automatic disassembly.

References

1. Boothroyd G, Altling L (1992) Design for assembly and disassembly. *CIRP Ann—Manuf Technol* 41(2):625–636
2. Chiodo J (2005) Design for disassembly guidelines. *Active Disassembly Res.* http://www.activatedisassembly.com/downloads/ADR_050202_DFD-guidelines.pdf
3. Dufloy JR, Seliger G, Kara S, Umeda Y, Ometto A, Willems B (2008) Efficiency and feasibility of product disassembly: a case-based study. *CIRP Ann—Manuf Technol* 57(2):583–600
4. Gungor A, Gupta SM (1998) Disassembly sequence planning for products with defective parts in product recovery. *Comput Ind Eng* 35(1–4):161–164
5. Lambert AJD (2003) Disassembly sequencing: a survey. *Int J Prod Res* 41(16):3721–3759
6. Kroll E, Beardsley B, Parulian A (1996) A methodology to evaluate ease of disassembly for product recycling. *IIE Trans (Institute of Industrial Engineers)* 28(10):837–845
7. Mok HS, Kim HJ, Moon KS (1997) Disassemblability of mechanical parts in automobiles for recycling. *Comput Ind Eng* 33(3–4):621–624
8. Gupta M, McLean CR (1996) Disassembly of products. Paper presented at the 19th international conference on computers and industrial and engineering, Computer Industrial Engineering
9. Lambert AJD, Gupta M (2005) Disassembly modeling for assembly, maintenance, reuse, and

- recycling. CRC Press, Boca Raton, Fla.
10. Gungor A, Gupta SM (2002) Disassembly line in product recovery. *Ann Rev Control* 40(11):2567–2589
 11. Tumkor S, Senol G (2007) Disassembly precedence graph generation. In: SAM 2007—IEEE international symposium on assembly and manufacturing, pp 70–75
 12. Bourjault A (1984) Contribution à une approche méthodologique de l'assemblage automatisé: élaboration automatique des séquences opératoires (Contribution to a systematic approach of automatic assembly: The automatic generation of operation sequences). Université de Franche Comté, Besançon, France
 13. Fazio D, Whitney TL (1987) Simplified generation of all mechanical assembly sequences. *IEEE J Robot Autom* RA-3 (6):640–658
 14. Homem De Mello LS, Sanderson AC (1990) AND/OR graph representation of assembly plans. *IEEE Trans Robot Autom* 6(2):188–189

15. Woller JD (1992) A combinatorial analysis of enumerative data structures for assembly planning. *J Des Manuf* 2(2):93–104
16. Kara S, Pornprasitpol P, Kaebnick H (2006) Selective disassembly sequencing: a methodology for the disassembly of end-of-life products. *CIRP Ann—Manuf Technol* 55(1):37–40
17. Lambert AJD (1999) Linear programming in disassembly/clustering sequence generation. *Comput Ind Eng* 36(4):723–738
18. Martinez M, Pham V-H, Favrel J (1997) Dynamic generation of disassembly sequences. In: IEEE symposium on emerging technologies and factory automation, ETFA, pp 177–182
19. Zussman E, Zhou M, Caudill R (1998) Disassembly Petri net approach to modeling and planning disassembly processes of electronic products. In: IEEE international symposium on electronics and the environment, pp. 331–336
20. Wang Y, Li F, Li J, Chen J, Jiang F, Wang W (2006) Hybrid graph disassembly model and sequence planning for product maintenance. Paper presented at the IET conference publications
21. Gungor A, Gupta SM (1997) An evaluation methodology for disassembly processes. *Comput Ind Eng* 33(1–2):329–332
22. Lambert AJD, Gupta SM (2008) Methods for optimum and near optimum disassembly sequencing. *Int J Prod Res* 46(11):2845–2865
23. Shan H, Li S, Huang J, Gao Z, Li W (2007) Ant colony optimization algorithm-based disassembly sequence planning. In: IEEE international conference on mechatronics and automation
24. Shih L-H, Chang Y-S, Lin Y-T (2006) Intelligent evaluation approach for electronic product recycling via case-based reasoning. *Adv Eng Inform* 20(2):137–145
25. Kaebnick H, O'Shea B, Grewal SS (2000) A method for sequencing the disassembly of products. *CIRP Ann—Manuf Technol* 49(1):13–16
26. Tang Y (2009) Learning-based disassembly process planner for uncertainty management. *IEEE Trans Syst Man and Cybern—Part A: Syst Humans* 39(1):134–143
27. Turowski M, Tang Y, Morgan M (2005) Analysis of an adaptive fuzzy system for disassembly process planning. In: IEEE international symposium on electronics and the environment, pp 249–254
28. Grochowski DE, Tang Y (2009) A machine learning approach for optimal disassembly planning. *Int J Comput Integr Manuf* 22(4):374–383
29. Veerakamolmal P, Gupta SM (2002) A case-based reasoning approach for automating disassembly process planning. *J Intell Manuf* 13(1):47–60
30. Gao M, Zhou MC, Tang Y (2005) Intelligent decision making in disassembly process based on fuzzy reasoning Petri nets. *IEEE Trans Syst Man Cybern B Cybern* 34(5):2029–2034
31. Salomonski N, Zussman E (1999) On-line predictive model for disassembly process planning adaptation. *Robot Comput-Integr Manuf* 5(3):211–220
32. Lee K-M, Bailey-Van Kuren MM (2000) Modeling and supervisory control of a disassembly automation workcell based on blocking topology. *IEEE Trans Robot Autom* 16(1):67–77
33. ElSayed A, Kongar E, Gupta SM, Sobh T (2012) A robotic-driven disassembly sequence generator for end-of-life electronic products. *J Int Robot Syst: Theory Appl* 68(1):43–52
34. Feldmann K, Trautner S, Meedt O (1996) Innovative disassembly strategies based on flexible partial destructive tools. *Ann Rev Control* 23:159–164
35. Desai A, Mital A (2003) Evaluation of disassemblability to enable design for disassembly in mass production. *Int J Ind Ergon* 32(4):265–281
36. Kara S, Pornprasitpol P, Kaebnick H (2005) A selective disassembly methodology for end-of-life products. *Assembly Autom* 25(2):124–134
37. Nevins JL, Whitney DE (1989) Concurrent design of products and processes: a strategy for the

- next generation in manufacturing. McGraw-Hill, New York
38. Pornprasitpol P (2006) Selective disassembly for re-use of industrial products. University of New South Wales
39. Seliger G, Keil T, Rebafka U, Stenzel A (2001) Flexible disassembly tools. In: IEEE international symposium on electronics and the environment, pp 30–35

40. Braunschweig A (2004) Automatic disassembly of snap-in joints in electro-mechanical devices. In: the 4th international congress mechanical engineering technologies'04, Varna
41. Karlsson B, Järrhed J-O (2000) Recycling of electrical motors by automatic disassembly. *Meas Sci Technol* 11(4):350–357
42. Reap J, Bras B (2002) Design for disassembly and the value of robotic semi-destructive disassembly. In: ASME design engineering technical conference, pp 275–281
43. Uhlmann E, Spur G, Elbing F (2001) Development of flexible automatic disassembly processes and cleaning technologies for the recycling of consumer goods. In: IEEE international symposium on assembly and task planning, pp 442–446

6. Related Work

Chapter 3

Disassembly Automation

Abstract Disassembly is one of the key steps in the efficient treatment of End- Of-Life products, however has generally been neglected in industry practice due to cost constraints. This is because disassembly has traditionally been restricted to manual labour. Automation assisting or substituting for human workers may be a lower-cost option. However, the associated technical challenges need to first be addressed. This chapter presents the principle of disassembly automation, as well as the basic elements needed in a disassembly system. A number of research works in

regard to automated disassembly systems and innovative disassembly tools are also described.

3.1 Introduction

Nowadays, automation plays an important role in the modern manufacturing industry. This automation was introduced to the manufacturing industry almost a century ago, with the introduction of feedback control [1, 2]. Automation has in general proved to be more cost-effective than human workers. A major advantage of automation is the ability to handle repetitive tasks quickly with high precision and accuracy. In addition, robots can be applied to hazardous tasks and in working environments that are undesirable for the human workers due to e.g. contamination or radiation [3]. The skilled labour required to achieve the high accuracies, or the measures required to meet health and safety requirements for human workers lead to high operating costs, particularly in developed countries. The primary advantages that automation has given to the manufacturing process can be summarised as: (a) cost effectiveness, (b) efficiency and accuracy, and (c) the ability to distance human workers from hazardous activities.

Many similar issues are faced in the disassembly process. Economical infeasibility is a major reason behind disassembly being generally neglected by the EOL industry [4]. It is difficult to economically justify the implementation of manual disassembly due to the generally-low economic returns from EOL treatment. The costs are high,

as the process is generally labour-intensive, and is often met with complications due to the variations and uncertainties in EOL products returned. The process may also be hazardous depending on the subject of disassembly. For example, the disassembly of electric vehicle batteries deals with risks due to the chemical composition and any charge remaining in the cells [5]. Automation has a high potential to reduce these problems in the disassembly process [6]. A number of attempts have been made to implement automation in the disassembly process. However, apart from manufacturers' own efforts to remanufacture their own products, the latest developments are still in the research phase and have not yet been implemented at the industrial level. The problem relates not only to the technical but also the economic aspects. Scharke [7] summarises the challenges of automated disassembly as follows:

- the variety of manufacturers and different product types,
- variety in design and product structure,
- small lot sizes and insufficient collection logistic,
- varying product condition after the usage phase,
- changes in standard components,
- insufficient disassembly tools,
- changes in legislation, and
- changes in market demand and prices.

The uncertainties due to the variations in returned products make the disassembly process and facility hard to design and economically justify. This makes the process plans and operations more complicated than those required for assembly. The variety in product condition may occur due to component replacement, upgrading/ downgrading, and damage during the usage phase. In general, an optimised disassembly process can be planned according to sufficient product-specific information, e.g. computer aided design (CAD) model, bill-of-materials (BOM), or precedence relations between disassembly operations. Unfortunately, this knowledge is only available for a small subsection of products at the EOL stage.

For disassembly, manual labour has traditionally been preferred over automation due to the human's advantages in (a) *perception*, (b) *dexterity*, and (c) *intelligence*. These aspects allow a human to interact flexibly with its environment. *Perception* refers to the sensor systems ability to perceive information about the process. *Dexterity* relates to the proficiency of

carrying out the physical operation. *Intelligence* relates to the ability to plan and control the process according to the knowledge of the product. Unlike the assembly process, information about the product is often inaccurate or incomplete at beginning of the process, but is revealed during the process [8]. The successful implementation of disassembly automation requires significant improvement in these attributes, which may be achieved with the aid of the human worker.

In practice, to limit the variations encountered in the process, each system is designed for disassembling a particular product type or family. Therefore, the technical requirements of the dismantling tools, relating to e.g. the type of techniques, product size and weight, can be limited. A specifically-designed system can satisfy the cost constraint given a sufficiently-large lot size. However, the lot size of

returned EOL products is unpredictable and may lead to difficulty at the strategic planning level. In addition, the profit depends on the market value of the recycled and reused parts, which changes according to demand and legislation. Therefore, strategic planning regarding these aspects should be done before developing the automated system.

In summary, automation has the potential to increase the economic feasibility of the disassembly process, in reducing operating costs and risk to the human worker. However, the technical requirements must first be met: the system needs to be flexible and robust enough to deal with uncertainties in the product and process. Due to the high level of intelligence, perception, and dexterity required in the system, humans may still need to play specific roles, e.g. in supervision, high-level planning, or performing sophisticated tasks.

This chapter explains the concept of disassembly automation and state-of-the-art developments within the field. An overview of disassembly automation and the integration of the various components are explained in Sect. 3.2. An overview of aspects relating to mechanical design is given in Sect. 3.3. Finally, existing developments in disassembly automation are explained in Sects. 3.4 and 3.5. It is noted that the aspects relating to perception are described in Chap. 4 and cognitive robotics, the artificial intelligence technique used in this research, is explained in Chap. 5.

3.2 Principle of Disassembly Automation

Disassembly automation may refer to a single disassembly cell that is able to complete the key steps of the disassembly process, such as dismantling and separation, or a system comprised of many (potentially more specialised) cells. Configuring a disassembly system as a *modular system* is recommended by [9] to fulfil the economic and technical requirements, since a modular system allows the flexibility of future adaptations to other products with little modification in the hardware and software. Modules may be added, removed, or modified in order to suit the new products.

Products that have certain similar features, e.g. size and connection methods, may be grouped into *disassembly families*, as proposed by [10]. As the same operations are required to disassemble products in the same family, the same disassembly tools can be used. This method reduces the technical requirements of the modules needed for the disassembly system. In regard to a modular system, [6] summarise the basic modules as follows:

- industrial robots or manipulating devices;
- disassembly tools specially designed for robots and tasks;
- gripping devices;
- feeding systems of the supply products;
- transport systems;
- fixture systems;
- storage systems for parts and tool;
- manual disassembly stations;

- vision systems;
- sensor systems;
- intelligent control units; and
- product databases.

This comprehensive summary can be taken as a guide for the modules required in the disassembly system. Not all are necessary in every system, and should be adapted on a case-to-case basis.

Excluding any manual input to the process, the development of a disassembly automation system can be considered in three broad subsystems: (a) *mechanical design and control*, (b) *vision and sensors*, and (c) *planning and artificial intelligence*. For this research, these subsystems are implemented as distinct *operating modules*. Figure 3.1 illustrates the implementation and interconnections between the operating modules. An overview of the uncertainties and variations expected to be addressed by each of these modules is shown in Table 3.1.

Mechanical Design and Control

The disassembly system physically interacts with the products via its mechanical units. The mechanical design and control of the system focuses on the physics of these units and includes the machine design, kinetics, and kinematics of the units. Motion controllers control the system at the sensor-actuator level. Theory in the *control of dynamic systems* or *feedback control* [11] should be considered if the low-level development of these components is necessary.

A cell for disassembly automation consists of a number of mechanical components. The main component is a *robot* for the manipulation of disassembly tools. Various kinds of *disassembly tools* can be used, depending on the connections to be disestablished. These tools can be traditional hand or power tools, e.g. screwdriver, drill or grinder, or innovative disassembly tools designed specifically for the efficient disestablishment of connectors (e.g. snap-fits [12]). After the connectors have been disestablished, a *gripping device* suitable for the various geometry and dimensions of the parts is used to remove detached parts. A *tool changer* may be needed if the system requires multiple disassembly tools and/or grippers for the one manipulator. Products need to be fixed to the system via a *fixture* that is capable of dealing with the various geometry and dimensions of the products. Other mechanical supporting

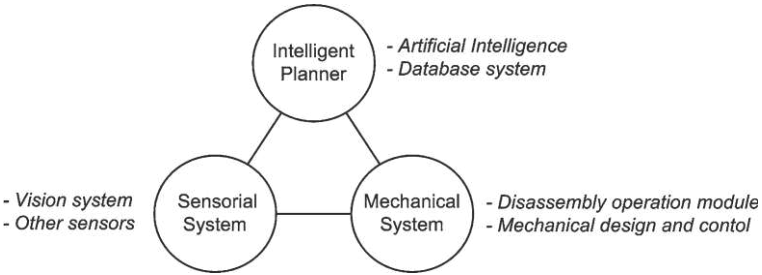


Fig. 3.1 Overview of the system

Table 3.1 Uncertainties and variations addressed by the three operating modules

Primary uncertainty	Specific issue	Related operating module		
		Intelligent agent	Vision system	Disassembly operation unit
Uncertainty in EOL condition	Modified product	•		
	EOL condition of product	•		
	EOL condition of main component	•		
	EOL condition of connective component	•		
Diversity of the supplied products	Main product structure	•		
	Physical appearance of the components		•	
	Quantity of the components	•	•	
	Location of the components		•	
	Variation in manufacture (quality)	•	•	•
Process and operations	Disassembly sequence plan	•		
	Disassembly operation plan (prior actions)	•		
	Disassembly process parameters	•		
	Capability of detection technique	•	•	•
	Precision of robot’s sensors—actuators	•	•	•
	Wear of disassembly tool	•		•
External factor	Technology and design change	n/a		
	Market driven factor			

components deal with the logistics of the system, including transportation, feeding, and storage of the products. *Transportation* is generally performed using conveyor systems, which can be designed to suit the configuration of the workstations.

Vision System and Sensors

Due to the uncertainties and variations in the supply, information about a product to be disassembled is generally at first incomplete, and revealed during the disassembly process. Sensor systems are used to perceive this information during the process. A sensor system capable of sensing the scene at a distance, e.g. using cameras or distance sensors, is generally used to obtain general information regarding the existence and location of the product and components. Other sensors, such as force and torque sensors, support the system by providing additional information regarding specific tasks [6]. The sensed information can be used at the planning and operational levels to adapt the process accordingly.

Planning and Artificial Intelligence

The planning and artificial intelligence subsystem is responsible for managing the data flow within the system, controlling the physical process to appropriately respond to the information received from the sensors and prior knowledge. This

can take the form of an *intelligent agent*. The process needs to be carried out according to a disassembly sequence plan (DSP) and disassembly process plan (DPP), which may be predefined or automatically generated by the agent. A number of artificial intelligence techniques have been applied in automated disassembly systems, e.g. genetic algorithms [13], Petri-nets [14] and cognitive robotics [15]. The technique defines the behaviour of the agent controlling the physical system.

Data is abstracted at various levels to reduce the complexity of the planner. In this case, the agent controls the system by supervising other operating modules, which handle details at a lower level. As a result, the desired behaviour of the agent can be better formulated and generalised. The levels of control are a key aspect of the system architecture and should be determined prior to the implementation of this subsystem. As an example, the control layers used in this research are shown in Table 3.2. This system consists of three levels of control: (a) *high-level*, (b) *mid-level*, and (c) *low-level*.

The intelligent agent or planner controls the system at the *high-level*. The agent generates the disassembly sequence plan and process plan. The process plan is a sequence of operations that can be divided into tasks, which can be executed by the respective operating modules. Plans are generated according to the knowledge base and the information sensed during the disassembly process. As a result, the system is able to conduct complex procedures, responding to prior knowledge and sensed information according to specified behaviour.

At the *mid-level*, the intelligent agent controls the information flow in the system. This level concerns detail at the operation level. The agent communicates between the vision and mechanical systems respectively regarding information such as the location of components and desired operations.

Table 3.2 Tasks separated into levels of control and operating modules

Control	Module		
	Intelligent agent	Vision system and sensors	Mechanical system
High-level	–Sequence planning	n/a	n/a

		-Process planning -Task planning -Knowledge base -Behaviour control	
Mid-level	-Information flow	-Object localisation	- Operation procedure
		-Object recognition	-Path planning
		-Signal interpretation	-Trajectory control
Low-level	n/a	-Image pre-processing	-Motion control
		-Camera control	-Force-torque control
		-Image grabber	
		-Data acquisition	
		-Signal processing	

The tasks of the vision system and sensors are to sense the data from the external world, interpret this data, and communicate the relevant abstract information to the intelligent agent upon request. The intelligent agent calls the sensing procedures, e.g. to locate an object and to sense the force exerted on the tool. The *low-level* acquires the relevant data, e.g. a pre-processed image or voltage signal associated with a force, which may be processed using hardware. Algorithms in the *mid-level* then interpret this data to answer the request of the agent.

The tasks in the mechanical system relate to the physical enactment of actions as requested by the agent. The functions in the *mid-level* generate the desired tool trajectory according to the command of the intelligent agent. The desired motion or force-torque control, which concerns actuator control with respect to end-effector position, speed and/or external forces, is then executed at the *low-level*.

The implementation of the operating modules within this research is further explained Sects. 3.5 and 6.4.1.

3.3 Mechanical Design and Control

In a similar way to assembly automation, disassembly automation carries out the process by physically interacting with the products to be disassembled as well as the working environment. Mechanical design concerns the development of tools that are capable of performing the required task. Tools relevant to the disassembly process can be categorised into the following three types: (a) *manipulators*, (b) *disassembly tools*, and (c) *handling devices*.

3.3.1 Manipulator

The manipulator can be a robot arm or other device that facilitates the movement and/or use of various components in the disassembly process, such as disassembly tools, products and sensors. Disassembly systems may consist of a single or multiple manipulators performing different functions and working together. A robot arm is generally selected as the manipulator in disassembly automation due to its versatility. An industrial robot is normally a stand-alone system including a controller sufficient to fully control the robot at the *low-level*. Force-torque control, path generation and basic motion control are typically built-in features of the robot. Physical requirements of the disassembly tasks, e.g. workspace, payload, accuracy and accessibility of the disassembly tool need to be considered when designing or choosing a robot.

Each manipulator needs to have sufficient *degrees of freedom* (DOF) to perform the tasks. In general, 3–7 DOF industrial robots are used according to the task required. 3–4 DOF robots, such as SCARA, Cartesian, and gantry robots, are usually sufficient for pick-and-place tasks. 4–6 DOF robots, e.g. articulated and

delta-star robots [3], are typically used for more complex handling of the disassembly tool or object. A 6 DOF robot generally allows the manipulation of an object to an arbitrary position and orientation within the robot's workspace. 7 or more DOF robots may be selected to improve reachability or eliminate problems regarding singularities in movement.

The requirements in *workspace* and *payload* can be determined from the weight and expected forces from the disassembly tools and size of the product to be disassembled. The configuration of the robot should also be selected according to the requirements of the task. Robot configurations can be considered to fall into two main categories: (a) *serial robots* and (b) *parallel robots*.

Serial Robots

Serial robots have been used in industry for many decades and are actuated via links joined in a serial structure. Serial robots generally have a large workspace and high dexterity; however, suffer from a low payload due to the location of the actuators. Examples of serial robots include Selective Compliant Articulated Robot Arms (SCARA), articulated, gantry and Cartesian robots (see Fig. 3.2).

Parallel Robots

On the contrary, parallel robots have only been recently introduced to industry, and display the opposite characteristics. In comparison to serial robots with similar size, weight, and number of actuators, parallel robots operate in a very limited workspace and dexterity. However, due to the high rigidity of the parallel structure, parallel robots can handle considerably larger payload with high accuracy. Examples of parallel robots include the Delta and Steward Platform (see Fig. 3.3).

3.3.2 *Disassembly Tools*

The purpose of disassembly tools is to disestablish fasteners connecting the main components. The main components will be detachable after all

corresponding connectors have been disestablished. The tools can be selected based on specific

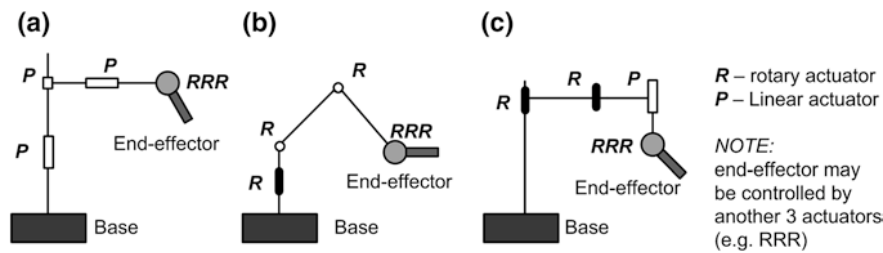


Fig. 3.2 Example of serial robots. **a** Cartesian robot. **b** Articulated robot. **c** SCARA robot

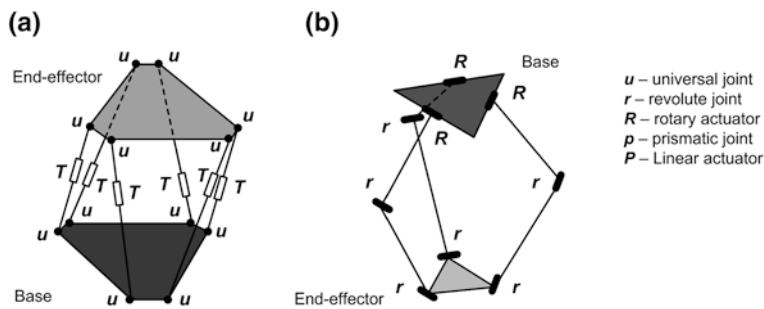


Fig. 3.3 Example of parallel robots. **a** Stewart platform. **b** Delta robot

tasks required for disestablish each type of fastener (see summary in Table 2.3). However, the required tool set differs between manual and automatic disassembly.

In manual disassembly, the human operator is flexible enough to utilise a limited set of primitive hand tools or power tools to disestablish a large variety of fasteners [8]. This is achievable thanks to our highly-developed senses and understanding of tool use: humans easily learn to control the tool appropriately in response to feedback, experiment to solve new problems using the fundamental knowledge of the tool's actions, and adapt their knowledge to new disassembly objects from past experience. Examples of such primitive tools include the screwdriver, pliers, cutters, grinder, saw and drill.

On the contrary, flexibility is quite limited in automated disassembly because of limitations in sensing and perception. Primitive disassembly tools may be able to use with disassembly automation by incorporating with proper sensors and control techniques. These can be complicated in most cases. Therefore, a number of attempts have been made to overcome this inflexibility with automatic tools that are specifically designed for each type of connectors. Lower-level control can be greatly simplified with tools that inherently required a straightforward method of application. Some innovative examples of such tools are as follows:

Automatic Screw Removal

Screws are a main category of fasteners and found in the majority of products. A number of special tools for the removal of screws have been developed. Screws can be removed by (a) *non-destructive*, (b) *semi-destructive*, and (c) *destructive* means.

For the *non-destructive approach*, an analysis of the unscrewing operation is done by Apley et al. [16]. The configuration of the prototype device is simple. A screwdriver is attached to an electric motor and a potentiometer is used for measuring the torque. The torque signal is continuously monitored to assess the condition of the unscrewing operation: whether the screw is coming out, the screw-driver is slipping on the head of the screw, or has completely missed the screw.

The condition is identified by using a least-square-based algorithm to analyse the measured torque signal. The success of the non-destructive operation relies on the accuracy in locating the screws, determining the type of screw head, and controlling the torque applied on it. These variations result in technical difficulties in automated disassembly. Even though the screws can be removed without damage, (semi-) destructive approaches are usually advantageous economically.

The *semi-destructive approach* is more feasible for automated disassembly due to its greater reliability. Seliger et al. [17] developed an automatic tool capable of unscrewing various types of screws, including those with damaged heads. The tool employs a pneumatic impact unit to create slots on the screw head. These slots become new active surfaces where torque can be transmitted, hence, allowing the unscrewing operation regardless of the shape and type of the screw. This tool is depicted in Fig. 3.4a, and the corresponding operation steps in Fig. 3.4b. Similarly, the “Drilldriver”, developed by Feldmann et al. [18], utilises a similar principle of creating new active surfaces to overcome the problem of differing or damaged screw head shapes.

In the *destructive approach*, the screw head or surrounding component is intentionally damaged to disestablish the connection. This can be achieved using various methods of material removal, including with a chisel, milling, grinding, sawing or drilling.

Automatic Removal of Snap-Fits

Snap-fits are a low-cost option commonly used as fasteners according to the principle of Design for Assembly (DFA) [20]. Hook-like elements built into the main component elastically deform when pressure is applied in the assembly direction, and snap into place when the component is correctly positioned. In this manner, connections between main components can be established without using additional connective components and the assembly process is simplified. No special tool is required for assembly and the workspace may be smaller [7].

On the other hand, the difficulty arises in the disassembly process—especially for non-destructive disassembly. To non-destructively disassemble components connected by snap fasteners, multiple forces are required at different locations and in differing directions. Components that are designed

to be disassembled may have a button which releases the corresponding snap-fit fittings. Where this is not the case, the tool end-effector must be small and the operation high in precision. Snap-fits are usually hidden and difficult to precisely locate. Manual detachment may be done by pulling, prying or deforming the components. However, these methods can easily damage components unless the worker is careful and skilled. Due to these difficulties, detaching components joined with snap-fits non-destructively is generally infeasible.

Automatic non-destructive disassembly currently only appears feasible in specific circumstances. Many have proposed the incorporation of new design elements in snap fits to facilitate disassembly (for further information, see *disassembly-embedded design* [21]). Fitting a magnet anchor on the flexible part of a snap-fit (see Fig. 3.5) allows the snap-fit to be pulled from outside. This principle can then be exploited using an automatic tool, as developed by Braunschweig [22].

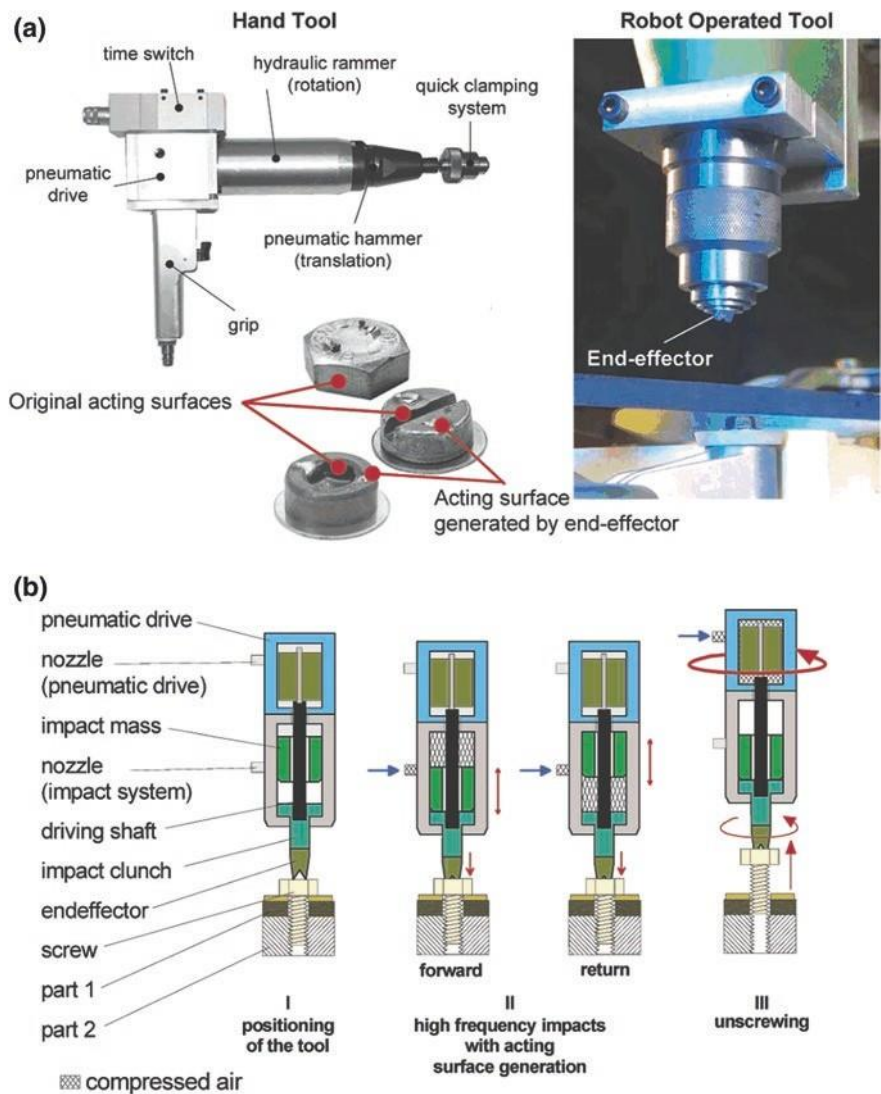
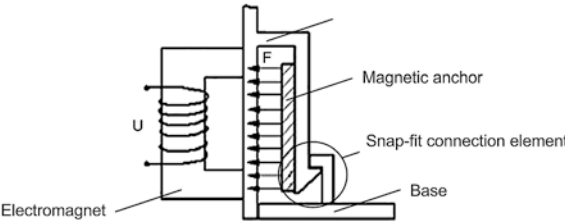


Fig. 3.4 Automatic unscrewers working by cutting a slot used as a new acting surface. **a** Two versions of tool developed from this concept: *a1* Hand tool—a hand held automatic unscrewers designed for manual use. *a2* Robot operated tool—an automatic unscrewers fully operated by a robot. **b** Operation steps for automatic unscrewing: positioning, creating acting surface, and unscrewing

Schumacher and Jouaneh [12] developed a prototype tool for disassembling snap-fit covers. The snap-fits in this case—a battery cover of electronic product—are not hidden and can be located by the vision

system from the top view. The movement of this tool will be controlled by a robot. To detach snap-fits, the tool pushes the flexible part of the snap-fits until they are fully deflected. The exerted force is monitored by a force-sensing resistor (FSR) (see Fig. 3.6).

Fig. 3.5 Automatic snap-fits removal with electromagnetic trigger [22]



Other Semi-Destructive and Destructive Approaches

An aim of disassembly is to obtain some main components undamaged. (Semi-) destructive approaches, on the other hand, offer a reliable, robust and flexible method of separating materials joined by fasteners or attaining access to more valuable components that can be obtained non-destructively. Examples of automatic semi-destructive and destructive disassembly operations include using an angle grinder with an abrasive disc in the disassembly of LCD screens [15], and using plasma cutting to cut through the metal cover of washing machines [23].

3.3.3 Handling Devices

Handling devices refer to fixtures, grippers, and logistics systems used to control the motion of the parts and products according to the steps in disassembly process. The logistic system is used to convey the product into and out of the disassembly

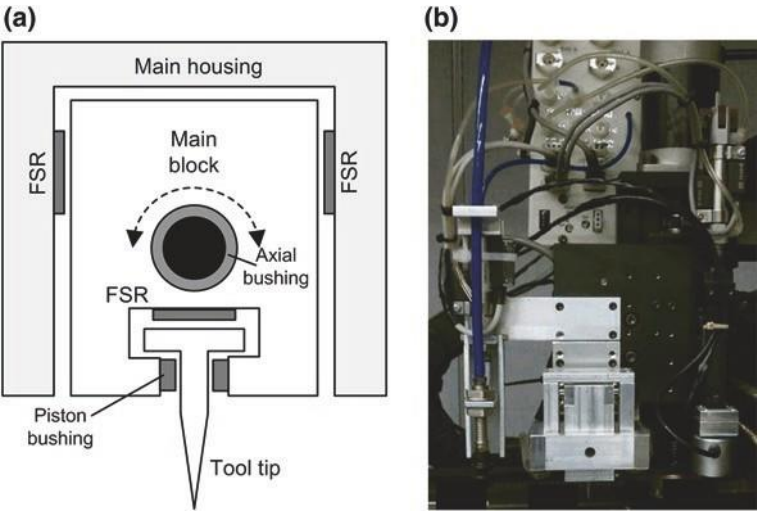


Fig. 3.6 Automatic snap-fits removal with FSR tool tip [12]. **a** Sketch of the disassembly tool. **b** Disassembly module

rig where the key activity of disassembly takes place. A variation of the designs of fixtures and grippers are used to overcome the uncertainties in the disassembly process. Meanwhile, the logistic system, e.g. conveyor belt and product storage, is straightforward due to its relatively simple function. Therefore, this section focuses on the fixture and gripper as opposed to the logistic system.

Product Fixtures

Fixtures are used to securely locate the product in specific locations and orientations during the disassembly process. Firm placement is essential in order to achieve high accuracy physical operations, and allow the necessary forces to be transmitted to the product. The product can be held with clamping techniques, e.g. using hydraulics or vacuum suction. If the product can be accurately positioned and is stationary with respect to the fixture, the exact position and orientation of the product can be exploited by the automatic system.

Alternatively, the fixture may be actuated to benefit the accessibility of tools in the disassembly operation. However, the difficulties are computation needed for coordination mapping and accumulating error from the structure of the moving fixture's components. A moving fixture is used in a number of systems, e.g. a rotating worktable that allows the robot to approach the back side of product (see Fig. 3.7a). A robot arm may also be used as the fixture [24] (Fig. 3.7b). In this case, there is a higher degree of freedom in the product location, however the product size and weight is limited by the specifications of the robot arm.

The fixture should be flexible enough to handle various product models, at least in the same product family. The components of the fixture should not obstruct the detection and removal of the product components and fasteners. One example of a product fixture designed for the disassembly application is the *flipping table* that is used in this research (see detail in Sect. 3.5).

Grippers

Various kinds of gripper are used in both assembly and disassembly processes to grasp and handle desired objects. A number of techniques by means of *hardware* and *software* are developed to overcome the uncertainties in geometry and operation.

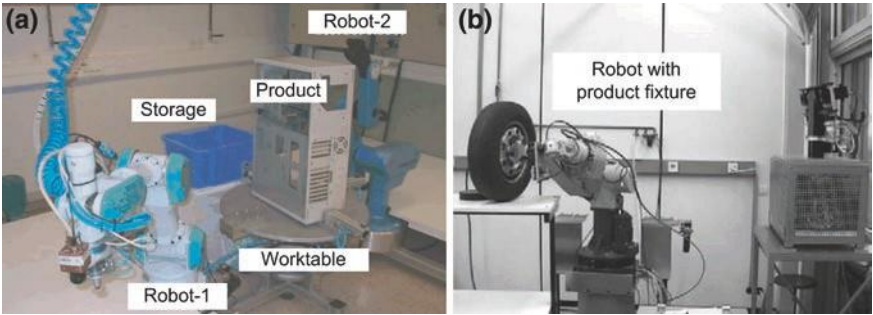


Fig. 3.7 Fixture (adapted from [24, 25]). **a** Rotating worktable with a pneumatic clamp used for holding a product (computer case). The product can be rotated with 1-DOF. **b** Robot arm handling a product (wheel). The product is more accessible due to the 6-DOF robot’s movement

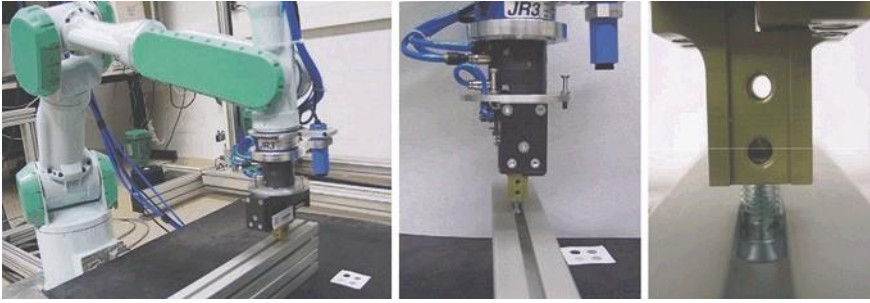


Fig. 3.8 Removal of a screw from a slot using force control [25]

In regard to the *hardware*, specific mechanisms are designed to suit requirements such as the specific geometry and features of the objects to be grasped. *Software*, in terms of control strategies, tends to relate to generic classes of grippers, e.g. 2-fingered-grippers. Inputs from vision and other sensor system are heavily used in this case. Focusing on the disassembly process, grippers are mainly used in two ways: (a) to disestablish connections and (b) to remove parts or components that have been detached.

For disestablishing connections, grippers are usually associated with action “pull” to detach the connectors from the remaining parts, for example pulling electrical/electronic cables [26]. A *fingered gripper* with force-torque control is needed because an accurate magnitude and direction of force and torque is crucial to avoid damage to the removed object. An example of this is in the removal of a screw from a rail slot in [25] (see Fig. 3.8).

For removing the detached parts or components, the gripper type is typically selected based on the geometry of the object. Much research has been done in developing strategies for handling objects with various shapes. The orientation of the gripper may be crucial in achieving a firm grasp and avoid collisions with other components. Visual input, together with force feedback, may be used to determine the size and shape of the object. Some work in this area is described as follows:

Schmitt et al. [5] developed the flexible gripper in Fig. 3.9 for the disassembly of Lithium-Ion batteries. The challenge here is to design a

gripper to be flexible for grasping various geometries of battery cells. Two parallel jaw grippers are attached to a standard rail, providing the ability to adjust to the dimensions of the battery cell in two axes. In addition, voltage and resistance measurement at the battery contacts is done via the conductive contacts in the grippers' jaws.

Seliger et al. [27] developed a disassembly tool with a "screwnail" end-effector for removing the cover of products. The key advantage is to overcome the geometrical uncertainties of products. The operation procedure is shown in Fig. 3.10. The sharp "nail" tips secure the tool on the component rotationally, while the "screw" is driven into the material, securing it in the lateral direction. The screw is turned in the opposite direction to separate the tool from the product cover. This connection is strong enough to transmit the required forces and torques for disassembly.

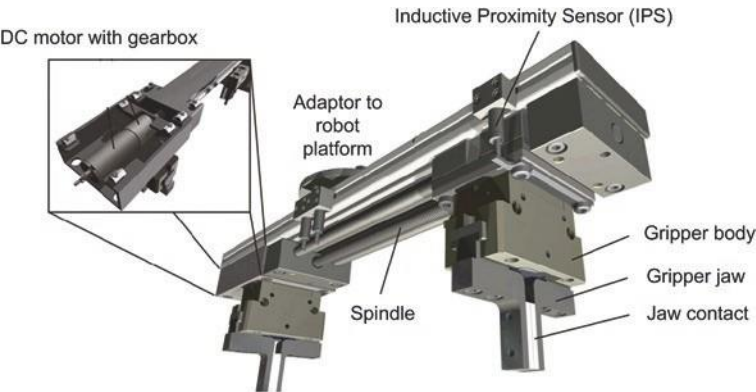


Fig. 3.9 Gripper for disassembly of Lithium-Ion battery [5]

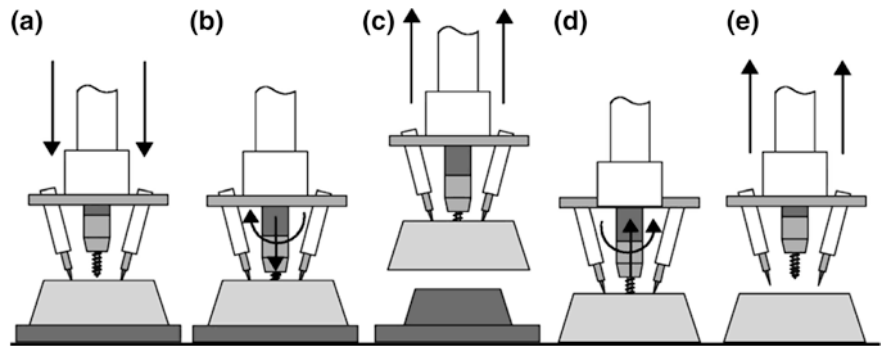


Fig. 3.10 Gripper for flexible disassembly using screwdriver end-effector [27]. (a–b) Screwdriver indentation process. (c–d) Task execution process. e Separating process

In the area of software, Fernandez et al. [28] developed an algorithm to select appropriate grasping points for a multiple-fingered gripper to grasp objects of different geometries. Potential grips are assessed considering the centre of gravity of the object and friction cones calculated from the positioning of the fingers on the object (see Fig. 3.11). The system is learning-based and behaviour rules are extracted from the given training samples.

Disassembly automation also employs various types of grippers commonly used in object handling process. In addition to fingered grippers, *vacuum grippers* are commonly used in pick-and-place operations.

Conventional robotic vacuum grippers consist of one or multiple air inlets connected to a source of negative pressure, which must be adequate to hold the object to the tool. Holding an object via one vacuum cup is susceptible to rotational moments, for example, caused by lack of alignment with the centre of gravity of the object. Using an array of cups or inlets eliminates this problem, however, requires additional assumptions regarding the shape and dimensions of the surface to be gripped.

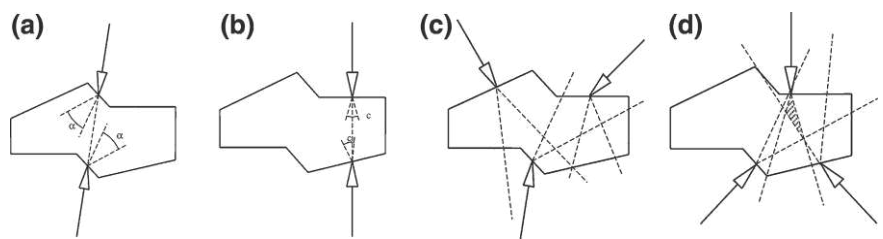


Fig. 3.11 Grasping point selection with two and three points contact [28]. **a** Two-point. **b** Two-point with force enclosure. **c** Three-point. **d** Three-point with force enclosure

3.4 Degree of Autonomy

3.4.1 Semi-automatic Disassembly

The *semi-automatic* or *hybrid disassembly system* consists of *automatic work-stations* and *manual workstations* working in some kind of collaboration. The automatic workstation is equipped with sensors and operated by robot arms equipped with disassembly tools; meanwhile, human operators are employed at the manual workstation and carry out tasks that are more difficult to automate. A conveyor system can be used for transporting the products between the worksta- tions. The main process flow can be controlled by an automatic planner or manu- ally by the operators. The tasks are distributed to workstations according to the problems that need to be resolved. An advantage of this kind of system is in reduc- ing workers’ exposure to dangerous situations such as with heavy lifting and the removal of hazardous parts. On the other hand, their contribution allows the sys- tem to carry out processes that cannot be achieved with automation alone.

It is evident that this approach is, robust, cost-effective, and flexible for various product families [29]. Franke et al. [30] state that the hybrid disassembly process is economically justified; the manual operation is essential due to the inability of automated systems to operate with a non- determined disassembly sequence and react to varying product conditions.

The, manual operation is, therefore, the fall- back option when the automatic operation has failed. Some research work with regard to hybrid disassembly systems is presented as follows.

Hybrid System for Disassembly of Various Product Families

At the *Institute for Machine Tools and Factory Management, Technical University of Berlin*, a hybrid system that is flexible to disassemble a wide range of product families has been developed. This work is presented in Kim et al. [29]. The study focuses on the automatic generation of plans and the control sequence of a system consisting of three robot arms and conveyor belts. The robots are responsible for heavy duty tasks, e.g. plasma cutting of the side wall of washing machines. Before the process starts, the system evaluates the degree of autonomy of the overall task from the product information and the availability of various systems.

Consequently, the tasks can be properly distributed to the manual and the automatic workstations.

Kim et al. [31] extended this concept to develop a disassembly line specific for LCD screens. At the manual workstation, the operator makes decision and disassembles monitor stands that interfere with other connectors that need to be removed. The LCD screen is carried to the automatic workstation by the robot. For the automatic workstation (see Fig. 3.12), the SCARA robot disassembles the components, i.e. screws, back cover, metal covers, printed circuit boards (PCBs), and cable connections, by using a two-fingered gripper. If the operations have failed, the product is forwarded to the next manual workstation to be further disassembled.

Hybrid System with Modular Concept

The *Institute for Handling Devices and Robotics (IHRT)*, Vienna University of Technology, Austria has developed a multi-cell system designed for disassembling obsolete PCs. The work is presented in Kopacek and Kronreif [33]. The system consists of 2 robots and a manual workstation. Due to the problems in accessibility and various connection techniques, the operators manually detach the cables, the cover, and any valuable parts. The robots—equipped with a gripper and a screw-driver—remove other components in PC including the hazardous components. The planning is carried out by a semi-automatic planner, in which the specific operations and tool selections are made in advance.

The modular concept is applied to the semi-automatic system by Zebedin et al.

[34] and Knoth et al. [6], which describes a system designed for extracting embedded components from PCBs. The operating modules (e.g. robot arms, part feeders, fixtures, desoldering system, and quality control system) are grouped as a subsystem. The disassembly cell controller is used to supervise the communication and co-ordination tasks in and between each subsystem. The cell controller is configured using hierarchical control and information distribution of the disassembly process. Machines can operate automatically and a human operator can command and monitor the system through the user interface.

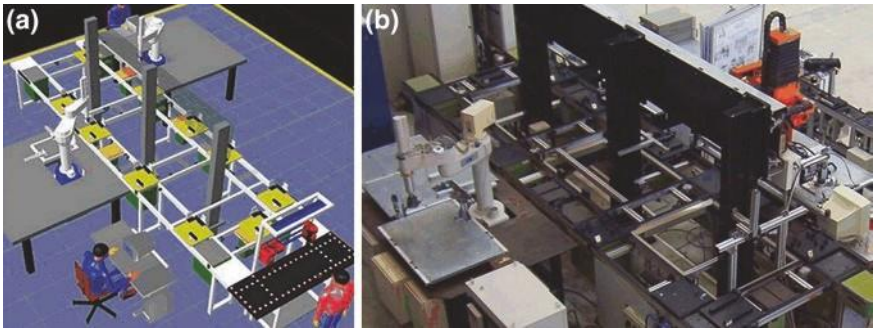


Fig. 3.12 Hybrid system for LCD screen disassembly [32]. **a** Verification in virtual system. **b** Automatic workstation in disassembly system

This system is also implemented as a part of the disassembly system for mobile phones described by Kopacek and Kopacek [9, 35], Kopacek [36]. The system employs 5 automatic workstations and a manual workstation. Automatic workstations carry out the tasks of cover removal, milling, drilling, and PCBs removal. Product feeding is done manually by the operator.

Disassembly Factories for Electrical and Electronic Products

Two disassembly systems at the *Institute for Machine Tools and Factory Management in Berlin, Germany* are presented in Basdere and Seliger [19].

First, a system for disassembling washing machines is presented in Uhlmann et al. [37]. The system consists of 2 stationary robots, 1 mobile robot, and 3 manual workstations (see Figs. 3.13 and 3.14). The automatic and the manual workstations are located in separated areas due to safety concerns. The products are transported between the workstations via a conveyor system. The robots perform plasma-cutting of metal covers and pass the product to the operators. Manual disassembly is done when the robots fail or the process is too complicated.

Second, a system for the disassembly of mobile phones is presented in Kniebel et al. [38]. The system consists of a 4-DOF SCARA robot and a manual workstation. The operator removes the battery then passes the product to the automatic workstation. The robot is equipped with a number of flexible tools, e.g. an automatic unscrewdriver, a vacuum gripper with a pin to open snap-fits, and a flexible clamp. The system is flexible enough to handle various models of mobile phones. A vision system is used to recognise the barcode in order to determine the model.

Other Selected Systems

Other interesting research is reviewed by Wicndahl et al. [39] and Scharke [7], which also illustrates the layouts of these systems.

Fig. 3.13 Hybrid system for disassembly of washing machines [19]



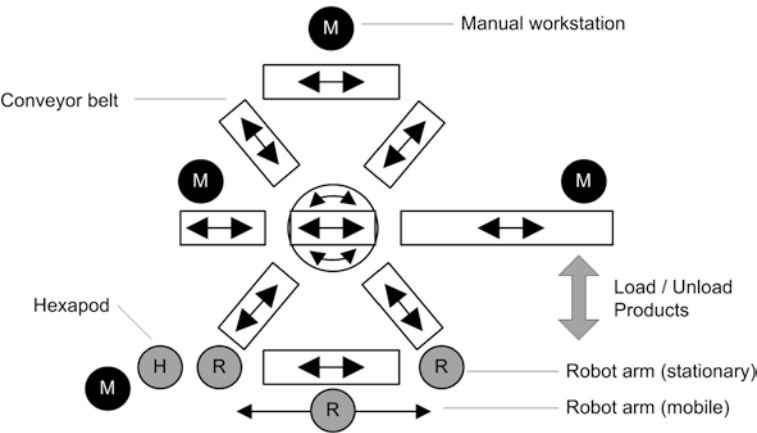


Fig. 3.14 Layout of the hybrid system for disassembly of washing machines (adapted from [39])

In the *Cleantech Project in Bochum, Germany*, a hybrid system for disassembling electrical plane and high pressure cleaning machines has been developed. This work is presented in Schnauber et al. [40]. The system consists of 2 manual workstations and an automatic workstation connected with a two-line structure conveyor system. The robot unfastens the screws at fixed locations. The first operator removes, separates, and sorts the parts according to reusability and material, while the second operator checks and cleans the parts. All disassembly plans are defined a priori.

At the *Fraunhofer Institute IML in Dortmund, Germany*, Jünemann et al. [41] presents a disassembly system consisting of 2 manual workstations and an automatic workstation. The products are transferred between the workstations with a closed-loop conveyor system. The robot is used to remove the cover of products, e.g. microwave ovens. The operators manually separate the valuable parts from the product.

At the *Institute for Production Automation and Handling Technology at the TU-Braunschweig, Germany*, a system for separating valuable and hazardous components from PCBs was developed. This work is presented in Hesselbach et al. [42], Friedrich [43]. The system consists of an automatic workstation and a manual workstation. The operator detaches

cables and connectors, and then passes the PCB to the automatic workstation. At the automatic workstation, a SCARA robot separates the components using a special gripper for microchips and an unsoldering tool. A laser scanner identifies the embedded components on PCBs based on a component database. Subsequently, the robot tasks are generated by a planning and control system.

At the *Technion's Center for Manufacturing Systems and Robotics, Haifa, Israel*, a system has been developed consisting of 2 automatic workstations and a manual workstation. The automatic workstations are operated by a 6-DOF robot and a 4-DOF robot. The manual workstation is used to deal with complicated tasks.

The system is designed to work with multiple pallets at the same time where the pallet can be identified using electronic non-contact badge. The related research in vision systems and advanced planning is also described in Zussman [44].

3.4.2 Fully Automatic Disassembly System

The heightened degree of autonomy in *fully automatic disassembly systems* is generally achieved by incorporating (a) better sensor modules, (b) prior knowledge of products, and (c) a high-level task planner. Ideally, the intelligent planner is able to control the system to carry out the operations and address all uncertainties during the process, therefore eliminating the need for human intervention.

However, due to the variations in EOL products and the uncertainties in the disassembly process, achievement of the required flexibility and robustness has posed to be a great challenge. As one cannot guarantee that all problems can be resolved solely by automation, it is expected that minimal human intervention may be involved. However, unlike semi-automatic disassembly, operators in a fully automatic disassembly system do not carry out tasks on the product directly but monitor and maintain the system and provide instructions at the planning level. This can be considered a type of human-machine collaboration. The relevant research work is presented as follows.

Co-operative Multi-sensorial Disassembly Cell

The *Department of Physics, Systems Engineering and Signal Theory, University of Alicante, Spain*, Torres et al. [45] has developed one of the most advanced disassembly cells for disassembling computers to date (see Fig. 3.15). This disassembly cell consists of two industrial articulated robots equipped with force-torque sensors and selected interchangeable

disassembly tools. This project comprises three major contributions: (a) co-operative operation, (b) vision system and (c) multi- sensorial system.

Both robots work co-operatively through a task planner that automatically generating paths and trajectories based on a graph model proposed by Torres et al. [46]. In terms of the vision system, work has been described in the detection of occlusions [47] and various components including screws and cables [25]. For the multi-sensorial system, Gil et al. [25] combines information between a tactile sensor and the vision system in order to perform visual servoing of the robot. This system was tested by removing a bolt from a straight slot (see Fig. 3.8).

This system solves the problem of uncertainties at the operational level by using an integrated sensor system. However, at the higher level, all disassembly sequence planning is based on precedence among assemblies [48]. Hence, the user is still required to provide this specific product structure information a priori. Input from the vision system is only used to determine the detailed geometry for the operation level.



Fig. 3.15 Multi-sensorial cooperative robots—demonstrate removing CD drive [25]

Automatic System for the Disassembly of EOL Personal Computers

At the University of Bridgeport, USA—EISayed et al. [13], as system has been developed for the selective disassembly of EOL personal computers (PC) for reuse and recycling purposes. The system consists of an articulated industrial robot and a camera system, and deals with uncertainties in the product structure using (a) a vision system and (b) an *online disassembly sequence planner using genetic algorithms* (GA).

The vision system consists of a 2D-camera and a laser range sensor. Camera images are used to recognise and locate the components, matching regions of the input image with predefined 2D templates of the components supplied in the bill-of-materials (BOM). The BOM of electrical/electronic products are subject to change during the usage stage due to modifications. PCs are one example where the location and the existence of the main components (e.g. random access memory (RAM)) and connectors (e.g. screws) are likely to change according to upgrading, repairing, or personal preference. This creates uncertainties in the product structure at the point of disassembly.

These uncertainties are addressed by the online disassembly sequence planner). The supplied BOM contains all expected components and the precedence relations between them. At any stage, and according to any

modifications to the product, only a select number of these components are detected by the system. From the detected components, the GA implementation generates a (near-)optimal solution for the disassembly sequence, minimising a certain criteria, e.g. travel distance.

The key feature of this system is its ability to adapt the plan according to the current situation perceived by the vision system. However, a BOM representing the precise product structure and a template of specific components needs to be supplied a priori.

Automatic System for the Disassembly of Automotives

At *Heinz Nixdorf Institute, University of Paderborn, Germany*, a disassembly system for wrecked cars has been developed. The work is presented in Bükér et al. [24]. This research is part of Project DEMON founded by the German Ministry for Education and Research (523-4001-01 IN 506 B 2). This research focuses on the disassembly of automotive wheels with variation in the size of the wheels, the number of the bolts, and the position of the wheel. The system employs active stereo cameras and a robot arm equipped with a tool for unscrewing (see Fig. 3.7b).

The active stereo cameras are used to reconstruct the 3D structure of the product. Principle component analysis (PCA) is used to identify components that are difficult to recognise due to uncertainties in the EOL condition, e.g. due to rust. A knowledge-based approach with a neural network is used to address the problem of occlusion in complex scenes [49].

Other Selected Systems

At the *Automation and Control Institute, Vienna University of Technology, Austria*, Merdan et al. [50] proposes an ontology-based architecture with a multi-agent system (MAS) for the disassembly of digital cameras. The robot disassembles the product using a screwdriver and grippers. The vision system detects the components and links to the knowledge base. The ontology is used to describe the tasks and their requirements according to the operating modules and level of control. The disassembly plan is automatically generated from the hierarchical product structure described in the ontology. Subsequently, the operation plan with optimised tool-paths can be generated.

At the *Department of Manufacturing and Mechanical Engineering, Miami University, USA*, Bailey-Van Kuren [51] proposed a strategy for the real-time generation of cutting tool paths which need to follow the complex surface of a product. A prototype system consists of a robot equipped with a cutting tool, a vacuum gripper, and the smart vision system based on structured light [52]. The mobile phone is the case-study product in this research.

At the *Fraunhofer Application Centre for Logistic System Planning and Information System, University of Technology, Cottbus, Germany*, Scholz-Reiter et al. [14] conducted research in collaboration with *Prielog Logistik GmbH* [53] where the disassembly system is developed. The system designed for the disassembly of obsolete TVs and monitors into their components within 5–7 min. The system is operated by two robots. The first one is a 6-DOF robot equipped with various disassembly tools for destructive and non-destructive disassembly. The second is a 4-DOF robot equipped with different grippers for handling parts. Vision is used to recognise variations in the product and component condition. Flexible disassembly plans are generated according to the sensed information and the existing database. The method to generate reactive and adaptive disassembly plans from the data is described in Scharke [7].

At the *Fraunhofer Institute for Information and Data Processing (IITB), Karlsruhe, Germany*, a system for disassembling used cars is developed based on MAS. The work is presented in Gengenbach et al. [54]. The system consists of 2

robots: one for disassembling and the other for monitoring. The disassembly robot is equipped with an electric screwdriver and gripper.

The “eye-in-hand” approach is implemented on the monitoring robot, which manipulates a miniature stereo camera system, called MiniVISTAs.

Tonko and Nagel [55] focused on the vision system, developing an algorithm for the model-based tracking of a non-polyhedral car engine. At the *Fraunhofer-Institute for Production and Automation in Stuttgart, Germany*, two systems were developed. First, a system for disassembling tele-phones is presented in Kahmeyer [56]. The robots are equipped with various tools: a flexible parallel gripper, a vacuum gripper, an unscrew, a tool for detaching snap-fits, and a drill. The system is able to carry out low-level planning and control. It can be adapted to different phones with manual modification. Second, a system for disassembling roofs and windows of cars is presented in Rupprecht [57]. A vision system is used to locate the position of these components, which are

removed by two robots equipped with special disassembly tools.

At the *Chair of Manufacturing Automation and Production at University Erlangen-Nürnberg, Germany*, a system for separating the embedded components on PCBs by flexible interconnected disassembly cells has been developed. The work is presented in Feldmann and Scheller [58]. The system uses a laser scanner to identify the embedded components. The valuable components are disassembled non-destructively while the hazardous components are disassembled using cutting techniques.

3.5 System Setup in This Research

In this research, at the *School of Mechanical and Manufacturing Engineering, The University of New South Wales, Australia*, Vongbunyong et al. [15] has developed a low-cost automated disassembly system in which LCD screens are used as the test case. A (semi-) destructive approach is implemented to selectively disassemble to a module level. The principle of cognitive robotics is used to resolve the uncertainties in product and

process. As a result, the system is flexible to disassemble various models of products regardless of the exact information in product structure and geometry. In addition, the system is capable of learning from past experience, which results in an increase of performance of the process.

The system is controlled by the *cognitive robotic agent* (CRA) which is an intelligent planner with cognitive functions. The CRA is part of the cognitive robotic module (CRM). The CRM also consists of a knowledge base (KB), where learned information is stored. A vision system module (VSM) is designed for detecting the components and assesses the success or failure of an operation. The VSM acquires and processes images from a colour camera and a depth sensor. The mechanical components and their low-level control will in this case be called the disassembly operation unit modules (DOM). The system employs a 6-DOF robot equipped with an angle grinder to perform (semi-)destructive disassembly. To reduce the complexity of using a tool changer, the *flipping table* is designed to function as the fixture and the means of removing a detached component from the product.

For the overview, the architecture of the system according to the operating modules and level of control is shown in Fig. 3.16. The communication system and information flow are further explained in Sect. 6.4.1 (Fig. 3.17).

Robot Arm

The robot arm is a stand-alone module used as the primary component of the DOM. This module consists of 2 elements: (a) *IRB-140*, a small-sized 6-DOF articulated industrial robot and (b) *IRC-5* robot controller. The system can be programmed using the high-level procedural language RAPID. The main task is to control the disassembly tool to perform the (semi-) destructive disassembly operation according to the primitive action requested by the CRA. The robot has built-in sensors used for preventing an overload on the motor at each joint. *MotionSupervision* [59] is implemented using these sensors and allow the system to recognise collisions. This collision awareness is used in finding a set of process parameters that allow the disassembly tool to approach the required cutting destination.

Flipping Table and Grinder

The *flipping table* is a device used for handling the sample to be disassembled and removing the detached components. This device is a low-cost substitution to classical grippers, designed to remove the detached parts from the product without

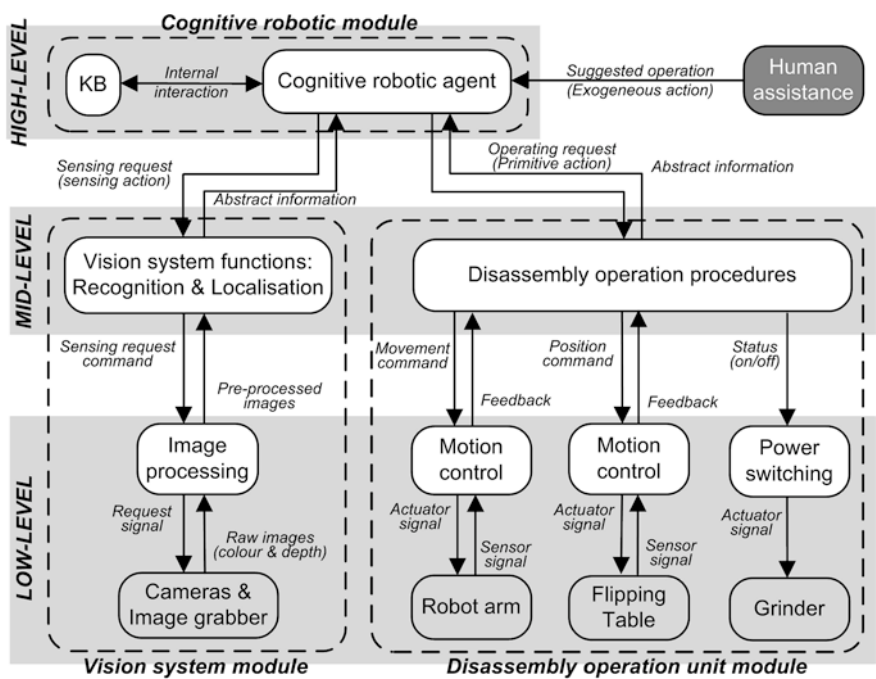


Fig. 3.16 Example of system architecture—levels of control and operating modules

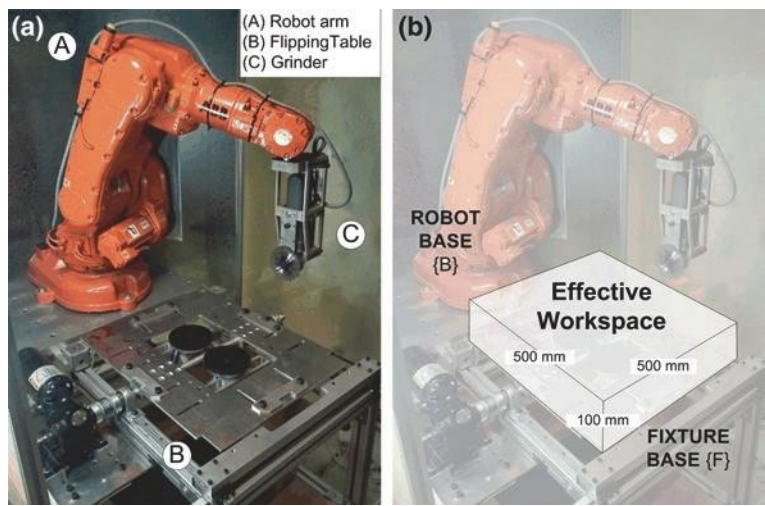


Fig. 3.17 Disassembly rig for disassembly of LCD screens. **a** Complete setup excluding cameras.

b Effective workspace

gripping. This solution addresses the complexity of picking up objects of varying geometry amidst clutter; however, the ability of the gripper to selectively pick an object or to apply forces in specific directions is lost.

An LCD screen sample is initially placed firmly on the fixture plate and gripped from the front side using two suction cups. The suction cup generates the axial force while the fixture elements situated on each side prevent any rotation or transverse movement (see Fig. 3.18a). The fixture plate is designed to support 16"–19" LCD screen. By gripping the sample from the front side, it is ensured that the LCD module is the last component remaining on the workstation after disassembly. The flipping table is activated after each operation cycle by flipping the entire product by almost 180°. As a result of the flipping force and gravity, any components and detached parts are unloaded into a container or conveyor situated underneath.

Disassembly Tool

The angle grinder is used as the disassembly tool for the cutting operation. (Semi-) destructive disassembly has been implemented due to its high

success rate and economic feasibility. A grinder equipped with a multi-purpose abrasive cut-off disc is a versatile cutting tool capable of cutting all materials found in LCD screens (e.g. steel, aluminium, polymers; see Fig. 3.18b) without losing its sharpness. However, a major drawback is tool wear. As a result, the tool-tip position changes non-uniformly according to the material, cutting speed, and feed speed. The vision system is used to sense the current tool-tip position in order to compensate for the change.

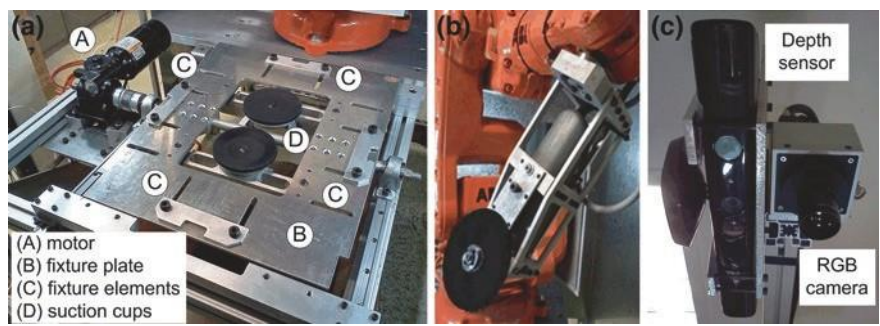


Fig. 3.18 Disassembly operation module (a, b) and vision system module (c). **a** Flipping table. **b** Robot with a grinder. **c** Camera system

The cutting operation destroys the fastener connections without any need to adapt the physical tooling, allowing the system to handle screws, wires as well as hidden snap fit connections by cutting through main components at less critical locations near the border. For this case-study product, a vertical cutting direction is sufficient for the disassembly process.

In conclusion, instead of grippers and multiple disassembly tools, a small set of devices has been selected in order to reduce cost and computing resources. The added complexity of force-torque control and tool-change has been eliminated while still allowing the system to be robust and flexible to physical uncertainties. However, a major drawback is component damage from the disassembly operation; this solution is suitable for recycling purposes only.

3.6 Conclusions

Disassembly automation generally refers to an automated system consisting of robot arms, equipped with disassembly tools, handling

devices, and sensors. Automation is implemented in the disassembly process to address the problem in economic feasibility directly related to high labour costs, especially in developed countries. Disassembly automation can also be used to aid or replace the operator in performing undesirable tasks, for instance handling heavy parts or those that contain hazardous substances. Human operators are still required to various extents to aid the system in planning and/or carry out activities that the automation lacks the flexibility for.

In *semi-automatic* or *hybrid disassembly*, human operators at manual workstations work in collaboration with the automatic workstation in executing physical tasks on the product or component. The major advantage is the high flexibility in dealing with uncertainties and variations in the products. Economic feasibility may be achieved with the automatic workstation performing certain tasks more cheaply

and efficiently, while the human operators provide the flexibility to recover from problems where the automation fails. However, in this situation the operators are still directly exposed to the product, which in some cases may be harmful.

Automatic disassembly systems carry out the process more autonomously, and are aimed at executing disassembly processes with minimal human involvement. Greater autonomy at various levels can be achieved by employing more intelligent planners and smarter sensor systems; this remains an active research area. The role of human operators becomes supervisory; exposure to potentially-harmful conditions can be reduced. Problems due to the variations in EOL products and uncertainties in the disassembly process still remain not completely addressed. Most existing research has focused on low-level operations, e.g. tool path generation. At the planning level, it is clear that prior knowledge of the product structure always needs to be supplied in various forms, e.g. CAD models, a BOM, and/or precedence relations. The high-level planner needs this specific information for each product model to generate the DSP and DPP. The flexibility of the system to handle variations in models and products is limited as this information is generally unavailable before a product has been disassembled.

In conclusion, it is desirable, both in the interest of preserving the safety of human workers and of achieving economic feasibility, to continue the path of realising fully automatic disassembly systems. This research aims to increase the flexibility of fully automatic systems to the variety of product models, and their robustness, to the uncertainties in the process. Plans need to be generated by the intelligent agent regardless of uncertainties in the specific model information. In this book, the principle of cognitive robotics is implemented to resolve these uncertainties (see Chaps. 5 and 6).

References

1. Rifkin J (1995) The end of work: the decline of the global labor force and the dawn of the post-market era. Putnam Publishing Group, New York

2. Bennett S (1993) A history of control engineering 1930–1955. Peter Peregrinus Ltd., London
3. Craig JJ (2005) Introduction to robotics: mechanics and control, 3rd edn. Prentice Hall, Englewood Cliffs
4. Dufloy JR, Seliger G, Kara S, Umeda Y, Ometto A, Willems B (2008) Efficiency and feasibility of product disassembly: a case-based study. *CIRP Ann Manuf Technol* 57(2):583–600
5. Schmitt J, Haupt H, Kurat M, Raatz A (2011) Disassembly automation for lithium-ion battery systems using a flexible gripper. Paper presented at the IEEE 15th international conference on advanced robotics: new boundaries for robotics (ICAR 2011)
6. Knoth R, Brandstötter M, Kopacek B, Kopacek P (2002) Automated disassembly of electr(on)ic equipment. In: IEEE international symposium on electronics and the environment
7. Scharke H (2003) Comprehensive information chain for automated disassembly of obsolete technical appliances. GITO-Verlag, Berlin
8. Lambert AJD, Gupta M (2005) Disassembly modeling for assembly, maintenance, reuse, and recycling. CRC Press, Boca Raton

9. Kopacek P, Kopacek B (2006) Intelligent, flexible disassembly. *Int J Adv Manuf Technol* 30(5–6):554–560
10. Kopacek B, Kopacek P (2001) Semi-automatised disassembly. In: *The 10th international workshop on robotics, Alpe Adria Danube region, RAAD 01, Vienna*, pp 363–370
11. Golnaraghi F, Kuo BC (2009) *Automatic control systems*, 9th edn. Wiley, London
12. Schumacher P, Jouaneh M (2013) A system for automated disassembly of snap-fit covers. *Int J Adv Manuf Technol* 1(15):2055–2069
13. ElSayed A, Kongar E, Gupta SM, Sobh T (2012) A robotic-driven disassembly sequence generator for end-of-life electronic products. *J Intell Robot Syst Theor Appl* 68(1):43–52
14. Scholz-Reiter B, Scharke H, Hucht A (1999) Flexible robot-based disassembly cell for obsolete TV-sets and monitors. *Robot Comput Integr Manuf* 15:247–255
15. Vongbunyong S, Kara S, Pagnucco M (2013) Application of cognitive robotics in disassembly of products. *CIRP Ann Manuf Technol* 62(1):31–34
16. Apley DW, Seliger G, Voit L, Shi J (1998) Diagnostics in disassembly unscrewing operations. *Int J Flex Manuf Syst* 10(2):111–128
17. Seliger G, Keil T, Rebafka U, Stenzel A (2001) Flexible disassembly tools. In: *IEEE international symposium on electronics and the environment*, pp 30–35
18. Feldmann K, Trautner S, Meedt O (1996) Innovative disassembly strategies based on flexible partial destructive tools. *Annu Rev Control* 23:159–164
19. Basdere B, Seliger G (2003) Disassembly factories for electrical and electronic products to recover resources in product and material cycles. *Environ Sci Technol* 37(23):5354–5362
20. Boothroyd G, Alting L (1992) Design for assembly and disassembly. *CIRP Ann Manuf Technol* 41(2):625–636
21. Masui K, Mizuhara K, Ishii K, Rose C (1999) Development of products embedded disassembly process based on end-of-life strategies. In: *The EcoDesign'99: 1st international symposium on environmentally conscious design and inverse manufacturing*, Tokyo, pp 570–575
22. Braunschweig A (2004) Automatic disassembly of snap-in joints in electro-mechanical devices. In: *The 4th international congress mechanical engineering technologies'04, Varna*
23. Uhlmann E, Spur G, Elbing F (2001) Development of flexible automatic disassembly processes and cleaning technologies for the recycling of consumer goods. In: *IEEE international symposium on assembly and task planning*, pp 442–446
24. Büker U, Drüe S, Götze N, Hartmann G, Kalkreuter B, Stemmer R, Trapp R (2001) Vision-based control of an autonomous disassembly station. *Robot Auton Syst* 35(3–4):179–189
25. Gil P, Pomares J, Puente SVT, Diaz C, Candelas F, Torres F (2007) Flexible multi-sensorial system for automatic disassembly using cooperative robots. *Int J Comput Integr Manuf* 20(8):757–772
26. Kernbaum S, Franke D, Seliger G (2009) Flat screen monitor disassembly and testing for remanufacturing. *Int J Sustain Manuf* 1(3):347–360
27. Seliger G, Stenzel A, Rebafka U, Zuo B-R (2002) A novel disassembly tool with screw nail end effectors. *J Intell Manuf* 13(3):157–163
28. Fernandez C, Reinoso O, Vicente MA, Aracil R (2006) Part grasping for automated disassembly. *Int J Adv Manuf Technol* 30(5–6):540–553
29. Kim H-J, Harms R, Seliger G (2007) Automatic control sequence generation for a hybrid disassembly system. *IEEE Trans Autom Sci Eng* 4(2):194–205
30. Franke C, Kernbaum S, Seliger G (2006) Remanufacturing of flat screen monitors. In: *Brissaud DTS, Zwolinski P (eds) Innovation in life cycle engineering and sustainable development*, pp 139–152
31. Kim H-J, Chiotellis S, Seliger G (2009) Dynamic process planning control of hybrid disassembly systems. *Int J Adv Manuf Technol* 40(9–10):1016–1023

32. Kim HJ, Kernbaum S, Seliger G (2009) Emulation-based control of a disassembly system for LCD monitors. *Int J Adv Manuf Technol* 40(3–4):383–392
33. Kopacek P, Kronreif G (1996) Semi-automated robotized disassembling of personal computers. Paper presented at the IEEE symposium on emerging technologies and factory automation (ETFA 2)

34. Zebedin H, Daichendt K, Kopacek P (2001) A new strategy for a flexible semi-automatic dis-assembling cell of printed circuit boards. *IEEE Int Symp Ind Electron* 3:1742–1746
35. Kopacek P, Kopacek B (2003) Robotized disassembly of mobile phones. Paper presented at the IFAC workshop: intelligent assembly and disassembly (IAD'03), Bucharest
36. Kopacek P (2005) Semiautomatized disassembly—some examples. In: *IFAC proceedings volumes (IFAC-PapersOnline)* 16:146–151
37. Uhlmann E, Haertwig JP, Seliger G, Keil T (2000) A pilot system for the disassembly of home appliances using new tools and concepts. In: *Proceedings of the third world congress on intelligent manufacturing processes and systems*, Cambridge, MA, 28–30 June 2000
38. Kniebel M, Basdere B, Seliger G (2004) Hybrid disassembly system for cellular telephone end-of-life treatment. In: *The joint international congress and exhibition electronics goes green*, Stuttgart, pp 281–286
39. Wicndahl H-P, Scholz-Reiter B, Bürkner S, Scharke H (2001) Flexible disassembly systems-layouts and modules for processing obsolete products. *Proc Inst Mech Eng Part B J Eng Manuf* 215(5):723–732
40. Schnauber H, Kiesgen G, Slawik F (1997) Das element produkt-wiederverwendung im qualitätskreislauf. Ergebnisse des EUREKA Forschungsprojektes CLEANTECH (EU 1104)— concept for logistical and environmental disassembly technologies. *Forschungszentrum Karlsruhe GmbH, Karlsruhe*
41. Jünemann R, Hauser H, Moukabary G (1997) Sensor- und informationssystemein der demontage. Paper presented at the in colloquium on closed-cycle economy and disassembly, Berlin, Germany, 30–31 Jan 1997
42. Hesselbach J, Friedrich R, Schütte G (1994) Automamization in dismantling of printed cir-cuit boards. Paper presented at the international seminar on life cycle engineering, RECY'94, Erlangen, Germany
43. Friedrich R (1996) Identifizierung elektronischer bauelemente und dren gezielte demontage. *VDI-Verlag*
44. Zussman E (1995) Planning of disassembly systems. *Assembly Autom* 15(4):19–22
45. Torres F, Gil P, Puente ST, Pomares J, Aracil R (2004) Automatic PC disassembly for compo-nent recovery. *Int J Adv Manuf Technol* 23(1–2):39–46
46. Torres F, Puente S, Díaz C (2009) Automatic cooperative disassembly robotic system: task planner to distribute tasks among robots. *Control Eng Pract* 17(1):112–121
47. Gil P, Torres F, Ortiz FG, Reinoso O (2006) Detection of partial occlusions of assembled components to simplify the disassembly tasks. *Int J Adv Manuf Technol* 30(5–6):530–539
48. Torres F, Puente ST, Aracil R (2003) Disassembly planning based on precedence relations among assemblies. *Int J Adv Manuf Technol* 21(5):317–327
49. Büker U, Hartmann G (1996) Knowledge based view control of a neural 3-D object recogni-tion system. Paper presented at the 13th international conference on pattern recognition
50. Merdan M, Lepuschitz W, Meurer T, Vincze M (2010) Towards ontology-based automated disassembly systems. In: *Industrial electronics conference (IECON)*, pp 1392–1397
51. Bailey-Van Kuren M (2006) Flexible robotic demanufacturing using real time tool path gen-eration. *Robot Comput Integr Manuf* 22(1):17–24
52. Bailey-Van Kuren M (2005) A demanufacturing projector–vision system for combined man-ual and automated processing of used electronics. *Comput Ind* 56(8–9):894–904
53. Hucht A (1996) Automatisierung: Roboter zerlegen Elektronikschrott. *Umwelttechnik* 2:36
54. Gengenbach V, Nagel H-H, Tonko M, Schaefer K (1996) Automatic dismantling integrating optical flow into a machine vision-controlled robot system. In: *IEEE international conference on robotics and automation*, pp 1320–1325
55. Tonko M, Nagel H-H (2000) Model-based stereo-tracking of non-polyhedral objects for

- automatic disassembly experiments. *Int J Comput Vis* 37(1):99–118
56. Kahmeyer M (1995) Flexible Demontage mit dem Industrieroboter am Beispiel von Fernsprech-Endgeräten. IPA-IAO Forschung und Praxis. Springer, Germany
57. Rupprecht R (1998) Flexibel automatisierte demontage von fahrzeugdachern. IPA-IAO—Forschung und Praxis. Springer, Berlin

58. Feldmann K, Scheller H (1994) Partial automated disassembling of used electronic products and their components. Paper presented at the IEEE international symposium on electronics and the environment, San Francisco, USA
59. ABB (2004) RAPID overview the heart of robotics—RAPID reference manual, 3HAC16580-1 Revision C edn.

Chapter 4

Vision System

Abstract Any system that needs to make dynamic decisions based on a variable situation requires some form of perception. The system must first determine what the current situation is before it can appropriately react to this situation. The system of sensing, recognising, and localising objects of disassembly, its components and/or fastening elements at a distance is referred to here as the *vision system*. This chapter provides an introduction to the requirements and considerations for the implementation of a vision system for autonomous disassembly.

4.1 Introduction

4.1.1 Why a Vision System?

Experience in the field of robotic disassembly has indicated that robotic disassembly cannot simply be considered as the reversal of assembly [1], due to two main factors: the use of irreversible fasteners and the presence of a higher degree of uncertainties. While the former can only be handled with (semi-) destructive disassembly and an appropriate choice of hardware, appropriate compensation for the latter falls largely within the domain of the vision system.

An accurate world model is required in order to respond appropriately to the given situation. If the existence, location, condition and tolerances of all components can be guaranteed, as may be the case in *assembly*, it may be sufficient to employ an autonomous system that is “blind” and relies entirely on internal knowledge. For *assembly*, because lot sizes are known in advance, it may be feasible to reduce job complexity and increase accuracy by designing appropriate jigs and fixtures specific to the product design. A high emphasis on precision leads to an almost deterministic execution of programs. Small deviations may be compensated using force and torque sensors; however, one expects that all fundamental knowledge, including the effect of actions is correct.

It is often uneconomical to design a *disassembly* system to suit a particular product design. Flexible methods of fixing the product in the workspace may mean that errors in product location are higher. A variety of products are expected, in unpredictable lot sizes, arriving with varying end-of-life product conditions. A commonly-successful disassembly operation may fail due to rust or damage during the usage phase or subsequent transport. External appearance is often insufficient to predict such results ahead of time. Due to repairs or modifications during the usage phase, components may also be exchanged, added or missing. These problems can be broadly summarised by saying that there is an increased degree of *uncertainties* in disassembly. A smarter system is required, capable of reacting to deviations in the object of disassembly, rather than relying entirely on static knowledge [2].

In this book, the broad term *vision* is used to describe all scene sensing techniques, including those that provide or use distance data instead of light intensity. The employment of a vision system enables the system to deal with such uncertainties by updating the agent with knowledge specific to an object to be disassembled. A vision system consists of:

- A method of obtaining *actual geometric data* of a scene from a distance
- The processing of raw sensory data to *semantic information* regarding the contents of the scene, e.g. detected components and their location.

The automatic localisation of products and components enables the system to be more flexible to positioning errors. Automatic identification of products reduces the effort required in setting the system up for each product's arrival. Automatic detection of components allows the system to dynamically respond to product modifications. The success or failure of operations can be verified by *seeing* whether the goal state has been achieved, and in the case of failure, back-up measures can be taken. Furthermore, the ability to detect relevant disassembly elements opens the possibility for the disassembly of previously-unknown product variants, using only knowledge of the generic structure of a product type.

However, vision processing is a non-trivial task and a continually-developing research area. Potential solutions are often limited by high processing times and/ or inappropriately high failure rates. The following sections provide an overview of the considerations involved in the implementation of a vision system for disassembly robotics.

Foreseeable alternatives for a “blind” disassembly system also exist, with a lower degree of autonomy, lower speed and higher tooling and labour requirements. For completeness, one such alternative is as follows:

- A generic product is manually positioned; the hardware setup ensures that the product remains in a known position for the entire procedure.
- A human worker inputs the product ID, and if necessary, the current stage of disassembly.
- The robot is pre-programmed with “skills” such as gripping and unscrewing, which are defined entirely by an initial tool position and subsequent contact sensing.

- To program or “teach” new action instances, the robot is manually positioned for the first execution of the skill. Parameters (e.g. location of contact surfaces) are saved for future execution.
- If the robot has already been programmed for this disassembly stage, the programmed actions are automatically executed.
- When unexpected forces are detected, the program terminates automatically with a call for assistance. Touching the “wrong” surface with the tool must be inconsequential.

However, such a solution requires for the entire procedure to be manually taught with the introduction of each new product variant, and is not robust to product modifications, inducing additional labour costs. Every incoming product must be manually identified. Manual positioning may be both slow and physically demanding. A lack of vision constrains the robotic system from further autonomy.

4.1.2 General Requirements

The function of the vision system is to obtain relevant scene information from the real environment, so that the system is robust to the aforementioned uncertainties. This consists primarily of the *detection* of disassembly-relevant objects, which entails

- *Recognition*: identifying or classifying the respective object; and
- *Localisation*: obtaining the coordinates of the object in 3D space.

A product may contain multiple instances of each main or connective component. In this case, each individual element needs to be detected. Additionally, since the result of disassembly actions is nondeterministic, it is important for the system to be able to assess the success or failure of performed operations. This is referred to as *execution monitoring*. Detection of *state change*, the effect of removing a component from the product, is an essential element of execution monitoring.

Simultaneously, the vision system must be insensitive to factors which are irrelevant to disassembly, but nonetheless cause variations in the raw image data. These factors include system-related variables such as changes in ambient lighting and the positioning of the product or camera. Such problems may be resolved either at the system end (e.g. with controlled lighting) or accounted for in the software implementation. While the former may involve more detailed planning and higher setup costs, the latter is often more prone to error. In any case, slight differences in product appearance such as varied placement of stickers or the presence of dirt or rust should have minimal impact on the vision system's ability to detect and localise objects.

From a case-study of disassembly of used cars investigated by Tonko et al. [3], the authors summarise the vision-based challenges commonly encountered in the disassembly process: (a) detection of rigid objects, (b) objects located in front of a complex background, (c) partial occlusion, (d) 6-DOF estimation of objects, and (e) low contrast image due to some covering mixture e.g. oil and dirt.

Table 4.1 General requirements of a vision system for autonomous disassembly

High sensitivity	Low sensitivity
• Product (type and location) • Main components (type, location, and number) • Connective components (type, location and number) • State change	Variations in physical appearance of the same product, component or connective element, e.g. • Dust, rust, wear • Location of stickers, wires Variations of appearance originating from the disassembly environment, e.g. • Product/camera location • Sensor noise • Ambient lighting • Background

The desired qualities of a vision system for autonomous disassembly are summarised in Table 4.1.

Furthermore, since the relevant information must be gathered *on-line* during the disassembly process, execution time for detection procedures should remain minimal. Optimally, all vision processing tasks should be integrated into the process such that they are completed during the time required for physical movements, hence requiring no additional time overhead [4].

Figure 4.1 demonstrates some typical challenges that may be faced by a disassembly vision system: complicated scenes; shiny surfaces affected by background and ambient lighting; the presence of dirt and scratches; and disassembly elements such as cables and tape, which may vary in appearance between products of the same model or even during the disassembly process.

4.2 System Setup

The first step in implementing a vision system is appropriate choice of hardware. Hardware must be chosen in view of the eventual requirements

of recognition and localisation. Following is a list of relevant factors influencing hardware choice.

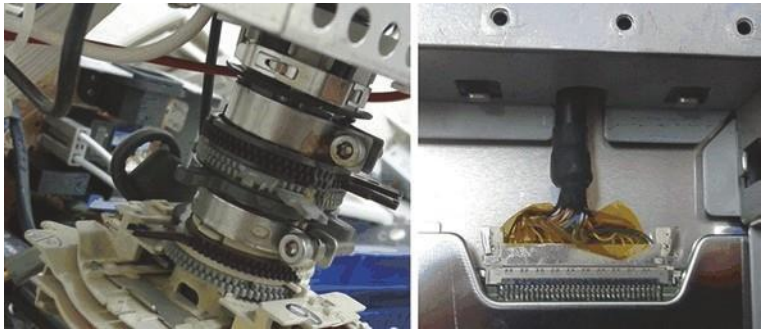


Fig. 4.1 Examples of challenges faced by a disassembly vision system

Technique and Data Collection

The fundamental technology in obtaining data controls the type and limitations of the data received. Various sensor technologies are reviewed in the following section. Furthermore, the physical setup may significantly impact the subsequent processing requirements and results gained. A camera mounted to a robot (an *eye-in-hand* camera) allows viewing of components from controlled distances and angles; however requires additional processing time for image acquisition, and has an additional error source due to robot positioning.

Resolution

The resolution describes the amount of detail in the received data. Camera resolution typically refers to the number of pixels obtained in each image. For depth cameras, the *depth resolution* (the minimum change in depth that can be sensed) may also be given, and may vary depending on distance. Since the technologies reviewed here view a scene from an approximate point, lateral resolution on objects can be increased by moving the camera closer to the object.

Sufficient resolution is required both for recognition and localisation. However, these measures of resolution do not guarantee any amount of accuracy or detail on sensed objects; noise, blur and sensor-specific weaknesses may severely reduce the useful information present in an image. Resolution comes at a trade-off with speed; image processing at excessively high resolutions only expends unnecessary time.

Range

The range of the sensor should be considered with regard to the product size and constraints in the physical setup. The field of view determines the minimum distance required between sensor and object to hold the entire object in view. The clarity of optical images also depends on focus; the focusing system inside a camera also limits the distances at which clear images can be taken. Range sensors tend to have a limited range defined in the specifications corresponding to the specified accuracy.

Frequency and Speed

Speed should be considered in terms of the entire scene acquisition process. If the disassembly system has to wait for scene acquisition, this

time required directly impacts disassembly efficiency. This is primarily apparent when actuation is a part of the acquisition process, e.g. when multiple images need to be taken using one sensor at different locations to view an entire scene.

Size and Weight

The size and weight of a camera system are primarily only relevant in eye-in-hand systems, where physical size impacts the workspace and dexterity of the robot and limits its ability to undertake other tasks.

Cost

Only low-cost systems can be economically justified, due to the low gains expected from disassembly and recycling.

4.2.1 Sensor Technologies

Conventional Cameras

Similar to human's eyes, conventional cameras sense a disassembly scene by measuring the reflection of ambient lighting in the visible spectrum by the objects to be disassembled. This is by far the most common camera technology found in literature, due to their price and availability compared to other sensors, as well as the rich information that can be gained. A camera captures 2D images of a scene; multiple overlapping images taken at different locations may be used to reconstruct a 3D scene (stereo vision) at the cost of additional processing. A monochrome camera only captures light intensity; a colour camera captures the intensity at multiple wavelengths, typically red, green and blue. Sufficient ambient lighting is required to minimise required exposure time and the effect of noise. Movement during camera exposure causes blurring; hence, images should only be taken when the camera is stationary or moving slowly.

Due to the advent of the internet age, consumer webcams are now low-cost (from about \$40), widely available, and offer sufficient frame rate (15–30 frames per second) and resolution for most purposes. However, the image quality is usually lower than that of high-end cameras used in specific applications, e.g. medical, surveillance camera, etc. Advantages of these specialised cameras may include higher resolution, higher frame rate, more accurate colour, less noise, less lens distortion and control of camera parameters.

Stereo vision can be implemented via the use of multiple cameras (which may be moveable or stationary) or one moveable camera. Bükér et al. [5] describes a technique that reports a depth accuracy of 2 mm using two moveable cameras with controlled focus, zoom and aperture (see Fig. 4.2). Using a fixed-focus webcam in an eye-in-hand system to take multiple images along an axis, Fontes and Brandão [7] reports a depth accuracy of approximately 5 mm. In this instance the focal length was estimated in the calibration process, as such information for the webcam was unavailable. The authors attributed this inaccuracy to the focal length estimation process. Regarding recognition, difficulties result from the varied

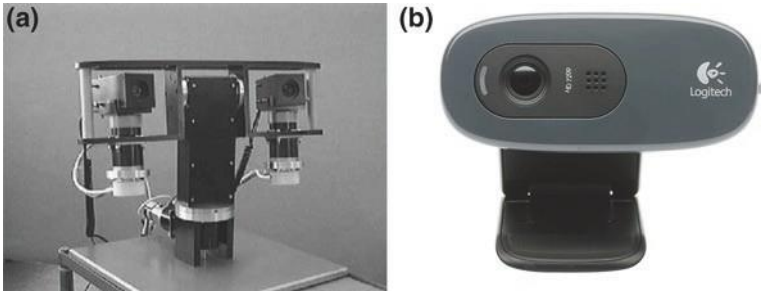


Fig. 4.2 Types of cameras used in vision system [5, 6]. **a** Stereo cameras. **b** Webcam

appearance of objects due to (a) uncontrolled ambient lighting, (b) reflection (particularly by shiny surfaces) and (c) shadows.

To minimise the effects of the ambient light, it is recommended to control lighting levels where feasible. Reflective surfaces display large alterations in appearance when the relative positions of the objects, camera and lighting are altered. Furthermore, the appearance of other objects may be reflected from a shiny surface. In our setup, we have observed that shiny surfaces reflect the orange colour of our robot. At the cost of process time, it is possible to reduce this effect by attaching a plain, neutral-coloured shield to one side of the robot and turning the robot to this side during image acquisition.

Shadows also lead to significant appearance variations in different relative positions of the objects and lighting, in particular as larger components are removed from the product. Shadowed regions have altered and potentially insufficient lighting, hindering object recognition. The effect of shadows may be reduced by using diffuse lighting sources from multiple angles, or attaching a light source as near as possible to the camera. Colour information is largely independent of lighting, however is not applicable in all contexts.

Hyper-Spectral Imaging

Information regarding the reflected spectrum of materials may aid recognition and facilitate material identification. Serranti et al. [8] describe a method for classification of recycling materials using hyper-spectral imaging in the visual and near infra-red spectrums. It was shown that it is possible to distinguish between granules of polyethylene (PE) from polypropylene (PP) in the 1,000–1,700 nm wavelength range, as well as PE PP mix from wood, aluminium and foam in the 400–1,000 nm wavelength range. Similarly, Freitag et al. [9] report an ability to distinguish between five common types of polymers found in computer industry waste using their spectra in the 800–1,700 nm range after smoothing and differentiation. The described approaches used spectrographs with precise lighting control. A spectrograph measures the spectrum of a beam of light, precisely and at high resolution, by scattering it into its component wavelengths. However, as only one line is scanned at a time, this method requires some means of actuation and is slower than desired.

Using specialised filters atop a conventional camera sensor, one can also measure light intensity at different (albeit less precise) wavelength bands. Imec recently unveiled such a camera capable of outputting images in 32 channels in the 600–1,000 nm range at 256 pixel resolution [10]. This technology allows a coarse spectral analysis of the entire scene from one image, however is not commercially available at the current time. Alternatively, it is possible to measure reflectance spectra using controlled lighting, by taking multiple images while emitting different frequency light, for example from a selection of coloured and infra-red LEDs. It is uncertain whether the spectral resolution of such methods is sufficient for material classification. Additionally, such approaches may be hindered if surfaces are covered in paint, dirt or dust.

Active Depth Sensors

An active sensor takes measurements by emitting its own source of energy and measuring the environment's response. The use of an integrated active depth sensor reduces the implementation effort and computation time required for stereo vision. In comparison to stereo vision, active sensors are more resistant to changes in ambient lighting and have an advantage in localising plain surfaces lacking in features; however may be less accurate with complicated scenes where there are rapid changes in depth.

Originally developed as a gaming controller, the Microsoft Kinect [12] uses *structured light* (a projected and reflected pattern of light) to create a depth image, which is provided alongside a colour image. Due to its affordable price, integrated setup, and ease at which depth information can be obtained, this soon saw widespread application in indoor robotics. The accuracy and resolution of the Kinect is investigated in [13]. Results showed a quadratic increase in error and decrease in resolution with distance. Shiny surfaces cause areas of overexposure, leading to gaps in the depth data which are considered *blind areas*. Similar systems include the PrimeSense Carmine

[11] and the ASUS Xtion PRO LIVE [14], both of which are smaller than the Kinect and require less power. In particular, the Carmine 1.09 (see Fig. 4.3) is designed for short-range sensing and may be more appropriate for automated disassembly. An external camera may also detect the projected light patterns if it is not shielded from infra-red light. This may interfere with object detection using external cameras.

A *time-of-flight* (ToF) sensor measures the time taken for a beam of light to travel the distance to and from the target. As a member of this category, 2D laser scanners have demonstrated reliable performance and are now widely accepted both in research and industry [15]. However, the need to actuate such a laser scanner to produce a 3D image again causes acquisition to be more complex and slower than desired. A lock-in ToF sensor (e.g. the MESA SR-4000) measures distance by emitting a modulated signal and measuring the phase change of the signal received at each pixel. An entire depth image is acquired in each scan, albeit at lower resolutions. A review of lock-in ToF sensors is presented in [16]. Range is limited by the modulation frequency, whereby in certain models, out-of-range points may be indistinguishable from the in-range counterpart that has a phase difference of 360°. This effect is reported to be suppressed in

the MESA SR-4500 [17]. Both lock-in ToF sensors and 2D laser scanners suffer an error whereby pixels lying on edges, where emitted light is partially reflected from multiple surfaces, report false values that lie between the two depth values. Stoyanov [15] reports the results from a comparative evaluation suggested that the Kinect and SR-4000 perform (on average) equally well at “low range” (<3.6 m). The SR-4000 displayed better

Fig. 4.3 The Carmine

1.09 integrated camera and structured-light depth sensor [11]



performance when greater distances are included but was found to be susceptible to the aforementioned range ambiguity. Performance was measured by the number of (positive and negative) reference points correctly and incorrectly classified.

4.2.2 Comparative Summary

A summary comparing examples of aforementioned sensor technologies are presented in Table 4.2. The maximum values are presented for resolution and frame rate. These settings are often variable. Exposure time for cameras is required for the sensor to gather enough light; some conventional cameras or camera software may automatically lower frame rate to compensate for low lighting levels. The frame rate for ToF cameras is dependent on integration time.

4.2.3 Overview of Configuration

In this research, the vision system is designed to be low cost, while functionally meeting the requirements for automated disassembly (see detail in Sect. 4.4). The hardware was selected from the options described in previous section. As a result,

Table 4.2 Specifications and market prices of examples of aforementioned sensor technologies at the time of writing this book

Sensor	Resolution (pixels)	Range	Frame rate (fps)	Size (mm)	Apprx. cost (USD)
Webcam (logitech HD C270)	1,080 × 720	Fixed focus (manually adjustable)	30	84 × 60 × 32	\$40 [6]
Webcam (logitech HD C615)	1,920 × 1,080	Autofocus	30	69 × 40 × 34	\$70 [18]

High performance camera for industrial imaging (PixelINK PL-B776)	$2,048 \times 1,536$	Focus determined by lens (standard mount; not included)	4,000 at 8px resolution 12 at max resolution	$102 \times 50 \times 41$	\$1,095 [19]
Structured light + colour camera (PrimeSense Carmine)	640×480	Colour camera: auto-focus 1.08: 0.8–3.5 m 1.09: 0.35–1.4 m	60 30 at max resolution	$180 \times 25 \times 35$	\$200 [11]
Lock-in time-of-flight sensor (MESA SR4500)	176×144	0.8–9 m	30	$119 \times 75 \times 69$	\$4,650 [17]

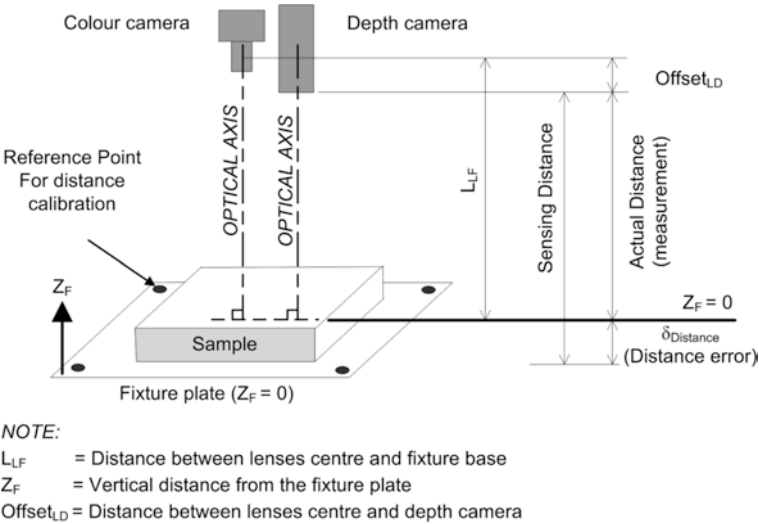


Fig. 4.4 Configuration of the cameras over the fixture plate and distance calibration

the system consists of two cameras, including a colour camera¹ and a depth camera.² The two cameras are mounted parallel and adjacent to each other at

1.2 m above the fixture plate which is set as $z_F=0$ (see Fig. 4.4). The mounting location is selected to minimise the effects from lens distortion and perspective as illustrated in Fig. 4.5 (Distortion map in Fig. 4.5a is illustrated in [13]).

At this mounting distance, colour camera images can be captured with the highest precision possible (mm/pixel) while still maintaining view over the entire fixture plate (see Fig. 4.6). This position is located above the robot’s working area, hence avoiding any possible crashes. Carefully adjusting both cameras to be parallel simplifies image perspective calibrations.

Controlling lighting conditions simplifies the problem of visual recognition. To reduce the effect of shading, daylight light bulbs (18 W, 6,400 K) are installed, projecting light at 45° onto the horizontal plane. Consequently, the majority of areas are lit by at least one light. High intensity light allows the colour camera to obtain better quality images

with lower noise and wider depth of field. Since colour is crucial in the recognition algorithm, the setup is calibrated for colour

¹ *The colour camera* captures a 1,000 × 1,000 pixel single channel 11-bit image encoded with the Bayer filter, which is decoded in pre-processing. The resulting colour image is appropriate for observing small details and colour-based features. The precision in the horizontal xy-plane at $z_F = 0$ is 0.57 mm/pixel.

² *The depth camera* which is a Kinect sensor is used to generate 2.5D depth images. This simplifies localisation by measuring the distance to the object. The easily-obtained 3D object geometry also aids in recognition. The precision of the 640 × 480 pixel depth image is 1.74 mm in the

horizontal xy-plane at $z_F = 0$ and 3.37–4.39 mm in height within the operational range.

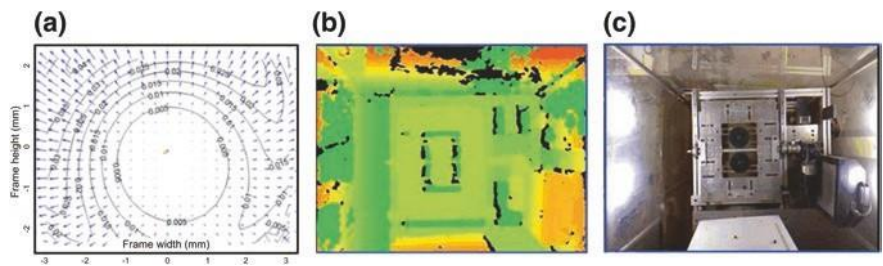


Fig. 4.5 Raw images and distortion field of the Kinect sensor. **a** Distortion of depth image. **b** Depth image. **c** Colour image

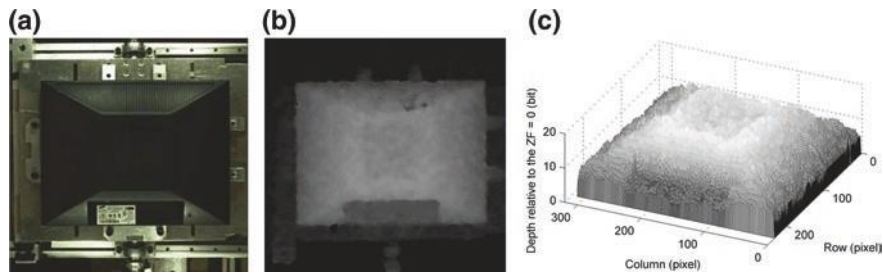


Fig. 4.6 Images from *top* view. **a** Colour image. **b** Transformed depth image. **c** 2.5D depth map

balance [18]. The hardware configuration, as well as the calibration process, are explained in detail in the following section.

4.2.4 Calibration and Localisation

Each camera senses information pertaining to an object’s position with respect to its own coordinates. Information may need to be combined from multiple sources. In order to direct physical actions, the coordinates then need to be transformed into a coordinate system known by the robot.

Calibration is required to discover the correct parameters for the coordinate transformations. These issues are explained with respect to our system in the following sections.

Calibration of the Depth Image

The 640 × 480 pixel depth image needs to be aligned to the 1,000 × 1,000 pixel colour image, which is the main image. The optical axis of each camera is fixed perpendicular to the fixture plate (see Fig. 4.4) so that the image planes are parallel to the fixture plate. The calibration is performed in two parts: (a) image alignment, and (b) distance calibration.

For *image alignment*, an affine 2D transformation is applied to the depth image (source image coordinates) in order to geometrically align the pixels to the colour image (destination image coordinates). The affine transformation matrix $\mathbf{M}_{Affine}$ is a 2×3 matrix containing three types of geometric transformation parameters: (a) 2D rotation, (b) scaling, and (c) translation. These parameters are represented as a 2×2 rotational matrix and 2×1 translational vector inside the 2×3 transformation matrix. The source image maps to the destination image by warping as according to Eq. (4.1) [19]. Points in the source and destination images are represented by $\mathbf{X}'_{src} [c_{src} \ r_{src} \ 1]$ and $\mathbf{X}_{dst} [c_{dst} \ r_{dst}]$ respectively. These locations are represented in image coordinates (c, r) , where c is the column index and r the row index. The origin point $(c, r) (0, 0)$ is located at the top-left corner of the image. The elements in $\mathbf{M}_{Affine}$ are numerically solved from pairs of corresponding points in both images.

$$\mathbf{X}_{dst} = \mathbf{M}_{Affine} \mathbf{X}'_{src} \quad (4.1)$$

Distance calibration is performed by comparing sensed distances (D_{sense}) to measured actual distances (D_{actual}). The sensed distances are calculated according to Eq. (4.2) [20] from the corresponding 11-bit pixel value of the depth image (ranging 0–2,047). The actual distance between the depth camera and the upper surface of the fixture plate is physically measured. The distance used for calibration is the average distance of four reference points located near the corners of the fixture plate (see Fig. 4.4). In this research, vertical distance is represented by z_F , the vertical distance above the fixture plate. z_F will be used for object localisation. $z_F(c, r)$ at a particular coordinate is computed from Eq. (4.3).

$$D_{sense}(c, r) = 123.6 \times \tan \sum \frac{PixelValue(c, r)}{2843.5} + 1.1863 \quad (4.2)$$

$$z_F(c, r) = L_{LF} - Offset_{LD} - D_{sense}(c, r) + D_{actual}$$

Ma
ppi
ng

Camera Configuration and

of Coordinate Frames

(4.3)

The relation between the image plane (spatial sampling) and operational space is determined by the coordinate mapping process. The mapping process is applied to both images after they are geometrically aligned. The frame mapping is performed based on a camera calibration matrix containing two types of parameters: (a) *intrinsic parameters*, and (b) *extrinsic parameters* [21]. The *intrinsic parameters* represent the characteristics of the lenses and the image sensor,³ and consist of focal length (f), scale factor in both directions (α_x and α_y), and the offset of the image coordinate with respect to the optical axis (X_0 and Y_0). The *extrinsic parameters* represent the relation of position and orientation between each

³ In this research, the approach to obtain intrinsic parameters is simplified by two assumptions:

(a) the camera is equipped with low distortion lenses, and (b) the physical position and orientation of the camera are accurately adjusted.

coordinate system. Therefore, a position in the operational space can be written as a function of the image space and the aforementioned parameters as seen in Eq. (4.4), where $\{L_{Offset}\}$ represents the constants involved in the translational and rotational transform between the origins of the coordinate frames.

$$position(x, y, z) = H c, r, z_F | \alpha_x, \alpha_y, f, X_0, Y_0, L_{Offset} \quad (4.4)$$

The configuration of the system is illustrated in Fig. 4.7. The system consists of four physical components resulting in four physical coordinate frames: (a) *robot base frame* {B}, (b) *fixture base frame* {F}, (c) *tooltip frame* {T}, and (d) *lenses centre frame* {L}. In addition, two virtual coordinate frames are set up to derive the geometrical relations inside the colour camera: (a) *a spatial sampling frame*

{S} and (b) *an image plane frame* {I}.

In addition, the *product coordinate frame* {P} is defined in order to describe the geometry and disassembly operation parameters with respect to each product. This is the primary coordinate frame used to store product-specific information, including robot movement paths, in the disassembly process. Since the location of

{P} changes according to each disassembled sample, this is not considered in the calibration phase. The conversion between {B} and {P} is done during the disassembly process.

The relation between these coordinates observed from the top-view is shown in Fig. 4.8. In summary, the coordinate frames in this system are listed in Table 4.3.

The perspective transformation in the camera is illustrated in Fig. 4.9. The spatial sampling frame defines the 2D position of a pixel on the image sensor. The

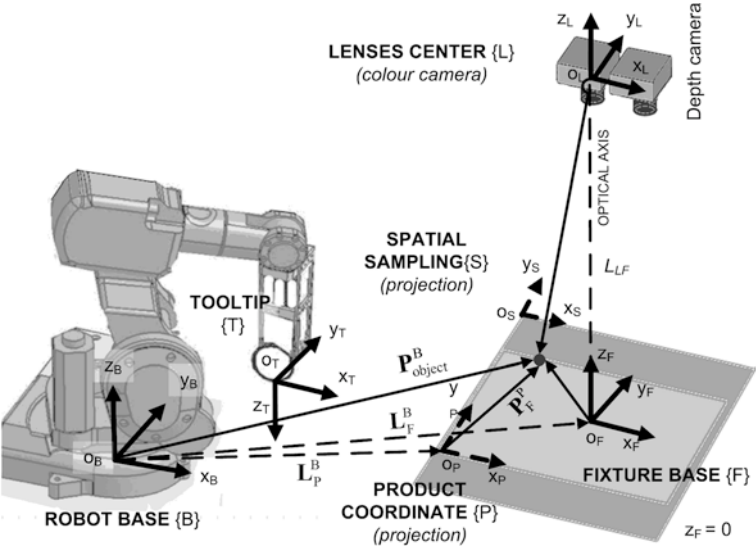


Fig. 4.7 Configuration of the disassembly cell

Fig. 4.9 Perspective transformation in the camera

captured image from the colour camera is a projection of the objects onto this xy - plane. The origin of the spatial sampling frame is located at the top left of the image. Pixel positions are represented by x_s and y_s which correspond to c and r on

the image plane respectively.⁴ From Fig. 4.9, the relations between the physical position of the object ($\mathbf{p}_{object}^L$) and the variables obtained from the cameras (c , r , and z_F) are presented in Eqs. (4.5–4.7).

$$c = x_S = - \frac{\alpha_x f \cdot p^L_{x,object}}{p^L_{z,object}} + X_0 \tag{4.5}$$

$$r = -y_S = - \frac{\alpha_y f \cdot p^L_{y,object}}{p^L_{z,object}} + Y_0 \tag{4.6}$$

$$z_F = L - \frac{p^L_{z,object}}{f} \tag{4.7}$$

Referring to the relation shown in Eq. (4.4), the position of the object relative to {B} can be obtained by computing the transformation matrices shown in Eq. (4.8). $\mathbf{p}_{Object}^L$ can be derived from Eqs. (4.5–4.7).

$$\mathbf{p}_{object}^B = \mathbf{T}^B \mathbf{T}^F \mathbf{p}_{FL object}^L \tag{4.8}$$

wher
e

$$\mathbf{T}_F^B = \left[\begin{array}{ccc|c} -1 & 0 & 0 & L_{F,x}^B \\ 0 & -1 & 0 & L_{F,y}^B \\ 0 & 0 & 1 & L_{F,z}^B \\ \hline 0 & 0 & 0 & 1 \end{array} \right]; \mathbf{T}_L^F = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_{LF} \\ \hline 0 & 0 & 0 & 1 \end{array} \right]; \text{ and } \mathbf{P}_{object}^L = \left[\begin{array}{c} P_{x,object}^L \\ P_{y,object}^L \\ P_{z,object}^L \end{array} \right]$$

The resulting $\mathbf{P}_{Object}^L$ is shown in Eq. (4.9). In general, calibration can accurately determine the value of all parameters ($\alpha_x, \alpha_y, f, X_0, Y_0, L_{LF}, L_{F,x}^B, L_{F,y}^B$ and $L_{F,z}^B$) using numerical methods. Alternatively, the parameters can be directly measured or obtained from the physical system, according to the aforementioned assumptions. The parameters, variables and the methods of obtaining the necessary data are summarised in Table 4.4.

$$\mathbf{p}_{object}^B = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_f} (L_{LF} - z_F(c, r)) (c - X_0) + L_{F,x}^B \\ \frac{1}{\alpha_f} (L_{LF} - z_F(c, r)) (-r + Y_0) + L_{F,y}^B \\ z \\ c, r + L_{F,z}^B \end{bmatrix} \quad (4.9)$$

⁴ The image plane and the spatial sampling frame are equivalent concepts, with a slightly different choice of coordinate system. Image plane coordinates (r row, c =column) are here preferred due to the definition of the image used in most image processing libraries, e.g. OpenCV.

Table 4.4 Summary of parameters and variables for calibration

Parameter/variable	Definition	Approach for obtaining the value	Unit
L_x, L_y, L_z	Offset between {B} and {F}	Physical measurement	mm
L_{LF}	Offset between {L} and {F} along the optical axis	Physical measurement	mm
X_0, Y_0	Offset between {I} and {S}	Measurement from captured image	pixel
α_x, α_y	Scale factor	Calibrate using the measured physical size of object at $z_F = 0$ and its size on the captured image $\alpha_i = \frac{f_{LF} \sum \Delta P^B_{i/object} [\text{mm}]}{\sum \Delta P^{S/I/object} [\text{pixel}]} ; i = x, y$	$\frac{\text{mm}}{\text{pixel}}$
f	Focal length of the lenses	Lenses specification	mm
c, r (variable)	Position on the spatial sampling coordinate frame	Output from processing the captured image	pixel
z_F (variable)	Vertical distance of the object from {F}	Processing from the depth image	mm

Localisation and Product Coordinate {P}

After the calibration process, the system is able to accurately determine the location of the object relative to the robot base coordinate. The locations of the object in 3D operational space are presented in two coordinate frames, {B} and {P}. The robot coordinate frame {B} describes robot movement paths within the scope of the program operated by the robot controller. The product coordinate frame {P} is used in the rest of the system. Using this frame, locations of product-specific features are described relative to the product itself. This information can therefore be used to generalise between samples of the same model or product—an appropriate frame of reference for a learning process (see Chap. 5).

The location relative to the product coordinate P^B can be obtained by multiplying the transformation matrix ${}_{mul-}^{Object}$ as seen in Eq. (4.10). According to the configuration of the system in Fig. 4.7, the final result is shown in Eq. (4.11).

$$\begin{matrix} \mathbf{p}^P \\ \text{objec} \end{matrix} = \mathbf{T}^P_B \mathbf{p}^B_{\text{objec}} = \mathbf{T}^P_B \mathbf{T}^B_F \mathbf{T}^F_L \mathbf{p}^L_{\text{objec}} \quad \Sigma \quad (4.10)$$

$$\begin{matrix} \mathbf{p}^P \\ \text{object} \end{matrix} = \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_f} (L_{LF} - z_F(c,r)) (c - x_0) + L^B_{F,x} - L^P \\ 1 (L_{LF} - z_F(c,r)) (-r + y_0) + L^B_{F,y} - L^P \\ -L \end{bmatrix} \quad (4.11)$$

In conclusion, the calibration process is conducted to determine the extrinsic parameters of the system. A careful consideration of physical system design can reduce the number of calibration parameters and therefore simplify the calibration process. The disassembly process operates according to positions sensed

in 3D space. These positions can be calculated using the calibration parameters and variables c , r , and z_F which are directly obtained from the images. The product coordinate $\{P\}$ is used in majority of the process since it represents the geometry of the product itself. However, the robot coordinate frame $\{B\}$ is also necessary in order to control the movement of the robot.

4.3 Recognition Techniques

The human vision system is excellent at recognising arbitrary objects. A myriad of work has been done in object recognition, yet the current best solutions in computer vision do not compare with the speed and accuracy of our own vision. The problem remains an active research area; this section only seeks to provide a very brief overview of the topic with specific emphasis on prior work in disassembly vision. The problem of recognition roughly consists of the steps of (a) methods of extracting and describing features, and (b) classification.

A *feature* is any measure that can be obtained from the raw data, used to identify objects of interest, e.g. a pixel (colour/lightness intensity), a location, an edge, a corner or any combination of such measures. Some method of *classification* then decides whether or not a region in the image constitutes an instance of the object of interest. At some point, regions of the image may also be *segmented* into different regions to be separately analysed. At least one known instance of the object of interest is required to allow comparison. A larger sample size is typically used to account for natural variations in appearance.

The appropriate choice of techniques largely depends upon application context. Common techniques, particularly those previously used in disassembly robotics, are described in the following sections.

4.3.1 *Thresholding*

An image obtained from a camera is represented as a series of numerical values signifying the intensity of light received at each pixel, i.e. from a particular direction. *Thresholding* is the conversion of such an image into one where each pixel only has a binary value by using some kind of limit or threshold.

A colour camera measures the intensity of light in multiple wavelengths. This information can be described in various *colour spaces* depending on the required functionality. The RGB colour space describes the representation in terms of light intensity in the red, green and blue channels respectively. The Hue-Saturation-Value (HSV) colour space is commonly used to isolate the chromatic properties from brightness. Colour can thus be considered independent of ambient illumination [22]. Converting between colour spaces is simply a mathematical transformation of the channel values for each pixel.

Where the object or background is of a known brightness or colour, the pixel values themselves may be used as the primary method of classification. A threshold colour may be set: pixels with values lying on one side of the threshold are considered to belong to the object of interest; those on the other, as the background. The object locations are then extracted from the resulting binary image. Adaptive thresholding is where the threshold is not constant, but rather derived from the properties of the image or regions within the image.

Thresholding is commonly-used in human skin tone detection [24]. Multiple instances of objects detected through thresholding can further be grouped into regions of conjoined or nearby pixels (see Fig. 4.10). This falls under the area of *clustering* or *blob detection*. Thresholding is also common as a pre-processing technique (e.g. [25]) to simplify the image for further processing.

Thresholding is also useful in the interpretation of depth images. Background clutter can be automatically rejected if it is known that the object lies within a specific height or distance. Similar to colour blob detection, objects can also be segmented using the criterion of minimum depth change or distance from other objects.

4.3.2 ***Edge Detection and Contour Geometry***

Contours or ***edges*** are lines separating regions of contrasting colour or intensity. On a camera image, edge detection is commonly performed by approximating the derivative of the image (*image gradient*), which has greatest magnitude in regions of intensity change. The direction of the image gradient can be used to determine the direction of the edge. The *Canny edge detector* [26] (see Fig. 4.11) builds on these principles in a technique that outputs a binary image where edges appear as single-pixel-wide lines.

The shape of an edge is naturally descriptive of the object and its geometry, however this data must be further processed into a form that can be compared or analysed. *Hough transforms* locate known and geometrically-defined objects within an image by considering each point as a vote towards all valid transformations of the object in which the point can participate. Votes are tallied in discrete

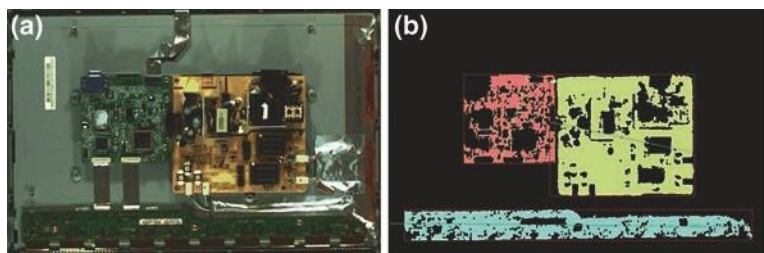


Fig. 4.10 Thresholding and blob detection for the detection of PCBs [23]. **a** Captured image.
b Blob detection on morphed positive area

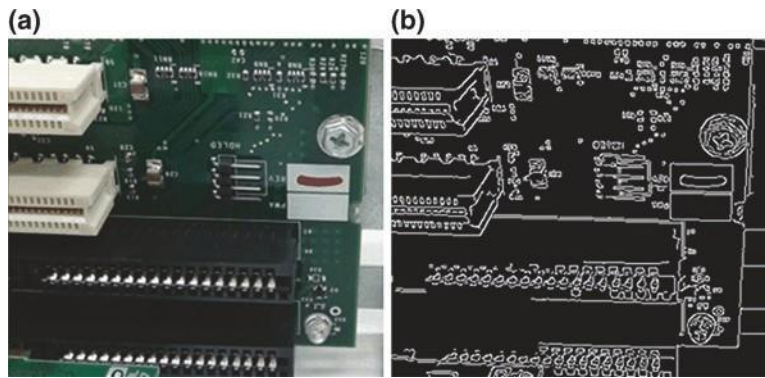


Fig. 4.11 OpenCV Canny edge detector applied to an image of a PCB. **a** Captured image.
b Canny edge detection

bins. Actual occurrences of the objects within the image cause local maxima in the tallies, from which the parameters of the representation (e.g. distance and angle for a straight line) can be directly obtained.

The *Ramer-Douglas-Peucker* or *split-and-merge* algorithm approximates the representation of a line of unknown shape to a series of segments. If the distance between a point and the proposed segment is greater than a threshold, the line is split and a shorter segment extending only to the problematic point is proposed, continuing until all points in the line lie within the given tolerance to the fitted segments.

Using a depth image, [27] uses edge detection to recognise grasping surfaces for a parallel gripper approaching the objective module in a camcorder from above. Parallel, straight edge sections are found after applying the split-and-merge algorithm using circular segments. Candidates are subsequently analysed to ensure the edges belong to the same object and provide enough space for the approach of the gripper.

Inspired by the human vision system, B ker et al. [5] uses Gabor filters to recognise contours at various resolutions and orientations. A General Hough Transform is then used to determine the location of nuts. This method is robust to background clutter, however the presence of dirt and rust meant that nuts “did not exhibit enough contours to determine their position accurately”.

After contour image extraction, Gil et al. [25] uses a Progressive Probabilistic Hough Transform to estimate straight-line sections of wires, from which cutting positions can be estimated. The split-and-merge algorithm using straight-line segments was proposed for the detection of screws. In the displayed results an edge detector appears to have been applied twice. Geometric features of edges, namely lengths, orientations and their distributions, as well as template matching, are used in component recognition.

Scale space filtering analyses edges and intervals (the areas separated by edges) in different *scales*, i.e. after different amounts of smoothing. Considering edges as the zeros of the second derivative, Witkin [28] observes a correspondence

between the perceptual salience of an interval with its stability to changes in scale. A “top-level” description or segmentation of the intervals can be obtained by finding the first local maxima in stability with decreasing scale (as the image becomes increasingly detailed). An edge found at a coarse scale can be localised at a finer scale. Hohm et al. [29] suggests scale space filtering as an appropriate means for the detection of cables.

4.3.3 *Template Matching*

Template matching is the recognition of objects via comparison with a template (i.e. example or model image). The simplest form is pixel-by-pixel comparison of regions within a scene to the template. This may be performed on the raw image or after pre-processing (e.g. edge detection). Template matching is sufficient provided that the objects of disassembly are known and the distance and orientation of the camera are either maintained or the variations compensated for. To reduce the number of comparisons required, it is common to *subsample* the image, i.e. perform the comparison on images of reduced resolution. *Image pyramids* are typically used as an efficient method for subsampling.

This is the method described in Elayed et al. [4] to recognise components within a personal computer after “appropriate scaling, rotation and histogram-equalization” following gross scene segmentation using a depth image. Following the localisation of ceiling beams using a Hough Transform, Rolando Cruz- Ramírez et al. [30] generates a new trajectory for the acquisition of images, maintaining a fixed distance between camera and beam. Template matching is then used to generate hypotheses for the locations of screws. In both aforementioned examples, knowing the location of the containing structure limited the required search space. For component recognition, Gil et al. [25] uses template matching aided by image pyramids, whereby candidate locations are initially found on a low-resolution image. These potential matches are then verified and refined at higher resolutions.

4.3.4 Keypoint Feature Matching

Keypoints are distinctive locations within an image which can be computed and used to recognise objects in a scene (see Fig. 4.12). A keypoint detector extracts locations of interest that optimally occur at repeatable positions in images from different viewpoints, and the descriptor aims to provide a concise and easily-comparable description of the area around each location. The keypoints, instead of every individual pixel, are then compared, often reducing the number of total comparisons that need to be made. Provided that enough keypoints exist on the object, such methods are more efficient than direct template matching when the scale and/

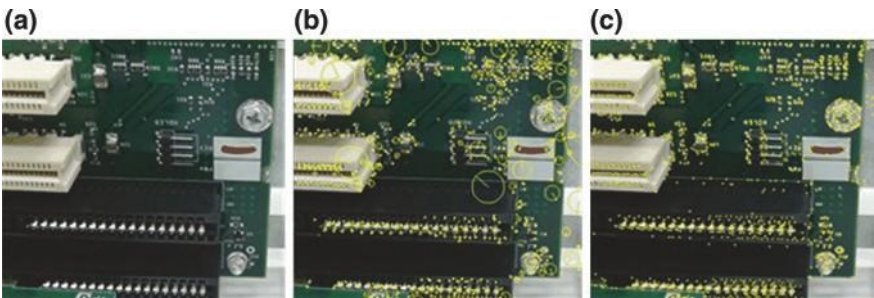


Fig. 4.12 Keypoint detection using SIFT and FAST algorithms. **a** Captured image. **b** SIFT. **c** FAST

Table 4.5 Table of common keypoint detectors and descriptors [31–39]

Blob detectors	Corner detectors	Other descriptors
• SIFT* [34]	• Harris* [38]	• BRIEF [40]
• SURF* [35]	• FAST [39]	• FREAK [41]
• CenSurE/STAR [36]		

or orientation of an object are uncertain. Examples of popular keypoint detectors and descriptors are given in Table 4.5. Blob detectors are typically more stable (robust to changes in viewpoint) than corner detectors; however, corner detectors have the advantage of speed and greater accuracy in localisation [31].

4.3.5 Semantic/Relational Features

Humans recognise objects, not only from the appearance of each object alone, but rather, in context of the scene and prior knowledge. External

information, such as size and location with respect to other components, may produce constraints for where objects can be found, and aid in disambiguating objects of similar appearance. A survey of general approaches to obtain and exploit context-based visual information in computer vision is found in Galleguillos and Belongie [40].

As opposed to general image classification, proposed disassembly systems operate in constrained contexts. In such contexts, it is common to first localise the product to be disassembled, then restrict the search space for smaller components to the area encompassing this product [4, 30]. Jørgensen et al. [41] uses a hierarchical tree model of products, whereby each node represents a component and the root nodes represent product classification. Nodes contain information both regarding normalised expected product position and a weighting factor based on the probability of occurrence within the product, which are used in determining the best identification for detected components. Karlsson and Järrhed [42]

describes a vision system for the disassembly of electric motors, whereby the regions of interest (ROI) for screws are taught manually by a human operator for every model. In future appearances of these models, only the ROIs are searched to verify the presence of screws. The existence of screws in the pre-learned locations is then used to confirm the identification of the motor. For bolt detection on car wheels, B ker et al. [5] further specifies that the bolts lie in a symmetrical arrangement; this is integrated into their Generalised Hough Transform approach.

4.3.6 Further Classification Methods

It is difficult to determine the exact qualities or feature values defining the identification of an object. Statistical and machine learning techniques are commonly used for the classification of a new instance with respect to a known data set. Following is a brief overview of some such techniques.

Fuzzy measures allow measurement of the correspondence of a new instance to a template by first calculating the *membership* or agreement of each measured feature to those of the template. In order to identify incoming products (electric motors), Karlsson and J rrhed [42] manually selected a set of 8 features calculated from a binary image containing the silhouette of the product: area, perimeter, moments of inertia I_{xx} , I_{yy} and I_{xy} and the measurements of the bounding rectangle (long side length, short side length and side ratio). Initial testing showed Gaussian-shaped fuzzy measures to be most appropriate. Each feature in each template (i.e. from one motor type) is assigned a Gaussian membership function, characterised by its mean and standard deviation. This measures how well a new instance correlates with the template according to this individual feature. To produce a fuzzy measure for the template, feature membership values are then fused using Eqs. (4.12) and (4.13).

$$f(\mu_1, \dots, \mu_N) = \frac{1}{K^N - 1} \sum_{i=1}^N -1 + K_i G^{\sum} \tag{4.12}$$

$$G = \frac{1 + \sum_{i=1}^N (K - 1)\mu_i}{K - (K - 1)\mu_i} \quad 1 < K < \infty \tag{4.13}$$

μ_i = membership to feature; N number of features; K a constant parameter [42]. Artificial *neural networks* are methods for function learning and approximation, inspired by biological systems of neurons, often used for classification using multiple inputs. Jørgensen et al. [41] proposes a component recognition system consisting of multiple pre-processing filters (including adaptive thresholding, histograms and edge and corner detectors), feature selection using a “principle component like algorithm”, and input of selected features into a RAM-based neural network used for component recognition. Each neuron in a neural network is

simple function with multiple inputs. These outputs can further be used in successive levels of neurons, allowing approximation of more complex functions. In order to obtain the parameters of the neuron functions, the neural network must first be trained with examples.

The *support vector machine* (SVM) is a similar technique whereby each instance for classification is considered as a point in multidimensional space defined by the value of each feature. Classification occurs according to a function (of defined type) that separates positive and negative training examples by the largest possible margin. After finding template matching alone to be insufficient for the detection of low-resolution screws, Rolando Cruz-Ramírez et al. [30] obtains greater accuracy by further assessing candidates using a SVM classifier.

Decision trees are a machine learning technique whereby the value of individual features are chosen as branching criteria which are successively applied to classify the object of interest. *Boosting* allows more accurate classifiers to be trained by using the votes of multiple weaker (less accurate) classifiers. For object detection, Viola and Jones [43] uses the boosting algorithm AdaBoost on Haar-like features, coarse features which are efficiently calculated and describe the relative brightness of adjacent regions within an image. These features are easily scaled for comparison with a scene image, however are too numerous for comparison of all features. Small classifiers are trained to provide high detection rates, which, during classification, are applied in consecutive stages (i.e. as a *cascade*) to quickly reject windows that are not of interest. For facial recognition, this method was shown to be comparable to existing literature in accuracy, with a significant increase in speed such that it is suitable for real-time applications. However, a large training set (perhaps a few thousand positive and negative labelled images) is required to train such a machine learner, making this method only suitable for standardised and commonly-appearing components.

4.3.7 Conclusions

There currently exists a wide selection of tools and techniques for the purpose of object recognition in computer vision. It is difficult to compare the performance of the various methods in the disassembly vision system due to the variety of tasks, including the image quality and difficulty of the tasks, and system setups described. Additionally, many publications lack information important for the well-informed implementation of an image processing system. Such information includes (where relevant) the algorithm used and the relevant parameters, image resolution, true and false positive rates, accuracy and execution time (with respect to the computer system).

There is so far no general solution for the recognition of disassembly objects, due to the broad scope of the required tasks. Solutions reporting positive results typically employ a range of techniques in succession, and exploit information regarding the relationship between objects. The choice of features needs to be

suited for the task, such that (a) the features are capable of making the necessary distinctions and (b) the number of necessary comparisons does not lead to prohibitive execution time. In order to achieve this with commonly-available technology, and as is the pattern in literature, we suggest an approach whereby the disassembly vision system is considered as an amalgamation of separate tasks, according to different levels within the product. In this way, the requirements of each task can be specifically addressed, and specific characteristics exploited.

4.4 Requirements and Functionality

A *flexible* disassembly vision system requires its functions to be applicable to a large number of product models, not only those that are already known, but with the potential to easily integrate future models. However, constraining problems to specific applications is beneficial for reducing complexity and improving the robustness of a vision system. It is recommended to separate the problem into tasks which are as general as possible with regards to disassembly object, yet limit the scope and requirements of each individual task. This book hence considers the four following categories: (a) *product detection*, (b) *main component detection*, (c) *connective component detection*, and (d) *state change detection*. Their respective requirements and characteristics are detailed in the following sections.

4.4.1 Product Detection

The product is the largest and highest-level object that needs to be detected in a disassembly vision system. Segmenting the product from the

background can be simplified by controlling the background colour (or lighting the background to take a silhouette image), thresholding by depth or location, or determining the difference between the occupied and empty rig. High resolution is generally available: general dimensions and shape can generally be obtained in high detail; however the challenge is to generate a sufficiently small set of features for quick comparison with a potentially-large database of previously-seen products.

A given product has a standard set of components, however may be altered due to repairs and maintenance. When the components are fairly standard, these, or the location of fastening elements may be used to confirm or reject hypotheses for product identification. When multiple products have a similar outer appearance, or multiple possible internal configurations exist for the one product, it may be possible to begin disassembly with multiple possibilities on hand. The ambiguity can be automatically cleared once more of the internal structure is revealed. Further product-specific features that may aid in recognition are identifiers such as logos, text and barcodes. However, Kopacek and Kopacek [44] reported that the barcode itself is insufficient for the full identification of mobile phones.

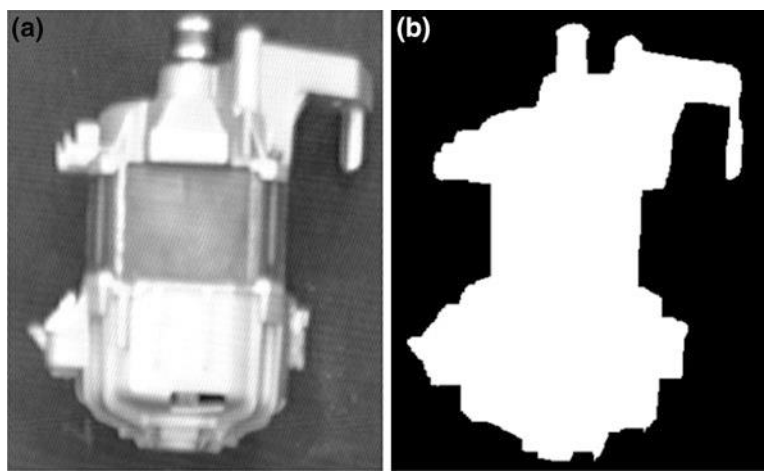


Fig. 4.13 Image processing for motor model identification [42]. **a** Captured image. **b** Processed image

Using controlled background and lighting, thresholding and morphological operations, Karlsson and Järrhed [42] obtains the binary silhouette of electric motors for disassembly (see Fig. 4.13). From these shape profiles, the values of 8 hand-picked geometric features are calculated and compared to the database using fuzzy measures (see Sect. 4.3.6). The screw locations are further used to confirm the classification. This method reported 95 % accuracy from two perpendicular viewpoints, leading to a total accuracy of approximately 98 %.

In this research, SURF is used to recognise the model of LCD screen samples. SURF keypoint features are extracted from the image of the back cover and compared with that of candidate models in the database. The model is recognised if a sufficient number of keypoints is matched. From a preliminary test on 37 different models of LCD screens, this detector achieves 95 % accuracy in which the screens were classified by the threshold of 15 % of keypoints matching. For the same models of screen, keypoints matching varied between 17–100 % according to the noise and little different in lighting condition. An example is shown in Fig. 4.14.

4.4.2 Main Component Detection

The problem of component detection can be approached from two perspectives:

(a) identifying the component by *model* or (b) identifying the component by *type*. The former is more appropriate when the exact component models are known, for example when the goal of disassembly is to obtain specific pieces for reuse. In this case, a strategic use of template matching is sufficient [4, 25]. Alternatively, the goal of component detection may be to recognise components by *type*. Products within the same product family tend to contain the same types of components,

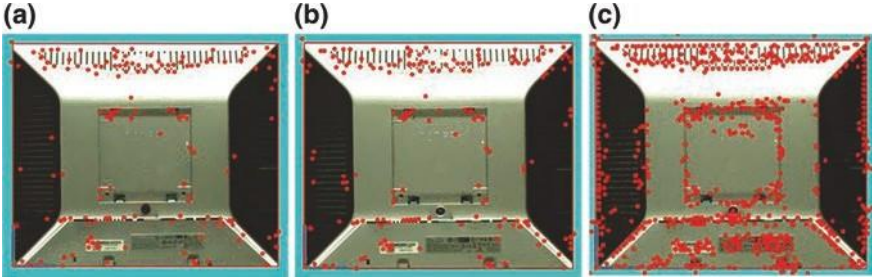


Fig. 4.14 SURF keypoints detected in a sample and a candidate model. **a** Sample. **b** Matched keypoints. **c** Detected keypoints

however with varying physical appearances, due to varying specifications and designs. If *types* of components can be recognised, the system is theoretically able to disassemble even unknown models, with the specific goal of treating each component appropriately. This is more appropriate for recycling and waste treatment. However, a more flexible classification technique is required. In this book, the concept of *common features* for detecting main component type is proposed.

The *common features* of a component type are physical characteristics which are shared between most or all components of this particular classification. Despite design differences, models within a component type often share consistent similarities, as particular features are directly related to the functionality of the component. These common features are used to formulate the rules for component recognition, which in this book will be expressed using logical notation. It is assumed that a particular component is detected at the current state if the corresponding set of component-specific rules is satisfied. The format of the general rules is shown in Eqs. (4.14) and (4.15), which state that an object x is a component type y if the object x satisfies all rules corresponding to the component type y .

$$rule_1(x,y) \wedge rule_2(x,y) \wedge \dots \wedge rule_n(x,y) \supset component(x,y) \quad (4.14)$$

$$component(x,y) \equiv object(x) \wedge componentType(y) \wedge (x \in y) \quad (4.15)$$

The accuracy of detection depends directly on the number of significant common features and how well they distinguish between the various components. A significant number of samples need to be observed in order to define the component types, select the common features and define the appropriate parameters for classification in view of the capability of the vision system. The algorithm to calculate the truth value of each abstract rule also needs to be developed.

The most appropriate set of common features differs according to product type. The following methodology demonstrates the application of this concept to the detection of components in LCD screens, the case study presented in Chap. 6. 37 different models of LCD screens have been observed based on the proposed camera configuration. The common features can be categorised into three main groups: (a) *geometry*, (b) *colour range*, and (c) *texture and connected regions*.

Geometry

The geometry of components can be observed directly from the colour and the depth images. The size, location and shape of an observed component can be roughly described by its *minimum bounding box* (MBB), the smallest box which contains the entire component. MBB size, aspect ratio and height (distance from the front of the LCD screen) are observed to fall within consistent ranges for each type of component. The corresponding rule is then formulated by giving the possible range according to these properties. This rule is shown in Eq. (4.16) where *prop* can be substituted for size, aspect ratio, and height.

$$rule_{prop}(x, y) \subset [component(x, y) \wedge \{prop_{min}(y) \leq prop(mbb(x)) \leq prop_{max}(y)\}] \quad (4.16)$$

A good example for using geometry as a distinguishing common feature is the LCD module, which largely determines the size of the product as a whole. Parameters for this rule can be obtained from standards [45]. The diagonal size of the LCD modules to be disassembled in the disassembly rig ranges from 15" to 19". The height of LCD modules ranges between 10 and 20 mm and their aspect ratio between 4:3 and 16:10.

Colour Range

Due to functional material specifications and production techniques, only minor colour variations are found in most component types. It is assumed that the component is detected if a sufficiently large region of connected pixels within the corresponding colour range is detected. Pixel colour is expressed using the HSV colour space [46]. Blob detection [47] is used to locate regions of interest, that both satisfy the colour criteria and have area greater than Φ_{Blob} [see Eqs. (4.17–4.18)]. If x is a component of type y , the colour criteria (associated with h, s, v) is met for pixel l belonging to the area of blob detected in x if it has both hue and saturation values within the range defined for component y .

$$rule_{colour}(x, y) \subset [component(x, y) \wedge \{area(blob(x)) \geq \Phi_{Blob}(y)\} \wedge satColourPixel(l, h, s, v, x, y)] \quad (4.17)$$

$$\begin{aligned} & satColourPixel(l, h, s, v, x, y) \\ & \equiv \sum_{component(x, y) \wedge pixel(l, colour(h, s, v))} \\ & \wedge \left[H_{min}(y) \leq h(x, l) \leq H_{max}(y) \wedge S_{min}(y) \leq s(x, l) \leq S_{max}(y) \right] \sum \end{aligned} \tag{4.18}$$

Colour range can effectively distinguish printed circuit boards (PCBs) from other main components in LCD screens. PCBs are generally green or yellow, while other components are gray. The gray colour is due to non-colour-coated metal. Parts made of hi-strength and thick plate steel, e.g. carriers, are matte gray. Parts used purely for covering purposes, e.g. the back of the LCD module and PCB cover, are made from light plate steel which is light gray. Significant difference in H and S channels can be clearly shown as histograms in Figs. 4.15 and

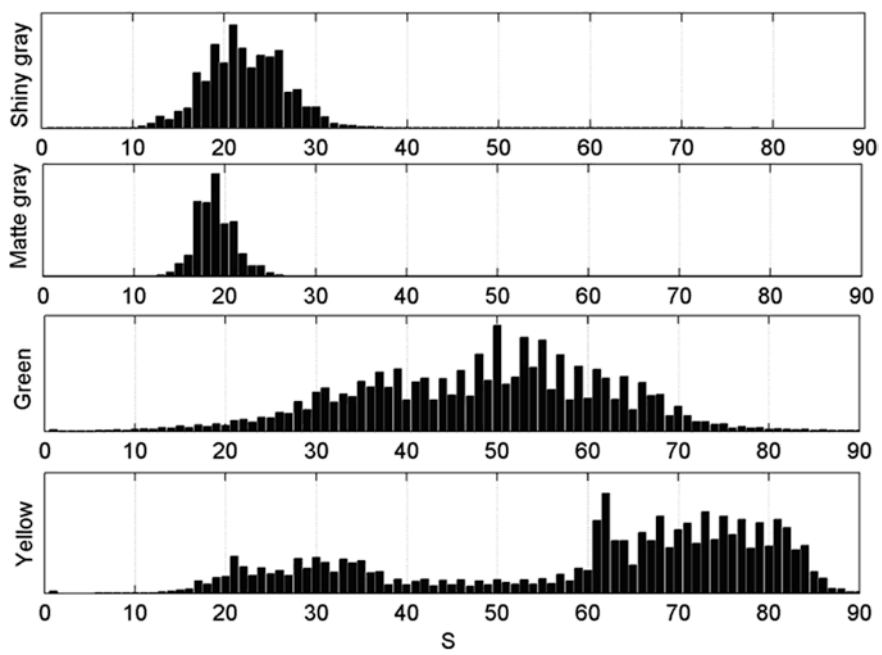


Fig. 4.15 Histogram of the base colour in the S-channel as collected from the samples

4.16 where the colour pixels are collected from the observed samples. The colour range of each component is summarised in Table 4.6. Classification between the components is done using fixed-level thresholding.

Texture and the Connected Region

The observed surface texture is also directly related to the function of the component. This can be classified into two categories: (a) *homogeneous* and (b) *non-homogeneous*. *Homogeneous* surfaces are large connected regions with minor variations in colour and depth due to small features such as ventilation holes. These are typically observed on the metallic main components, e.g. PCB covers, carriers, and LCD modules. Components having *non-homogeneous* texture, e.g. PCBs, usually contain sub-components which are noticeable due to their distinctive colours and depth variations from the base.

A connected region is recognised as a component using the rule in Eq. (4.19) associated with two key indicators: (a) *homogeneity* and (b) *significant size*. The *homogeneity* of the detected area is determined by the ratio between the blob cluster and the size of its minimal bounding rectangle (MBR) [see Eq. (4.20)]. The MBR is used as the region is considered in 2D. The MBR must have significant size in comparison to the area of the entire product. This can be determined by Eq. (4.21). The threshold Φ is used to justify both rules.

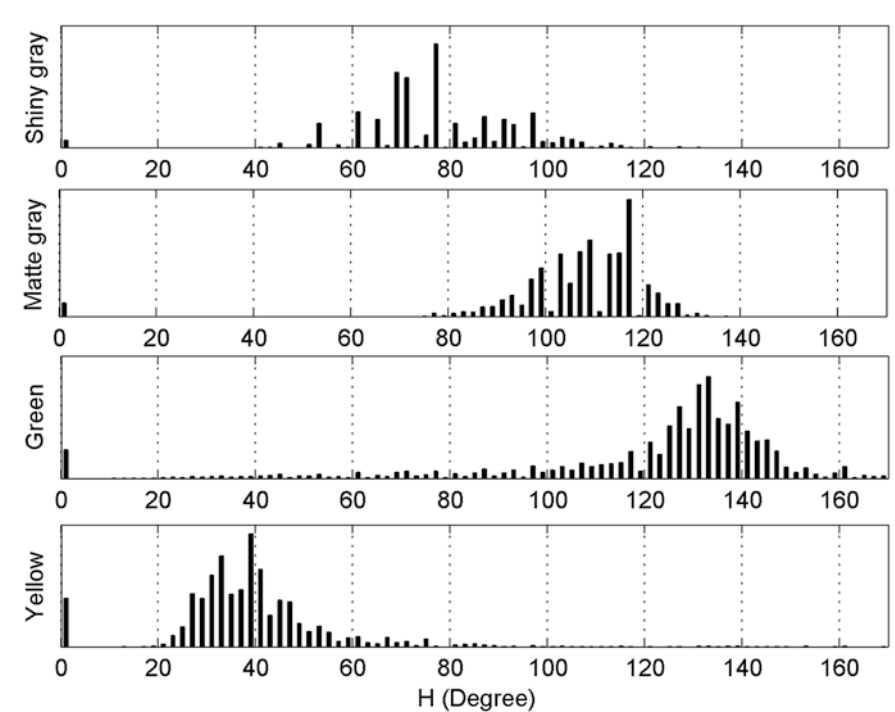


Fig. 4.16 Histogram of the base colour in the H-channel as collected from the samples

Table 4.6 Colour ranges of the components in LCD screens

Component	Colour name	HSV colour range			
		H (0,360°)		S (0,100)	
		Min	Max	Min	Max
Back cover	n/a	–	–	–	–
PCB cover	Matte gray	73°	135°	10	27
	Light gray	40°	128°	9	35
Carrier	Matte gray	73°	135°	10	27
PCBs	Green	70°	200°	35	80

$$rule_{connectedArea}(x, y) \subset component(x, y) \wedge homogeneity(x, y) \wedge significantArea(x, y) \tag{4.19}$$

$$homogeneity(x, y) \equiv \frac{area(blob(x))}{area(mbr(x))} \geq \Diamond(y) \tag{4.20}$$

$$significantArea(x) \equiv \frac{area(mbr(x))}{area(Product)} \geq \Diamond_{Mbr/Product}(y) \tag{4.21}$$

In addition, the surface roughness rule in Eq. (4.22) holds if the surface roughness value (R_a) as measured from the depth image lies within the maximum acceptable roughness for the corresponding component ($R_{a,Max}$). The capability of the depth camera must be taken into account in determining this threshold.

$$rule_{roughness}(x, y) \subset component(x, y) \wedge R_a(x) \leq R_{a,Max}(y) \quad (4.22)$$

In conclusion, this section explained the principle of using common features for detecting main components in disassembly. The detection rules can be formulated from the common features and parameters obtained by observing a number of product samples. An example of a set of rules and common features used to classify between types of main components was shown using the case-study of LCD screens. Further details regarding the detection of main components in LCD screens are given in Sect. 6.6.1.

4.4.3 Connective Component Detection

Automatic detection of connective components, or *fasteners*, is highly desirable in a disassembly vision system, both to verify the existence, type and location of the fasteners before executing an operation, and for the potential of executing these fundamental operations on unknown product variants. However, this is at best a challenging task due to their small size and the varying appearance of connections. Table 4.7 qualitatively lists common fasteners by their ease of detection and disassembly [48]. Due to the significantly-varying nature of the different fastener types, detection of different categories should be considered as separate tasks within the vision system.

Ease of detection was evaluated on the basis of the standardisation of the components and their appearance, as well as their general visibility. Snap fasteners, welds and glued joints are non-standard and commonly hidden from view. Automatic detection of these appears infeasible for

current disassembly automa- tion. In this case, one can only resort to using databases of known products, case- by-case teaching or bulk destructive disassembly. On the other hands, screws, bolts and rivets are of standard geometry, and are easy for a human to recognise. Screws and bolts may be lost or exchanged during product maintenance.

Table 4.7 List of common fasteners by ease of detection and disassembly with their best practice

	Disassembly method		
Detectability	Non-destructive	Semi-destructive	Destructive
High	Screws, bolts, etc.	Rivet	–
Medium	Spring, staple, cotter pin	Solder, cable, bundler, tape	Seam
Low	–	–	Snap fit, weld, glue/ seal

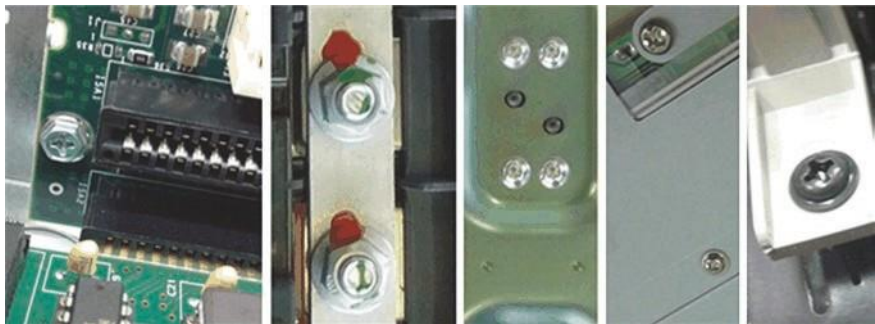


Fig. 4.17 Various screw-type fasteners found in end-of-life products

Cables, bundlers and tape are generally visually salient; however pose a challenge due to their lack of fixed shape. Springs, pins and staples have a more defined shape; however tend to be less distinguishable from other components. Due to their common occurrence within electronic products, the detection of screws and cables will be examined in more detail.

Screws, Bolts, and Nuts

Screws, bolts and nuts are defined by their shape, being typically circular with a particular standard pattern that forms surface contact with a tool (see Fig. 4.17). The main challenges posed by the detection of screws and bolts are their small size and often shiny surface. Due to their small size, either a high resolution camera or a movable camera (e.g. mounted on a robot) is required to obtain the raw images. The detail of the screw head is generally undetectable using currently-feasible active depth sensors. The shiny surface means that greater variation in appearance is expected from the reflection of the lighting, background and other components, as well as surface variations. As such, a method is required that is either edge-based and robust to extraneous salient edges (caused by varying light reflections caused by altered location/orientation or screw head shape); or very fast, enabling comparison to a large database of known examples. Unlike product and component detection, screw detection must be insensitive to changes in colour; such fasteners are made from a range of materials and colours, and may change colour due to rust.

Encouraging results for nut and screw detection were reported in [5], who reported detection a rate of 98 % for screw detection. Rolando Cruz-Ramírez et al.

[30] also suggests that high detection rates may be attainable after the integration of multiple images taken from different perspectives. Both approaches demonstrated a strategic use of contextual information, first finding the containing element (the car wheel [5] or the ceiling beam [30]), then using this information to selectively take higher resolution images for the detection of the small fasteners. Figure 4.18 shows the detection of bolts in [5], whereby both the shape of the bolt and the symmetrical geometric arrangement is used, hence ruling out the multitude of other candidate locations that fit the shape criteria alone. The accuracy in [30]

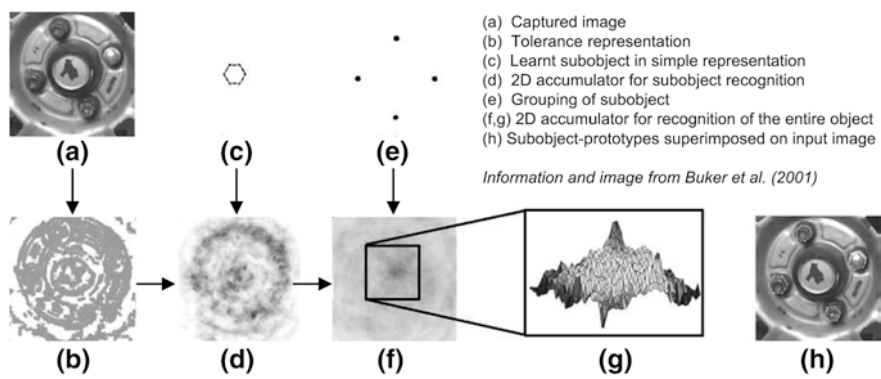


Fig. 4.18 Bolt detection, presented by [5]. **a** Captured image. **b** Tolerance representation. **c** Learnt subobject in simple representation. **d** 2D accumulator for subobject recognition. **e** Grouping of subobject. **f** 2D accumulator for recognition of the entire object. **g** 2D accumulator for recognition of the entire object. **h** Subobject-prototypes superimposed on input image

is reported for “temporal multi-image integration”: 30 images are taken per metre along the ceiling beam, and the data from multiple images fused to produce the final result.

Cables

Cables are generally salient within an image due to their contrasting colour, how- ever are flexible and only fixed in specific locations, being otherwise free to move. Cables are therefore expected to change shape during use and disassembly; such changes should neither affect their detection nor the detection of the product or other components. Thinner cables are often undetectable or difficult to detect using currently-feasible active depth sensors. In general, cables are characteris- tic in having (a) a constant colour, (b) a constant, limited width and (c) a length often significantly longer than the width. However, exceptions to these rules (when viewed from one perspective) include ribbon cables and twisted pairs. As cables are commonly found in a large range of colours, a method insensitive to changes in colour is desirable. The current literature presents few cases showing conclu- sive results for cable detection. Gil et al. [25] employs a Hough Transform-based method that only detects straight sections of wires (see Fig. 4.19). This is suffi- cient for suggesting some locations for cutting, however detects each wire as many small sections and is incapable

of detecting wires that are entirely curved. Hohm et al. [29] suggests scale-space filtering for the detection of cables.

Apart from their flexibility (and hence often-irregular shape), a major challenge in cable detection is that many non-cable objects—e.g. painted lines, light-reflecting edges—share the same distinguishing features of cables. Figure 4.20 shows the results of a detector targeting thin areas located between edges. This is more suitable for the detection of curved sections of wires, however displays an unacceptable rate of false positives due to the aforementioned problems, which are also unsolved by the aforementioned approaches from other literature.

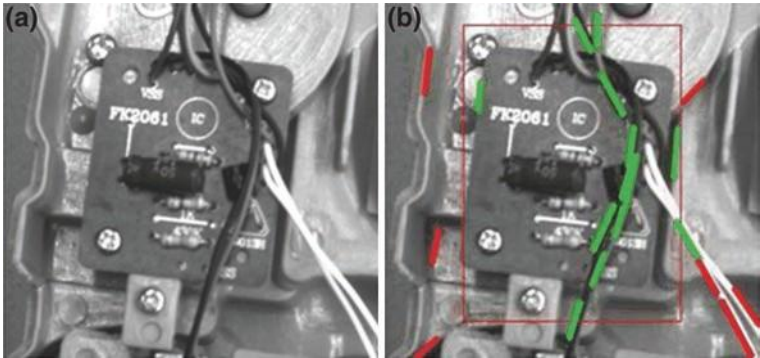


Fig. 4.19 Cable detection with Hough Transform based method [25]. **a** Captured image. **b** Processed image

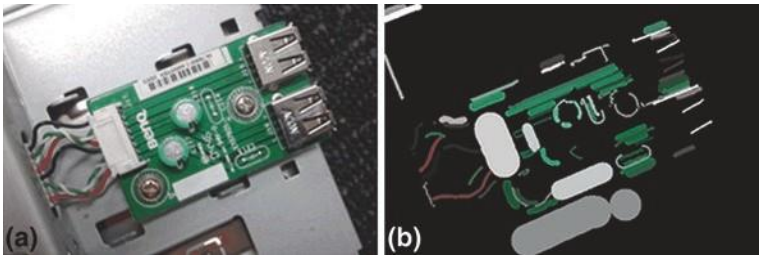


Fig. 4.20 Cable detection targeting thin consistent regions between edges. **a** Captured image. **b** Processed image

Further measures are needed to attain the required robustness for application in industry. Many of the above false positives can be rejected by determining that they lie on a flat surface or form a regular shape. A potential solution also lies in exploiting the knowledge that cables are flexible, by actively testing the flexibility of candidates. Ultimately, the existence or location of wires should be learned with respect to their attachment points, which can also be determined using an active approach.

4.4.4 State Change Detection for Execution Monitoring

A state of disassembly is defined according to the existence of particular main components on the disassembly rig. A state transition occurs when an entire component or significant portion thereof has been detached and significantly moved from its original location. Disassembly state change is a key measure of the success or failure of the current disassembly operation, and is expected to occur when sufficient disassembly operations have been performed. The

detection of state change is used for the cognitive function of execution monitoring (further explained in Chap. 5). This book proposes two possible candidate approaches for measuring state change: (a) the *absolute approach* and (b) the *relative approach*.

For the *absolute approach*, the detection of a particular component is repeatedly performed to recheck the properties of the component including its location and existence. State change is indicated if an entirely new set of properties have been detected. This method allows greater flexibility in the case of a more complicated product structure. However, logical ambiguity can arise in case that some parts of a component remain after a significant portion is removed. Remnants may be incorrectly recognised as a new component leading to misperception of the product structure.

For the *relative approach*, the measurement is performed relative to originally-detected properties. The component is detected once at the beginning. Afterwards, incremental change is measured. This approach resolves the ambiguity problem found in the absolute approach. The relative approach is therefore recommended for (semi-) destructive disassembly, where parts of a component may still remain after sufficient operations have been carried out. Only this approach is presented in this book, according to the LCD screen case study.

Detection Algorithm for the Relative Approach

Using the relative approach, state change is detected when a significant amount of change is observed between the current condition and the original condition detected at the beginning of the previous state. Depth information is intuitively more decisive since it represents the physical geometry of the component and is less affected by external factors. However, due to (a) the inaccuracy of the depth sensor, (b) the comparatively insignificant height of some components, and (c) the occurrence of blind areas in the depth data, the colour image is also taken into account to compensate for these limitations. Once a new state is detected, this state is flagged and the new properties stored as a benchmark. Subsequent conditions are compared to this benchmark until the next state change. In the condition definitions, the flagged benchmark

will be referred to using the subscript *flag* and the images associated with the current condition with *check*. To disregard the irrelevant surroundings, the difference is only measured within a local *region of interest* (ROI) enclosing the main component.

Equation (4.23) expresses the condition used to determine state change. State change is detected when the measured difference from either the depth or colour images exceeds the thresholds Φ_{depth} and Φ_{colour} respectively. The difference measures are explained as follows.

$$stateChange \equiv diff_{depth} \geq \Diamond_{depth} \vee (diff_{colour} \geq \Diamond_{colour}) \tag{4.23}$$

For the *depth image*, one indicator for disassembly state change is the appearance of regions in the second image with height (*z*) value lower than the corresponding regions on the flagged image [see condition ϕ_1 , Eq. (4.25)]. These regions represent the volume of the component that has been removed. The depth sensing

technique is also prone to the occurrence of blind areas due to reflective surfaces (see Active depth sensor in Sect. 4.2.1). Although the depth itself is not sensed in these areas, change in blind area is taken as an additional indicator for state change [see conditions ϕ_2 and ϕ_3 , Eqs. (4.26) and (4.27)]. A significant change in blind area implies a change in surface to one with a contrasting reflectivity property. The total measured difference with respect to the depth image ($Diff_{depth}$) is the ratio between the number of pixels satisfying at least one of the aforementioned change conditions and the number of pixels within the local ROI [see Eq. (4.24)].

$$Diff_{depth} = \frac{I[\phi_1 \vee \phi_2 \vee \phi_3]}{S_I} \quad (4.24)$$

$$\phi_1 \equiv z_{i,flag} > z_{i,check} \quad (4.25)$$

$$\phi_2 \equiv z_{i,flag} \notin \phi_{blind} \wedge z_{i,check} \in \phi_{blind} \quad (4.26)$$

$$\phi_3 \equiv z_{i,flag} \in \phi_{blind} \wedge z_{i,check} \notin \phi_{blind} \quad (4.27)$$

where ϕ_i = condition i ; I = pixel of the specific ROI; S_I = size of ROI; z_i = height (in direction z_F); and ϕ_{blind} = blind area on the surface of component.

The state difference with respect to the *colour image* ($Diff_{colour}$) is measured using a colour-based histogram comparison in the HSV colour space. The histogram (H_k) is constructed in two channels, H and S, in order to reduce the effects of illumination. $Diff_{colour}$ is obtained by Eqs. (4.28) and (4.29), which are derived from the correlation equation for measuring histogram similarity in [49].

$$Diff_{colour} = 1 - \frac{\sum_i H_{flag}(I) - \bar{H}_{flag} \sum_i H_{check}(I) - \bar{H}_{check} \sum_i H_{flag}(I) + \bar{H}_{flag} \bar{H}_{check} \sum_i 1}{\sqrt{\sum_i H_{flag}(I) - \bar{H}_{flag} \sum_i 1} \sqrt{\sum_i H_{check}(I) - \bar{H}_{check} \sum_i 1}} \quad (4.28)$$

$$- \sum_i H_{flag}(I) - \bar{H}_{flag} \sum_i 1 \sum_i H_{check}(I) - \bar{H}_{check} \sum_i 1$$

$$\frac{1}{H}$$

$$k = \frac{1}{N} \sum_{l=1}^N H_k(l) \tag{4.29}$$

where N = number of histogram bins; l pixel of the local ROI; H_{flag} = histogram of the original condition; and H_{check} = histogram of the current condition.

For the case-study, preliminary tests were conducted by non-destructively removing main components from LCD screen samples. With regards to Eq. (4.23), thresholds of Φ_{depth} 50 % and Φ_{colour} 75 % are sufficient to correctly distinguish state change in 95 % of the samples. An example of state change is shown in Figs. 4.21 and 4.22. However, these criteria are subject to slight changes according to the geometry of the components.

Results and Conclusions

The depth criteria are sufficient to identify state change in the majority of the samples. However, due to the limitations of the depth sensor, these criteria fail in the

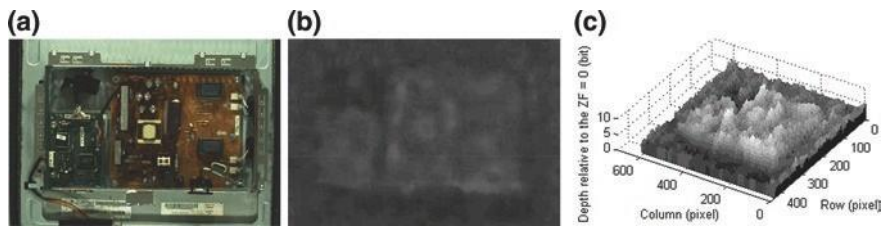


Fig. 4.21 State change—original condition (flagged images). **a** Colour image. **b** Depth image. **c** 2.5D depth map

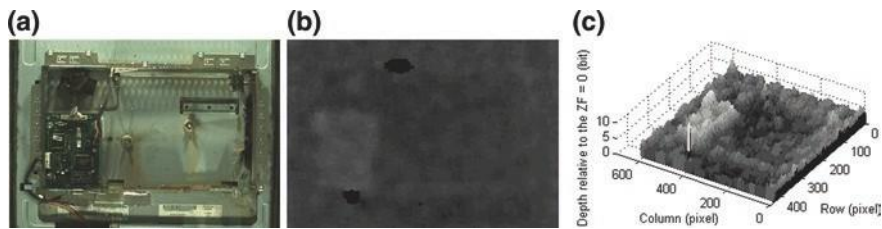


Fig. 4.22 State change—the component is removed. **a** Colour image. **b** Depth image. **c** 2.5D depth map

following two circumstances: (a) when removed components have insignificant height, in comparison to sensor resolution or noise; and (b) when a product contains multiple reflective components, with significant portions lying under blind areas. Under these circumstances, an insufficient number of pixels may be counted resulting in false negatives. The colour criterion is robust for state change detection if there is an adequate colour difference between each pair of component types. In the case study, this weaker criterion was found capable of resolving the issues encountered by the depth criteria alone. However, inaccurate assessment was found to occasionally occur during destructive disassembly, as the sensed colour is affected by the presence of dust and fumes. In conclusion, both colour and depth criteria were found to be useful in state change detection, both displaying technical limitations which can be overcome by the other.

4.4.5 Extensibility

Only the requirements with respect to the process of disassembly have been discussed thus far. Due to the rapidly-evolving product market, a disassembly system (and corresponding vision system) should not be static in time, but rather, easily extensible to meet future requirements. Software development should occur with the intention of reducing the effort required to implement new functionalities. Modular programming is useful in this context. Additionally, the ability to extend

Table 4.8 Requirements of a GUI with extensibility in mind

Information	Functions	Other requirements
• Raw sensory data • Results from detection • Intended action	• Add a positive example • Flag an error/classify a negative example • Manual control (Advanced: associate a new action to a sensed or detected feature)	• Simple to understand • Simple to operate simple to add new • Sensors/detectors/actions

functionality should preferably remain as open as possible: this is the underlying motivation for a learning-by-demonstration system, and can be applied to the vision system as well. No vision system is guaranteed to perform perfectly, and much time can be saved if *anyone* can make the necessary adjustments, without expert knowledge.

When a database is used for template matching or in training a machine learner, the ability to add new examples to the database, flag errors or add corrections can easily be granted to the operator. For this, the operator also requires access to the information the robot perceives. A *graphical user interface* (GUI) is required, to display the information sensed from the environment and the robot’s intentions, as well as allowing user input. Intended actions can be communicated by superimposing action locations onto sensed visual information. This is additionally useful for safety and debugging. Some potential requirements of such a GUI are summarised in Table 4.8.

4.5 Conclusions

The primary advantage of a vision system, with respect to disassembly automation, is in allowing the disassembly system to be robust to the large number of uncertainties present during disassembly. These uncertainties include lot size and product model, as well as potential damage or modification during the usage phase. A vision system allows the

system to potentially detect products and variants, components and fasteners, as well as verify the effect of its own actions. This allows the disassembly system to automatically react to sample deviations, unsuccessful actions and potentially unseen models.

However, the implementation of a vision system is a non-trivial task. An overview of sensor technologies was presented, followed by a literature review of common techniques previously used in the field. Those reporting positive results used a clever combination of techniques which take advantage both of the features and requirements of the specific task, as well as relational information with regard to other components or features. Finally, the last section presented an overview of the features and requirements of the primary detection tasks, as well as the requirements for extensibility of such a disassembly vision system.

References

1. Weigl-Seitz A, Hohm K, Seitz M, Tolle H (2006) On strategies and solutions for automated disassembly. *Int J Adv Manuf Technol* 30(5–6):561–573
2. Scholz-Reiter B, Scharke H, Hucht A (1999) Flexible robot-based disassembly cell. *Robot Comput Integr Manuf* 15(3):247–255
3. Tonko M, Gengenbach V, Nagel H-H, Schäfer K, Picard S, Horaud R, Mohr R (2009) Towards the integration of object recognition and visual servoing for disassembly of used cars. In: IAR Jahresbericht/Rapport Annuel 1997 (ISSN 0947-0727), Deutsch-Französisches Institut für Automation und Robotik/Institut Franco-Allemand pour les Applications de la Recherche (IAR), IAR Annual Meeting '97, 20–21 November 1997, Gesamthochschule Duisburg/Germany, 1998, pp 79–84
4. ElSayed A, Kongar E, Gupta SM, Sobh T (2012) A robotic-driven disassembly sequence generator for end-of-life electronic products. *J Intell Robot Syst* 68(1):43–52
5. Büker U, Drüe S, Götz N, Hartmann G, Kalkreuter B, Stemmer R, Trapp R (2001) Vision-based control of an autonomous disassembly station. *Robot Auton Syst* 35(3):179–189
6. Logitech (2013) Logitech HD Webcam C270. <http://www.logitech.com/en-us/product/hd-webcam-c270>
7. Fontes YC, Brandão D (2010) Application of stereoscopic vision for disassembly line of electronics devices. In: Proceedings of the 9th IEEE/IAS international conference on industry applications
8. Serranti S, Gargiulo A, Bonifazi G (2011) Characterization of post-consumer polyolefin wastes by hyperspectral imaging. *Waste Manage* 31(11):2217–2227
9. Freitag H, Huth-Fehre T, Cammann K (2000) Rapid identification of plastics from electronic devices with NIR-spectroscopy. *Anal Lett* 33(7):1425–1431
10. Imec (2013) Imec first to introduce hyperpectral CMOS camera for snapshot and video. http://www2.imec.be/be_en/press/imec-news/imechyperspectralcamera2013.html
11. PrimeSense (2013) 3D Sensors—PrimeSense. <http://www.primesense.com/solutions/3d-sensor/>
12. Corporation M (2014) Xbox 360—Kinect. www.xbox.com/Kinect
13. Khoshelham K, Elberink SO (2012) Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors* 12(2):1437–1454
14. Inc. AC (2014) Xtion PRO LIVE. http://www.asus.com/au/Multimedia/Xtion_PRO_LIVE/
15. Stoyanov T, Mojtahedzadeh R, Andreasson H, Lilienthal AJ (2013) Comparative evaluation of range sensor accuracy for indoor mobile robotics and automated logistics applications. *Robot Auton Syst* 61(10):1094–1105
16. Foix S, Alenyà G, Torras C (2011) Lock-in time-of-flight (ToF) cameras: a survey. *IEEE SENS J* 11(9):1917–1926
17. RobotShop (2013) MESA imaging TOF camera SR4500 44 35° FOV
18. Viggiano JAS (2004) Comparison of the accuracy of different white-balancing options as quantified by their color constancy. In: Proceedings of the SPIE—the international society for optical engineering
19. Bradski G, Kaebler A (2008) Learning OpenCV—computer vision with the OpenCV library. Affine transform, 1st edn. O'Reilly Media Inc., California
20. OpenKinect (2011) Imaging information. http://openkinect.org/wiki/Imaging_Information
21. Siciliano B, Sciacivico L, Villani L, Oriolo G (2009) Vision Snsors. Robotics: modelling, planning and control. Springer, London, pp 230–255
22. Vezhnevets V, Sazonov V, Andreeva A (2003) A survey on pixel-based skin color detection techniques. In: Proceedings of the GraphiCon

23. Vongbunyong S (2013) Applications of cognitive robotics in disassembly of products. Ph.D. Dissertation, The University of New South Wales, Sydney, Australia
24. Kovac J, Peer P, Solina F (2003) Human skin colour clustering for face detection. In: Proceedings of the international conference on computer as a tool. The IEEE Region 8
25. Gil P, Pomares J, Puente T, Diaz C, Candelas F, Torres F (2007) Flexible multi-sensorial system for automatic disassembly using cooperative robots. *Int J Comput Integr Manuf* 20(8):757–772
26. Canny J (1986) A computational approach to edge detection. *IEEE Trans Pattern Anal Mach Intell* 8(6):679–698. doi:[10.1109/TPAMI.1986.4767851](https://doi.org/10.1109/TPAMI.1986.4767851)

27. Weigl A, Hohm K, Seitz M (1995) Processing sensor images for grasping disassembly objects with a parallel-jaw gripper. In: Proceedings of the TELEMAT telerobotics research conference & ERNET Workshop, Noordwijkerhout, The Netherlands
28. Witkin AP (1983) Scale-space filtering: a new approach to multi-scale description. In: Proceedings of the 8th international joint conference on artificial intelligence
29. Hohm K, Müller Hofstede H, Tolle H (2000) Robot assisted disassembly of electronic devices. In: Proceedings of the 2000 IEEE/RSJ international conference on intelligent robots and systems
30. Rolando Cruz-Ramírez S, Mae Y, Arai T, Takubo T, Ohara K (2011) Vision-based hierarchical recognition for dismantling robot applied to interior renewal of buildings. *Comput-Aided Civil and Infrastruct Eng* 26(5):336–355
31. Fraundorfer F, Scaramuzza D (2012) Visual odometry Part II: matching, robustness, optimization, and applications. *Robotics and automation magazine, IEEE*
32. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *Int J Comput Vision* 60(2):91–110. doi:[10.1023/B:VISI.0000029664.99615.94](https://doi.org/10.1023/B:VISI.0000029664.99615.94)
33. Bay H, Tuytelaars T, Gool L (2006) SURF: speeded up robust features. In: Leonardis A, Bischof H, Pinz A (eds) *Computer vision—ECCV 2006*, vol 3951. *Lecture Notes in Computer Science*. Springer, Heidelberg, pp 404–417. doi:[10.1007/11744023_32](https://doi.org/10.1007/11744023_32)
34. Agrawal M, Konolige K, Blas M (2008) CenSurE: center surround extremas for realtime feature detection and matching. In: Forsyth D, Torr P, Zisserman A (eds) *Computer Vision—ECCV 2008*, vol 5305. *Lecture Notes in Computer Science*. Springer, Heidelberg, pp 102–115. doi:[10.1007/978-3-540-88693-8_8](https://doi.org/10.1007/978-3-540-88693-8_8)
35. Leutenegger S, Chli M, Siegwart (2011) RYBRISK: binary robust invariant scalable key-points. In: Proceedings of the IEEE international conference on computer vision (ICCV), 6–13 Nov 2011, pp 2548–2555. doi:[10.1109/ICCV.2011.6126542](https://doi.org/10.1109/ICCV.2011.6126542)
36. Harris CG, Pike JM (1988) 3D positional integration from image sequences. *Image Vis Comput* 6(2):87–90. doi:[10.1016/0262-8856\(88\)90003-0](https://doi.org/10.1016/0262-8856(88)90003-0)
37. Rosten E, Drummond T (2006) Machine learning for high-speed corner detection. In: Leonardis A, Bischof H, Pinz A (eds) *Computer vision—ECCV 2006*, vol 3951. *Lecture Notes in Computer Science*. Springer, Heidelberg, pp 430–443. doi:[10.1007/11744023_34](https://doi.org/10.1007/11744023_34)
38. Calonder M, Lepetit V, Strecha C, Fua P (2010) BRIEF: binary robust independent elementary features. In: Daniilidis K, Maragos P, Paragios N (eds) *Computer vision—ECCV 2010*, vol 6314. *Lecture Notes in Computer Science*. Springer, Heidelberg, pp 778–792. doi:[10.1007/978-3-642-15561-1_56](https://doi.org/10.1007/978-3-642-15561-1_56)
39. Alahi A, et al. (2012) FREAK: fast retina keypoint. *IEEE conference on computer vision and pattern recognition (CPVR)*, pp 510–517
40. Galleguillos C, Belongie S (2010) Context based object categorization: a critical survey. *Comput Vis Image Underst* 114(6):712–722
41. Jørgensen TM, Weimar Andersen A, Sloth Christensen S (1996) Neural net based image processing for disassembling TV-sets. In: *Solving engineering problems with neural networks*
42. Karlsson B, Järrhed J-O (2000) Recycling of electrical motors by automatic disassembly. *Measur Sci Technol* 11(4):350
43. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the conference on computer vision and pattern recognition*
44. Kopacek P, Kopacek B (2006) Intelligent, flexible disassembly. *Int J Adv Manuf Technol* 30(5–6):554–560
45. Miller Technology NEMA specifications. http://www.millertech.com/Technical_Specs.htm. Accessed 18 Nov 2013
46. Vezhnevets V, Sazonov V, Andreeva A (2003) A survey on pixel-based skin color detection techniques. In: *Proceedings of the GraphiCon 2003*. pp 85–92

47. Chang F, Chen C-J, Lu C-J (2004) A linear-time component-labeling algorithm using contour tracing technique. *Comput Vis Image Underst* 93(2):206–220
48. Lambert AJD, Gupta SM (2005) *Disassembly modeling for assembly, maintenance, reuse and recycling*. CRC Press, Boca Raton
49. OpenCV (2014) Histograms. In: *The OpenCV reference manual—release 2.4.9.0*. pp 295–306

Chapter 5

Cognitive Robotics

Abstract Uncertainties and variations in returned End-of-Life (EOL) products result in complexity at the planning and the operation levels of automated disassembly. These have become critical obstacles in disassembly automation, which lacks the flexibility and robustness of manual disassembly. In this chapter, the principle of cognitive robotics is implemented in disassembly automation, to overcome these problems by emulating the behaviour of human operators. The methodology, framework, and cognitive functionalities with regard to the disassembly domain are explained in this chapter.

5.1 Autonomous Robot and Cognitive Robotics

Autonomous robots are intelligent robots with certain degrees of autonomy that performs tasks by itself with minimal or without human guidance. The tasks are carried out by the Intelligent Agent (IA) which makes decisions according to the information of the dynamic environment sensed. *Cognitive robotics* describes the high-level cognitive functions¹ [1] which allow a robot to reason, revise and perceive change in

unpredictable environments. This allows a robot to respond to its environment and complete goals in a robust and adaptive way [1]. Müller [2] illustrated characteristics of classical Artificial Intelligence (AI) and cognitive system in a complexity space which presents a relation between complexity of the tasks and complexity in environment (see Fig. 3.2: Example of serial robots). In this Figure, vertical axis is “flexibility” representing the ability of the agent to deal with complex environment; whereas, the horizontal axis is “specific task success” representing the ability to deal with complex tasks. The classical AI performs complex tasks effectively in non-complex environments, whereas the cognitive system exhibits the opposite characteristics. The cognitive functions allow the

¹ *Cognitive functions* (n.) An intellectual process by which one becomes aware of, perceives, or comprehends ideas. It involves all aspects of perception, thinking, reasoning, and remembering.

cognitive system to effectively perform simple tasks in a complex environment. Cognitive robotics, used in conjunction with classical AI, therefore allows a system to become more flexible and reliable in dynamic environments [2] (Fig. 5.1).

Cognitive robotics takes the form of various approaches. One practical approach is to use *knowledge representation and reasoning* (KRR) to solve problems in situations where the robot has incomplete knowledge [3]. The knowledge of the environment is represented in a format in which the robot can reason and control its behaviour. The robot interacts with its environment using sensors and effectors. Behaviours are expressed using a high-level programming language based on logic programming. The programs generate action sequences corresponding to conditions: initial states, prerequisites and effects of primitive actions, exogenous events, and the result of sensing. As a result, the robot can interact robustly and autonomously with the external world.

Another important feature distinguishing cognitive robotics from classical AI is the method of interaction with humans: interaction, rather than control, in order to increase the level of intelligence and flexibility of the system [2]. This architecture is based on a *closed perception-action loop* as shown in Fig. 5.2 [4]. The behaviour of the robot is organised by three main elements: (a) *learning and reasoning*,

Fig. 5.1 Complexity space for intelligent agent [2]

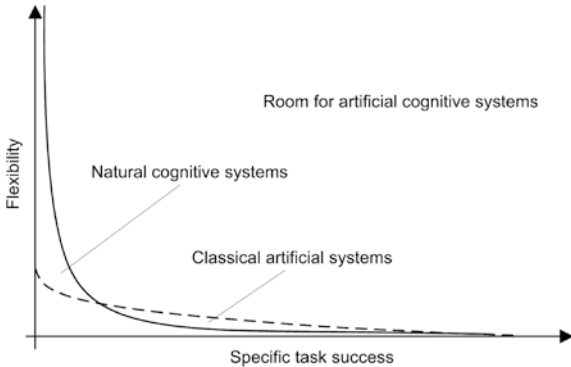
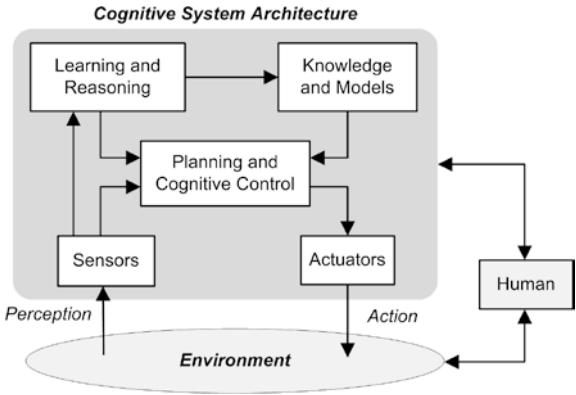


Fig. 5.2 Cognitive system architecture with closed perception-action loop.
Adapted from [4]



(b) *planning and cognitive control*, and (c) the *knowledge model*. Interaction with the external world occurs via the sensors and actuators, which are associated with perception and action respectively. The human is considered part of the external world and can interact with both the environment and the robot. This architecture was originally implemented as part of “*Cognitive Factory*” in which human operators work side-by-side with automation that is flexible, reliable, and safe.

Cognitive robotics has recently been presented in much research in many fields. The Cognitive Factory is one of the state-of-the-art applications of cognitive robotics in industry. This project is developed by CoTeSys (Cognition for Technical Systems), a company that applies cognitive robotics functions to industrial activities, especially production [5]. The project integrates the key activities of production: (a) system monitoring and planning; (b) material and parts condition; (c) work piece assembly; and (d) human-machine co-operation. Four main functions of cognitive robotics are implemented: *perception*, *learning*, *knowledge*, and *planning*. These functions allow the robots to monitor the product and carry out the assembly process with respect to prior knowledge of the assembly plan. A learning module makes the system self-optimize and derives a better assembly sequence from past incidents. Consequently, an optimal sequence plan can be autonomously found by the planning module. With respect to skill acquisition, self-adaptation, and self-modelling, the cognitive factory is flexible enough to operate with a variety of products. In comparison to classical manufacturing, the Cognitive Factory is hence capable of achieving higher productivity and flexibility in the production line, as illustrated in Fig. 5.3 [6].

In the field of disassembly, much research regarding automatic disassembly cells has been conducted using classical AI. However, the flexibility of the existing

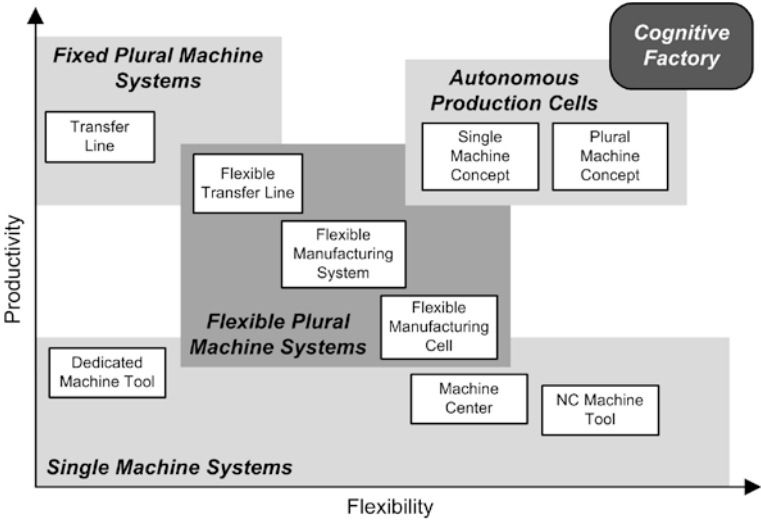


Fig. 5.3 Classification of various types of manufacturing with respect to productivity and flexibility [6]

system is very limited in dealing with variations in the EOL product in the planning and operational levels. Due to the aforementioned advantages of cognitive robotics, it appears beneficial to employ these concepts in handling the uncertainties in disassembly domain. The disassembly process can be considered a dynamic external world with which the cognitive robotics system interacts. This chapter presents the application of cognitive robotics in product disassembly.

5.2 Concept Overview

In comparison to automated systems, human operators are far more capable of dealing with uncertainties in the EOL products returned. Therefore, the principle of cognitive robotics has been used to emulate the behaviour expressed by a human operator during the disassembly process. These behaviours are expected to address the uncertainties in product and process that have thus far hindered the industrial application of robotic disassembly. Common behaviours of the human operator and their implementation using cognitive robotics principles are explained in this section.

5.2.1 Human Driven Disassembly

Human operators are expected to intuitively overcome the aforementioned uncertainties in disassembly. Decisions are taken based on prior expertise, by adapting past experiences to the perception of the currently-encountered situation. The process performed by human operators is *flexible* and *robust*. Firstly, the intuitively-performed disassembly process is expected to be flexible enough to handle any product model without prior knowledge of specific details, e.g. the

structure of the product and the number of components. This information can be obtained during the process itself. Secondly, with respect to robustness, manual disassembly has a high success rate because human operators are able to evaluate the success of each step. Hence, failures in specific strategies lead to further attempts using alternative methods until the task has been accomplished. These two characteristics allow human operators to disassemble a larger range of products with fewer restrictions.

A variety of product models are expected to be met with in the actual disassembly process. The efficiency of the process varies according to the prior knowledge of the operators on the models encountered. If the operator knows the relevant disassembly information in advance (*known model*) then the operation can generally be completed efficiently. The process can be carried out quickly with few fruitless attempts, since the steps, previously shown capable of disassembling this model, are already known. Minor physical uncertainties are expected but can generally be compensated intuitively.

On the other hand, difficulties may arise in the case that the model is new to the operator (*unknown model*). Since the specific information is unavailable, the operator attempts previously-known strategies to discover a suitable method for the new model. This process may be carried out awkwardly, state by state using trial and error, for the first few attempts to disassemble each new model. In each disassembly state, the operators may spend more time to locate or identify main components. A number of unsuccessful attempts may be made before the components are successfully removed and the operator can proceed to the next state.

Possible operations can be logically selected based on the operator's perception and past experience. Concurrently, during the process, the operator gathers information: product structure, component-related details, disassembly operations and relevant parameters. The operator learns an appropriate process for the new model by considering the relation between actions and their consequences: success or failure. The knowledge base (KB) is continuously built up and recalled when needed. This process is repeated in all disassembly states and finished when the goal state has been reached. After a certain number of samples have been successfully disassembled, the operator should have enough experience to effectively disassemble this particular model when it is again encountered. This human behaviour in disassembly process is illustrated in Fig. 5.4.

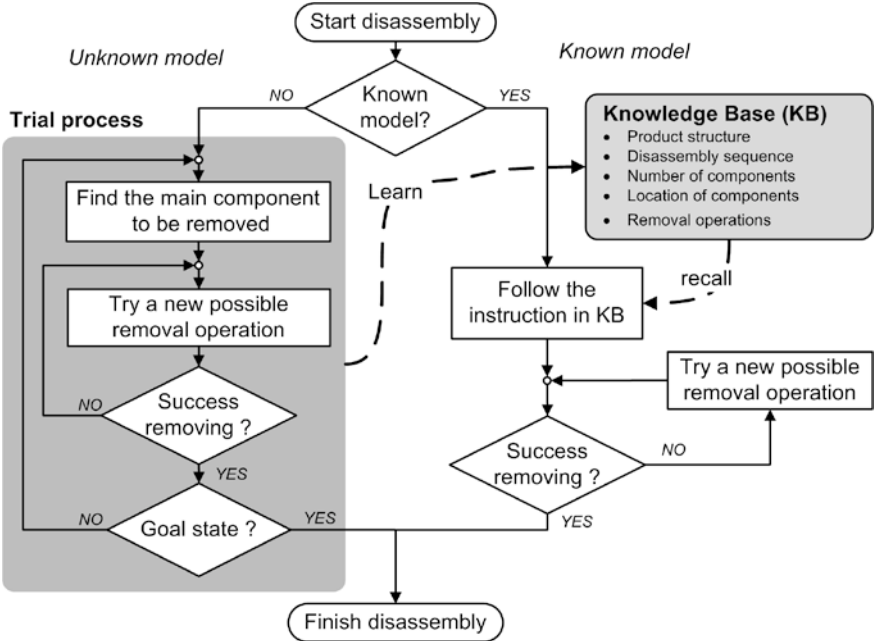


Fig. 5.4 Behaviour of human operators in the disassembly process

According to this behaviour, the characteristics of the human operator that influence the flexibility and the robustness of disassembly are summarised as follows:

- Ability to perceive product structure during the disassembly process;
- Ability to assess the outcome of each performed operation and perform alternative operations if the first attempt fails;
- Flexible operation schemes that can be adapted to other physically similar components; and
- Ability to learn from past experience and adapt to previously unseen cases.

These characteristics are emulated by using the principle of cognitive robotics which is explained in the following sections.

5.2.2 Agent Emulating Human Behaviour

The aforementioned human-like behaviour needs to be adapted to be suitable with the *closed perception action loop architecture* and capabilities of the automation. In considering the limitations of automation, assistance is introduced to the behaviour control in order to help the robot to handle unresolved situations. The cognitive robotic agent (CRA) first determines whether or not the model of each sample is already known in the existing knowledge base (KB). If the model is being seen for the first time (unknown), the agent² goes through the trial process in which actions are executed based on general operation plans according to the perceived components in the current state. The removal process is carried out with a number of attempts using different operation strategies and process parameters. When the alternatives available to the automation have been exhausted, assistance is requested. In the end, the model-specific successful process is learned by being stored in the KB. The agent is now able to recall this knowledge from the KB if this model of product is seen again (known). In this case, the agent follows the instructions in the KB on default. In case of failure due to variations and process uncertainties, the

agent requests for additional assistance to resolve the uncertainties and achieve the goal. The agent also learns the new additions and revises the existing KB (Fig. 5.5).

Regarding the process flow, the agent drives the disassembly system through the states of disassembly by generating a sequence of actions using its knowledge of the external world and the behaviours at its disposal. These actions are physically performed by the robot. The transition from one disassembly state to another occurs when a main component has been successfully removed from the original location, and occurs after a sufficient combination of disassembly actions have been performed. This process continues from the initial state of the product until the goal state has been reached.

² In this chapter, “agent” refers to “cognitive robotic agent (CRA)”.

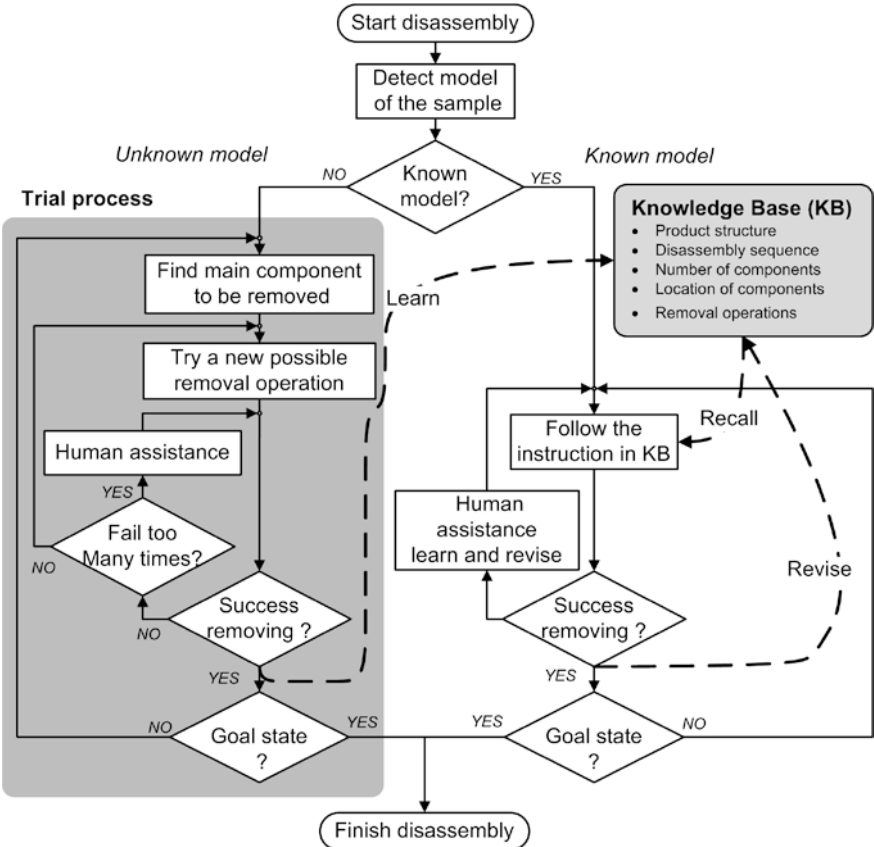


Fig. 5.5 Human behaviour emulated by cognitive robotic agent

In perspective of the information available to the agent, the physical disassembly operations have a non-deterministic effect on the world due to uncertainties in the process and product condition. Therefore, this automated process is considered an *open-world execution* since the incomplete knowledge of the world needs to be sensed on-line during the process. The interaction with the external world is described in the next section.

5.3 System Architecture and

Cognitive Functions

5.3.1 Operation Modules and Uncertainties

The uncertainties in the process are typically caused by the variations in the products to be disassembled. The types of uncertainties are expected to be effectively handled by the appropriate modules of the system. The connection between

the modules and the uncertainties to be addressed are illustrated in Fig. 5.6 and summarised in Table 5.1. The modules are explained as follows.

Cognitive robotic module (CRM)

This module is an AI agent that controls the behaviours of the system according to the cognitive functions and relevant knowledge of the product and process. The cognitive functions are expressed by the cognitive robotic agent, which interacts with the other modules and knowledge base (KB). The agent is expected to handle the variations in product structure relating to the quantity of components and their connections. These variations result in uncertainties in the disassembly sequence plan (DSP) and the disassembly process plan (DPP).

Vision system module (VSM)

Information regarding the physical world is typically obtained by the vision system. This module is designed to handle uncertainties in regard to the appearance variations of components. Its main function is to obtain information regarding the underlying components. The quantity and location of each component is observed.

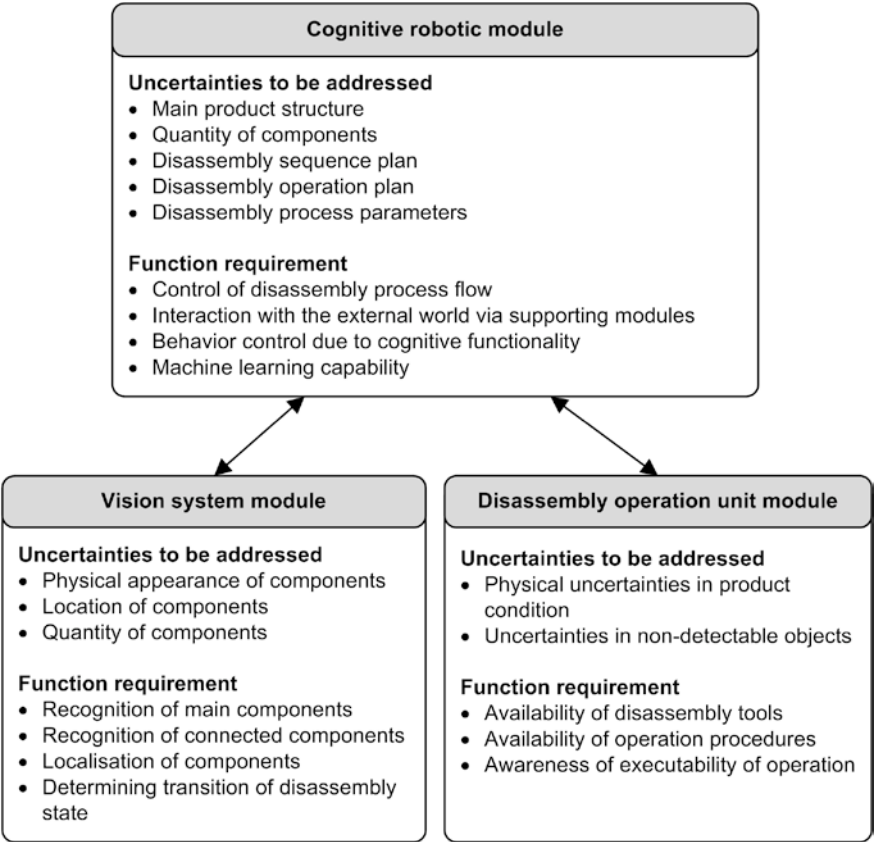


Fig. 5.6 Specification summary of the robotic disassembly system

Table 5.1 Uncertainties in disassembly process

Uncertainties	Specific issue of uncertainties	Module		
		CRM	VSM	DOM
EOL condition	Physical product conditions			•
Variety in the supplied products	Main product structure	•		
	Physical appearance in the components		•	
	Quantity of the component	•	•	
	Location of the components		•	
Complexity in process planning and operations	Disassembly sequence plan	•		
	Disassembly operation plan	•		
	Disassembly process parameters	•		
	Uncertainties in the non-detectable objects			•

In addition, visual input is used in the execution monitoring process in order to determine the success or failure of physical operations.

Disassembly operation unit module (DOM)

The operation units consist of actuators and sensors having direct contact to the product. The main functions are to detach the connections between the main components and remove these components from the remaining parts of the product. Robot arms equipped with interchangeable disassembly tools and grippers are generally used for these purposes. Force and tactile sensors are used for enhancing the system’s capability to perceive visually unobservable information important to the disassembly process. The destructive disassembly approach is less sensitive to variations in the EOL product condition and reduces the complexity in detaching connective components. However, the desired outcome of EOL treatment must be taken into account.

5.3.2 Architecture

The architecture of this system is developed based on the *closed perception action loop* architecture [4] in which the CRM interacts with the physical world via the disassembly cell and human input. The system architecture is shown in a diagram in Fig. 5.7. The CRM connects to the physical world through sensors and actuators which are the VSM and the DOM in the disassembly cell. The behaviour of the system is expressed through these modules. In case that the system cannot resolve the problems autonomously, human experts are able to assist by demonstrating the operations or revising the original belief in the product and process. This framework was earlier presented in [7].

The CRM consists of the CRA and KB, which interact internally with each other. The CRA plays an important role in controlling the behaviour of the system.

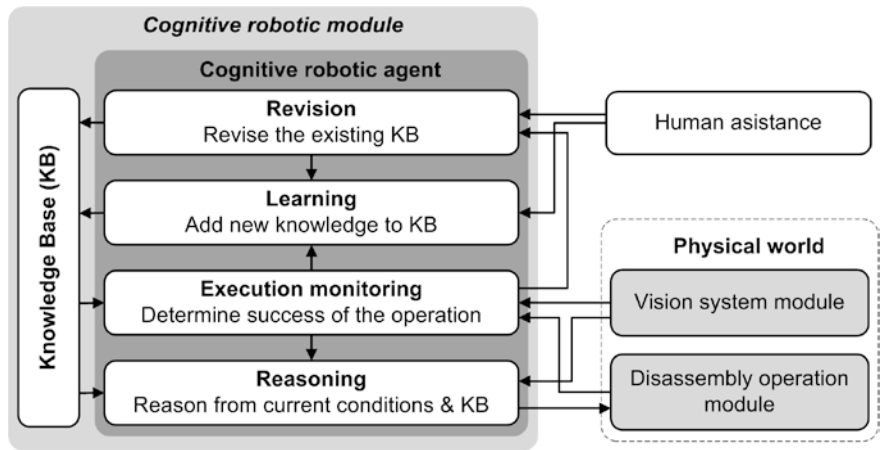


Fig. 5.7 System architecture in cognitive robotics perspective

This behaviour is expressed according to four cognitive functions: (a) *reasoning*, (b) *execution monitoring*, (c) *learning*, and (d) *revision*. These core functions allow the disassembly system to react to the dynamic external world in a flexible and robust way. The CRA typically interacts with the KB by consulting or modifying specific knowledge according to the product being disassembled. The CRA consists of generic procedures and rules, which are used for the general disassembly process. Meanwhile, the KB contains information that is more specific to the product models. The cognitive functions in CRM and the KB are explained as follows.

Reasoning

The disassembly process can be represented as *choice-points* through which the CRA drives the system until the goal is achieved. Reasoning is used to prune these choice-points according to the predefined rules, the disassembly state perceived by the sensors, operation success or failure as measured by execution monitoring, and the existing knowledge in the KB. As a result, the CRA can react logically to the external world via the DOM.

Execution monitoring

This function measures the accomplishment (success or failure) of the removal of components at the planning and operation levels according to the predefined rules. At the planning level, information regarding the change of disassembly state is supplied by the VSM. This information can be used to determine the main structure of the product as disassembly progresses. At the operation level, the outcomes returned by the actuators performing the assigned tasks further allow the reasoning function to perform parameter adjustments where necessary. Subsequently, the corresponding operations and plans are learned and revised.

Learning

This function typically deals with the *model-specific knowledge* which is information specific to a particular model of product. This knowledge is obtained during the disassembly process. Significant information contributing to a successful component removal is stored in the KB. Consequently, the CRA can use this knowledge in subsequent processes for the disassembly of this already-known model. This knowledge can be obtained from two sources, the reasoning process and demonstration by assistance.

Revision

This function is used to revise the existing knowledge in the KB. The aim is to improve the efficiency of the process of disassembling new samples of the previously seen model. Some improvement is expected to result from removing redundant operations from the previous process.

In addition, assistance may be used to correct a false original belief causing failures in component removal. The system autonomously builds its belief according to sensor perception. Since the sensors are not perfect, a certain degree of error is expected which results in imprecise belief regarding the product and operations.

In a more advanced approach, the CRA may also have the ability to modify its own program, revising its general (*non-model-specific*) behaviour according to the new knowledge received. However, it should be noted that this self-modification ability has not yet been implemented in disassembly automation.

Knowledge base

The KB contains knowledge associated with the DPP for both the general and the model-specific disassembly processes. The general knowledge is used in the case of disassembling an unknown model. This knowledge consists of constraints and parameters for using with the generic rules are stored. Conversely, the model-specific knowledge is applied when disassembling a known model. This knowledge consists of (a) *disassembly sequence plan (DSP)*, (b) *disassembly operation plans*, and (c) *process parameters*. The KB will be continuously expanded in regard to learning and revision as disassembly is carried out.

5.3.3 Language Framework and Interactions

In this book, the CRA is modelled using the cognitive robotics framework based on *Situation Calculus* [8]. *IndiGolog* is the programming language selected to model the system. The main advantage of this language is to support on-line execution. The language supports sensing and exogenous actions which allows the agent to deal with incomplete knowledge perceived from the external world [9, 10]. *IndiGolog* can be implemented on various language platforms; in this case, *Prolog* is selected [11]. *Prolog* is a logical and declarative language widely used in AI. *Prolog* also has an inference engine which is used to find the solutions from the given rules. This characteristic allows the researchers to develop

the system that is capable of dealing with complex problems with less and simpler code needed in comparison to imperative language. The program is more capable of dealing with the complex problem more. According to the characteristics of the high-level programming language IndiGolog, the CRA and the states of the world can be modelled using *behaviour specifications* and *domain specifications*.

First, the *behaviour specification* is defined as procedures for performing complex actions in order to achieve the desired goal. The control structure uses procedural language control structures (conditional statements, testing, loop statements, action sequences, and sub-procedures) together with nondeterministic constructs for planning and search (nondeterministic choices of argument, concurrency, interrupt, and searching). All of these components determine the behaviour of the agent and provide mechanisms to prune the search space of the nondeterministic choices of actions through the disassembly domain.

Second, the *domain specification* represents the history of the world as a first order term called a *situation* which is a sequence of actions executed inside and outside the CRA. The properties of the world are described by predicates called *fluents*. The change of the fluents is characterised by *successor state axioms* which describe the relation between fluents, actions, and preconditions. The value of the fluents is dynamically changed after the corresponding actions have been executed when the defined preconditions hold. The successor state axiom of fluent f can be

written in the form of $f(\bar{x}, do(a, S))$ where the free variables $\bar{x}$ will be changed by executing an action a in the situation S in which the preconditions hold. Actions

can only be executed when their preconditions hold. Actions are categorised into three types:

- Primitive actions,
- Sensing actions, and
- Exogenous actions.

The *primitive actions* are basic actions used for performing a task, in which the purpose can vary from internally modifying system variables to physical operations that affect the external world. *Sensing actions* are a type of action executed in order to obtain information, particularly from the

external world through sensors. *Exogenous actions* are actions, which are sent from the external world.

All of these components are used to systematically describe the behaviour of the system and the world. The CRM is modelled by both the domain and the behaviour specifications. The primitive actions are defined and used according to the behaviour specification. The fluents are updated according to performed actions and sensed information. Consequently, the CRM performs tasks according to the desired behaviour with respect to the sensed condition of the world.

As shown in Fig. 5.8, communication between the modules occur using abstract information in the form of actions and fluents.

Sensing actions are sent to the corresponding modules as the result of the internal process needing more information from the external world. Subsequently, the sensing result is sent back to the CRM in the form of fluents. The data structure of the fluents varies according to module and information conveyed. For example,

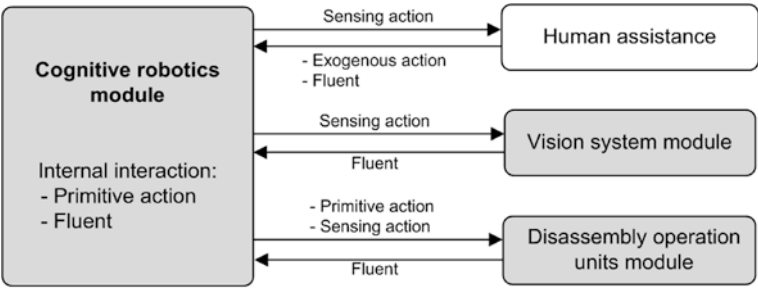


Fig. 5.8 Interaction with actions and fluents

the location of a component obtained by VSM may be expressed using $box(x_1, x_2, y_1, y_2, z_1, z_2)$. *Primitive actions* are generally sent to the DOM to request physical operations, e.g. cutting at a particular position $cut(x, y, z)$. Fluents are normally sent back to the CRM after the execution has been finished, and inform the CRM about the success or failure of the primitive actions. Thirdly, *exogenous actions* are sent to the CRM in the case of a human demonstration. A demonstration can also be in the form of a fluent.

The CRA only proceeds once the result fluent is received from the respective module, and continues in this fashion until the desired task is complete. This operation routine is shown in Fig. 5.9. In general, this control structure is sufficient to conduct complex processes such as automated disassembly. However, the control

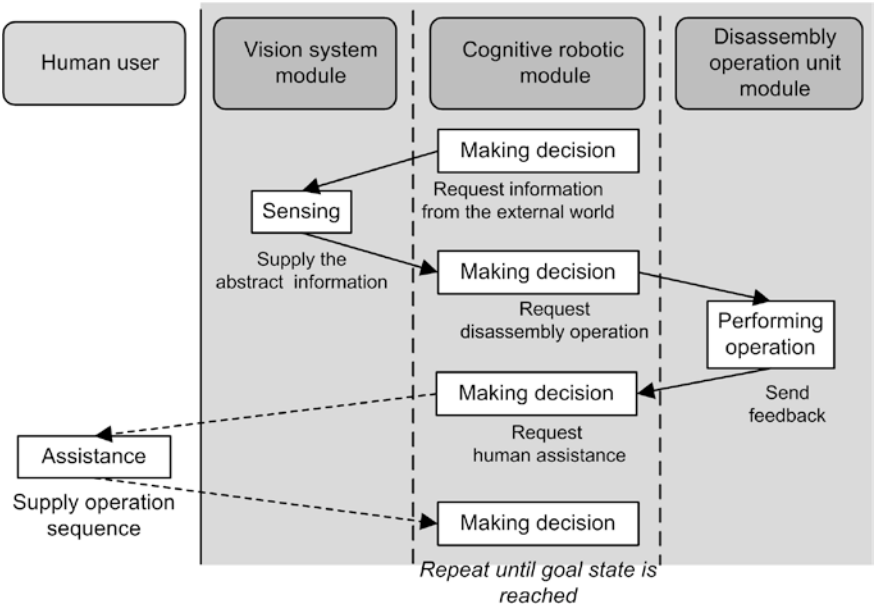


Fig. 5.9 An overview of the common operation routine

structure can be modified to be more efficient and complex by taking advantage of other features of IndiGolog, e.g. concurrency and interrupts.

In conclusion, a general framework of interactions between the modules has been presented in this section. Abstract information is transmitted in the form of actions and fluents which can be clearly defined according to situation calculus. It should be noted that this language framework is only a guideline in which other languages can be considered.

5.4 Basic Behaviour Control

The main function of the CRA is to prune the search space according to the cognitive functions of reasoning and execution monitoring. The search space describes the domain of disassembly processes consisting of variations of plans and operations. For a new product model, the information regarding the product and process is missing. Consequently, the search space has been limited to general strategies broad enough to apply to various models of the product. Specific operations, not encompassed by these general strategies, are further acquired as needed from human demonstration.

5.4.1 *Disassembly Domain*

The aforementioned uncertainties in Table 5.1 describe the variations in conditions that the CRA must handle while disassembling a new product model. These variations are represented as possible choice-points in the search-space regarding action sequences. Transitions in the disassembly state occur by executing a sufficient sequence of actions with appropriate parameters to remove a particular component. These choice-points can

be organised into two levels: (a) the *disassembly state level* and (b) *operational level*.

In the *disassembly state level*, the entire disassembly process is represented as a graph of disassembly states equivalent to a DSP (see Fig. 5.10a). The state of disassembly deals with the main structure of the product; decisions are made based on detected components. This can be considered a high level planner. The corresponding disassembly operations are conducted in the lower level within each state of disassembly. In the *operation level*, the choice-points are formulated by a hierarchy of operations and parameters relating to the type of main component (see Fig. 5.10b). The complete search space is explained as follows.

Product structure and types of component

The main product structure can be considered an arrangement of the main and the connective components. Variations in structure may be presented in different models within the same product family. The state of disassembly is defined here as the set of main components detected at a particular time.

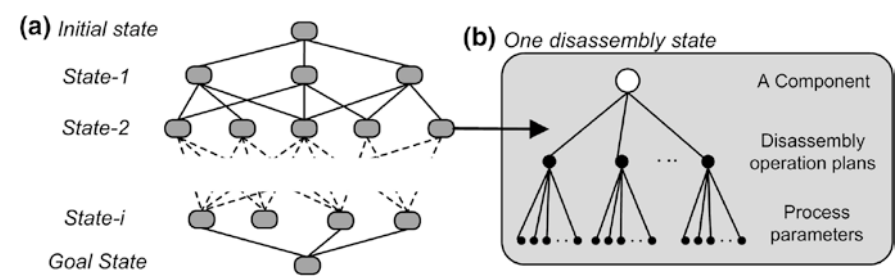


Fig. 5.10 Choice-points in disassembly domain. a Disassembly states, b operational level

The complete main product structure can be obtained by tracking the state of disassembly. The vision system observes the components belonging to each state of disassembly to obtain the following four categories of abstract information:

- Types of main component;
- Quantity of the main components;
- Types of connective components; and,
- Quantity of the connective components.

For example, the structure in the minimal form of a liaison diagram in Fig. 5.11b can be obtained from the detection results in Fig. 5.11a. This liaison diagram shows the connections between two pairs of main components. PCB1 and carrier are connected by 6 screws and PCB2-carrier connected by 5 screws. The accuracy of the reconstructed product structure depends heavily on the accuracy of the vision system in recognising the components.

Therefore, the specific product structure does not need to be known a priori. This flexibility reduces the need for a pre-defined product structure which is rarely known in advance by the EOL product treatment facility. Finally, the CRA executes suitable plans according to this visual input and the reasoning process explained in the following section.

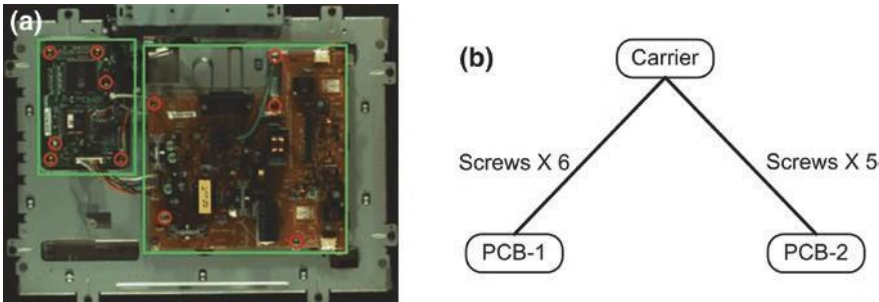


Fig. 5.11 Representation of a product structure in a disassembly state. **a** Detected components in a disassembly state, **b** minimal form of liasian diagram

Disassembly operation plans

The operation level handles the treatment of individual main components, aiming at removing them from the remainder of the product. The operation plans offer various alternative options for achieving this task. A disassembly operation plan is a procedure containing a sequence of primitive actions used to disestablish the connections attached to the main component. The underlying main component is expected to be removable after all corresponding connections have been disestablished. The main components in different models are fixed using different connective components at different locations, some of which are not visually detectable. Hence, multiple operation plans are developed for each type of main component, associated with varying levels of component damage and success rate of the removal process.

A general form of the operation plan is in Procedure (5.1) which represents one of the plan variations (plan- k) for the component c_i with a set of parameters t . During the disassembly process, the agent attempts to execute the operation plans one by one. Alternative sets of operation plans can be executed in different conditions, including type of components and success of previous operation plans. As part of trial-and-error strategy, the agent modifies the operation plan until the main component is removed (i.e. until a state change).

Proc <i>operationPlan</i> (c_i, k, t)	<i>// plan k with parameters t for component c_i</i>	
<i>primAction</i> ₁ (t_1);	<i>// execute primitive action with $t_1 \in t$</i>	
<i>primAction</i> ₂ (t_2);		
...		
<i>primAction</i> _{n} (t_n);		
endProc		(5.1)

Process parameters

The variation of process parameters also forms choice-points, generated during the process within the range limited by predefined constraints. The process parameters (e.g. cutting depth) directly relate to primitive actions where the low-level operations are conducted. In order to remove a component, the agent executes primitive actions within the operation plan,

varying the value of each related parameter until the component is removed. Such a trial-and-error approach is necessary if the parameter value cannot be predetermined. The critical value is generally unknown due to two sources of uncertainties: (a) *unobservable objects* and (b) *imprecision of the operation*. One clearly demonstrative example is in the destructive disassembly of a plastic cover. Hidden snap fasteners are located underneath the cover and need to be destroyed by a cutting operation. Since the fastener locations can neither be predetermined nor detected, trial-and-error is necessary to find the critical depth of cut. In addition, tool wear during the cutting process places uncertainties around the exact cutting destination. Therefore, it can be concluded that the ability to vary parameter values increases the robustness and flexibility of the operation.

Predefined constraints to parameter values include the deepest allowed cutting level, maximum offset from the component border, maximum feed and cutting

speed. Parameters can typically be varied over a continuous range. To reduce the number of possible choice-points, a discrete set of possible choices are provided. Equation (5.2) shows such choices expressed as a simple set. With respect to the logical reasoning of the agent, the choices can also be represented as a set of preconditions (initial and final value) and rules (strategy to step changing) [e.g. Eq. (5.3)].

$$\text{process parameters} = \{a_1, a_2, a_3, \dots a_n\} \quad (5.2)$$

$$z_{min} \leq z_{cut} \leq z_{max} \text{ and } z_{cut} = z_{old} + \Delta z \quad (5.3)$$

5.4.2 Reasoning

In the field of artificial intelligence, reasoning is one of the main functions used in automated planning. The intelligent agent autonomously carries out the process from the initial condition to the goal state under the given constraints, which model the agent and the external world. A number of reasoning techniques are widely used [12]. This book presents only the simple method of *rule-based reasoning*. The interaction with the information from the external world is also emphasised. More complex reasoning methods can be applied using this proposed framework.

According to the cognitive robotics architecture presented in Sect. 5.3, the agent prunes the search space by reasoning about the conditions at each state of disassembly. These conditions occur in four categories:

- The disassembly domain representing plans and operations;
- Condition of the current disassembly state as perceived by the sensors;
- Success or failure of removal operations; and
- Existing knowledge regarding the process for specific models of products.

The disassembly domain shown in Fig. 5.10 should be reconsidered due to these perspectives. As a result, the more specific version is shown in Fig. 5.12 where one state of disassembly is emphasised.

For an *unknown* model of a product, the agent performs disassembly based on trial-and-error throughout the disassembly domain. At the planning level, the agent reasons about the main structure and the main

components detected in each state. Relations between the combination of components and appropriate treatment plans must be defined. In this case, a rule-based heuristic is used. A set of rules are pre-defined by a human expert according to each product family as a broad guideline for the agent. To determine the correct level of specificity for the rules, one must consider the trade-off between the achievement rate and flexibility of the system. Highly specific rules may efficiently handle specific product models but are then less capable of disassembling unknown models. The proposed set of rules only takes the existence (presence or absence) of the main components into account. The existence of components is a priori incomplete knowledge that is obtainable

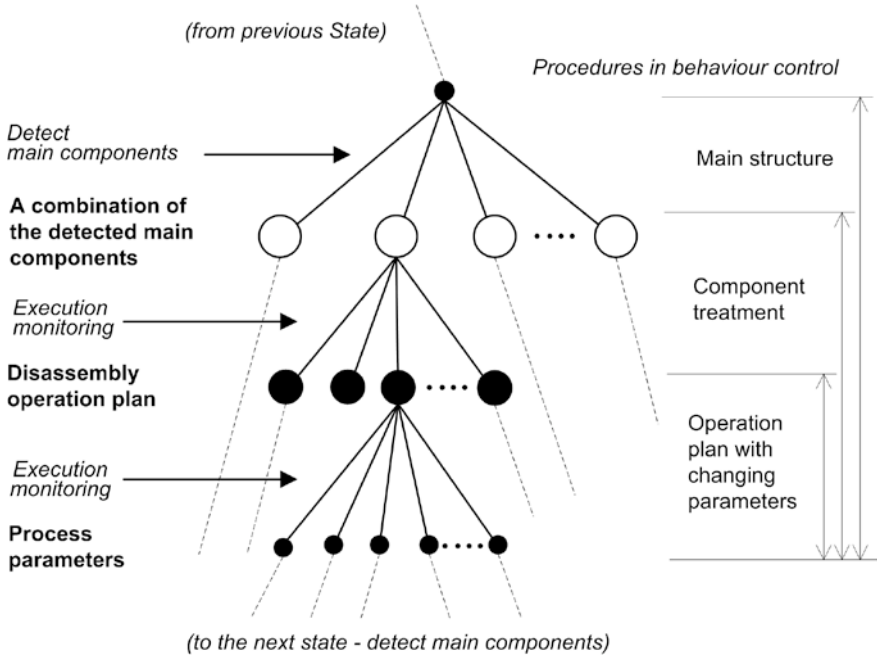


Fig. 5.12 Behaviour control in regard to the disassembly domain

via the sensors. The possibilities, regarding components that can be detected in each state, are also modelled.

In Statement (5.4), the component- k (c_k) is treated if the left hand side of $rule_k$ is satisfied, i.e. if the components c_i to c_n are detected and c_j to c_m are absent. A complete set of rules is required for each component to be treated. An example is given in the case-study in Chap. 6.

$$rule_k : (c_i \wedge c_{i+1} \wedge \dots \wedge c_n) \wedge \dots \wedge \neg c_j \wedge \neg c_{j+1} \dots \wedge \neg c_m \rightarrow treatment(c_k) \quad (5.4)$$

where $c_k \in c_i, \dots, c_n, \dots, c_j, \dots, c_m$.

After the rules are established, the CRA is able to reason about the correspond-
ing disassembly operation plans and their process parameters respectively. The CRA prunes the process parameter options belonging to a disassembly

operation plan until either the goal is achieved or all available options have been exhausted. The goal in each disassembly state is the removal of a particular main component. In the case that the component is not removed, the agent backtracks to the next available operation plan and prunes the corresponding process parameters on the new operation plan. This pruning can be considered a depth-first search. The *execution monitoring* process conducted after physical operation determines whether or not the component has been removed. After achieving the goal, the CRA returns to the planning level to decide the main component to be treated in the next state.

This process continues until reaching the ultimate goal, in which the entire product has been disassembled. This component treatment process can be written as Procedure (5.5).

```

Proc treatComponent( $c_i$ )           // treat component  $c_i$ 
  while( $\neg$ stateChanged)           // repeat operation if state not changed
    operationPlan( $c_i, k, i$ )       // execute plan  $k$  with parameters  $i$ 
    removeComponent
    detectStateChange              // execution monitoring - success/failure
  if ( $\neg$ stateChanged)
    then vary( $k$ ); vary( $i$ )         // try removing the component
  endIf
                                   // disassemble component
endWhile
endProc
```

(5.5)

For a *known* model, the agent performs the process according to the knowledge in the KB, built up from previous operations. The process is performed in a straight-forward manner from one state to another with minimal trial-and-error. However, the sensing of components and execution monitoring are still necessary to maintain the robustness and flexibility of the system. The agent needs to be capable of adapting to possible changes caused by process uncertainties and variations in the product that remain unknown until a certain disassembly state is reached.

A challenge occurs in reasoning regarding destructive disassembly, because it is an irreversible process. When only dealing with reversible processes, as may be the case in non-destructive disassembly, the agent can backtrack to other options in a straightforward manner. In this case, if a particular operation fails, the agent can return the external world to the same condition as before the failed operation was attempted. Then the next operation will attempt subsequently. For the destructive operations, permanent damage on the components can introduce new uncertainties resulting in more complex conditions for backtracking. These uncertainties can only be addressed by using additional strategies, e.g. statistical reasoning approaches, a physical model or sensing techniques for detecting damage, operation plans capable of handling a wider range of uncertainties. This issue will be explained in the case-study in Chap. 7.

5.4.3 Execution Monitoring

Execution monitoring is considered an important function of closed-loop control providing the feedback of the process and operation. In this case, the execution monitoring is performed at the planning and the operational levels where the outcome is either success or failure. The motion control in motor-sensor level is not included in this scope.

Execution monitoring at the planning level deals with the main structure in the disassembly domain (see Fig. [5.12](#)). In each state of disassembly, it determines whether the current main component has been detached within the component

treatment according to the reasoning process. As a result, the agent will proceed to the next state if the removal is successful. Otherwise, the trial-and-error process using other available options of the disassembly plans and process parameters will be continued. A general execution monitoring is shown in Procedure (5.5) where *detectStateChange* is executed. Vision system is generally used for assessing the removal process by detecting the change of the components. Our proposed technique (see Sect. 4.4.4) indicates that the process succeed if the volume and appearance are substantially changed from the beginning of the state.

At the operational level, the execution monitoring focuses on executability of the operation in regard to the physical constraints. Accessibility of the desired component has become difficult problems in complex products. In order to approach the object in exact position and orientation, complete knowledge of the model of the disassembly tool and the working environment, i.e. remaining of the product, are needed. This knowledge can be obtained by predefined geometrical models of the product and the tools. Actual information at the time performing the operation can be perceived from the sensors also. However, inaccuracy of this information potentially occurs due to the physical uncertainties in both cases. Therefore, the feedback indicating the successful approach is used to address this inaccuracy. Since this uncertainty involves the physical condition which is visually non-detectable, force and tactile sensors are used to perceive the information regarding these constraints. Subsequently, the agent will be able to try executing the current operation plan with the alternative available process parameters. If the execution can be completed without any crash, the next operation in planning level will be considered.

The execution monitoring is implemented in both unknown and known models cases. It is crucial in the unknown model case since less information of the process is known a priori. During the trial-and-error process, regarding the learning process, the KB will be built from critical information according to the executed plans, the operations, and the process parameters. The critical information is collected when the state has changed. In case of known model, the agent has the similar behaviour but execute less frequently since most knowledge of a particular model of product supposes to be already built from the first time seeing that model. Therefore, the learning happens in the form of revision. The learning and

revision are explained in the following section: Advanced behaviour control.

5.5 Advanced Behaviour Control

The advanced behaviour involves the learning and revision process of the CRA [13]. The main objective is to collect significant pieces of information extracted in the current disassembly process and reproduce the same disassembly outcome more efficiently. According to the basic behaviour, the advanced behaviours are expressed in every level of plan and operation execution during the disassembly process. As the learning process, the knowledge is initially obtained in the process

for the unknown model. Later on in the revision process, when the known model has been recognised, this existing knowledge is utilised and the agent continuously obtains more knowledge to revise the existing KB for increasing the process performance. The KB is revised repeatedly every time the known model has been disassembled. These activities are conducted autonomously during the process. In addition, human operators involve certain complex situations where the CRA cannot carry out by itself.

In this section, the knowledge base will be explained first followed by the two cognitive functions: learning and revision. Assistance will be explained as parts of learning and revision.

5.5.1 Knowledge Base

To achieve an efficient disassembly, information regarding product and process (e.g. the specification of the main components, their connective components, and detachment method) is expected to be accurately determined from the previous disassembly process. The agent will disassemble according to this information; therefore, the time consumption and redundant operations due to further trial-and-error will be minimised. The knowledge in the KB representing a successful sequence of choice-points in the disassembly domain is used.

The location of components and their corresponding tool executing position are ones of the important process parameters. To accurately determine the specific location, currently, the proposed methods for learning and revision are developed for modifying the *model-specific knowledge* rather than the *generalised rules*. The model-specific knowledge is used since the exceptional features and uncertainties vary from a model to another model. These locations are arbitrary depending on the design of each model and there are other relations presented. In addition, high success rate of disassembly can be achieved by using the model-specific knowledge since it is directly obtained from successful disassembly cases.

The improvement of the process is expected to be seen within first few revisions due to the promising success rate.

In the KB file, the knowledge is stored in the form of structured data according to each specific product model. In this book, the knowledge is presented in the form of Prolog facts due to compatibility with IndiGolog as mentioned in Sect. 5.3.3. The inference engine in Prolog is used to match the input queries to the corresponding facts stored in the KB. According to the prospective industrial scenarios where a massive number of models will be retrieved, the size of the KB is expected to be enormous and needs to be concerned. Therefore, the proposed KB is designed to be compact and represent sufficient information for reproduce the process. Only critical value of the parameters that contributes to the characteristics of individual operations and the entire disassembly process are stored. Considering IndiGolog syntax, the agent will treat each element in the Prolog facts as a fluent. According to the disassembly domain, the knowledge fact is classified

Table 5.2 Facts in knowledge base

Knowledge level	Input queries	Fact
Product-level	–Model	DSP
	–Revision version	Specific product structure
Component-level	–Model	Component location
	–Revision version	General plan (autonomously generated)
	–Main component instance	–Tool executing position
		–Other tool related parameters
		Add-on plan (from human assistance)
		–Customised primitive operation
		–Tool executing position
		–Other tool related parameters

into two main levels: (a) *product-level* and (b) *component-level*. The types of fact are summarised in Table 5.2. The explanation is as follows.

Product-level fact

This level represents an overview of the disassembly process in regard to the main product structure and DSP in which the agent can follow in order to achieve the goal. The agent uses the KB by searching for the corresponding knowledge according to the input queries. The proper set of knowledge can be determined based on two input queries: (a) *model* and (b) *revision*. The agent needs to recognise the model in order to follow the existing plans. The first revision version (*rev*) of the knowledge will be built if the agent has never seen this model before. On the contrary, if this model has been repeatedly disassembled for many times, the latest revision version of the knowledge will be used and modified. The proposed knowledge in the form of Prolog fact is shown in Expression (5.6).

$$dspInKb([model, rev], sequencePlan, structureType). \tag{5.6}$$

where $sequencePlan = [c1(1), \dots, ck(i)]$ and $ck(i) = compType_k(i)$.

Afterwards, these input queries will be matched to the corresponding knowledge of DSP. The DSP is represented by a fluent *sequencePlan* which

is a sequence of indexed instances of the main components to be treated. For example, *sequencePlan* [$c1(1)$, $c2(1)$, $c3(1)$, $c3(2)$, $c4(1)$, $c5(1)$] shows an order for removing 5 types of the main component where 2 pieces of the component type $c3$ are found. The $compType_k(i)$ represents the main component type- k with the index- i .

In addition, specific structure type (*structrueType*) is used to identify groups of the product models having similar characteristics, e.g. similar order of the main components. This knowledge provides extra information that helps the agent in reasoning process to achieve disassembly process more effectively. In Chap. 6, the case-study shows a clear example of the usage of this information to justify the order to treat the main component.

Component-level fact

This level stores the information regarding the treatment of a specified component. The knowledge in physical specification of the component and the related operation are given. The agent matches the corresponding fact according to three input queries, including (a) *model*, (b) *revision*, and (c) *compType*. These queries are related to the product-level fact where the current main component to be treated ($compType_k(i)$) is obtained from the *sequencePlan*.

The knowledge about the treatment process consists of three parts used for representing the location and the related operation. First, the location of the component relative to the product coordinate (*compLocation*) is used as a reference for the locating region of interest (ROI) which can be used in the process of detecting the state change. Second, the operations' details are stored as $plan_{general}$ and $plan_{addon}$ which represent the *general operation plan* and *add-on plans*, respectively. The knowledge for each main component is represented in Expression (5.7).

$$planInKb_k[model, rev], compType_k(i), compLocation, plan_{general}, plan_{addon} . \tag{5.7}$$

For the general operation plan, the knowledge is initially generated by the autonomous trial-and-error process. To execute the operation plan, the agent will use this knowledge incorporating with the existing predefined general plan. The knowledge regarding the general plans is presented in Expression (5.8). Only the critical process parameters are stored in order to minimise the size of the KB. For clarification, the knowledge is represented with two types of parameters, **8** and Γ , which are used to reproduce the primitive actions as originally executed.

$$plan_{general} = \frac{\sum \sum \sum \sum}{\sum \hat{W}} , \hat{W}_0 , \hat{W}_1 , \dots , \hat{W}_n , \dots , \hat{W}_n , \tag{5.8}$$

8 represents the executing position of the disassembly tool which is a critical value of the operation plan. The position relative to the main component can be indicated in a compact form according to primitive feature. $\hat{W}$ represents other tool's related parameters used when

executing the operation. It contains a set of parameters describing the approach of the tool in detail, e.g. orientation, feed speed, etc. The primitive action can be formulated by using these parameters; for example, a rectangular cutting path at the critical depth: $\Phi \text{ rect}(x_1, y_1, x_2, y_2, z_c)$ with tool orientation m with feed speed s . With the cutting contour rectangle operation $\text{cut-Contour}()$, the primitive action is $\text{cutContour}(x_1, y_1, x_2, y_2, z_c, m, s)$.

For the added-on plan, the knowledge is generated according to assistance given to resolve specific issues happened during the process. The primitive actions with the critical parameters are stored directly as $\text{primAction}(\Phi, \Gamma)$ as shown in Expression (5.9).

$$\text{plan}_{addOn} = \text{primAction}_1(\Diamond_1, \hat{W}_1), \dots, \text{primAction}_n(\Diamond_n, \hat{W}_n) \quad (5.9)$$

5.5.2 Learning

The learning process collects the knowledge contributing to the successful disassembly process of a particular model of the product. This knowledge is used for reproducing the same disassembly outcome for the previously seen models. The same disassembly outcome will be achieved based on assumption that the main components are treated in the same order; meanwhile the operations for each main component are assumed to be *order-independent*. Therefore, some variations occur at the operation level and there is a possibility of improvement. Essentially, in the learning process, all operations that have been performed need to be recorded even where the state has not been changed immediately after the operation has finished. The assumption is that all performed operations have contributed to the upcoming state change unless there is *a proof of redundancy* which can be considered by the revision process (see Sect. 5.5.4). The normal form of learning is performed only the first time that the new model has been recognised. Afterwards, the learning will occur in the form of revision. In this section, two normal forms of learning,

i.e. (a) *by reasoning* and (b) *by demonstration*, are described.

Learning by reasoning occurs throughout the trial-and-error process. The knowledge is obtained as the agent conducts the process autonomously according to the basic behaviours of the general plans and operations. According to the KB, all facts will be generated except the fact regarding *plan_{addOn}* which is generated by the learning by demonstration. Each type of the fact is generated in a different stage of the plans and operations.

For the *product-level knowledge*, all elements will be obtained after the entire disassembly process has been achieved. First, the component is continuously added to a list of the fluent *sequencePlan* once the new main component has been detected. Second, the *structure type* can be indicated after classification of the structure types if needed. The classification process can be different according to the variations found in each product. Currently, this classification strategy needs to be predefined by the user before starting the process. However, in the future, to increase flexibility of the system, an ability to add new structure types during the process is

possible by modifying the CRA using *learning by demonstration* strategy [14].

For the *component-level knowledge*, the critical value of parameters entailing the state change or the last action before proceeding to the next operation plan need to be learned due to the aforementioned discussion regarding the redundant operations. This information will be learned after the execution monitoring. If the state has changed, the current plan and parameters are recorded as the critical values and proceed to the next state. If the state has not changed, the current parameters are recorded and the agent will try alternative parameters or operation plans. In this case, the information will be revised and only the final values of the parameters of each operation plans will be kept.³ After all available general plans have

³The revision strategy is similar to one that is done for the human demonstrated primitive actions sequence. It is clearly explained in the section of learning by demonstration where the inaccurate detected location has been addressed [see Eqs. (5.14–5.15)].

been executed or the state has been changed, all required parameters for $plan_{general}$ will be stored in the KB file.

5.5.3 Learning by Demonstration

Tance is involved in this type of learning. The user demonstrates actions which are exogenous actions to address the unresolvable condition. The actions can be a sequence of primitive actions when the agent has struggled to remove the component after all available autonomous operations have been attempted. In addition, the action can be used to change original belief of the agent in case of initially misperception. Theoretically, the assistance is required only for the first time in disassembling the unknown model in order to handle complex situations. The agent supposes to learn from this demonstration and autonomously carries out the entire process the next time this model is found. However, according to the physical uncertainties in the product and the process, minor assistance may be needed in a few subsequent disassembly processes to resolve remaining minor uncertainties.

The problems are associated with the component level and the operational level. The unresolved conditions typically result from the imperfection of the sensors in various detection processes. The problem can be classified into three types, including (a) *false detection of components*, (b) *inaccuracy in localisation*, and (c) *non-detectable components*. Proper types of demonstration need to be given to resolve these problems after the autonomous operations have been finished. These conditions are summarised in Table 5.3.

Table 5.3 Unresolvable conditions to be addressed by human assistance

Level		Type of problem	Sensor

	Unresolvable conditions	False detection by vision system	Localisation inaccuracy	Non-detectable	Vision	Force / tactile
Component	Existence of components	•			•	
	Location of components		•		•	
	Detection of state change	•	•		•	
Operation	Tool executing position		•	•	•	•
	Remaining established connections	•	•	•	•	
	Physical collision with outer objects			•	•	•

False detection

The false detection directly relates to the vision system. The false positive or negative detection affects the reasoning process about the existence of the component and state change. After the information has been sensed, *belief* about the condition of the current disassembly state will be initially created in the agent. This wrong belief can result in logical and physical fault in the subsequent operations.

The *false positive outcome* results in executing the actions that do not suppose to be executed. For the main and connective components, excessive damage on the components possibly occurs due to improper treatment of the component. Subsequently, since the agent believes that a certain component exists, it will try many ways to remove the component which does not exist in reality. This trial can harm the product and waste time. For the state change, the agent will realise that the removal of the component has been successful while it does not happen in reality. Therefore, logical understanding of the main structure of the product will be wrong. The user can give action in Eq. (5.10) to manually indicate the state change resulting in skipping of this component.

$$stateChanged = true \quad (5.10)$$

On the contrary, the *false negative outcome* results in an ignorance of necessary actions needed to be executed. For the main components, they will not be treated properly and will not be removed. The disassembly state related to this main component will be skipped (human operator gives action in Eq. (5.11)). Therefore, this component will still attach to the product and preventing the system to access the components in the subsequent state. The perception about this main component in main product structure will be incorrect also. For the connective components, it is high possibility of having false negative result due to their small size and occlusion. The fault detection will lead to the connections untreated; consequently, the main component will be unsuccessfully removed. For the change of disassembly state, the agent will not realise that the component has been already removed. Consequently, it will keep trying to remove the non-existence component which leads to infinite loop of trials since substantial change will not be able to take place.

$$action = skipComponent \quad (5.11)$$

Inaccuracy in localisation

Localisation issue takes place after the components have been properly detected (true positive). Regarding the location of components, the location is determined by detection algorithm of the vision system. Inaccuracy is associated with the performance of the detector used and the EOL condition that can vary according to the usage phase. Therefore, location error within an acceptable range is likely to occur due to these uncertainties. The location of the component is an original belief that is used as a reference ROI for determining state change. In case that significant error occurs, success of the plan will be inaccurately measured. Therefore, assistance needs to define the new accurate component location (see Eq. (5.12)).

$$action = compLocation(< component\ location >) \quad (5.12)$$

The tool executing position involves at the operation level. It is a consequence of the reasoning process of the location of the component and the corresponding operation plans. Inaccurate executing position will result in a failure to remove the component. The assistance is needed in order to give extra operations to compensate this position error in addition to the agent's actions. Since the operation plan generally composes with a sequence of primitive actions, the assistance must be given in the same form as in Eq. (5.13).

$$action = primAction_1; primAction_2; \dots primAction_n \quad (5.13)$$

Each executed action needs to be stored in the KB individually due to the specificity. Therefore, the size will be extended according to the number of times assistance given. To reduce this size, a strategy to refine the critical value from a sequence of actions producing similar result should be implemented. Then, only a critical action will be kept in the KB. This is typically applied to the destructive operations where the change, i.e. damage, is explicit on the objects. For example, a demonstration that repeatedly drill at position (x_0, y_0) incrementally deep down to the critical depth. Multiple vertical drills at various depth $z_0, z_1, \dots, z_k$ have been demonstrated along the way. Given that $z_0 < z_1 < \dots < z_k$, only the primitive action with the deepest cut z_k should be kept. The original and the refined action sequences are shown in Eqs. (5.14–5.15), respectively.

$$action_{original} = [cutPoint(x_0, y_0, z_0), cutPoint(x_0, y_0, z_1), \dots cutPoint(x_0, y_0, z_k)]. \quad (5.14)$$

$$\text{if } (z_0 < z_1 < \dots < z_k) \text{ then } action_{refined} = cutPoint(x_0, y_0, z_k) \quad (5.15)$$

Non-detectable components

This category deals with non-detectable objects which are not expected to be detected or incapable to be detected by the sensors. In the unknown

environment according to the variations in products, disassembly operations depend on the sensing capability rather than the prior knowledge. If the environment cannot be fully described by the sensors, the agent will not be able to execute proper actions to achieve the goal of disassembly process. The non-detectable connections and the physical collision with other object are the major problem. Difficulty in determining the proper tool executing positions is a consequence of these problems. Therefore, a sequence of primitive action as in Eq. (5.13) must be given as assistance.

For the connections, one of the challenging problems is to the detectors that is capable of recognise and localise particular types of the connections. According to the explanation in Chap. 4, the connections are categorised into three types, including (a) *quasi-component*, (b) *virtual-component*, and (c) *non-components*. The quasi-components are highly possible to be detected since they are visually observable, e.g. screws, plastic rivet, snap-fits, cable, etc. However, the detection

rate can be low in certain circumstances if they are partially occluded or totally hidden by other objects. On the contrary, the virtual components are more difficult to be detected due to the size and observability, e.g. weld and soldering. For the non-component connections, these are not able to be detected visually since there is no extra connective components involve, e.g. press-fit, glue, mating, etc. In summary, the connections are not able to be properly detected and disestablished in most case. Consequently, the corresponding main component will be irremovable. Therefore, the assistance should be provided in order to locate and remove them.

In regard to the physical collision, when the robot tries to approach an object to be executed, the accessibility can be limited by physical constraints. It is possible that the disassembly tool or part of the robot will crash with other part of the product during the operation. This circumstance can be prevented if the product and the movement path of the automation are accurately modelled and planned. Therefore, force-tactile sensors are typically used in addition to the vision system to enhance the perception of the system. However, this modelling and planning can become more complex and challenging due to the computing resource with respect to the real-time internal and external information. Therefore, if the robot cannot resolve the collision while accessing the object, the human operator should demonstrate tool paths for a proper movement.

5.5.4 Revision

The revision process aims to optimise the disassembly process of the known model by removing redundant operations and reducing the size of the operation plan set in operation-level fact that have been learned. Therefore, the efficiency of the process in term of time consumption will increase. The reduction will be done by retracting the redundant operations that do not contribute to the removal of the main component. This process is expected to be carried out autonomously.

To find out the redundant operations, the treatment process of each main component needs to be repeated by executing the operations in a different order. The process with this variation is expected to be repeated multiple times to obtain the optimal set of operation plans. The smallest subset of the operations that achieve the disassembly goal can be found out by execution monitoring. The subset is obtained by retracting all redundant steps based on assumption that all performed operations contribute to the upcoming state change unless there is a *proof of redundancy*. This redundancy can be proved by not executing some of the previously recorded operations. If the main component is still successfully removed by executing the existing set of operations, the removed operation can be concerned redundant. Retraction of the redundant operation plans will be done subsequently. The KB will be repeatedly revised based on the latest version once the identical known model has been seen. Finally, the revised version of the facts will be stored in the KB.

An example for redundancy is shown in Fig. 5.13 where the final material removed from cuts are related to only $op(i,2)$ and $op(i,3)$ but all of them are recorded in the learning process. The $op(i,1)$ can be considered a redundancy since its cutting area can be covered by another two operations. Therefore, the operation $op(i,1)$ has been proved as redundancy which can be retracted (see Fig. 5.13c) or unnecessary to be recorded at the first place.

The retraction strategy focused on the knowledge associated with the general plans since they are created autonomously by trial-and-error. Therefore, it is high potential that the redundancy is generated. Unlike the added-on plans, operations demonstrated by the human operators are expected to be more specific and necessary for the components removal. It can be assumed that none of the redundant operations occur in this assistance. Therefore, only the general plans will be taken into account.

General approach

A diagram in Fig. 5.14 illustrates a general strategy for finding out the redundant operations by repeating the selected operations from previous process. Initially, to achieve removing a component in the first revision, the operations were executed as this order: $\vec{op}(c,1)$ $op(c,2)$ $op(c,3)$ $op(c,H)$. In the second revision, the executing order will change to many different cases. The $op(c,H)$ which is a

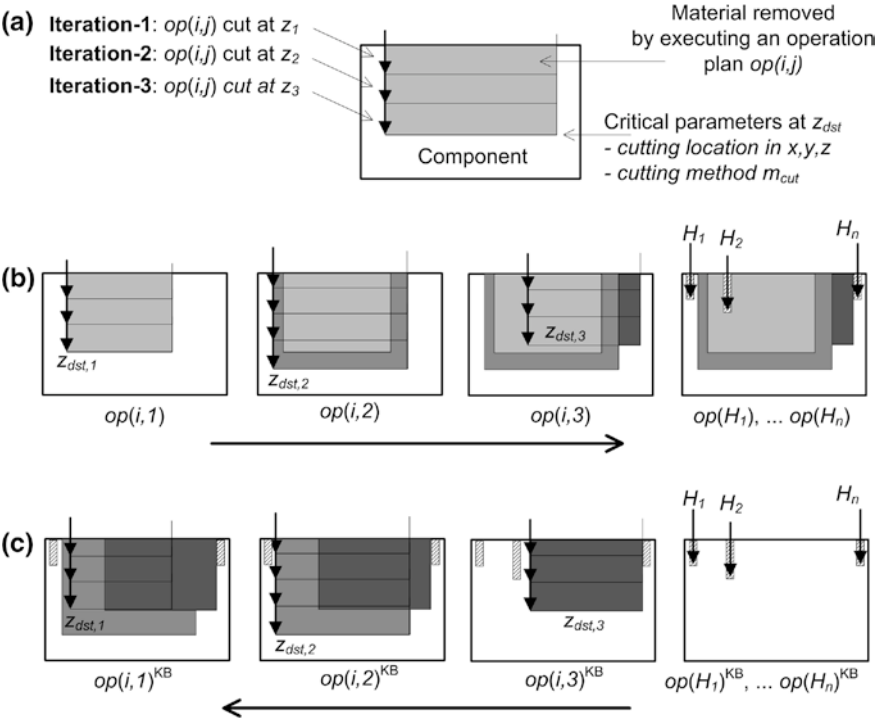


Fig. 5.13 Cutting operations in learning and implementation

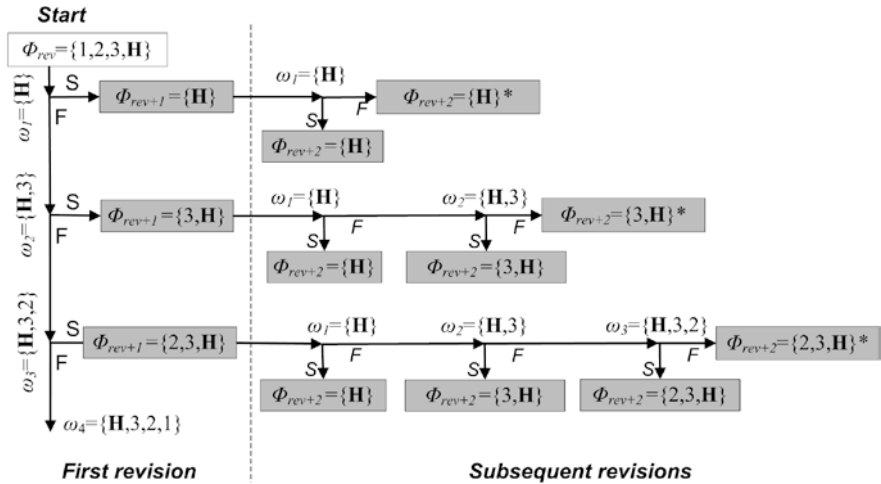


Fig. 5.14 Strategy for retracting the redundant operation

sequence of added-on plans will be executed in any cases and only one operation plan will be skipped in each process. If the component is successfully removed, it can be concluded that this operation is redundant and can be retracted from the KB. In subsequent process, one more operation will be taken out from the remaining valid operation. In case that the removal is failure, another operation plan will be considered. This process will be repeated until all valid combinations of the plans have been tested. This strategy is able to reduce the testing times by reducing the search-space.

Simplified approach: an example for heuristic approach

The aforementioned general approach expects to comprehensively check all valid combination of the operation plans. However, enormous time consumption due to the number of repetitions needed is a drawback. In this section, a simplified retraction strategy using heuristic approach is described. In this example, the order of the operation plans is designed to be directly related to the success rate of the component removal and the impact on the component, e.g. physical damage. Initially, the lower indexed operation has low impact but low success rate; whereas, the higher indexed operation has high success rate but high impact. The

assistance added-on plan is considered as the most specific actions and will not be retracted as always.

In this case, the success of removal is the highest priority of the heuristic strategy. Therefore, to achieve the time constraint, the agent will try executing the plan with the highest success rate first. In the first revision, the executing order is

$$op(c, 1) \rightarrow op(c, 2) \rightarrow op(c, 3) \rightarrow op(c, 4) \rightarrow op(c, H).$$

In the subsequent revision, the process will be carried out in a reversed direction:

$$op(c, H) \rightarrow op(c, 4) \rightarrow op(c, 3) \rightarrow op(c, 2) \rightarrow op(c, 1).$$

With the assessment from execution monitoring, the retraction can be simplified by retracting all plans with lower indices than the current operation without considering the intermediate plans. For example if the component has been removed after executing $op(c, H)$ and $op(c, 4)$, the remaining plans $op(c, 1)–op(c, 3)$ will be retracted without considering an effect of reordering among these plans as done in the general approach. In Procedure (5.16), the retraction for component c_i of the sample “model” is shown.

```

Proc generalPlanRetraction
  < verDsp = j >                                     // current revision j + 1 use previous verDsp = j
  executeAddOnPlan(model, ci);                       // execution the custom plan sequence in KB
  k = n;                                              // start with the highest plan index-n : k = n
  while ¬stateChanged ∧ k ≥ 1 do
    op(ci, k);                                       // execute plan-k in KB
    checkStateChange;
    if stateChanged then                             // if state changed, retract the lower index plans
      retractGeneralPlan([op(ci, 1), ..., op(ci, k - 1)]);
    else k = k - 1;                                  // proceed to the lower index plan
    endIf
    // if fail, add more custom plans and learn
  endWhile;
  if ¬stateChanged then callUser endIf
  writeToKb(planInKb([model, j + 1], ci,           // record the revised verDsp to KB
    compLocationKb', plan'general, plan'addOn))
endProc

```

(5.16)

In summary, this simplified approach can find and retract the redundant operation more quickly than the general approach. A drawback is that the redundant operation plan within the intermediate plans cannot be retracted in any case. However, this simplified approach is effective enough to improve the process performance given that the operation plans are in a proper order. This concept has been proved by the experiment of the case-study which is explained in the next chapter.

Cognitive robotic agent's self modification

Previously, the revision is done on the KB with is a separated part from the cognitive robotic agent. In this case, the cognitive robotic agent will modify its own program in order to revise the program structure which represents its behaviour. The agent will be more flexible to learn and

generalise the operation plans from the disassembly process. Eventually, the agent is expected to be less associated with the model-specific knowledge. With the generalised plans, the agent is expected to be capable of disassemble previously unseen models of product more effectively without assistance. According to the learning by demonstration proposed by Braun [14], the version-space is used for considering the self-modification of the Golog program according to the demonstrations and the existing program. As a result of the learning process, relations between input and output will be formulated. The input is sensible information of a type of component. The outputs are the specific non-detectable connections needed to be disestablished and the corresponding operations. Afterward, the agent will be able to perform proper operations as per information perceived.

Two major challenges are involved with this approach. First, to accurately identify relations is one of the major challenges of this concept. However, these relations have not been discovered yet; for example, there is no relation between the required cutting destination of a hidden snap-fit underneath the middle area of a back cover and the location of the back cover's border. Second, the cognitive robotic agents will be able to evolve in totally different ways according to the training samples. In actual disassembly scenario, the training set is variable and unpredictable due to the EOL product returned. Therefore, after the agents have been trained with a certain number of models, to merge the agent together in behavioural level can be complicated. Currently, this concept has not been successfully implemented by the time this book is being written.

5.6 Implementation Procedure

In order to integrate this cognitive robotic module in disassembly automation, the automation needs to satisfy the requirements regarding the functions needed by the agent. The requirements of each module is summarised in Fig. 5.6. Regarding the cognitive robotics module, the agent is designed based on the cognitive behaviour presented in this chapter. In addition, the functionality and performance of the supporting modules are taken into account. The cognitive robotic module will be created considering the integration procedure of the entire system (see Sect. 3.5). The procedure is summarised as follows:

- Analyse the product,
- Analyse the disassembly requirements;
- Design and assess performance of the functions needed VSM and DOM;
- Create disassembly domain;
- Determine the cognitive robotic functions;
- List the actions, fluents, precondition, effects required for each modules;
- Program the behaviours using Procedure due to complex actions; and
- Connect the modules together via actions and fluents.

In prior to create the cognitive robotic module, the product needs to be analysed and the desired disassembly strategy needs to be decided. The supporting modules also need to be assessed. For the cognitive module, the disassembly domain should be created first; therefore, the overview of the process can be seen more clearly. After that, the requirements in the disassembly process and operation will be implemented by the cognitive behaviours. The four cognitive functions are needed to be designed for performing these tasks. From the programming perspective, the operations are needed to be represented in the form of actions, fluents, preconditions, and effects. These relations can be written as successor state axioms and the complex actions can be defined as Procedures. These elements related not only to the CRM but also VSM and DOM. As a result, the communication among these modules is established and ready for performing the disassembly process.

5.7 Conclusions

The concept of cognitive robotics has been applied to many fields of research. This chapter presents the implementation of cognitive robotics in disassembly context. The main objective is to increase the flexibility and robustness of the system. Basically, the cognitive ability allows the system to handle incomplete knowledge of the world in more effective way. As a result, the problem regarding uncertainties in the product and process can be resolved.

Firstly, the information in disassembly process is needed to be represented in the form that the cognitive robotic agent can understand and utilise. KRR is involved in structuring the disassembly domain which represents the state of disassembly. In this case, the disassembly domain consists of three levels: the product structure, disassembly operation plan, process parameters. The CRA controls the system according to the basic and the advanced behaviour. For the basic behaviour, the agent can reason about the disassembly process and schedule the actions for each operating modules in the system. To make the process more robust, execution monitoring is used for assessing the outcome of the operations. When the operation fail, the alternative operations and parameters will be used in order to complete the goal. In addition, thank to the advanced behaviour, learning and revision strategy make the process becomes more efficient after repeating disassembly the previously seen model.

In conclusion, it is clear that the majority of the uncertainties in the disassembly process can be addressed by the cognitive robotic functions. However, in the actual process, the performance of the system also relies on other operating modules in which imprecision is present. These uncertainties occurred from those sources may introduce a new problem that is unable to be resolved by the cognitive robotic. Therefore, the assistance is still needed for handling unresolved problems. However, due to the learning capability, the human intervention is expected to be reduced after some revision. The system will be fully automated eventually. In the

next chapter, the validation of this concept will be clearly described through the case-study.

References

1. Moreno RA (2007) Cognitive robotics. <http://www.conscious-robots.com/en/conscious-machines/the-field-of-machine-consciousness/cognitive-rob.html>. Accessed Oct 2009
2. Müller VC (2012) Autonomous cognitive systems in real-world environments: less control, more flexibility and better interaction. *Cogn Comput* 4(3):212–215
3. Levesque H, Lakemeyer G (2007) Cognitive robotics. *Handbook of knowledge representation: foundations of artificial intelligence*. Elsevier, Amsterdam, pp 869–882
4. Zaeh M, Lau C, Wiesbeck M, Ostgathe M, Vogl W (2007) Towards the cognitive factory. Paper presented at the international conference on changeable, agile, reconfigurable and virtual production (CARV), Toronto, Canada
5. Beetz M, Buss M, Wollherr D (2007) Cognitive technical systems—what is the role of artificial intelligence? *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, vol 4667, pp 19–42

6. Bannat A, Bautze T, Beetz M, Blume J, Diepold K, Ertelt C, Geiger F, Gmeiner T, Gyger T, Knoll A, Lau C, Lenz C, Ostgathe M, Reinhart G, Roesel W, Ruehr T, Schuboe A, Shea K, Stork Genannt Wersborg I, Stork S, Tekouo W, Wallhoff F, Wiesbeck M, Zaeh MF (2011) Artificial cognition in production systems. *IEEE Trans Autom Sci Eng* 8(1):148–174 (art no 5524092)
7. Vongbunyong S, Kara S, Pagnucco M (2012) A framework for using cognitive robotics in disassembly of products. In: *Leveraging technology for a sustainable world—proceedings of the 19th CIRP conference on life cycle engineering*, pp 173–178
8. McCarthy J (1963) Situations, actions, and causal laws. Technical Report Memo 2, Stanford Artificial Intelligence Project. Stanford University
9. De Giacomo G, Lespérance Y, Levesque HJ, Reiter R (2001) IndiGolog-OAA interface documentation. <http://www.cs.toronto.edu/~alexey/ig-oaa/index.htm>. Accessed 16 July 2011
10. Lapouchnian A, Lespérance Y (2002) Interfacing IndiGolog and OAA—a toolkit for advanced multiagent applications. *Appl Artif Intell* 16(9–10):813–829
11. SWI-Prolog (2010) SWI-Prolog. <http://www.swi-prolog.org>
12. Ghallab M, Nau D, Traverso P (2004) Automated planning theory and practice. Morgan Kaufmann Publishers, Elsevier, San Francisco
13. Vongbunyong S, Kara S, Pagnucco M (2014) Learning and revision in cognitive robotics disassembly automation. *Robot Cim-Int Manuf*. <http://dx.doi.org/10.1016/j.rcim.2014.11.003>
14. Braun A (2011) Programming by demonstration using the high-level programming language Golog. Diploma, RWTH Aachen University and The University of New South Wales

7. Discussion

Chapter 6

Implementation and Case-Study

Abstract The automated disassembly system consists of three main operation modules presented in Chaps. 3–5. In this chapter, procedures for system integration and the development of each module are described in detail, with regard to a cognitive robotic disassembly system designed for disassembling a case-study product, the Liquid Crystal Display (LCD) screen. Finally, the performance of the proposed system is validated by experiments which demonstrate the flexibility and robustness of the system.

6.1 Implementation Overview

Economic feasibility is a major concern in the actual implementation of a disassembly system. This is the main reason why automation is proposed to replace the high-cost human labour. The main objective is therefore to develop a low-cost automated system that is both flexible and robust. The system needs to be capable of handling uncertainties and variations in product and process. To develop the automated disassembly system, the procedure is summarised into the following 6 main steps:

- Product analysis and disassembly strategy,
- System framework,
- Cognitive robotic module,
- Vision system module,
- Disassembly operation unit module, and
- Validation and performance testing.

Firstly, the product to be disassembled is examined to obtain the main product structure and components (see Sect. 6.2). This information is used to create the disassembly domain used by the cognitive robotic agent. The strategy to disassemble this product is needed to be identified (see Sect. 6.3). The information about the components is used for designing the functions of the vision module system and disassembly operating module.

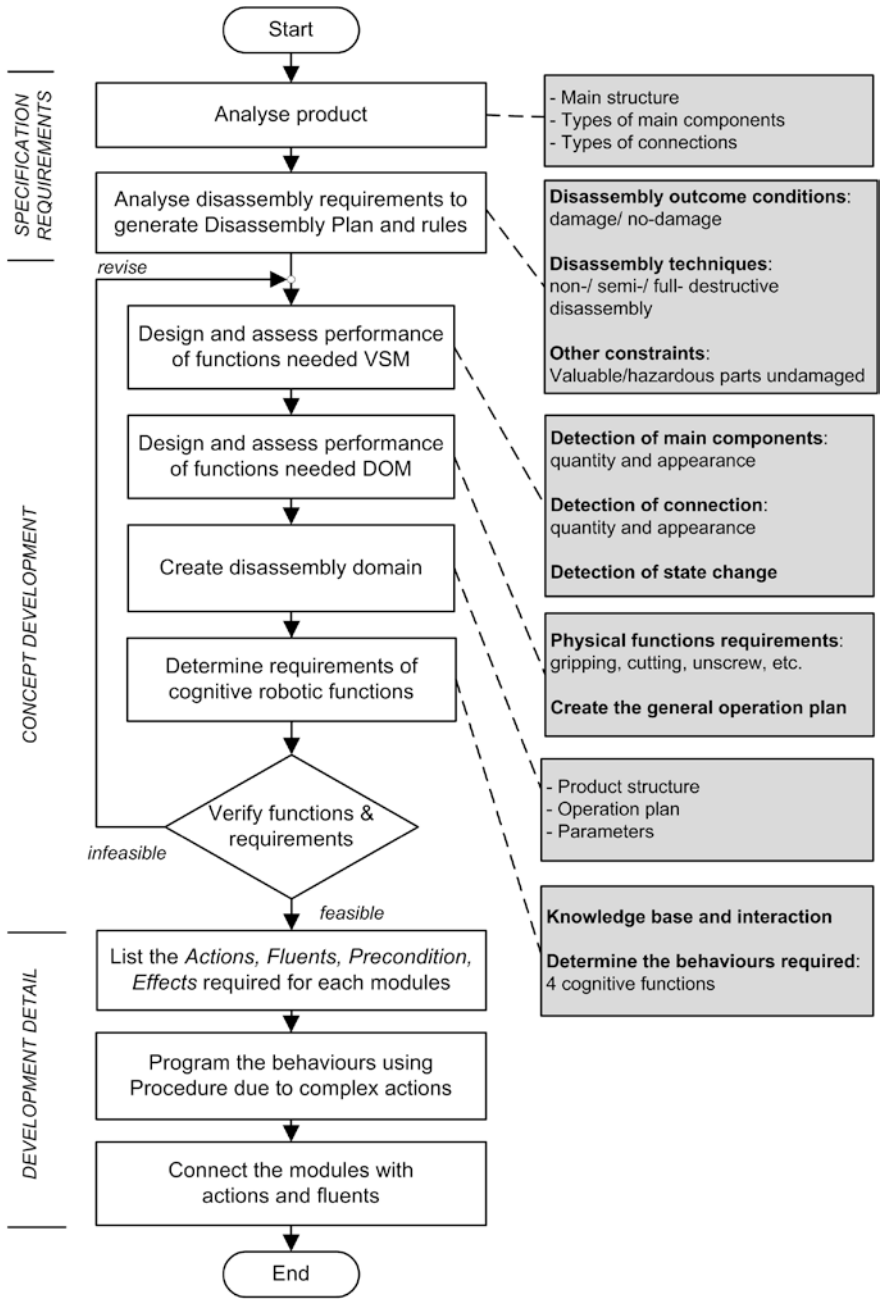


Fig. 6.1 Flowchart for developing cognitive robotic disassembly automation

The system framework (Sect. 6.4) provides an overview over the various control levels and interactions among the operating modules. Each operating module is designed to address the uncertainties in the disassembly process (see Sects. 6.5–6.7). Regarding the concept of cognitive robotics, the activities and behaviour of each module are subsequently represented in the form of actions and fluents. Finally, the performance of the system is validated according to the objective of the system. These procedures are summarised in Fig. 6.1 and explained in detail in the following sections.

6.2 Product Analysis

6.2.1 *Case-Study Product: LCD Screens*

LCD screens have dramatically replaced Cathode Ray Tube (CRT) monitors over the past decade. More than 120 million units of LCD screens were sold worldwide in 2008 and used in approximately 90 % of desktop computers in 2010 [1]. According to an earlier prediction, the sales of LCD screens would be US\$80 billion in 2012—approximately four times higher than that of other types of monitors [2]. Therefore, the number of disposals is continuously increasing. In Germany alone, it is expected that more than 4,000 tons of LCD screen monitor will be disposed of by 2012 [1]. The environmental impact of this product is significant and increasing. EOL treatment needs to be considered.

6.2.2 *End-of-Life Treatment of LCD Screen Monitors*

Treatment of EOL LCD screens is considered according to the European Directive on WEEE [3]. This directive indicates the recovery rate as well as hazardous substances that need to be taken into consideration: both aspects need to be considered in the design of the disassembly process. The minimum threshold requirements by weight are 75 % for material recovery and 65 % for recycling and reuse.

This study started with the investigation of a number of LCD screens, in order to gauge their material composition and the commonly-found components. The majority of the overall weight is contributed by the main components, i.e. lightbox casing, PCB mounting panel, back cover, and PCBs [2]. According to the material, the distribution is shown in Fig. 6.2 [4]. The target of 78 % material recovery by weight is achievable if material fractions, i.e. ferrous metal and Halogen-free plastic, are separated before recycling.

Regarding the treatment of hazardous substances indicated by the directive, the removal of three potentially hazardous components must be taken into account: (a) the Cold Cathode Fluorescent Lamps (CCFL), (b) the LCD glass panel, and (c) PCBs [2]. The disassembly of CCFLs (a) is difficult due to its fragility and strong

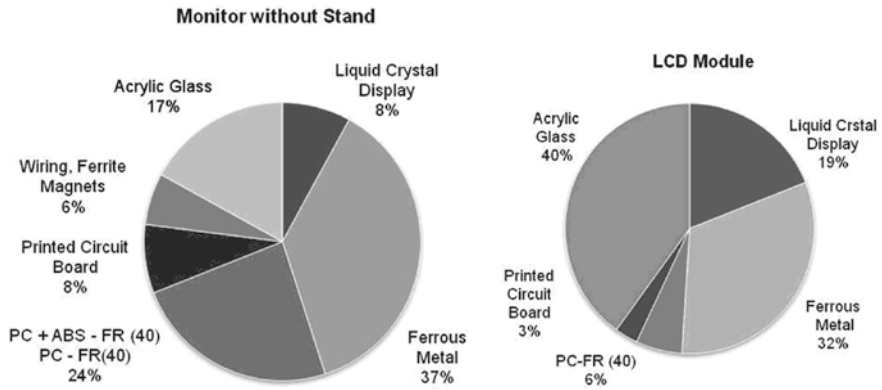


Fig. 6.2 Distribution of the material in LCD screen (adapted from [4])

connection with other components in the LCD module. Special treatment is required due to the small amount of Mercury contained in CCFLs. The amount of Mercury varies in different screen with approximately 4.8 μg per screen¹ on average [4]. The LCD glass panel (b) is a potential environmental risk. Any LCD larger than 100 cm^2 must be removed from any WEEE. Therefore, the LCD module must be disassembled to retrieve LCDs. It is usually carried out destructively [1]. Third, the PCBs need to be separated since they contain several types of metal and thermoplastic material which is difficult to recycle. Any PCB larger than 10 cm^2 must be removed from the WEEE [2].

It is clear that component separation is important in the EOL treatment process for LCD screens. The disassembly process needs to be designed well to be efficient and satisfy the constraints regarding the hazardous components. However, achieving economic feasibility via manual disassembly is still a challenging problem. It was decided to implement selective disassembly to the module level. The LCD module can then be further destructively disassembled but CCFL and LCD should be secured.

6.2.3 Structure Analysis and

Components

The basic structure of the product is needed in formulating the disassembly domain used by the cognitive robotic agent. Information regarding the components is needed for developing the functions for visual detection and operation plans.

¹ Estimate that 250–480 kg of Mercury potentially found in 80 million LCD screens are expected to be disposed by year 2010.

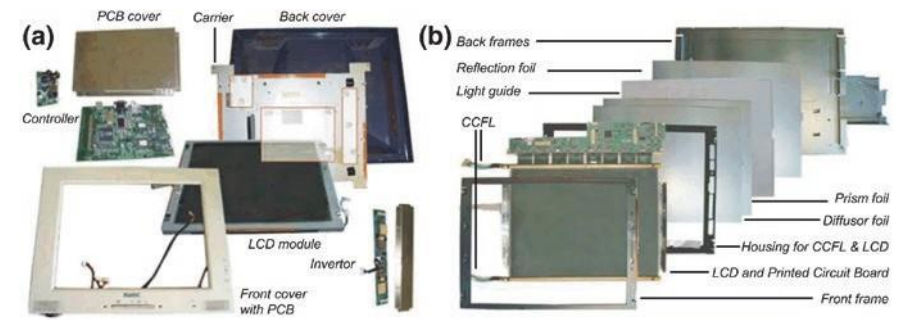


Fig. 6.3 Structure of LCD screen [5]. **a** Modules in LCD screens. **b** Components in LCD a module

A comprehensive study of the structure and components of LCD screen are reported in many research works [2, 5]. A common structure can be presented in two levels: (a) *module level* and (b) *component level* (see Fig. 6.3). Disassembly at the module level can be examined by the selective disassembly approach. This product consists typically of 6 types of main components: a back cover, PCB covers, PCBs, a carrier, an LCD module, and a front cover. Three types of PCBs are commonly found, i.e. a power supply—inverter board, a switch board, and a controller. This order of the component is from the front to the back of the screen. This general structure of LCD screens is similar among the different manufacturers. However, the location of the components, e.g. PCBs, screws, cables, can be significantly changed even within manufacturer families. With respect to the component level, the LCD module can be further disassembled into 9 components (see Fig. 6.3b).

As the disassembly system is aimed at material recovery, this solution considers the concept of component and structure in a slightly unusual way. Rather than the specific identification of components, only the component type needs to be determined. Such a definition is beneficial in this context because the system is then able to immediately begin operation on previously-unseen models. A more relaxed definition of product structure is also adopted, whereby only the attachment direction and relationships are taken into account. This solution exploits the advantages of the cognitive functions and operating modules, in par-

ticular the vision system. Figure 6.4 describes the product at the module level under this scheme.

The product structure can be classified into two types according to assembly direction: *Type-I* and *Type-II*. This classification is based on the configuration of components PCBs, PCB cover, and carrier. The PCB cover is a separate part in the Type-I structure, whereas it is integrated with the carrier for Type-II. In addition, Type-I can be further categorised into two sub-classes according to the material and appearance of the PCB cover. Type-Ia is visually distinguishable from the carrier by the vision system while Type-Ib is not. The CRA incorporated

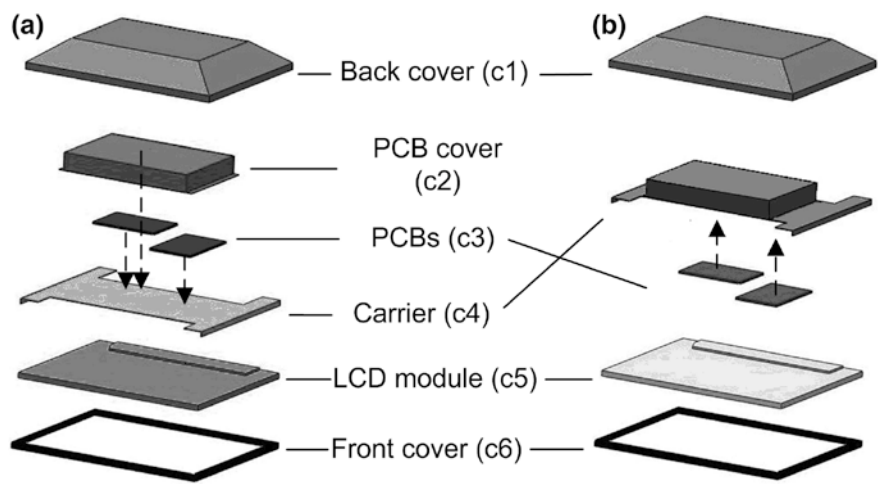


Fig. 6.4 Product structure of LCD screens and the main components. a Type-I. b Type-II

with physical disassembly operation is used to differentiate these two sub-classes. LCD screens typically consist of six types of main components and four types of connective components denoted by 'c' and 'cn', respectively. A list of components is as follows:

- Back cover (c1);
- PCB cover (c2);
- PCBs (c3);
- Carrier (c4);
- LCD module (c5);
- Front cover (c6);
- Screws (cn1);
- Snap-fits (cn2);
- Electric cables (cn3); and,
- Plastic rivets (cn4).

The variation of the structure and appearance of the components are the major challenges in the automated disassembly. Examples of the liaison diagrams of LCD screens with each structure types are shown in Fig. 6.5 and an example of a very complex structure is shown in Fig. 6.6. Using traditional approaches for effective automated disassembly, all relevant information regarding the product and process would typically be specified a priori. This

information is model-specific and unfortunately often unavailable due to the confidentiality of the manufacturers' design. Instead, the recognition of component by type and the principle of cognitive robotics are applied to resolve the variations and uncertainties.

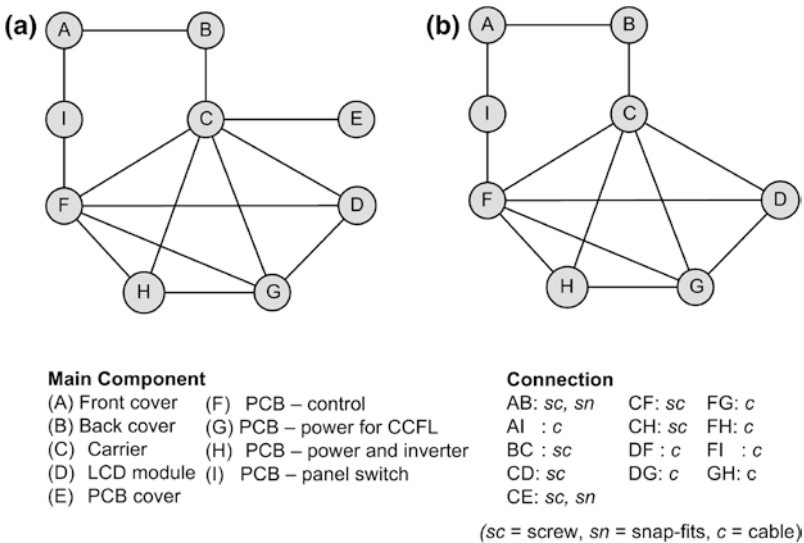


Fig. 6.5 Liaison diagram. a Type-I. b Type-II

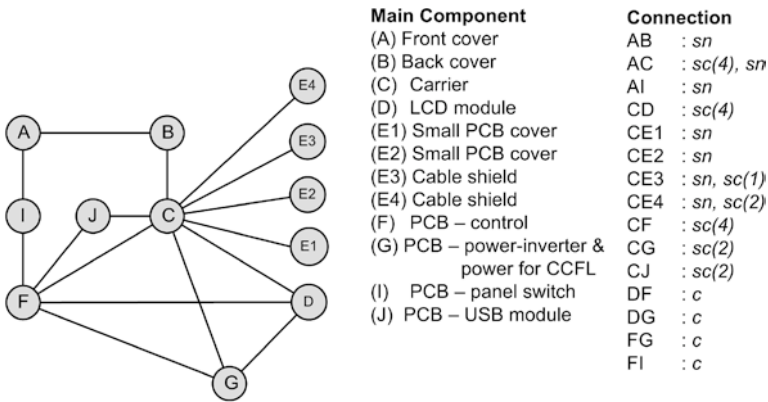


Fig. 6.6 Example of a complex structure LCD screen

6.3 Disassembly Requirements

Disassembly is a low-profit activity but nonetheless important for EOL treatment. Therefore, economical feasibility is a major concern. The automated system needs to be low-cost but flexible and robust enough to deal with a variety of the samples. According to the operation, (semi-)destructive disassembly is suggested due to its robustness and flexibility [6]. However, the damage to the disassembled parts is a major drawback. Therefore, this approach is appropriate for recycling but not

suitable for reuse or remanufacture. Regarding disassembly strategy, it is difficult to achieve economically feasible full disassembly due to the decreasing value returned from time consumed. As a result, the LCD screens are selectively disassembled to the module level [5]. In addition, valuable components, as well as the hazardous constituents according to the WEEE directive, need to be considered. Breakage of CCFLs results in leakage of mercury. Hence, the LCD module will not be further disassembled to avoid damage to the CCFLs. The PCBs are crucial for removal. In summary, the automated system is designed based on the following requirements.

- Low-cost design with sufficient flexibility and robustness;
- (Semi-) destructive disassembly;
- Selective disassembly; and,
- Special requirements for the hazardous and valuable parts.

6.4 System Overview

The control architecture of the system is designed based on the cognitive robotic architecture presented in Sect. 5.3.2 with respect to the simplified scope of the case-study product. The detailed configuration of the system in regard to level of control and operating module is illustrated as a schematic diagram in Fig. 6.7.

6.4.1 Levels of Control and Operating Modules

The system can be considered in two complementary perspectives: the level of control, and the operating module. The level of control specifies the degree of automation and level of data abstraction of a given process; the operating module is the functional system in which the process belongs.

One operating module can consist of multiple levels of control depending on the behaviour of processes needed for carrying out the module's functions.

Levels of Control

The levels of control relating to the individual operation modules are explained in detail in earlier chapters. This chapter only provides an overview such that the connections and information flows between the modules can be clearly seen. The system consists of three levels of control: (a) *high-level*, (b) *mid-level*, and

(c) *low-level* (see Fig. 6.7).

The *high-level* controls the top-level behaviour of the system and planning of the disassembly process. Commands and information transferred to other levels are in abstract form. The cognitive robotic module operates at this level.

The *mid-level* is used to manage the information exchange between the high-level and the low-level. It is at this level where data is interpreted into information

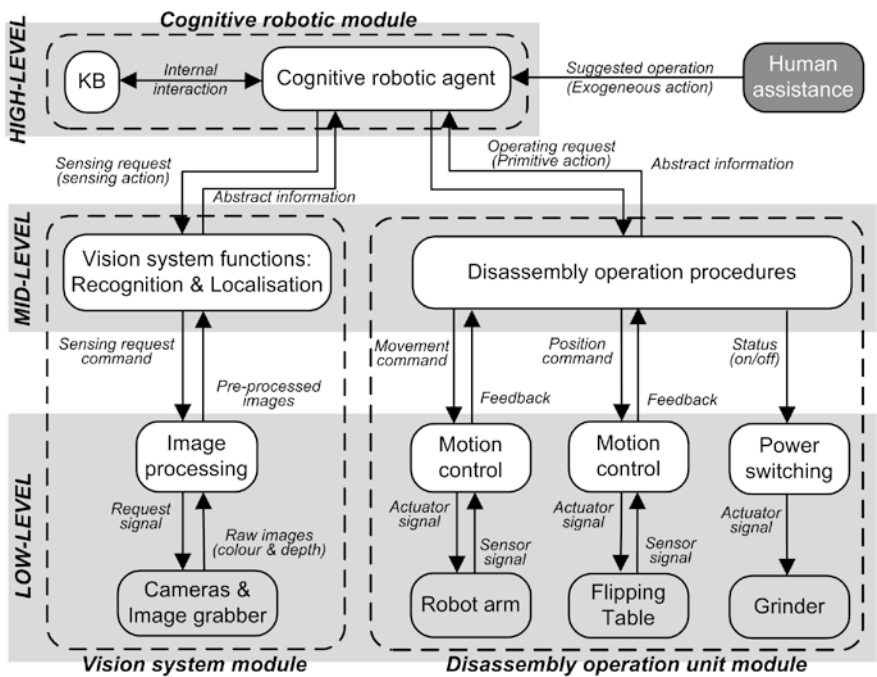


Fig. 6.7 System architecture—levels of control and operating modules

and abstract information into realisable actions. The detection functions of the vision system and the operation procedures of the disassembly operation unit modules operate at this level.

The *low-level* deals with machine-level hardware operation and signal processing, including sensor-actuator motion control and image acquisition and pre-processing.

Operating Modules

The system consists of three modules, which are designed based around specific functionalities and operate independently from each other. The operating modules and the associated hardware and functions is summarised in Table 5.2.

The *cognitive robotic module* (CRM) consists of cognitive robotic agent (CRA) and knowledge base (KB). The high level planning and interaction with the human operator are its main functions.

The *vision system module* (VSM) consists of the detection functions and two cameras—a colour camera and a depth camera.

The *disassembly operation unit module* (DOM) is responsible for physical interaction with the disassembly objects. It consists of a robot arm, a flipping table, and a grinder. The robot arm is used to move the grinder which is a disassembly tool and the flipping table is used to unload the detached components from the product (Table 6.1).

Table 6.1 Operating modules

Operating module	Associated hardware	Main functions	Language
CRM	Main computer	Cognitive robotic agent	IndiGolog
		Knowledge base	Prolog [7]
VSM	Depth camera	Detection functions	C/C++
	Colour camera		
DOM	Robot arm	Operation plan and primitive actions	RAPID [8]
	Flipping table		C/C++
	Grinder		

Communication System with a Client-Server Model

Internal communication is carried out in a straightforward manner within the scope of each operating module. A *data acquisition device* (DAQ) is needed in the case of the VSM and DOM since the main computer interacts with the hardware, namely the image grabber and I/O controller. The robot controller is comparable to the main computer and is capable of fully controlling the robot arm via the connected low-level hardware. An overview of the physical connections showing both internal and external communication is shown in Fig. 6.8.

The external communication among different modules is more complicated since the operating modules are developed on varying platforms. Therefore, the multi-platform client-server model [7] is selected to enable communication over the network. A communication protocol using Transmission Control Protocol and Internet Protocol (TCP/IP) based on the Prolog syntax is used to manage the information flow via socket-messaging. In this case, the messages are in the form of *actions* and *fluents* (see Chap. 5 for more detail). The protocol is designed to

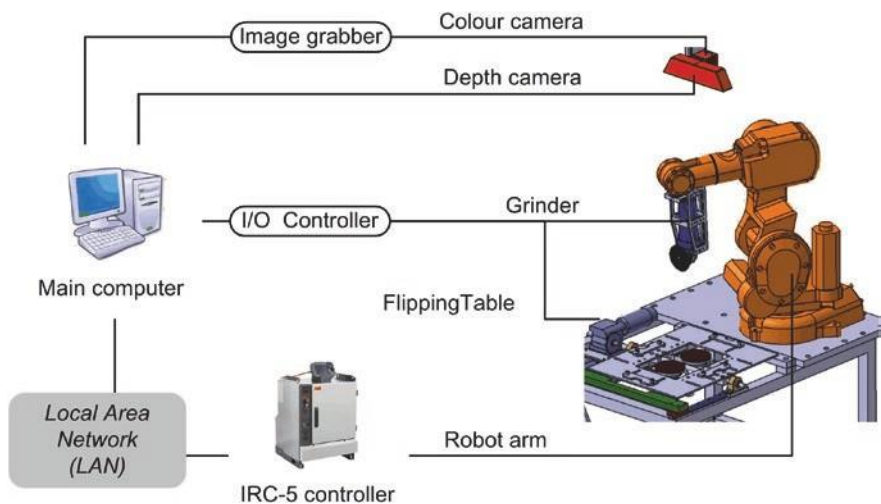


Fig. 6.8 Schematic diagram of the physical connections

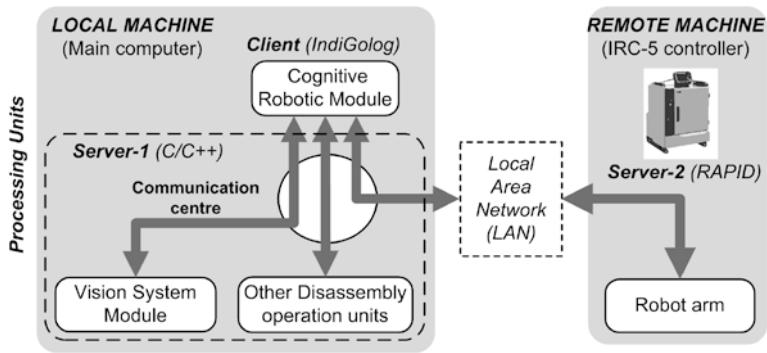


Fig. 6.9 Schematic diagram of communication network structure

be compatible with the cognitive robotic module, the planner which directs the operation of the other modules.

The network consists of three components: (a) *client*, (b) *server*, and (c) *communication centre*. In this case, the cognitive robotic module is a client and the other modules are servers (see Fig. 6.9). The client communicates with the servers by sending requests to the communication centre, either for sensing actions or primitive actions. Subsequently, the message is distributed to the corresponding module. Feedback is sent back from each module once the required information has been processed or the requested actions have been performed. The communication centre further adjusts the syntax as needed according to the different platforms. The communication system can be grouped, with respect to the location of the programs and language platform (see Fig. 6.9).

In Fig. 6.9, the client and Server-1 both operate on the local machine, located on the main computer. The client is the CRA which is the planner of the system. The TCP/IP connection is established via Prolog. C/C is used among many components on the local machine: the vision system, communication centre, and the disassembly operation module excluding the robot. These components are grouped together as Server-1 to simplify internal communication. Window Socket Library (winsock2) [8] is used to implement the communication protocol. Server-2 is the module located on a remote machine, the controller for the robot arm. TCP/IP for socket

messaging is implemented in RAPID. This server connects to the communication network through the Local Area Network (LAN).

Interaction Between Modules—Actions and Fluents

With respect to the integration of the system, the actions and fluents associated with each module are listed. The actions are classified into three types: (a) primitive actions, (b) sensing actions, and (c) exogenous action, which are described in Sect. 5.3.3. Example of interactions between each pair of operating modules are

given in Statements (5.6)–(6.3), where ‘s’ denotes a sensing action; ‘p’ a primitive action; ‘e’ an exogenous action; and, ‘f’ a fluent.

detectBackCover. // s: CRM asks VSM to detect backcover
box($x_1, y_1, x_2, y_2, z_1, z_2$). // f: VSM sends the detected location to CRM (6.1)

cutRect($x_1, y_1, x_2, y_2, z_1, m$). // p: CRM asks DOM to do operation
done. // f: DOM informs CRM of the completion (6.2)

senseHumanAssist. // s: CRM asks human operator for assistance (6.3)
(demo action). // e: human operation demonstrate an action

In conclusion, the system architecture is a composition of three operating modules that seamlessly connect with each other via the network system using a client-server model. The actions and fluents represent the commands and information used to interact among the modules. They are encoded in a form compatible with Prolog syntax, according to the client’s requirements. The messages are conveyed to the desired operating modules through the communication centre, which resolves the multi-platform compatibility problem. In the following sections, the operating modules are explained in detail. The design and the functional requirements according to LCD screens are described. The complete list of significant actions and fluents will be shown in the section of each module.

6.5 Cognitive Robotic Module

The cognitive robotic module is configured as presented in Chap. 5. In short, the CRM consist of CRA and KB. The CRA is formulated with the

behaviour specification and the domain specification. The behaviour specification is influenced by the four cognitive functions and the domain specification is formulated with the disassembly domain. As this module controls the behaviour of the system, an overview of the entire disassembly process will be given in this section.

6.5.1 Design and Functions

Domain Specifications

The disassembly domain consists of three levels: (a) the product structure, (b) operation plan, and (c) process parameters (see Sect. 5.4.1). The state diagram of the product structure is constructed specifically for LCD screens (see Fig. 6.10). The information about this variation was gathered from around 40 different model samples. The products are disassembled into the module level. As described earlier in Sect. 6.2.3, two main structure types with six main components are considered

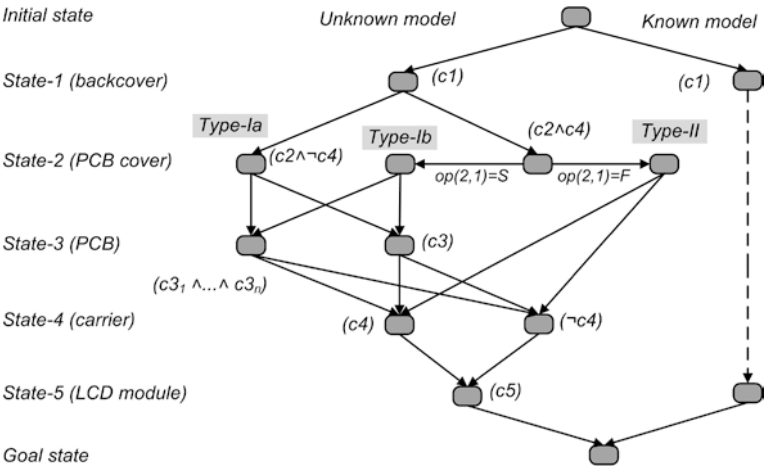


Fig. 6.10 Disassembly domain showing main structure of LCD screens

in developing this state diagram. Although significant variation in the ordering between PCB cover—PCBs—carrier is found, it is evident that the order of the main components is consistent in different models. The structure type can further- more be identified based on the order of components detected. This is seen from State-2 to State-4 in Fig. 6.10 where both the visual detection outcome² and exe- cution monitoring after performing a cutting operation³ are needed to identify the correct state.

Basic Behaviour

In accordance with the (semi-)destructive approach, operation plans and param- eters are designed based on the cutting operation using a grinder. Each type of component requires a different strategy for removal due to the varying types and locations of associated connective components. As a result, appropriate operation plans and process parameters need to be individually determined for each main component.

The operation plans are composed by the following parameterised primitive actions: (a) *cutContour*, (b) *cutCorners*, (c) *cutLine*, and (d) *cutPoint* (see com- plete list of actions in Appendix A). The inputs are fluents representing primitive geometries (i.e. *rect*, *line*, and *loc*) according

to each primitive action. The initial investigation suggested that due to the simple structure of LCD screens, it is

² In Fig. 6.10, notation in the bracket is the combination of the components detected or not detected at that state, e.g. $(c_i \wedge c_j)$ denotes the state that c_i is detected whereas c_j must not exist.

³ The executed j th-operation plan of component c_i is denoted with $op(c_i, j)$. 'S' and 'F' represent the execution result, success and failure, of removal the component.

sufficient to define all cutting actions on horizontal planes. The cutting location on this horizontal plane is parameterised with respect to the sensed component location, fixed via an offset according to the operation plan. The required depth of cut is initially unknown, and incrementally changes in each operating cycle until reaching a critical depth which results in main component removal. The feed speed and the orientation of the cutting tool are variable and considered process parameters. In short, 1–4 operation plans (see Eq. (6.4)) with 3 process parameters (see Eqs. (6.5) and (6.6)⁴) are used depending on the type of component. The set of possible variations can be seen from these definitions (as detailed in Sect. 6.7.2).

$$\text{operation plan} = \text{op}(ci, j) | i \in \{1, 2, \dots, 6\} \wedge j \in \{0, 1, \dots, 3\} \quad (6.4)$$

$$m \in \Theta_{\text{tool}} \times S_{\text{feed}} = \{'0', '1', \dots, '8'\} \quad (6.5)$$

$$z \in z_{\text{Fmin}} \leq z_F \leq z_{\text{Fmax}} | z_F \in \mathbb{Z} \quad (6.6)$$

$$Oz \in Oz_{\text{min}} \leq Oz \leq Oz_{\text{max}} | Oz \in \mathbb{Z} \quad (6.7)$$

Execution monitoring is implemented at the planning and the operation in a specific way. At the *planning level*, the removal of a main component is assessed by visually detecting the state change. The sensing action *detectStateChange* is executed after executing each cutting operation plan, followed by an attempt to remove the component. In this case, the primitive action *flipTable* flips the fixture holding the product in place, causing the component to fall and be collected if the threshold depth has been reached (see Sect. 6.7.2). Execution monitoring is used to classify the type of main structure as shown in State-2 of Fig. 6.10. The success or

failure of the operation $op(c2,1)$ differentiates between Type-Ib and Type-II.

The success of a cutting operation without collision is also an important source of information in the *operation level*. A successful completion of the cutting path implies that cutting tool is able to reach the cutting destination using the current cutting method m . The robot is hence capable of determining a suitable cutting method by itself. The successful cutting method is acknowledged by sensing action *checkCuttingMethod*. Subsequently, since the destination can be vertically approached, this m can be used in the next operation cycle to repeat the horizontal

⁴ The cutting method m represents a combination of tool orientation ϑ_{tool} and feed speed S_{feed} . In this case, the combinations are denoted with '0'...'8'. The z_F denotes the level relative to the fix- ture and Δz is incremental step changed. z_F and Δz are discrete value as they are integers.

cutting path with an adjusted depth. In summary, the operation routine for operation plan $op(c_i,j)$ with execution monitoring for this system is in Procedure (6.8)

```
Proc  $op(c_i,j)$ 
  while  $(\neg stateChange \wedge m \neq 0 \wedge (z_{Fmin} \leq z_F \leq z_{Fmax}))$ 
    offsetPrimitiveDepth           // incrementally change  $z_F$ 
    cutPrimitive                   // execute cutting operation
    checkCuttingMethod            // obtain fluent cuttingMethod
    flipTable                     // remove the detached part
    detectStateChange             // obtain fluent stateChange
  endWhile

endProc
```

(6.8)

Advanced Behaviour

The knowledge base, learning, and revision are implemented as explained in Sect. 5.5 but applied more specifically to LCD screens. Since the operation plans are pre-defined, the number of the operation plans is known a priori. Therefore, the agent’s program is simplified by limiting the learning to this known structure. According to Expression (5.7), the Φ_i store the primitive geometry of the horizontal path of the desired cutting destination. Γ_i are the parameters for the tool’s approach. The first part, representing the operation for connection disestablishment, refers only to screw removal.

As a result, a simplified version of the general plan is given in Eq. (5.8), where the maximum number of operation plans available for the main components is four. Γ_0 for screw cutting represents only the cutting method since the cutting depth for the screw head is quite consistent. On the other hand, because the thickness of components is unknown, the other Γ_i must be additionally specified with a starting depth z_{start} and end depth z_{dst} in addition to m .

$$plan_{general} = \sum_{i=0}^3 \sum_{j=0}^3 \sum_{k=0}^3 \sum_{l=0}^3 \hat{W}_{0,i,j,k,l}, \hat{W}_{1,i,j,k,l}, \hat{W}_{2,i,j,k,l}, \hat{W}_{3,i,j,k,l}, \hat{W}_{4,i,j,k,l}, \hat{W}_{5,i,j,k,l}, \hat{W}_{6,i,j,k,l}, \hat{W}_{7,i,j,k,l}, \hat{W}_{8,i,j,k,l}, \hat{W}_{9,i,j,k,l}, \hat{W}_{10,i,j,k,l}, \hat{W}_{11,i,j,k,l}, \hat{W}_{12,i,j,k,l}, \hat{W}_{13,i,j,k,l}, \hat{W}_{14,i,j,k,l}, \hat{W}_{15,i,j,k,l}, \hat{W}_{16,i,j,k,l}, \hat{W}_{17,i,j,k,l}, \hat{W}_{18,i,j,k,l}, \hat{W}_{19,i,j,k,l}, \hat{W}_{20,i,j,k,l}, \hat{W}_{21,i,j,k,l}, \hat{W}_{22,i,j,k,l}, \hat{W}_{23,i,j,k,l}, \hat{W}_{24,i,j,k,l}, \hat{W}_{25,i,j,k,l}, \hat{W}_{26,i,j,k,l}, \hat{W}_{27,i,j,k,l}, \hat{W}_{28,i,j,k,l}, \hat{W}_{29,i,j,k,l}, \hat{W}_{30,i,j,k,l}, \hat{W}_{31,i,j,k,l}, \hat{W}_{32,i,j,k,l}, \hat{W}_{33,i,j,k,l}, \hat{W}_{34,i,j,k,l}, \hat{W}_{35,i,j,k,l}, \hat{W}_{36,i,j,k,l}, \hat{W}_{37,i,j,k,l}, \hat{W}_{38,i,j,k,l}, \hat{W}_{39,i,j,k,l}, \hat{W}_{40,i,j,k,l}, \hat{W}_{41,i,j,k,l}, \hat{W}_{42,i,j,k,l}, \hat{W}_{43,i,j,k,l}, \hat{W}_{44,i,j,k,l}, \hat{W}_{45,i,j,k,l}, \hat{W}_{46,i,j,k,l}, \hat{W}_{47,i,j,k,l}, \hat{W}_{48,i,j,k,l}, \hat{W}_{49,i,j,k,l}, \hat{W}_{50,i,j,k,l}, \hat{W}_{51,i,j,k,l}, \hat{W}_{52,i,j,k,l}, \hat{W}_{53,i,j,k,l}, \hat{W}_{54,i,j,k,l}, \hat{W}_{55,i,j,k,l}, \hat{W}_{56,i,j,k,l}, \hat{W}_{57,i,j,k,l}, \hat{W}_{58,i,j,k,l}, \hat{W}_{59,i,j,k,l}, \hat{W}_{60,i,j,k,l}, \hat{W}_{61,i,j,k,l}, \hat{W}_{62,i,j,k,l}, \hat{W}_{63,i,j,k,l}, \hat{W}_{64,i,j,k,l}, \hat{W}_{65,i,j,k,l}, \hat{W}_{66,i,j,k,l}, \hat{W}_{67,i,j,k,l}, \hat{W}_{68,i,j,k,l}, \hat{W}_{69,i,j,k,l}, \hat{W}_{70,i,j,k,l}, \hat{W}_{71,i,j,k,l}, \hat{W}_{72,i,j,k,l}, \hat{W}_{73,i,j,k,l}, \hat{W}_{74,i,j,k,l}, \hat{W}_{75,i,j,k,l}, \hat{W}_{76,i,j,k,l}, \hat{W}_{77,i,j,k,l}, \hat{W}_{78,i,j,k,l}, \hat{W}_{79,i,j,k,l}, \hat{W}_{80,i,j,k,l}, \hat{W}_{81,i,j,k,l}, \hat{W}_{82,i,j,k,l}, \hat{W}_{83,i,j,k,l}, \hat{W}_{84,i,j,k,l}, \hat{W}_{85,i,j,k,l}, \hat{W}_{86,i,j,k,l}, \hat{W}_{87,i,j,k,l}, \hat{W}_{88,i,j,k,l}, \hat{W}_{89,i,j,k,l}, \hat{W}_{90,i,j,k,l}, \hat{W}_{91,i,j,k,l}, \hat{W}_{92,i,j,k,l}, \hat{W}_{93,i,j,k,l}, \hat{W}_{94,i,j,k,l}, \hat{W}_{95,i,j,k,l}, \hat{W}_{96,i,j,k,l}, \hat{W}_{97,i,j,k,l}, \hat{W}_{98,i,j,k,l}, \hat{W}_{99,i,j,k,l}$$

(6.9)

where

$$\Phi_0 = \sum_{i=1}^n loc(x_{i0}, y_{i0}, z_{i0}), \dots loc(x_{n0}, y_{n0}, z_{n0})$$

$$\hat{W}_0 = [m_{10}, m_{20}, \dots m_{n0}]$$

$$\begin{aligned} \diamond_1 &= rect(x_{11}, y_{11}, x_{21}, y_{21}, z_{dst1}) \quad \text{and} \quad \hat{W}_1 = [m_{11}, z_{start1}] \\ \diamond_2 &= line(x_{12}, y_{12}, x_{22}, y_{22}, z_{dst2}) \quad \text{and} \quad \hat{W}_1 = [m_{12}, z_{start2}] \\ \diamond_3 &= rect(x_{13}, y_{13}, x_{23}, y_{23}, z_{dst3}) \quad \text{and} \quad \hat{W}_1 = [m_{13}, z_{start3}] \end{aligned}$$

Regarding new plans contributed via assistance, demonstrated primitive actions are encoded as shown in Eq. (6.10). In the GUI console, the operator demonstrates an action by first selecting it from the provided options then drawing the

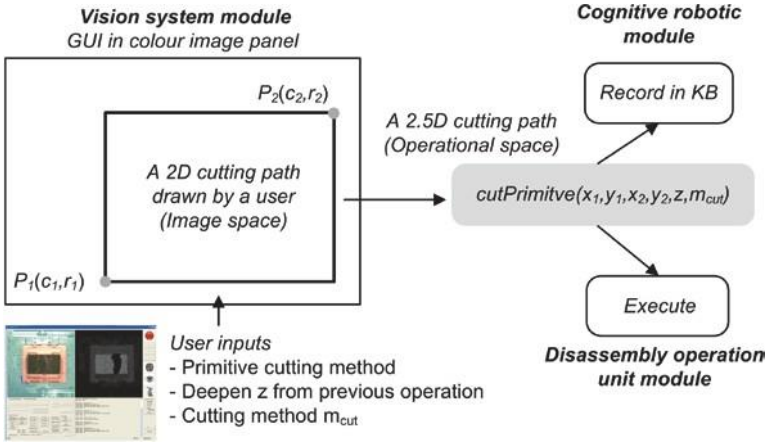


Fig. 6.11 User's demonstrated primitive cutting operation in GUI

cutting path on the 2D colour image in the GUI console (see Appendix B). This specified cutting path is then converted to a 3D path with the aid of the vision sys- tem (see Fig. 6.11).

$$plan_{add-on} = [cutContour(x_{11}, y_{11}, x_{21}, y_{21}, Z_{dst1}, m_{11}), \dots] \quad (6.10)$$

Other exogenous actions (e.g. *skipComponent*, *newComponentLocation*, etc) allow the operator to resolve inaccuracies in the system's original belief. These actions implicitly change the corresponding fact in the planning level KB and do not alter facts in the operation level.

Once a model has been learned in the KB, regardless of applied revision strat- egy, faster performance is achieved by reducing the number of operation cycles using larger cutting steps. Since the final cutting destination is known, the agent can simply approach this desired depth without frequently assessing the state change (a time consuming process).⁵ Significant time is reduced by assessing state change only once, after the final operation cycle at end of the operation plan.

For revision, the simple approach explained in Sect. 5.5.4 is implemented. The order of application of the operation plans is designed in regard to the impact on the components and success removal rate. The

cutting screw operation is first executed since it is expected to cause minimal damage to the component. Afterwards, higher impact operations corresponding to different offsets from the edge of the component are used. The result from this revision strategy will be clearly shown in Sect. 6.8.4.

⁵To assess state change, the action *flipTable* is executed in addition to visual detection. This physical operation takes around 8.5 s.

6.6 Vision System Module

The VSM addresses the uncertainties associated with the quality and quantity of the components. In particular, the vision system is designed to handle variations in (a) physical appearance in the components, (b) quantity of the component, and

(c) location of the components. As the system is not aimed for component reuse, it is sufficient and more robust to simply classify components by type. However, components of the same type typically appear different in each model, even within the same product family. The system must be capable of making the classifications despite of these variations. The CRA requests information using a sensing action. Subsequently, the vision system attempts to recognise the specified component and determines the number and location of each component detected. The uncertainties are suppressed by processing the raw detection outcome into abstract information. This information is sent back to the CRA via fluents for further decision-making. Sensing actions and associated fluents are listed in Appendix A.

6.6.1 Design and Functions

This module needs to perform the detection of the main and the connected components. The detection process can be described as two processes, (a) recognition and (b) localisation, which are used to address all relevant uncertainties. Moreover, the vision system is used to determine a transition of the disassembly state, facilitating the cognitive robotics task of execution monitoring.

Hardware Setup and Coordinate Mapping

The disassembly process is performed on the product in 3D space. Two cameras,

i.e. a colour camera and a depth camera, are used to obtain this information (see detail in Sect. 4.2.3). This depth camera also satisfies the low-cost constraint in contrast to the other 3D imaging technologies.

These two cameras are mounted stationary above the flipping table where the whole LCD screen can be seen from the back side. According to this setup, calibration is needed in three areas (a) colour, (b) image quality, and (c) coordinate mapping. All calibration methods are implemented in the low-level control layer. First, accurate colour perception is necessary since colour-based recognition techniques are used for some components. In this case, colour balance correction is performed [9]. The second issue, regarding the image quality, deals with noise and inaccuracy of sensed information. This problem is greater in the depth image, which is sensitive to edges and highly reflective surfaces and results in inaccurate data sensed from these areas. In addition, due to the depth resolution of approximately 3–4 mm, position error due to noise is significant. Therefore, noise filtering and interpolation techniques are needed to improve image quality.

Finally, coordinate mapping is crucial for the entire process due to the requirement for localisation of the component and positioning of the related cutting

operations. Therefore, the relation between the product coordinate frame {P} and the image space {S} need to be determined. In this case, the calibration process can be simplified by mechanical adjustment. The two cameras are aligned parallel to each other such that the lines of sight are normal to the fixture plate. As a result, the number of variables reduces significantly. Eventually, Eq. (4.11) is used for localisation, where position in the operational space (x_p, y_p, z_p) of the object relative to {P} is a function of the image space variables (c, r, z_F) . The c and r are obtained from the colour camera and z_F from the depth camera (see detailed explanation is in Sect. 4.2.4).

$$\mathbf{P}_{object}^P(x_p, y_p, z_p) = \mathbf{H}(c, r, z_F) \tag{6.11}$$

Detection of Components

According to the structure analysis of LCD screens and the actual disassembly process, 5 main components and 1 connective component need to be detected: (a) back cover, (b) PCB cover, (c) PCBs, (d) carrier, (e) LCD module, and (f) screws.⁶ The component types are recognised using rules based on *common features* as summarised in Fig. 6.4 (see detail of common features in Sect. 4.4.2). The information from both types of images is taken into account. In this case, the common features are determined from the 41 initial samples manufactured between years 1999 and 2011.

The presented values are subject to change when applied to newer models (Table 6.2). The *back cover* is the outermost part made of plastic that covers all other com-

ponents within the product. The size and aspect ratio vary according to the size of the LCD panel, and in this case, is limited to 15"–19" with a maximum aspect ratio of 2:1. The height of the back cover can be used to reject other small irrelevant objects, e.g. supporting fixture elements. From the samples, the thickness is 10–70 mm from the front of LCD screen. However, for newer design the LCD screen tends to be thinner (see detection example in Fig. 6.13a).

PCB cover is a metal box that covers PCBs. The height of the box used to identify this component. This component can be a separate component (Type-I) or integrated as part of the carrier (Type-II). The classification

between subclasses can be preliminarily done by colour criteria. The Type-Ia cover is typically made of reflective sheet metal, approximately 0.5–1 mm thick. Type-Ib and Type-II covers typically consist of a matte gray metal that is 1–3 mm thick due to the strength required for the carrier. The colour range for the reflective metal is $H(35^\circ, 130^\circ)$ and $S(12, 35)$. The colour range for the matte gray metal is $H(73^\circ, 135^\circ)$ and $S(10, 27)$ (see detection example in Fig. 6.13c).

PCBs are usually perfectly-rectangular plates found in two colours: green and yellow. The colour range criteria are, respectively, $H \in (70^\circ, 200^\circ)$ and $S \in (35, 80)$ for green, and $H \in (20^\circ, 70^\circ)$ and $S \in (35, 90)$ for yellow. Due to a number of

⁶ Note: the front cover is excluded from the list since it is expected to be detached according to operations in earlier states. Other types of connections, i.e. cables, snap-fits, and plastic rivet, are also expected to be disestablished by the operation plan. As a result, visual detection of these components is unnecessary.

Table 6.2 Common features for the detection of the components in LCD screens

Component	Component type (m—main and c—connective)	Recognition	Localisation	Common features							
				Colour image						Depth	
										Height	Surface roughness
				2D geometry			Colour range	Connected region (blob)	Haar-like feature	Height	Surface roughness
Size	Shape	Aspect ratio									

Back cover	m	•	•	•	•	•				•	
PCB cover	m	•	•	•	•		•	•		•	
PCBs	m	•	•	•	•		•	•			
Carrier	m	•	•		•		•	•			
LCD module	m	•					•	•		•	•
Screws	c	•	•	•					•		

embedded components mounted on the boards, the number of the connected pixels belonging to an area having valid a colour is taken into account to accurately determine the boundary of this component. Blob detection [10] is used to detect regions in which the area of the bounding rectangle lies between 1,000–40,000 mm². In addition, since multiple PCBs may be found in one disassembly state, an algorithm further separates the detected area into isolated PCBs (see detection example in Fig. 6.13e).

The *carrier* is a core structural component of LCD screens, to which other main components are mounted. It is designed to be high strength; a thick matte gray metal is always used. The colour criterion is used to identify this. The size of this component is large enough to support the LCD module. It is determined by a connected region of the specified colour, covering 45–90% of the LCD screen (see detection example in Fig. 6.13g).

The *LCD module* is the component that performs the main functionality of the product, and appears as a perfect rectangle covering more than 90 % of the entire area of the product. The back of LCD module is flat and usually consists of shiny sheet metal which can be identified using a colour criterion and/or surface roughness calculated from the depth image. The LCD module is then detected by determining the appropriate connected region satisfying the surface criteria. Detection of the LCD module is important because after this is achieved the CRA will acknowledge that the goal state has been reached (see detection example in Fig. 6.13i).

Screws are the only connective component to be detected in this system. In comparison with other fastener types, screws are rigid and well-defined, simplifying the visual detection task. In addition, most other connectors are expected to be automatically disestablished by the cutting operations. From the top view, the heads of M2–M3 flat button Phillips screws are detected using a classifier based on Haar-like features [11]. The classifier has been trained by approximately 800 positive images of screw's head and 7,000 negative images. A high detection rate is achieved, however alongside a significant number of false negatives; the size criterion is used in order to filter out some false negative detected objects. The location of the attached main components is also taken into consideration in order to limit the area to be searched. For example, in a process of removing a PCB, the region of

interest (ROI) will be set to cover the PCB and the screw detection will be performed only in this area.

To communicate abstract localisation information, a main component is represented by a minimal bounding rectangle (MBR) and a screw by a point in 3D operational space. The fluents are *compLocation* $box(x_1, y_1, x_2, y_2, z_1, z_2)$ and *screwLocation* $loc(x, y, z)$, respectively. The position in the operation space is obtained from Eq. (4.11). In case that multiple instances of a type of component is detected, the locations are represented in a list, e.g. *pcbLocation* $[box(x_{11}, y_{11}, x_{21}, y_{21}, z_{11}, z_{21}), box(x_{12}, y_{12}, x_{22}, y_{22}, z_{12}, z_{22}), \dots, box(x_{1n}, y_{1n}, x_{2n}, y_{2n}, z_{1n}, z_{2n})]$.

Detection of State Change and Other Utilities

The *detection of state change* is performed according to the algorithm presented in Sect. 4.4.4. The change of disassembly state is measured from the differences and

similarities between the original condition and the current condition of a disassembly state. The colour and the depth images are considered over the ROI. However, the depth image is more critical, as it represents the physical geometry of the component. The state change is assessed using Eq. (4.23). A depth difference threshold (Φ_{depth}) of 50 % and a colour difference threshold (Φ_{colour}) of 75 % are selected. Experimentally, this was found to successfully recognise state change in 95 % of the samples.

Other functions of the vision system include: (a) recognition of the product model and (b) height measurement z_F of features.

The product model detector is used to determine whether the current sample constitutes a previously-seen (known) model. This recognition is performed using SURF [12] on the colour image in order to match the keypoints of the current model to the existing database. It was experimentally found that a match percentage of greater than 15 % implies that two samples are identical.

The measured height z_F is used as the starting point of cutting operation for the known models (see Fig. 6.12), and ensures that the cutting operation starts from the top surface, preventing force overload on the cutting tool. Force overloading cut can occur when the cutting tool tries to quickly approach the target by cutting through thick layers of materials. This inappropriate movement will cause the cutting tool to be blunt, worn, or broken. Step cutting shown in Fig. 6.12 is one of the strategies to prevent this circumstance (Fig. 6.13).

6.6.2 Performance

The detectors were tested on 37 different models to validate the selected rules and common features. The performance in the recognition and localisation processes based on the non-damaged components is shown in Table 6.3. The recognition of

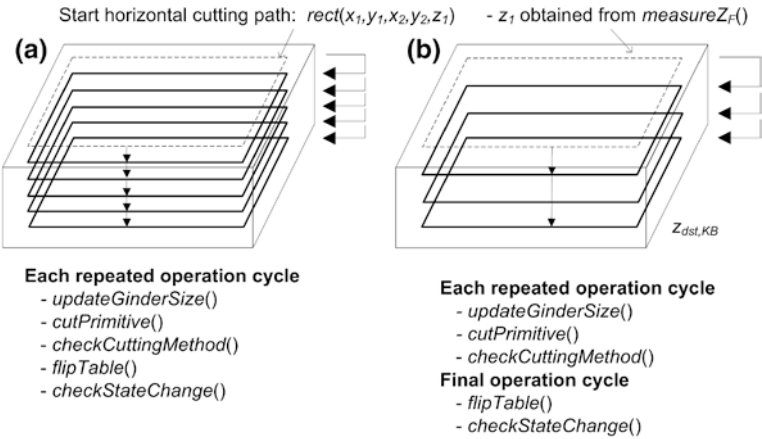


Fig. 6.12 Incremental cutting operations. a Unknown model. b Known model

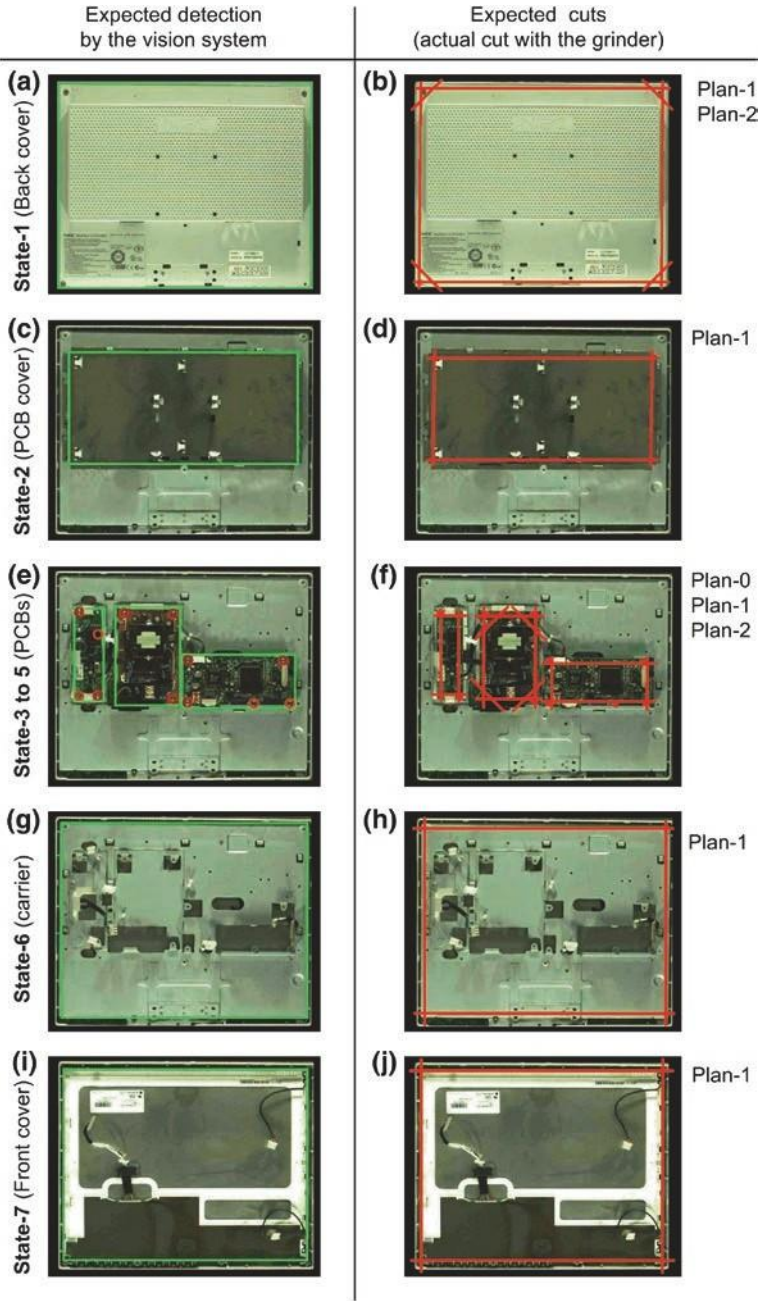


Fig. 6.13 Samples of components in LCD screens

Table 6.3 Summary of operation plans for removing the main components

Main component	Operation plan			
	Plan No.	Primitive cutting operation	Inner offset from the border (mm)	Intention or connection to be disestablished
Back cover (c1)	1	cutContour	5	Snap-fits around border area
	2	cutCorner	20	Screws at the corners
	3	cutContour	12	Screws around border
	H	*	*	– Press-fits at with the PCB cover – Uncut screws in the middle area – Correct the inaccuracy
PCB cover (c2)	1	cutContour	5–10	Remove the top plate
	2	cutContour	5 (outer)	Hidden cables underneath the carrier
	H	*	*	– Uncut hidden cables – Correct the inaccuracy
PCB (c3)	0	cutScrew	n/a	Screws at particular locations
	1	cutContour	5	External ports, cables, screws
	2	cutCorner	20	Screws at the corners
	3	cutContour	10	Screws around border area
	H	*	*	– Uncut connectors in the middle area – Correct the inaccuracy
Carrier (c4)	1	cutContour	5	Screws, snap-fits around border area
	H	*	*	– Correct the inaccuracy
Front cover (c6)	1	cutContour	5 (outer)	Snap-fits around border area
	H	*	*	– Correct the inaccuracy

Note * Based on the decision of the operator; *H* human assistance; Operation plan used in cognitive robotics denotes as op((MainComponent),(plan No.)), e.g. op(c1,1) represents operation plan-1 for treating a back cover

components generally performed with around 80 % accuracy, with an exception of screws and the LCD module, which had an accuracy of 65 and

50 % respectively. Regarding localisation precision, the position of detected components was determined with a Root Mean Square (RMS) error of within 6 mm. State change can be assessed with 95 % accuracy. In general, the error of the detection algorithms results from the depth image due to the Kinect sensor's sensitivity to reflective surfaces and edges. In addition, the colour range used to determine connected regions can be slightly inaccurate due to the reflection of colour from other nearby components.

In conclusion, the VSM is capable of detecting the components effectively. Errors resulting from the uncertainties in the component and non-detectable objects are expected to be further resolved by the operation plans and assistance.

6.7 Disassembly Operation Unit Module

The DOM addresses the problem in the disassembly operation due to the physical uncertainties of (a) EOL condition and (b) other non-detectable objects. These can be considered as missing pieces of information, which the sensing module alone is incapable of obtaining. Variations in EOL condition manifests itself as minor physical changes in the product returned, e.g. in the case of damaged parts. Problems caused by non-detectable objects occur in two primary categories: undetected connected components, and structures obstructing robot movement during the disassembly operation. The CRA can only control the process appropriately if such information can be perceived. In order to fulfil this functional requirement, the disassembly operation units need to appropriately respond to these uncertainties using hardware and return relevant information to the CRA.

In the case study, the system is design to be simultaneously low-cost, flexible and robust to uncertainties. Therefore, the disassembly rig, including a robot and disassembly tools, are designed to perform (semi-)destructive disassembly. In addition, operation plans based on common connection locations is used, designed to have a high chance of applicability and success even on unseen models.

6.7.1 Hardware Design and Functions

As earlier described in Sect. 3.5, the disassembly rig consists of three operation units: (1) a robot arm, (2) a flipping table, and (3) an angle grinder. The robot arm is a stand-alone module, used as the main element of the DOM. The main task is to control the disassembly tool to perform the (semi-)destructive disassembly operations according to the primitive action requested by the CRA. The flipping table is used as a device for

handling an LCD screen while the grinder is used as a disassembly tool (see Fig. 3.17). The CRA can simply activate these devices with the primitive actions *flipTable* and *switchGrinderOn/Off*.

The control architecture is more complicated for the robot arm module. In the *low-level control* layer, the robot controller controls the motion of the robot arm at the motor-sensor level to an accuracy of within 1.0 mm. The part directing the operation is in the *mid-level control layer*. The parameterised operation procedures that correspond to the primitive actions are programmed in this part. The operation procedures are *cutLine*, *cutContour*, *cutCorner*, and *cutScrew*. The supplied parameters are the cutting location and cutting method (i.e. feed speed and tool orientation).

The robot controller also sends feedback to the CRA, informing it of the parameters leading to successful execution. The operation succeeds if the entire cutting path can be cut without any collisions. At the end of each operation, the CRA requests *senseCuttingMethod*, at which the robot controller returns the cutting method *m*. This was earlier presented in Eq. (6.5). *m* is defined in

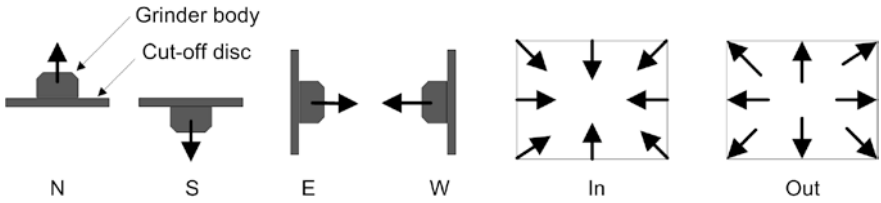


Fig. 6.14 Notation of tool's orientation seen from top view

Eq. (6.12). $M_{success}$ is encoded with the values '1'–'8' representing the tool orientation and feed speed used in the successful operation, whereas M_{fail} is represented with '0'. The successful cutting method is subsequently recorded in the learning process. The notation of tool orientation is shown in Fig. 6.14. The orientations {N, S, W, E} are used with *cutLine*, whereas only {In, Out} are used with *cutContour* and *cutCorners*.

$$m = \begin{matrix} M_{success} \\ M_{fail} \end{matrix} \in \Theta_{tool} \times S_{feel} \mid (\Theta_{tool} \in \{N, S, W, E\} \vee \Theta_{tool} \in \{In, Out\}) \wedge S_{feel} \in \{L, H\} \quad (6.12)$$

6.7.2 Function s

This section presents the *general operation plans*, procedures in the CRA containing sequences of parameterised primitive actions applicable to general (and previously unseen) models of LCD screens. The operation plans are designed based on strategies to disestablish the connections of each main component. In the treatment of a main component, the available operation plans are called one by one in an order specified by the CRA. In formulating the operation plans, three main factors are taken into account: (a) *common location of the connectors*, (b) *level of destruction*, and (c) *success rate of the removal process*.

The common location of fasteners is specific to each type of component. In general, connectors typically lie nearby the border of each main component. Only a small amount is found in other locations. The general operation plans are used in the former case and human demonstration in the latter, respectively. For the general plan, the *border offset* is a significant factor in determining the cutting location. Appropriate parameters can be obtained statistically, from observing the common locations of connectors from samples of different models [13].

Regarding the level of destruction, *semi-destructive disassembly* is preferred, whereby only the connective components are damaged without or minimally damaging the main component. However, the success rate is lower since due to the possibility of misdetection of the connective components. For the *destructive disassembly*, the cutting operation is performed directly on the main components. Both the success rate and the level of destruction are higher in this case. In contrast to semi-destructive disassembly, it is possible here to disestablish unseen

connectors that lie underneath the component, and separate the main fraction of a component while leaving inseparable parts behind. This latter is the case when the majority of the back cover is detached by cutting it away from the snap-fits located in the border. The size of the successfully detached part can be maximised by selecting the minimal border offset that provides an acceptable success rate.

The common locations of the connectors for each type of component, and the corresponding operation plans, are explained as follows. The connections and dis-assembly processes are analysed in the perspective of the view and operations available to the system. Therefore, the potentially-erroneous outcome from the vision system is taken into account.⁷

Back Cover

The back cover is generally connected to the front cover or carrier with 8–10 snap-fits within 5 mm around the border area. In addition, 4 screws are commonly found close to the corners within 10–12 mm of the border, and 1–2 additional screws are occasionally found in the central area for additional support. In certain cases, hidden press-fits also occur in the central area supporting the PCB cover.

The operation plans consist of cutting at different offsets from the border using *cutContour*, then cutting at the corners to bypass potential corner screws. The operations cut into the component to a limited depth with the aim of detaching the majority of the material from the product. The common location of the connections and the respective operation plans are shown in Fig. 6.15.

PCB Cover

For Type-I LCD screens, the PCB cover is generally a box that attaches to the carrier with snap-fits and screws around the base (see Fig. 6.16a). These connectors are not needed in Type-II structures where the cover is part of the carrier (see Fig. 6.16b). In both cases, there are cables and external ports joining parts of the PCBs that lie inside the cover. These cables and ports sometimes need to be dismantled in order to remove the PCB cover.

For the operation plan, *cutContour* with different offsets is used to cut the top surface and at the base level. This operation is needed to distinguish between two structure types. Plan-1 (*op(c2,1)*) cuts through the top

surface of the PCB cover. In a Type-I structure, the top section of the cover is now detached, since there are no remaining connections (see Fig. 6.16a). In contrast, for Type-II, the cover remains connected to the rest of the product via the PCBs and the external ports (see Fig. 6.16b). As a result, the product structure type can be identified using the combination of an action and execution monitoring (see basic behaviour in Sect. 6.5.1). For Plan-2, the cut is done at the base level in order to disestablish the connections including the hidden cable underneath the carrier. The common location of the connections and the operation plans are shown in Fig. 6.17.

⁷ Semi-destructive disassembly for removing screws is not used for most main components due to the low screw detection accuracy. Destruction of screws is only considered in the case of PCBs where the screw detection rate is high.

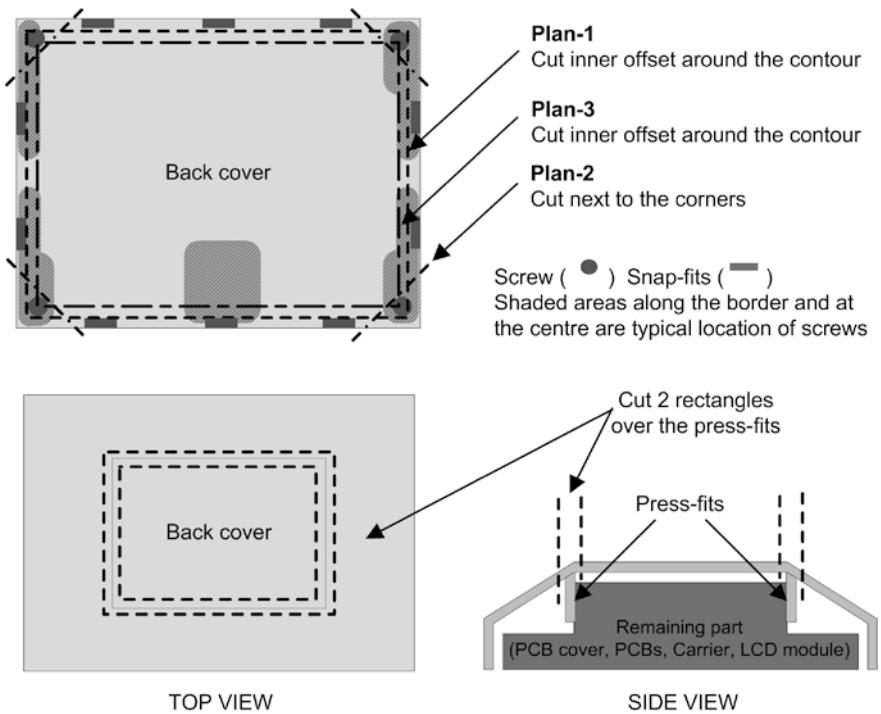


Fig. 6.15 Operation plans of the back cover

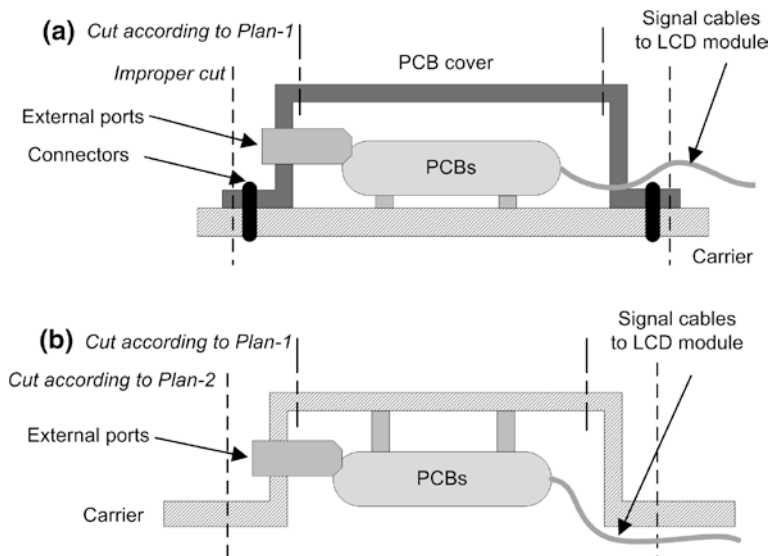


Fig. 6.16 Cross section of PCB covers in the different structure types. **a** Type-I. **b** Type-II

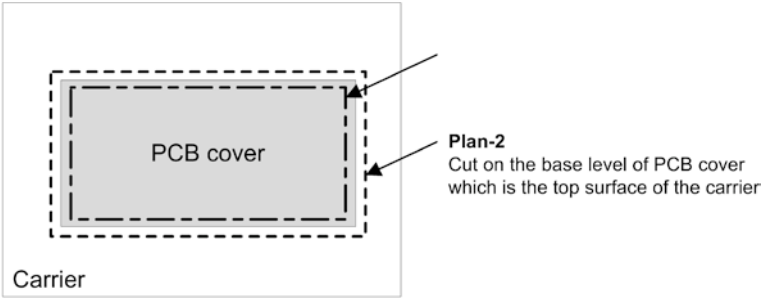


Fig. 6.17 Operation plan for PCB cover

PCBs

PCBs connect to the carrier via 4–5 screws within 10 mm of the component border. In some cases, 1–2 plastic rivets additionally support the central area of the board. Cables and input ports are also found around the border. The cables connect PCBs with each other and the LCD module. In general, the configuration is as follows: Power cables connect from a power supply board to a controller board and the CCFLs in the LCD module. Signal cables connect from the controller board to a switch board at the front panel, the power supply board, and another PCB that is part of the LCD module. The cables connecting to the LCD module are normally hidden underneath the carrier. The configuration of PCBs is different in Type-I and Type-II as shown in Fig. 6.18.

For the operation plan, both semi-destructive and destructive approaches are applied to this component. To minimise damage to the PCB, *cutScrew* is performed first, the screws on this component are easily detected and this operation causes minimal damage to the PCB. Afterwards, *cutContour* and *cutCorners* with various offsets are performed. The first cut occurs very close to the border in order to cut the cables and ports with minimal damage to the PCB. The second cuts the corners where screws are commonly located. Lastly, *cutContour* with a larger offset is performed. The common location of the connection and the operation plans are shown in Fig. 6.19.

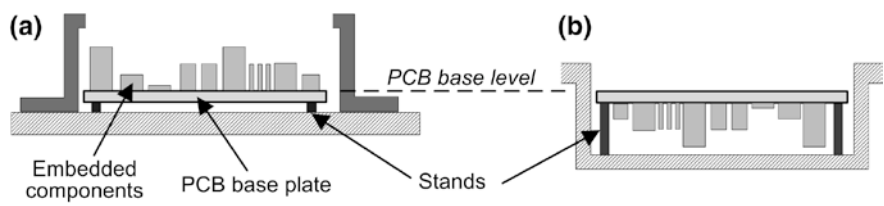


Fig. 6.18 Configuration of PCBs. **a** Type-I. **b** Type-II

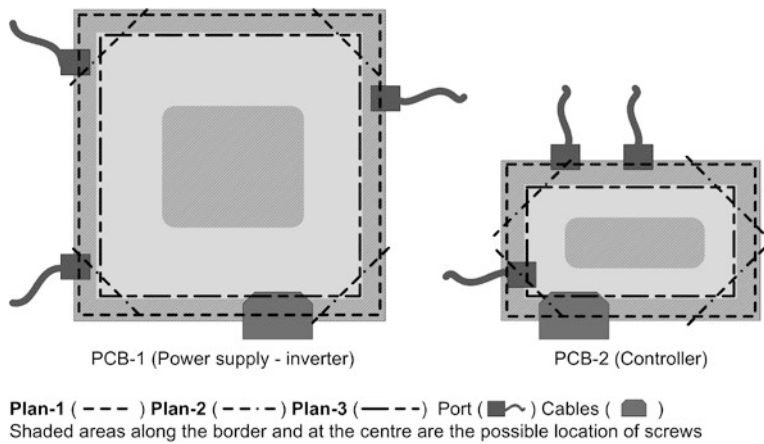


Fig. 6.19 Operation plans of the PCBs

Carrier

The carrier connects the PCBs to the PCB cover, and attaches to the LCD module via 6–8 snap-fits and 4–8 screws around the border. In most cases, the cables from the PCB are attached to the carrier via clips and screws, which also need to be disestablished. Furthermore, in the Type-II structure, where the carrier also covers the PCBs, input ports located around the covering area also need to be detached. However, this should have already occurred during removal of the PCB cover.

Regarding the operation plan, *cutContour* is performed, cutting near the border of the carrier to detach the majority of the component material. One of the challenges of this operation is to identify an appropriate depth of cut that avoids damaging the LCD module underneath. In this case, the preferred depth of cut is 2–3 mm measured from the carrier’s top surface. The common location of the connection and the operation plans are shown in Fig. 6.20.

LCD Module and Front Cover

The LCD module is typically connected to the carrier, as well as to the front cover via snap-fits around the border. At this state of the disassembly process, all connections are expected to have been disestablished by earlier operations. However, in case that the LCD module is still attached

to the front cover, an extra operation can be done to cut around the contour of the front cover, destroying all remaining connections. This is the final state of the disassembly process. At this stage, the LCD module can finally be separated.

In conclusion, general operation plans are used to disestablish connections in the treatment process of each main component. The plans are summarised in Table 6.4. The CRA will request each operation using primitive actions. The general plans are designed to be flexible and effective in detaching the main component, typically via a destructive approach. The strategies developed are

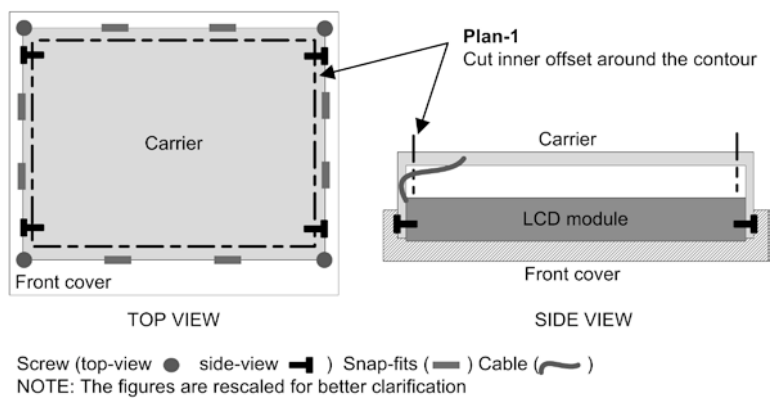


Fig. 6.20 Common location of the connections belonging to the carrier

Table 6.4 Success rate of the plans for removing main components

Main component	Rate of success to remove the main component (%)				
	General plans conducted autonomously				Human assistance
	Plan-0	Plan-1	Plan-2	Plan-3	
Back cover	n/a	0	0	12.5	100.00
PCB cover	n/a	46.67	33.33	n/a	100.00
PCB	0	0	0	17.65	100.00
Carrier	n/a	16.67	n/a	n/a	100.00
LCD module	n/a				

based on the common location of the components and corresponding connections as sensed by the vision system. Hence, the parameters, i.e. border offset values, implicitly account for errors in visual localisation. The uncertainties in regard to visually non-detectable connections and inaccuracy detection are also addressed.

6.8 Experiment

6.8.1 Process Flow

This section aims to demonstrate the operation of the system by describing the process flow relating to initial disassembly, learning and revision of the KB. The descriptions focus on key situations that emphasise the characteristics of the cognitive robotics approach. An example of the KB in regard to learning and revision process is shown, as well as the visual inputs and demonstrated actions. Images showing a typical disassembly operation are shown in Fig. [6.21](#).

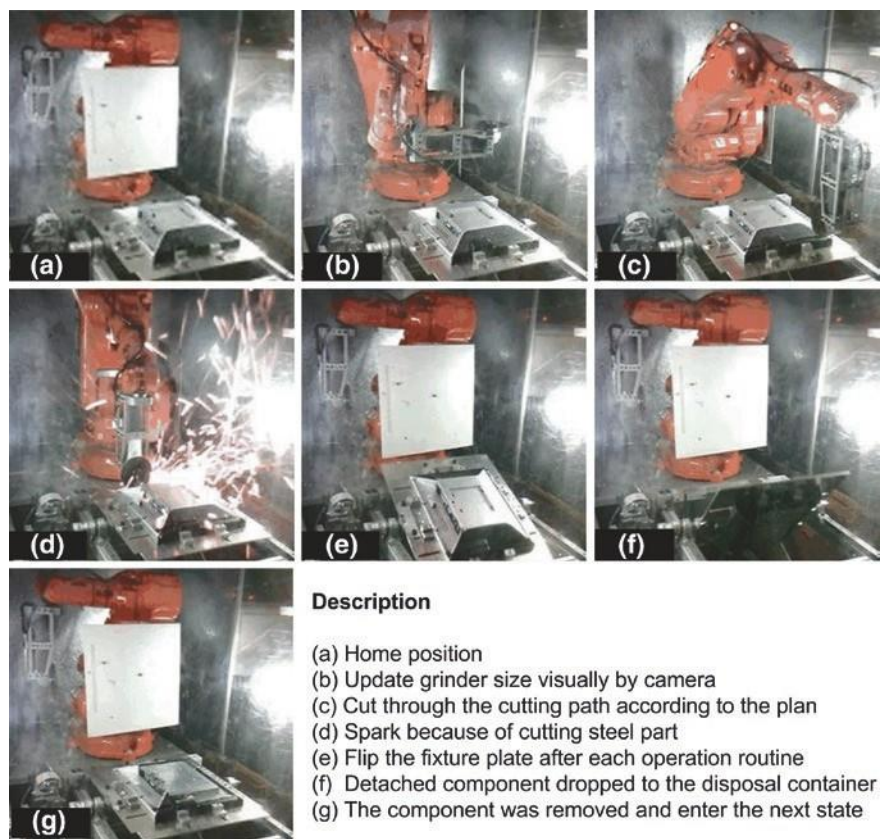


Fig. 6.21 Snapshots of the disassembly process captured from the test

Type-I Versus Type-II

Due to the variations in product structure, the disassembly process varies between Type-I and Type-II LCD screens. As shown in Fig. 6.10, a major difference occurs at the states representing the removal of PCB cover, PCBs and carrier. As a result, Type-I samples can be completely disassembled in a single run, whereas Type-II samples generally need to be loaded a second time into the rig to remove PCBs inaccessible in the original direction. An overview of this section of the disassembly process for both structure types is shown in Fig. 6.22.

For Type-I, the PCB cover is generally removed first, followed by the underneath PCBs. Considering the assembly direction, the PCBs are located above the carrier; the connections are easily seen, accessible, and can be disestablished from above (see Fig. 6.16a). The disassembly process is carried out until reaching the LCD module. On the other hand, for Type-II, the PCBs are generally connected to the carrier, which also covers the PCBs (see Fig. 6.16b). In this configuration the PCBs are inaccessible from above. After performing the operations that cut

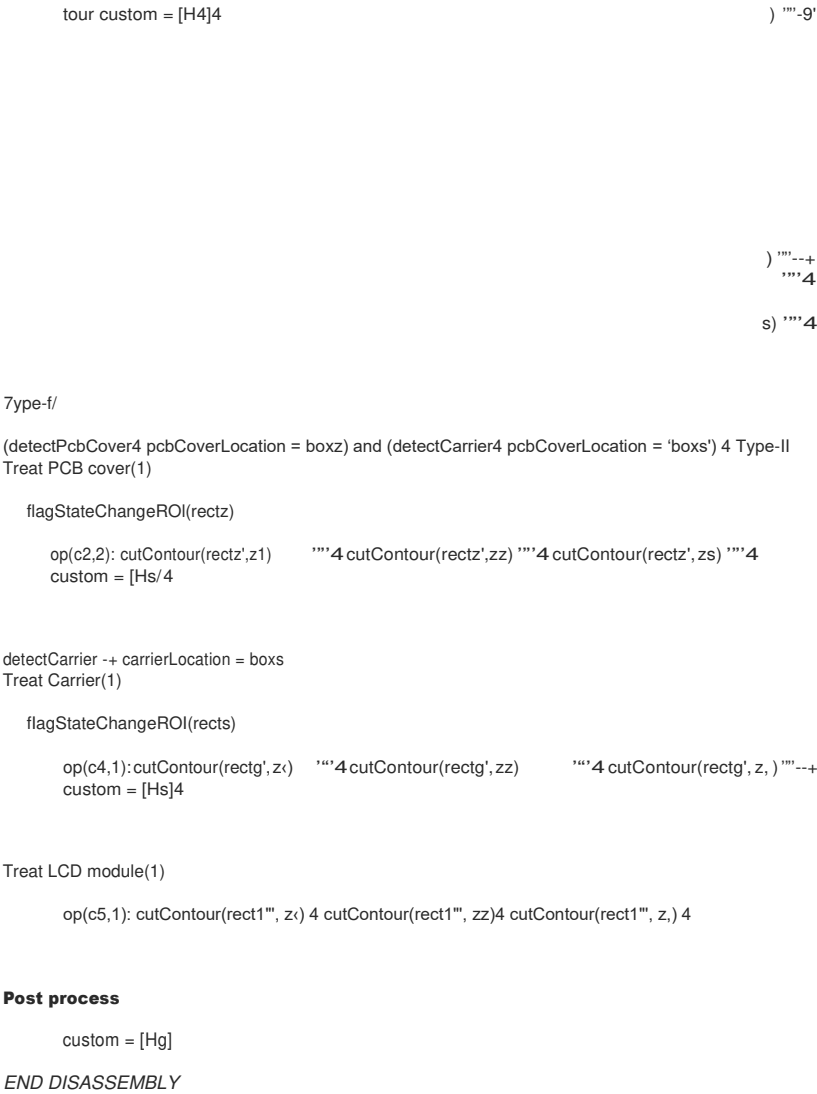


Fig. 6.22 Disassembly process overview of a Type-I and b Type-II

around this covering area ($op(c2,1)$), this part of the carrier is separated from the remaining product with the PCBs still attached. In order to further separate the PCB, this part must be re-loaded into the rig in the opposite direction, in a second disassembly run after completion of the original process.

Carrying out this extra process is inefficient due to time consumption, however usually unavoidable for an unknown model. In a subsequent appearance of this model, the CRA will recognise this structure and try to improve the process. The strategy to achieve disassembly in a single run is to cut the screws from the back of the carrier. However, it is difficult for the VSM to detect the screws from this side due to their small size. A one-off human demonstration is required for the localisation of these screws. The CRA learns the demonstrated locations and executes this step automatically in subsequent processes. As a result, the PCBs can be also detached from the carrier in Type-II screens without reloading, with the consequence that the screws remain attached to the PCBs.

Unknown Model

In this example, the disassembly process at the state of removing PCBs is emphasised (see Fig. 6.22 for at the highlighted section). At the beginning of the disassembly process, the CRA requests the detection of the product model via comparison with the existing database. The result shows that the model is unknown; therefore, this disassembly is conducted as a trial-and-error process. The operations for the detected main components are executed according to each disassembly state. In the state that PCBs are expected, the CRA sends the sensing actions *detectPCB*. In reply, the VSM sends a fluent containing a list of PCBs: *pcbLocation* [*box*($x_{11}, y_{11}, x_{21}, y_{21}, z_{11}, z_{21}$), *box*($x_{12}, y_{12}, x_{22}, y_{22}, z_{12}, z_{22}$)]. The disassembly operation for the first PCB is started. The knowledge learned by the CRA, with regard to the operation in component-level, is represented by the highlighted fluents in Fig. 6.22. It can be seen that only the critical locations are collected from the general operation plans; meanwhile, all demonstrated action sequences from assistance are recorded.

Upon first entering the state, the original condition of the PCB is noted for benchmarking. This focuses on a rectangular ROI belonging to the PCB represented by a fluent *rect*($x_{11}, y_{11}, x_{21}, y_{21}, z_{11}$).⁸ The action used for observing the original condition is *flagStateChangeROI*($x_{11}, y_{11}, x_{21}, y_{21}, z_{11}$).

Subsequently, semi-destructive disassembly is attempted via the disestablishment of screw connections. The CRA requests the sensing action *detectScrews*. Subsequently, the VSM returns a fluent containing the list of detected screws *screwLocation*[(x_{11}, x_{21}, z_{11}), ... (x_{n1}, x_{n1}, z_{n1})]. The remove screw operation *op(cs,0)* repeatedly performs the screw cutting operation at each location until the end of

⁸ *rect*($x_{11}, y_{11}, x_{21}, y_{21}, z_{11}$) is denoted by *rect*₁. The offset contour is denoted by *rect*'₁. The operation plan adjusts the cutting level to a particular depth *z*.

the list. The execution monitoring procedure is performed after each operation.⁹ If the state changes, the system proceeds to disassemble the next PCB. However, an example treatment for the first PCB demonstrates the case that the component always fails to be removed by any general plan.

The general operation plan for PCB $op(c3,1)$ uses the rectangle contour with offset as a cutting path and cutting method m . The cutting operation starts from the top surface at z_1 and incrementally cuts deeper in each operation cycle until z_3 where the boundary condition is reached. This operation is repeated for $op(c3,2)$ and $op(c3,3)$ with the respective offsets and cutting methods. If, after execution of all general operation plans, the PCB still fails to be removed, human demonstration provides a sequence of actions represented as *custom*. The complete information of demonstrated sequences is recorded, including the path x , y , z and a specific cutting method m . In this case, the demonstrated actions constitute cutting 2 lines, after which this component is detached.

Afterwards, the system proceeds to treat the second PCB. The example shows the successful removal of this PCB after the second trial of the general plan $op(c3,1)$. The disassembly proceeds to the next state for treating the next main component. The system goes through the subsequent states until reaching the goal component, the LCD module. The relevant knowledge about the product and process has been learned and is now ready to implement when the same model is seen again.

Known Model

This section focuses on the knowledge in the KB that has been learned from previous disassembly process. In Fig. 6.23, the learning and revision in KB and the operation at each state are shown. This example shows the entire disassembly process whereby 6 main components are separated. Initially, during the first time disassembling this model, the CRA obtains the knowledge in both the product-level (see Fig. 6.23a: I) and component-level (see Fig. 6.23a: II–VII). This process is denoted as Revision-1.

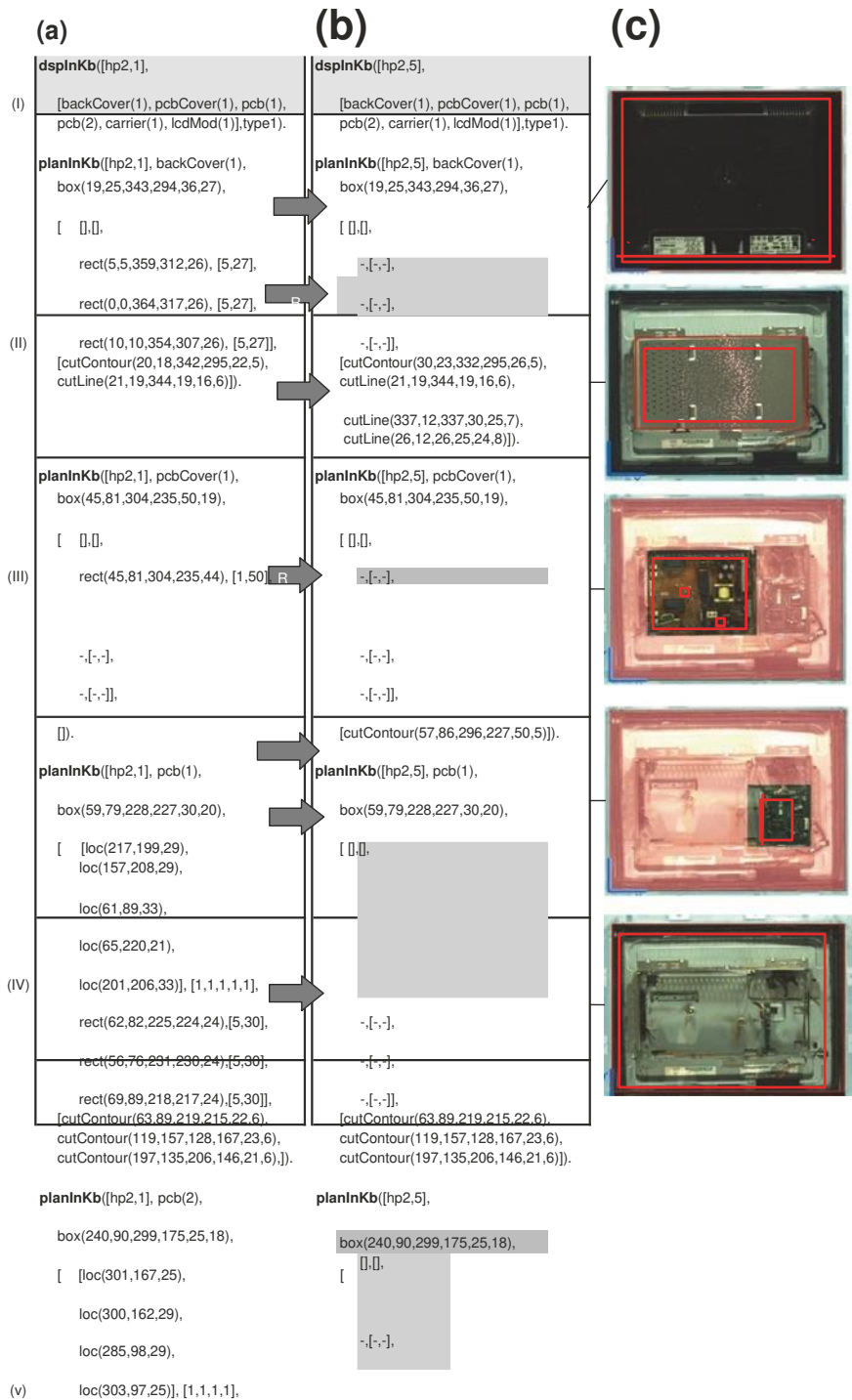
After this model has been disassembled multiple times, significant changes have been made to the KB (Revision-5, Fig. 6.23b). Modification of the KB in the revision process occurs in three general categories: (a) *retraction of a plan*,

(b) *learning of new component locations*, and (c) *learning of primitive cuts*.

Retraction occurs via removing the redundant general operation plans as described in Sect. 5.5.4, in effect making the process more efficient by reducing the time consumed. *New component locations* are demonstrated by the user to resolve errors in detection caused by the VSM. This process improves the accuracy of the cutting operation and the assessment of state change based on this location. *Additional primitive actions* can be demonstrated even when disassembly has previously been achieved in a given model. Extra cutting operations can resolve problems

⁹ Regarding notation, the operation sequence is connected with the transition symbol `<code>→`.

The code denotes the detail of the action executed in the transition state where $F = \text{flipTable}$, $S = \text{checkStateChange}$, and $M = \text{checkCuttingMethod}$.



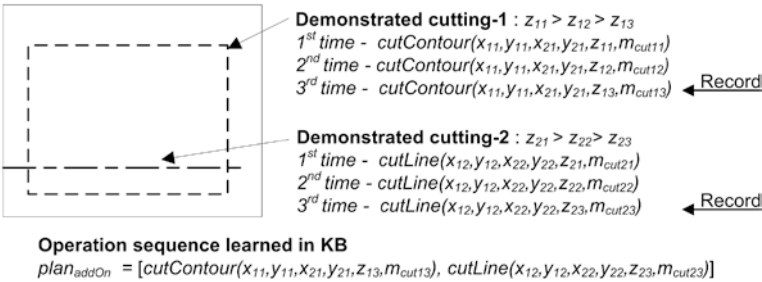


Fig. 6.24 Refining process for learning cutting operation for add-on plan

However, the operating time also slightly increases due to these additional actions. Through such revisions of the KB, the system is capable of improving its performance, hence reducing process time and the need for assistance.

In Fig. 6.23c, the xy-cutting paths of the human-demonstrated cutting operation at each state are shown. These operations are recorded in the added-on plan in the KB. The xy-cutting path is repeated multiple times at different depths since the human-operator does not know the final depth a priori. Trial-and-error is conducted, as directed by the user. Therefore, key primitive actions are only learned after the entire sequence of actions is executed (see Sect. 5.5.3). The learning of two demonstrated cuts (see Fig. 6.23c: I) is illustrated in Fig. 6.24. From 6 demonstrated cutting actions, only 2 primitive actions, representing one contour and one line cutting path, are recorded.

6.8.2 Key Performance Indices

This system was validated by performing the disassembly process. The experiments are designed to assess the performance of the system in two perspectives:

(a) flexibility and robustness and (b) performance improvement, with respect to learning and revision. The assessments are done based on 3 key performance indices (KPI):

- Completeness of disassembly
- Time consumption
- Need of assistance

Completeness of Disassembly

Completeness of disassembly is considered in two aspects: (a) process completeness and (b) material separation efficiency. First, the *process completeness* describes the capability of the system to separate all desired components. The disassembly outcome is indicated as complete if every main component has been detached from one another. It also relates to the types of product structure where the number of the process needed is different.

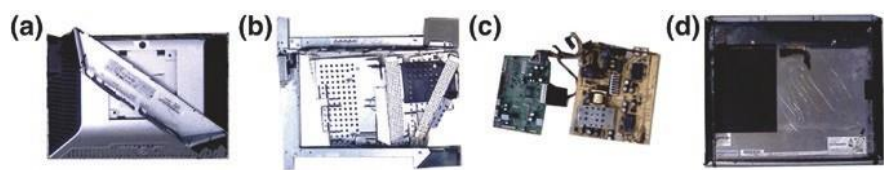


Fig. 6.25 Detached components classified by type of material. **a** Plastic. **b** Steel. **c** PCBs. **d** LCD module

The *material separation efficiency* describes the purity of the resultant separation of the various material categories. This metric is designed for (semi-)destructive disassembly where the output is in the form of lumps and scraps instead of complete parts. This is measured by *weight* in comparison with the *upstream* condition. Measurement is carried out after the disassembly process is completed.¹⁰ The weight comparison is observed according to four groups of material: (a) plastics, consisting of the back cover and front cover, (b) steels, consisting of the PCB cover and carrier, (c) PCBs, and (d) a compound component, the LCD module. An example of the detached parts is shown in Fig. 6.25.

Time Consumption

Time consumption is commonly used for assessing system efficiency due to its direct relationship with operation cost. Time consumption is measured from the beginning to the end of each process, ignoring the lead time for the manual loading and unloading of the samples.

Need for Assistance

The amount of assistance required in achieving a task is an indirect measurement of the level of autonomy, which is more difficult to measure. Because each demonstration is a discrete application of just one motion primitive, the number of demonstrations is an objective measure of the amount of assistance required. The number of demonstration instances is counted from the beginning to end of the disassembly of each LCD screen.

6.8.3 Performance Testing—Flexibility

of the System

The flexibility and robustness of the system describe its ability to handle the expected uncertainties and variations presented by previously unseen models. The behaviour of the CRA in handling unknown models is particularly relevant.

¹⁰ The KPIs are measured at the end of the first run for *Type-I* and at the end of second run for *Type-II* structures. A constant penalty of 5 min is given when the PCBs-carrier connected part requires reloading in Type-II screens.

In order to assess flexibility, the results of the first-time disassembly process of a broad selection of LCD screen models¹¹ were collated. The overall system performance is affected by the performance of each operating module, summarised as follows:

Vision System

Performance in recognition and localisation is slightly reduced from the ideal case (see Table 6.3) due to the damage incurred on the main components from the (semi-)destructive disassembly method. The cutting operations occasionally destroy the common features necessary for recognising the components. As a result, recognition accuracy was generally reduced by 5–10 % in comparison with the ideal case. The localisation accuracy also changed within 1 mm from the ideal case. The accuracy of the state change detection remained unchanged.

General Disassembly Plans

The general disassembly plans were executed autonomously to assess their robustness in handling unknown models. In theory, the parameters are developed based on the assumption that the vision system can provide accurate detected locations. Therefore, due to the overall position error of maximum 6.5 mm for xy-cutting path¹² and approximately 3 mm for the depth, complete component removal was sometimes difficult to achieve automatically. However, the required level of accuracy of vision system is hard to be achieved, so that additional operations, e.g. assistance, will also need to compensate this error.

Failures are caused by (a) inaccurate xy-cutting path, (b) insufficient depth of cut, and (c) non-detectable connections. Regarding the depth of cut, it is in some cases intentional to predefine a narrow range of cutting depths in order to decrease the chance of erroneously cutting through multiple components. For example, if the back cover is cut too deep, the border of the carrier may also be cut at this state, resulting in the simultaneous detachment of the carrier, PCB cover, and PCBs, which are still connected to one another and need to be further disassembled in a second run. This decision results in a higher failure rate due to insufficient depth of cut.

It is obvious that the success rate of the PCB cover is quite high in comparison with other components. This can be attributed to the CRA's

execution of the plan with multiple cutting offsets. This variation of cutting path compensates for the localisation error from the vision system. The high success rate is necessary, as the outcome is used to determine the product structure. The trial process

¹¹ The samples consist of 24 different selected models from 12 manufacturers. The size varied from 15"–19", and the year of manufacture ranged over 1999–2011. With respect to the structure type, 15 monitors of Type-I and 9 monitors of Type-II were used. The Type-I sample was further classified into 8 of Type-Ia and 7 of Type-Ib.

¹² The *location accuracy* refers to the root mean square (RMS) error measured at the border. The error occurs on each side of the bounding box will be accumulated.

Table 6.5 Success rate of the plans for removing main components

Main component	Rate of success to remove the main component (%)				
	General plans conducted autonomously				Human assistance
	Plan-0	Plan-1	Plan-2	Plan-3	
Back cover	n/a	0	0	12.5	100.00
PCB cover	n/a	46.67	33.33	n/a	100.00
PCB	0	0	0	17.65	100.00
Carrier	n/a	16.67	n/a	n/a	100.00
LCD module	n/a				

costs extra operating time, however allows the system to learn the successful cutting path and implement a more efficient process in subsequent appearances of the model.

Overall, successful removal rates for the general plans were quite low but all components could be removed after assistance (see Table 6.5). Unsuccessful operation plans often indirectly contributed to the removal process by destroying the significant connections. The subsequent assistance allowed completion of the process.

Completeness of Disassembly

The system successfully identifies Type-I and Type-II via the proposed execution monitoring strategy (see Sect. 6.5.1). However, some misclassification occurs due to failure to remove the PCB cover part of the carrier. Approximately 70 % of all samples could be completed in the first run, with the remaining 30 % completed after the second run. The CRA directly learned the process for the first group. For the second group, the screw cutting strategy is taught in the subsequent revision and learned after that.

Regarding material output, around 97.36 % of the detached parts turned out as lump, with the remaining 2.64 % as scrap material. Considering the entire product, 98 % of the material was separated and collectible while other 2 % turned out to be small scraps and dust according to the cutting process. For each category, the efficiency of detaching plastic and the compound material (LCD module) were more than 93 % and the efficiency of PCB and steel were around 85 %. One of

the critical issues in disassembly of LCD screens is the risk of damaging the Cold Cathode Fluorescent Lamps (CCFL) lying within the LCD module. From this experiment, minor damage occurred to the LCD module in most cases due to the cutting operations for the carrier. However, due to the predefined constraints, there was no single case where the CCFL were damaged. Therefore, it can be concluded that no toxic substances were leaked from the CCFL.

Time Consumption

The duration of the process depends on the complexity and the size of the screen. On average, around 97 % of the total time consumed was due to physical

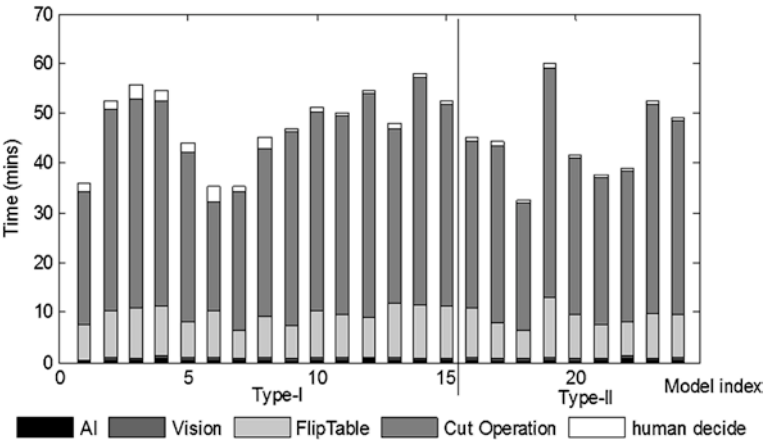


Fig. 6.26 Time consumption of the disassembly process by each operation

operations,¹³ 1.5 % due to visual sensing and artificial intelligence (AI),¹⁴ and

1.5 % due to data transfer activity. The autonomous trial-and-error process performed 67 % of the entire disassembly process while the remaining 33 % was directed by human demonstration.

The time required to complete an entire disassembly was on average 48 min, varying between 32 and 60 min according to the complexity of the samples. This process time is very long in comparison to traditional manual disassembly. However, the process for each individual model is expected to be further optimised via the learning and revision strategy. Firstly, the redundant operations for trial-and-error and learning will be reduced: the flipping table routine, reloading for second run and assistance. The time spent by each type of operation is clearly shown in Fig. 6.26 where most of the flipping table routine and cutting operations were redundant. At least an average of 11–16 min will be regained by removing these redundancies. Successful cutting methods will also be known, further reducing the time required testing cutting methods and recovering from collisions. Secondly, the redundant cutting operations will be retracted. Lastly, as the cutting depth is known, larger increments can be taken in order to make the process

¹³ The operation routines in a operating cycle:

- Flipping operation (8.45 s) and checking state change (2.45 s);
- Updating the grinder's size: movement and visual checking (4.19 s); and,
- Cutting operations (average 33.75 s). It varied from 5.09 to 186.91 s (95 % of the data were within 0–100 s) depending on the size of the cutting path and variation of the cutting method trial.

¹⁴ The activity of the AI includes the decision-making process of the human operator during demonstration steps: approximately 5 s per demonstrated action.

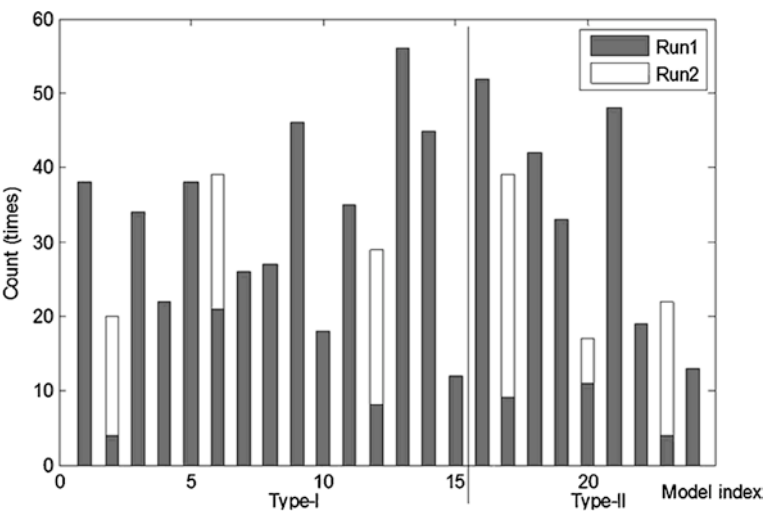


Fig. 6.27 Assistance count in the disassembly process

quicker. The time regained is dependent on the characteristics of the operations required. From results in the following section, the time consumption is finally reduced to around 24 min.

Need for Assistance

99.9 % of the assistance provided consisted of the demonstration of a primitive cutting path. Less than 0.1 % was for correcting an error in state or component detection. The overall amount of assistance for each model is shown in Fig. 6.27 where *Type-I* and *Type-II* are shown separately. The counts for *Type-I* and *Type-II* were similar, averaging 32 instances and ranging from 12 to 56. The majority of counts in *Type-II* occurred due to the demonstration of the strategy for cutting screws and/or during second runs.

Summary

In conclusion, the general operation plans were capable of accomplishing most of the required work in disassembling unknown models, however were unable to complete the process without assistance. Assistance resolved all remaining uncertainties, compensating for the limitations in

the perception of the system. Since component separation was achieved in every case, it can be concluded that the disassembly is achievable via the physical configuration and vertical cutting direction. However, the time required needs to be improved in order to compete with the traditional manual disassembly. Even though the operation was not fully automatic in the first-time disassembly of unknown models, the process was learned. The system is expected to increase in its degree of automation with more experience disassembling each given model.

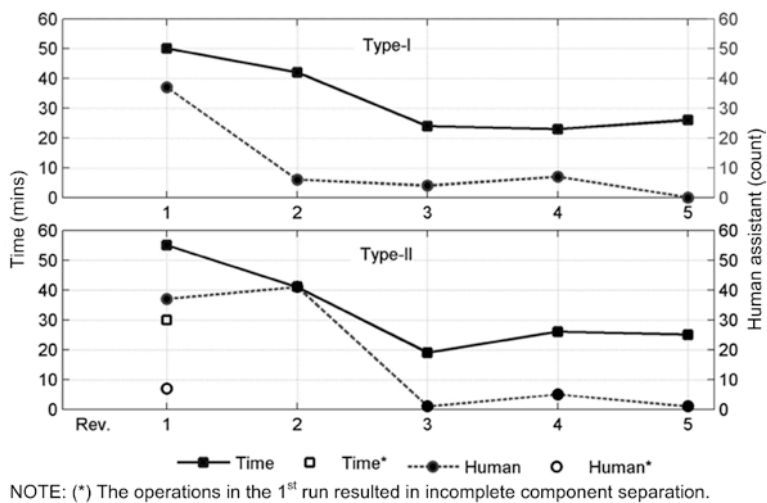


Fig. 6.28 Disassembly performance with respect to multiple revisions

6.8.4 Performance Testing—Learning and Revision

This experiment aims to assess the capability of the system both in learning and in revising the knowledge extracted from previously-executed disassembly processes. These behaviours are expected to improve the performance of the disassembly process. The experiment was executed on two models that embody the typical characteristics of the two structure types. Each model was disassembled five times, each time building upon the knowledge learned from previous disassembly processes, in order to see the trend of performance improvement due to learning and revision.

Results showed a dramatic improvement in the performance of the system during the first few revisions, with regard to time consumption and the amount of assistance required. System performance then levels out, remaining roughly constant with small fluctuations due to the variability in the vision system and the disassembly operation processes. These trends can be seen in Fig. 6.28. Detailed observations regarding each model are

explained in the following. Significant differences are observed between the two structure types.

Type-I Structure

The entire disassembly process is expected to be completed in a single run. This condition was satisfied from the first-time disassembly of this model. In the first disassembly (*Rev-1*), the overall time consumption was 47.9 min with 37 instances of assistance. Around 70 % of the assistance was given for deepening cuts that were insufficient to remove the components. The rest were used to locate non-detectable connections. The revisions were done later on until *Rev-5*. In comparison to the first disassembly, the time consumption decreased to about

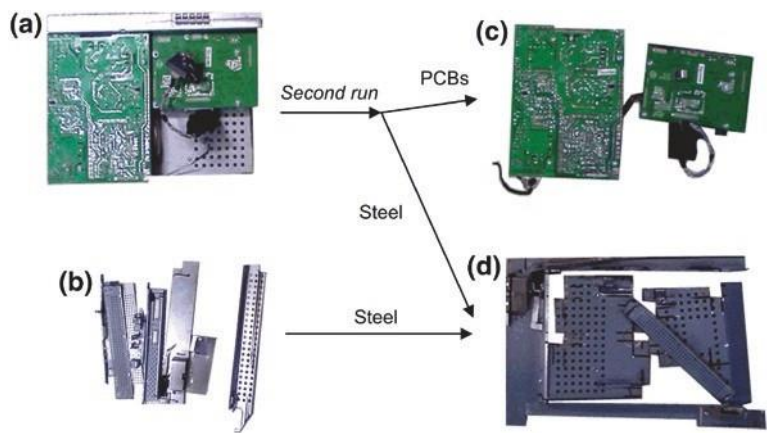


Fig. 6.29 Incompletely detached carrier and PCBs and the second run. **a** Incompletely detached PCB and carrier. **b** Cut-off steel part from carrier. **c** Completely detached PCBs. **d** Completely detached carrier

89 % in *Rev-2* and 52 % in *Rev-3*. Small fluctuations of within 5 % occurred between *Rev-3* and *Rev-5*. A similar trend occurred regarding assistance. The assistance count dropped dramatically to 16 % in *Rev-2*, maintained this level within 3 % fluctuation, and then finally dropped to 0 % at the *Rev-5*.

In summary, the performance increased as the process was revised. The final disassembly time was 25.7 min and no further assistance was needed in the final test. There was no significant difference in the material separation efficiency amongst the revisions, which remained at approximately 98 %.

Type-II Structure

Due to the configuration of the PCBs under the carrier, a second run was initially needed to complete disassembly (*Rev-1*). However, this problem was resolved in the second-time disassembly (*Rev-2*), where only a single run was needed, since the strategy for cutting screws from outside the PCB cover had been taught. The incompletely detached carrier and PCBs, and the material outcome after performing the strategy taught in the second run are shown in Fig. 6.29.

In the first revision, the total time consumption (sum of the first run, second run, and reload penalty) was 28.1 + 22.0 + 5.0 = 55.1 min. The total amount of assistance in the first and the second runs was 8 + 30 + 38 instances. As with the *Type-I* case, most assistance in *Rev-1* was given for deepening existing cuts. In *Rev-2*, the CRA realised that multiple runs were previously required—a suboptimal solution. Consequently, it asked for the user's assistance for employing the screw-cutting strategy. As a result, the time consumption decreased to 74.5 %. However, more counts of assistance were required, increasing to

107.9 % of the first run. The problem of cutting depth was almost completely resolved during *Rev-1*; more than 90 % of assistance in *Rev-2* were given to direct the screw-cutting strategy since the VSM is unable to detect the screws

in this circumstance. The knowledge obtained in this revision now satisfies the more efficient single-run requirement. In *Rev-3*, both values decreased dramatically, whereby time consumption decreased to 34.5 % and assistance to 2.6 % according to the first revision. Both values marginally increased by about 10 % in *Rev-4* due to process uncertainties (see following section) and maintained about this level until the end of *Rev-5*. At the final revision, the time consumption was 25.1 min (45.5 % of the first revision) and no assistance was required. The material separation efficiency was around 90 %, with no significant difference observed between the revisions. The disassembly outcome is shown Fig. 6.25.

Uncertainties in Process

The results showed a fluctuation in performance in some revisions, e.g. *Type-I Rev4-5* and *Type-II Rev3-4*. These fluctuations were caused by the uncertainties in vision system and disassembly operation.

Firstly, significant uncertainty in the vision system is caused by the inaccuracy of *measureZF*, used to locate the top surface for cutting. The precision of *measureZF* suggests that the measured top cutting level lies within approximately 3 mm from the actual top surface. As illustrated in Fig. 6.30, this inaccuracy may cause extra deepening iterations to be required. The CRA cuts the object from the start point to a destination depth z_{dst} learned from the previous operation. When the sensed top surface is too high, an extra cut is necessary (see Fig. 6.30b). This extra cut results in extra operating time. This variation is limited by the precision of the sensor.

Secondly, the disassembly operation involves an imperfectly-repeatable cut. An identical action, successful in detaching a component in a previous process, may fail in subsequent processes. The problems are caused by two factors of uncertainty: (a) non-uniform wear rate of the abrasive cutter, resulting in minor differences in cutting location, and (b) uncertain physical responses of materials to different process parameters. For instance, plastic may be melted when using a higher feed speed or larger cutting depth, causing components to become stuck together. Extra assistance is required to resolve these minor uncertainties, with extra time consumption as the consequence.

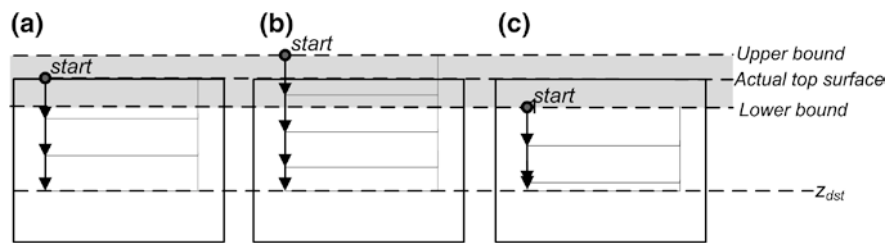


Fig. 6.30 Uncertainties due to the variation of the starting level for cutting. **a** Ideal case. **b** Too high measured z . **c** Too low measured z

Summary

In conclusion, due to the implementation of learning and revision, the performance of the system increases to a certain level as more disassembly processes are conducted on each model. This performance improvement is particularly seen with regard to the Type-II structure, which can be completed with a single run in the second and subsequent revisions. The time consumption of the disassembly process decreased dramatically in the first few revisions, to a final disassembly time of about 25 min, approximately 50 % of the time for the first run. After the revised KB becomes stable after the first few revisions, the process is expected to be performed autonomously without any or with only minimal assistance. Minor uncertainties in the process cause the performance to fluctuate within bounds. This is expected to be further suppressed by improving the accuracy of each individual operating module.

6.8.5 Conclusion and Future Improvement

These experiments validated the performance of the system, proving it to be flexible and robust to the variation and uncertainties found in the case-study EOL products. The autonomous process together with some assistance was able to complete the disassembly of every given sample. In addition, it has been proved that the performance of the system can be improved via the learning and revision strategy. After a few revisions, the system is able to complete the process within a shorter time and fully autonomously in good conditions. These cognitive abilities are important for the development of an automated disassembly system where flexibility, robustness, and cost minimisation are crucial. This system has successfully proved the principle that cognitive robotics is able to overcome the uncertainties and variations in the disassembly process.

However, in comparison to traditional manual disassembly, the process time consumption is still too high, rendering the system economically

infeasible. The time consumption needs to be further reduced to 6.2 min/screen through the prospective improvements which are as follows. Firstly, the disassembly time can be reduced to 9–11 min if the cutting tool can approach the destination depth in one operation cycle instead of incrementally deepening the cuts. More powerful and reliable hardware is required for this purpose. Secondly, all physical movements can further be optimised: e.g. feed speed, robot movement, flipping table, reducing the time required for each primitive action.

Moreover, the dependence on the human operator should be minimised, since this is directly associated with operation cost. The general plans should be improved to become more capable of carrying out the process autonomously. Even though the proposed concept of common location of the connectors is able to identify an appropriate border offset, visual localisation must be improved in order to achieve a higher success rate.

6.9 Conclusions

In this chapter, an automated disassembly system with cognitive abilities is described in detail. This system is designed for disassembling LCD screens, selected as the case study product. To improve economic feasibility, (semi-)destructive selective disassembly is implemented. For flexibility and robustness, the disassembly strategy, the principle of cognitive robotics and assistance together address the uncertainties and variations in products and process.

Firstly, a significant number of LCD screens were examined. The typical variations and uncertainties in the products were found in the different models of LCD screen. It was found that the product can be classified into 2 main structure types, *Type-I* and *Type-II*, according to the configuration of the main components. Typically, LCD screens consist of 6 main components and 4 connective components. This information was used in designing each operating module in the system.

The tasks, designed to handle the types of uncertainties expected in the process, are distributed between three operating modules. The CRA is the high-level planner that controls the disassembly process according to the desired cognitive behaviours. This addresses the variations in the product structure and associated disassembly plan. Using this methodology, the disassembly process can also be learned and improved later on. The second module, the VSM, recognises and locates the relevant main and connective components. This module takes into account the variations in physical appearance of components in different models. The third module, the DOM, performs the physical operations on the samples. The general operation plans are designed to address uncertainties in the cutting operations where the visual information and the related knowledge are not precise. Moreover, the human operator uses this module to perform the demonstrated actions, allowing the system to complete disassembly when the automatic process fails.

Finally, the operating modules not only perform their own specific tasks but also help in addressing the unresolved problems of other modules. For example, in case of the connective components occasionally misdetected by VSM, the general plans in DOM are performed multiple times according

to the CRA's decision to increase the success rate of component removal. In conclusion, as a whole, the system was able to experimentally demonstrate its ability to automate the disassembly of a significant number of different models of LCD screens.

References

1. Kernbaum S, Franke D, Seliger G (2009) Flat screen monitor disassembly and testing for remanufacturing. *Int J Sustain Manuf* 1(3):347–360
2. Ryan A, O'Donoghue L, Lewis H (2011) Characterising components of liquid crystal displays to facilitate disassembly. *J Clean Prod* 19(9–10):1066–1071

3. Parliament (2003) Directive 2002/96/EC of the European Parliament and of the council on waste electrical and electronic equipment (WEEE) of 27 Jan 2003
4. Franke C, Kernbaum S, Seliger G (2006) Remanufacturing of flat screen monitors. In: Brissaud D, Tichkiewitch S, Zwolinski P (ed) *Innovation in life cycle engineering and sustainable development*, pp 139–152
5. Kim HJ, Kernbaum S, Seliger G (2009) Emulation-based control of a disassembly system for LCD monitors. *Int J Adv Manuf Technol* 40(3–4):383–392
6. Uhlmann E, Friedrich T, Seliger G, Harms R (2005) Realization of an adaptive modular control for a disassembly system. In: *The 6th IEEE international symposium on assembly and task planning: from nano to macro assembly and manufacturing*
7. Reese G (2000) Distributed application architecture. In: *Database programming with JDBC and Java*, 2nd edn. O'Reilly & Associates
8. MSDN M (2011) Windows sockets 2. [http://msdn.microsoft.com/en-us/library/ms740673\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms740673(v=vs.85).aspx)
9. Viggiano JAS (2004) Comparison of the accuracy of different white balancing options as quantified by their color constancy. In: *SPIE—The International Society for Optical Engineering*, pp 323–333
10. Chang F, Chen C-J, Lu C-J (2004) A linear-time component-labeling algorithm using contour tracing technique. *Comput Vis Image Underst* 93(2):206–220
11. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: *IEEE Computer Society conference on computer vision and pattern recognition*, pp 1511–1518
12. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). *Comput Vis Image Underst* 110(2):346–359
13. Vongbunyong S, Kara S, Pagnucco M (2015) General plans for removing main components in cognitive robotic disassembly automation. In: *The 6th international conference on automation, robotics and applications (ICARA 2015)*, Queenstown, New Zealand, 17–19 Feb 2015

9

Managing and Maintaining the Code

Just creating an automation project is not enough. It is important to have your project organized in a proper way—whether it is deciding which layout to use, or naming your steps properly. A project can also be reused in a new one, making it very convenient for the user. This chapter explains ways in which we can reuse projects. We will also learn about configuration techniques and see an example. Finally, we will learn how to integrate the TFS server.

Listed are the topics that will be covered in this chapter:

- Project organization
- Nesting workflows
- Reusability of workflows
- Commenting techniques
- State Machine
- When to use Flowchart, State Machine, or Sequence
- Using config files and examples of config files
- Integrating the TFS server

Project organization

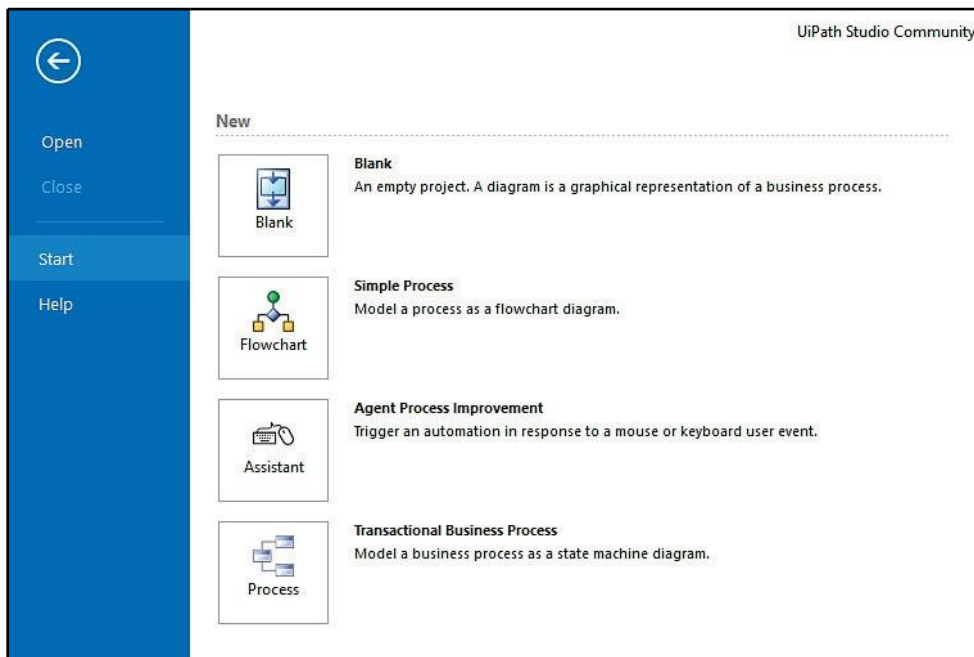
While working on any automation project, it is very important to work with a proper set of rules so that the project can be organized in an efficient way. In UiPath, the following are some of the best practices considered while working on a project:

- Pick an appropriate layout for each workflow
- Break the whole process into smaller parts Use
- exception handling
-
- Make your workflow readable
- Keep it clean

We will now elaborate on each of the best practices.

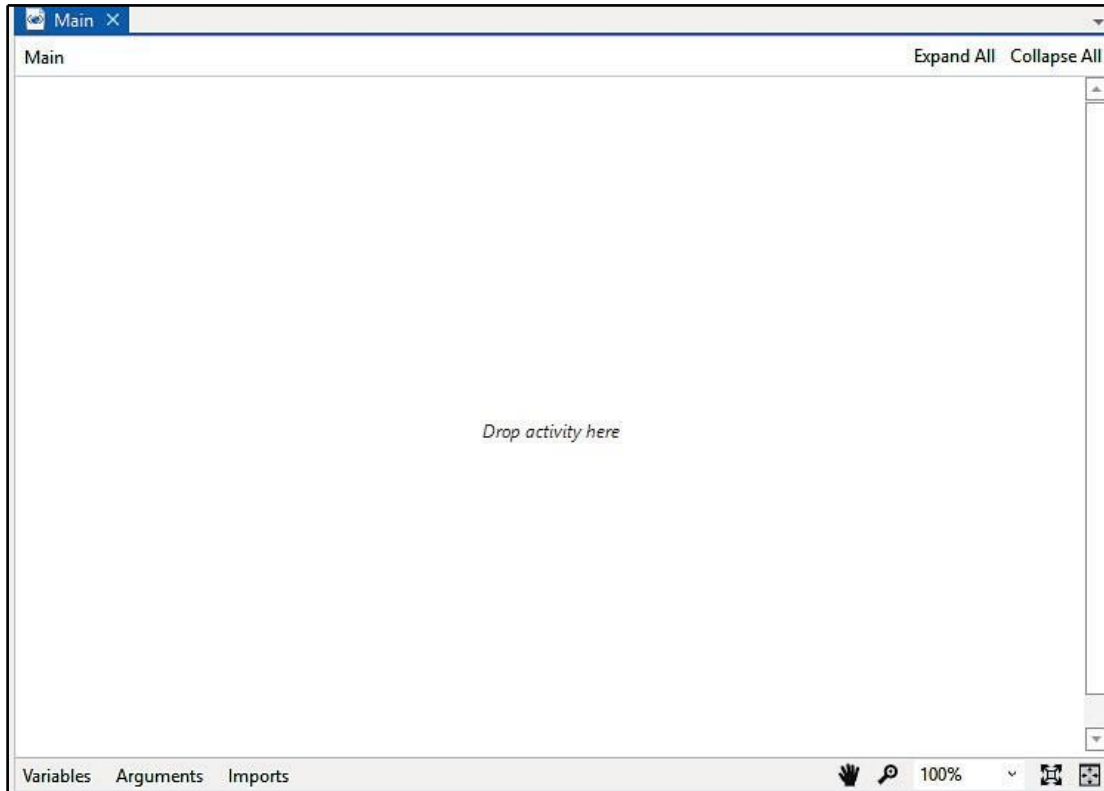
Picking an appropriate layout for each workflow

There are various layouts available while creating a new project. Among those layouts, we have to choose the best option on the basis of the type of automation process we are undertaking. All the layouts are shown in the following screenshot:



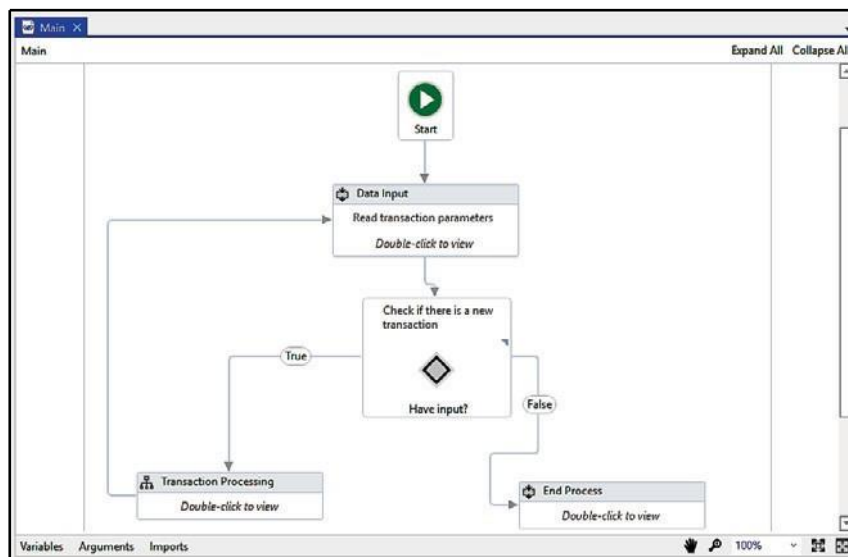
Blank

A Blank project is simply a blank page on which you can create the type of layout you want. That is, you can simply start with a Sequence activity if your workflow is in a single order/sequence or you can use a Flowchart activity if you have a bigger or more complex workflow to be designed. It depends on the needs of the user or the type of automation to be undertaken. The following screenshot shows a Blank project:



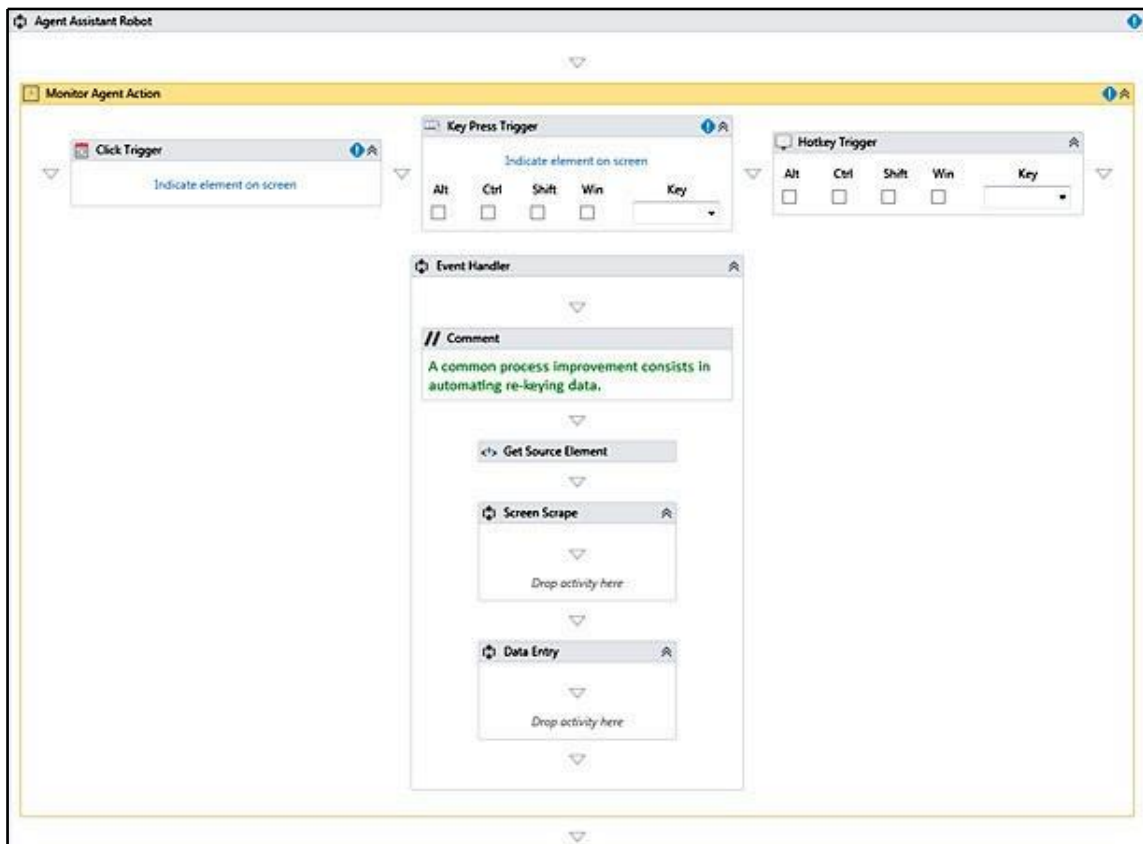
Simple process

The simple process is a layout that is used to model a process as a flowchart diagram in which there is space for user input. Inside this, we can use a sequence that processes the required input in a further transaction process. If there is no new input for the transaction, it will end the process; inside the transaction process, we have to make a workflow that can be used to automate it. This is by default a generated process that can be deleted or changed if required. An example of a simple process has been shown in the following screenshot:



Agent process improvement

This triggers the automation in response to a mouse or keyboard user event. It is basically used when the user is automating processes that involve typing or clicking actions. A simple layout that appears in this process is shown in the following screenshot:



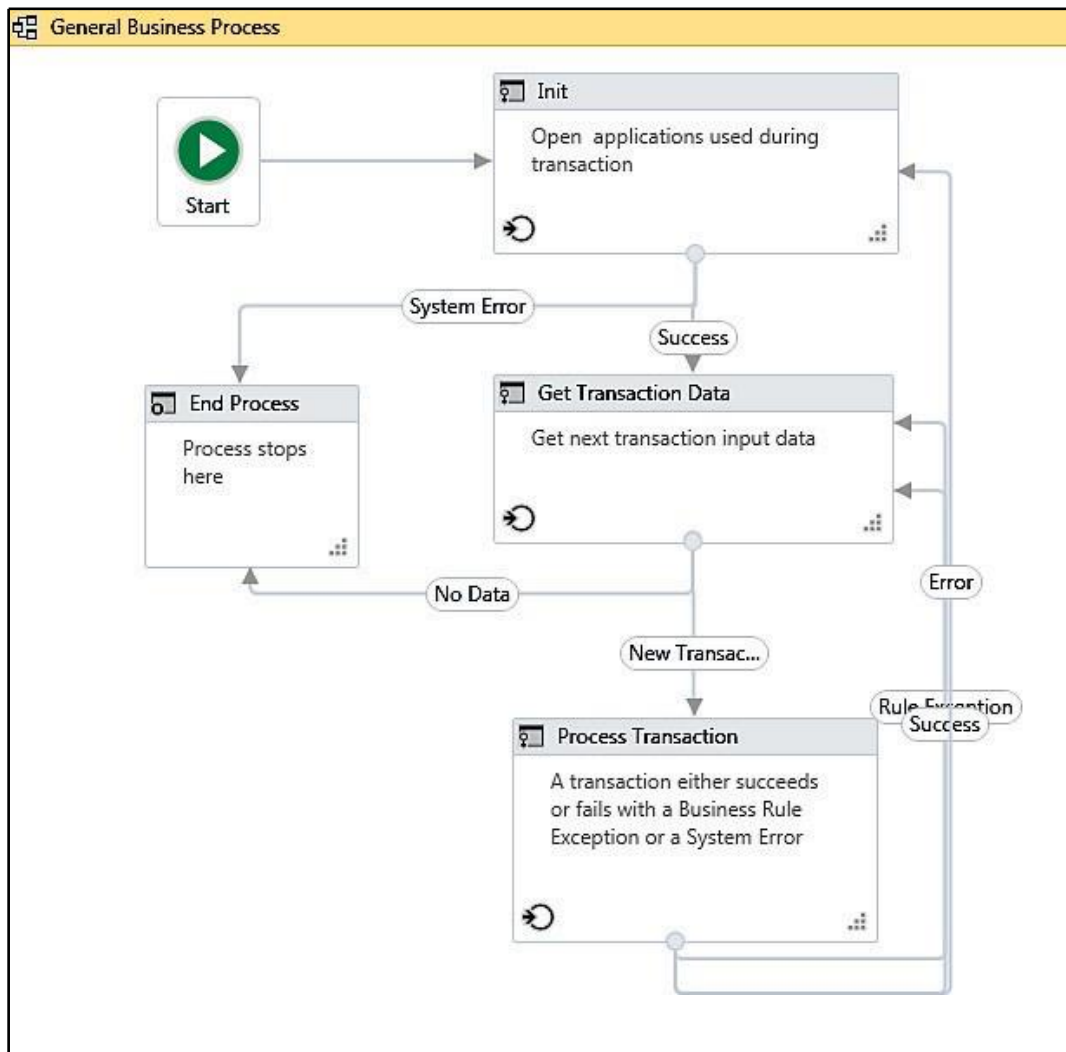
Transactional business process

We will use this layout if we want to model a business process as a State Machine diagram. It is basically a demo of how transactional business process automation works. If we want to build a better Robot to automate such processes, it is better to use this layout.

This layout is categorized into different states:

- **Init:** In **Init** state, we have to configure our settings, credentials (if any), and initialize all the variables that are going to be used in this transaction. All configuration files of the applications (being used in this transaction), are read and taken into account by the robot. The **Init** state also invokes all the applications that are used in the transaction.

- **Get Transaction Data:** In this state, all the transaction data is fetched from the **Init** state. If there is no transaction data, then it transfers the control to **End Process** state.
- **Process Transaction:** In this State, all the transaction data is processed.
- **End Process:** This state ensures that all the processes are completed and there is no transaction data available. It also closes all the applications that are used in the transaction:



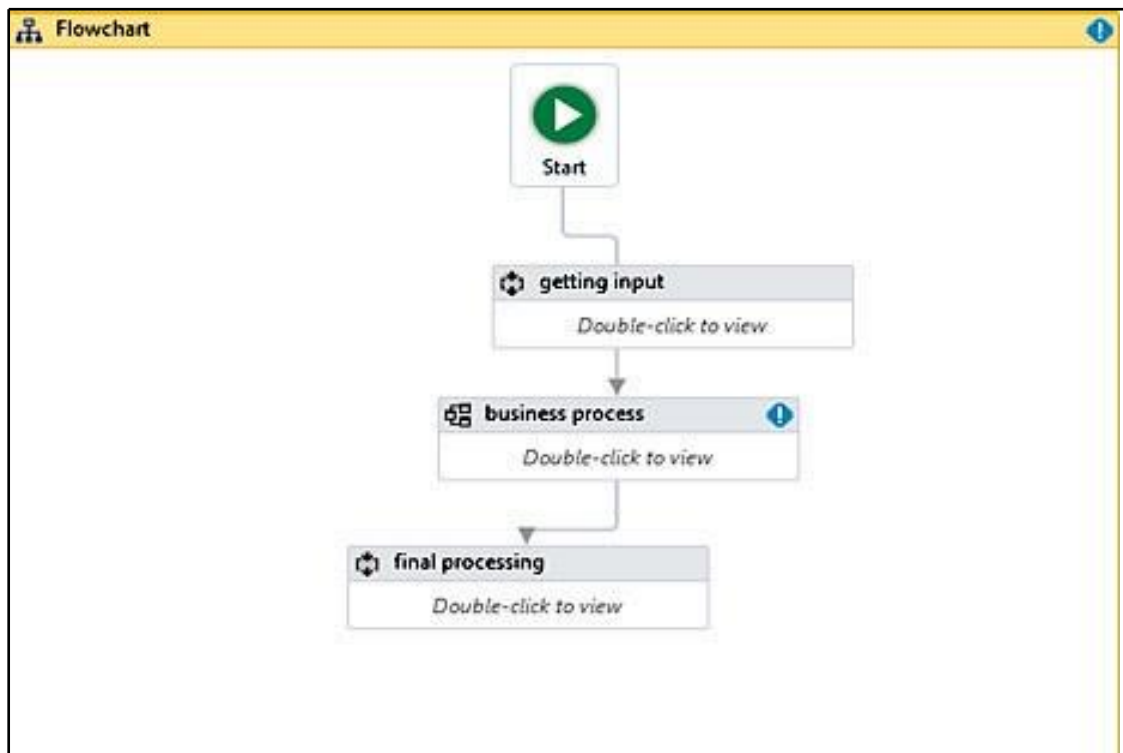
Breaking the process into smaller parts

To build any project, we have to use various activities. But using too many activities makes the project clumsy and it is not readable. We have to design our project in such a way that each independent part resides alone. We can achieve this by using workflows.

We should put each independent part of the project inside a single workflow. We can invoke all the workflows inside the project at the appropriate position.

Dividing the project into workflows makes the project cleaner and more maintainable. Now, if any developer wants to debug your code, they can check the different workflows and easily pinpoint in which workflow a particular error occurred. If the project is not divided into workflows, it will be a nightmare for the developer to fix any error.

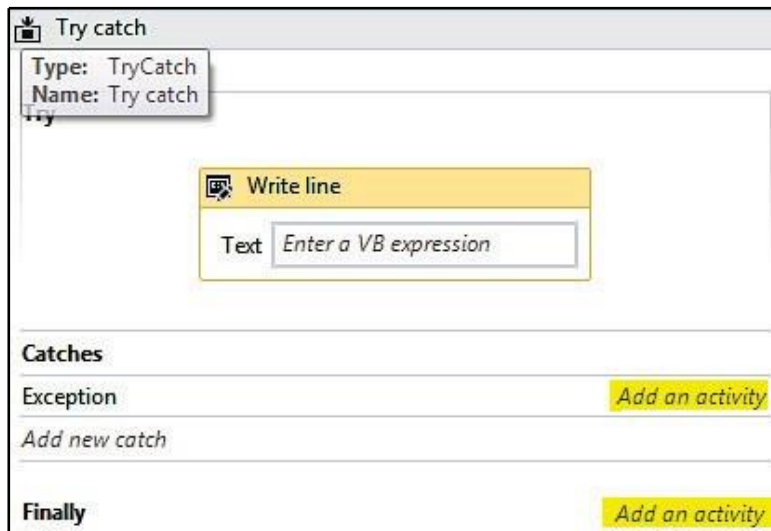
Thus, breaking an automation into smaller parts enables easy debugging, as well as the workflows across projects:



Flowchart of breakdown of automation and reusing workflows across projects

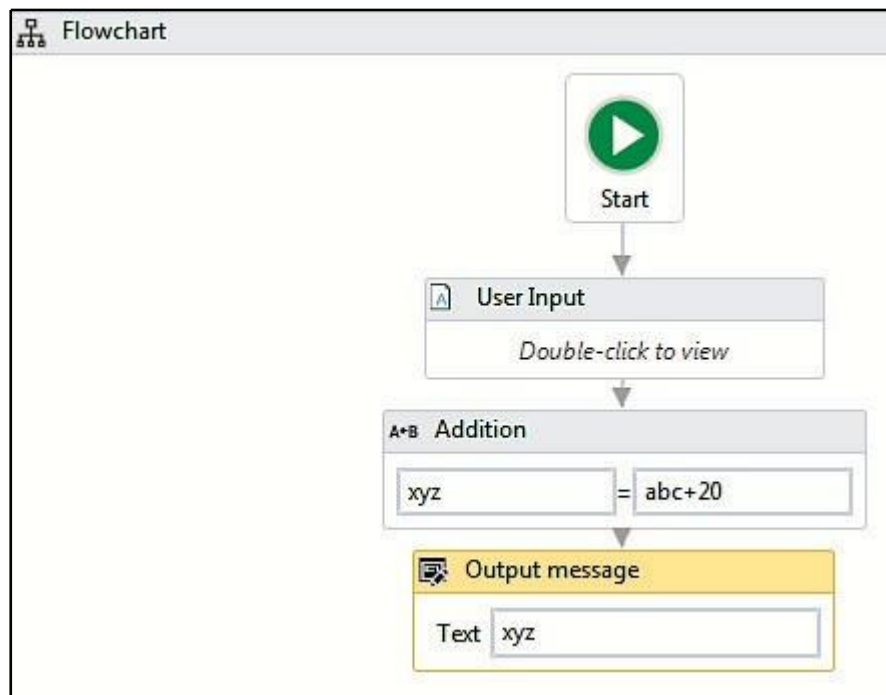
Using exception handling

While working on a project, it is better to use exception handling because it reduces the risk of errors. For instance, using the **Try catch** block can give you a proper error message, which helps us handle exceptions. There are various exception handling techniques that have been explained earlier and that are very useful while working on a project. An example featuring using the **Try catch** activity to handle exceptions has been shown in the figure. Here, we have used the **Write line** activity to display messages in the case of any error detected by the **Catch** block or the **Finally** block (as highlighted in the screenshot):



Making your workflow readable

It is good practice to name activities on the basis of the operations they perform to ensure that when we return to the workflow, we can easily identify each and every step used in it. This becomes very helpful while finding and resolving errors as it specifies the process when showing an error during debugging. If activities are properly named, we get to know exactly which part of the workflow is not working. For example, we will create a workflow that will ask the user to guess a number, on the basis of which we will perform an addition and finally display the answer. The following screenshot shows the proper naming of the steps involved in the process:



Keeping it clean

Just as writing in a clean and understandable manner is the quality of a good coder, the same holds true for an RPA developer. Clean code helps us understand the whole process very easily—you and whoever is reading it.

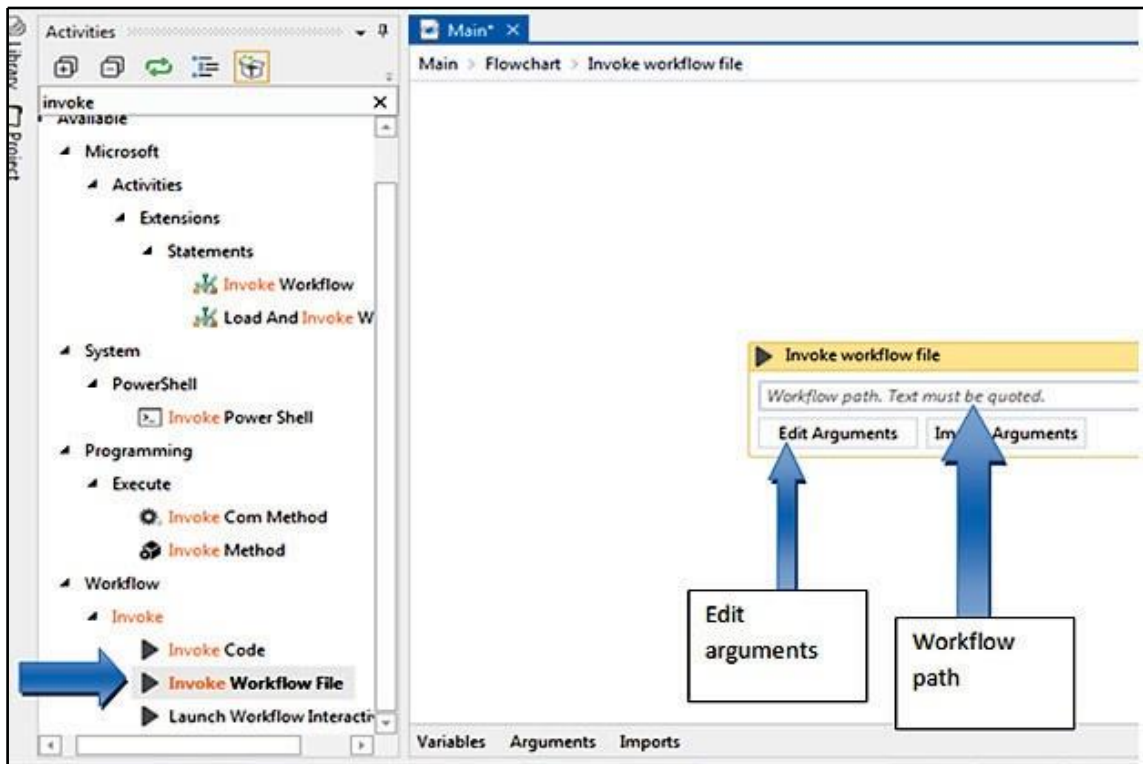
Nesting workflows

While working in UiPath, it is better to divide the whole process into smaller parts and then nest these workflows into a larger one or the **Main** workflow. This can be done using the **Invoke workflow file** activity given in the **Activities** panel. There are several steps involved in nesting a workflow or many workflows into a single workflow.

How to nest a workflow inside a single workflow

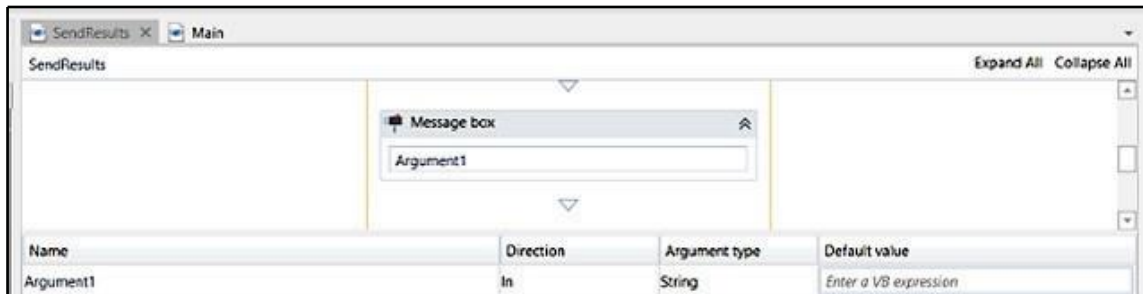
Say we have two workflows. In this example, we will invoke one workflow into the other:

1. Add an **Invoke workflow file** activity to the first workflow:



Screenshot of Invoke workflow file

2. Click on the **Edit Arguments** option available.
3. Define an argument and type it in the Invoke workflow arguments that appear:



Invoke workflow arguments

4. In the **Arguments** panel in the second workflow, create an argument with the same name as the first workflow. You will now be able to use that argument as any other variable.

Reusability of workflows

Reusing workflows makes the automation process easier and better since we can use earlier created workflows in our project that we are trying to use for automation. There are two methods for this:

- Invoke workflow file
- Templates

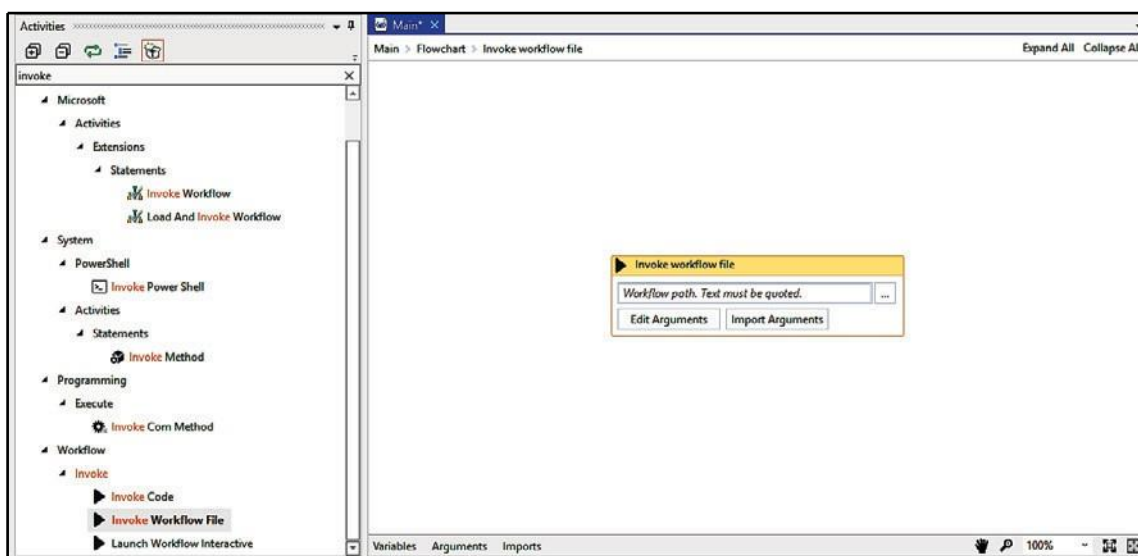
Invoke workflow file is a good process if we have a complex automation project. We can divide it into smaller parts. By using the **Invoke workflow file** activity, we can invoke all those files in our project and collect all these smaller parts in a single workflow. However, if we want to invoke a previously created workflow in our project and make changes to the latter, the former will also get affected. Hence, it is recommended to use the **Invoke workflow file** activity only when we have a complex workflow:

image to be added here

As shown in the preceding screenshot, the **Invoke workflow file** activity requires the path to its associated XAML file.

Invoke workflow file

Invoke workflow file is a good process if we have a complex automation project. We can divide it into smaller parts and then, by using the **Invoke workflow file** activity, we can collect all these smaller parts in a single workflow file. However, if we want to invoke previously created workflows in our new workflow and make changes to the new workflow, the previous workflow will also get affected. Hence, it is recommended to use the **Invoke workflow file** activity only when we have a complex workflow and we want to divide the process into smaller parts and then use them together. There is another property for that, what we need here; it's as follows:



As shown in the preceding screenshot, the **Invoke workflow** activity requires a variable expression. We can create a variable and set a timeout that is required for the **Invoke workflow file** activity.

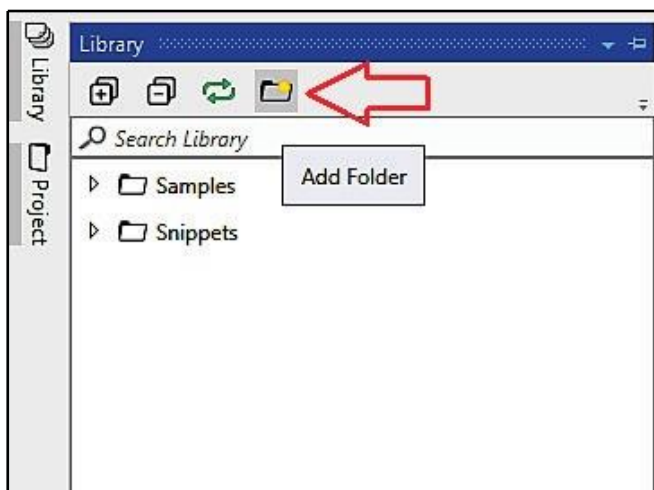
Templates

Saving the workflow as a template helps you preserve the original workflow file. So whatever modifications you made in the template, no changes will be made in the original workflow. We often use templates when creating small pieces of common automation that are reusable and applicable in multiple workflows. So you can use templates if the workflow does not change over time. The most common example is when you create your own reusable snippets using data, data tables, and `.xml` files.

Adding a workflow as a template

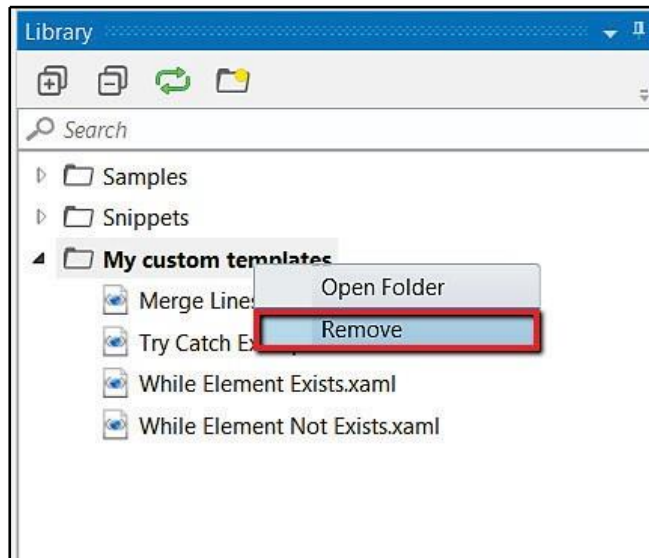
Follow the steps given to add a workflow as a template, which is explained as follows:

1. Add a new folder in the **Library**:



2. After clicking on the **Add Folder** icon you can browse for your file containing the workflows. Just select folder from the list that contains all the workflows. Now the folder can be used anytime in any workflow from the **Library** panel.

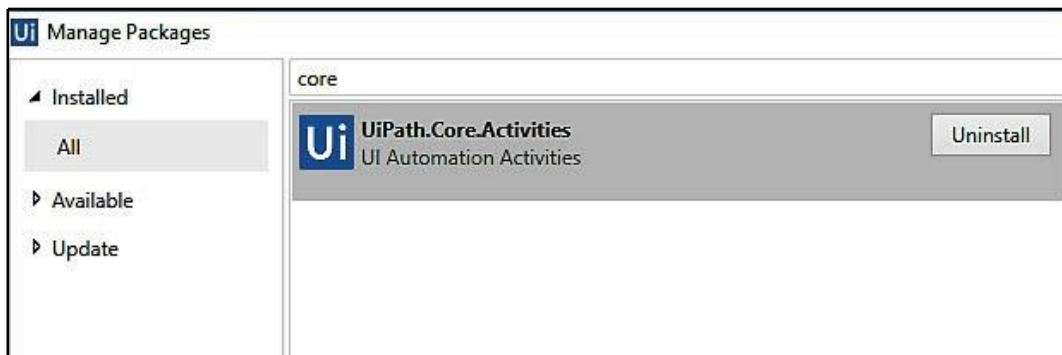
3. We can also remove an added file by just right-clicking on it and then selecting the **Remove** option, as shown in the following screenshot:



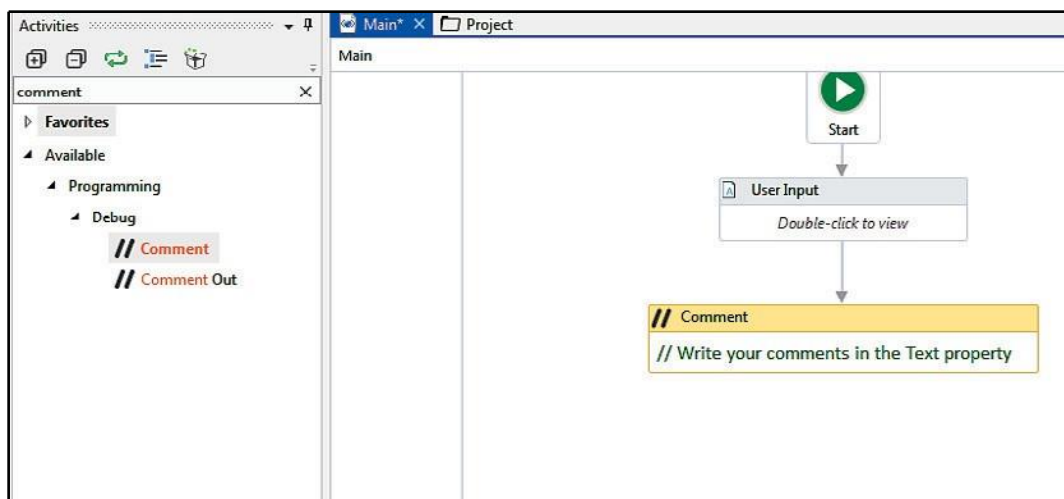
Commenting techniques

Using comments in workflows is considered a good practice as it can give a better step-by-step notification of what is done in the workflow. Therefore, commenting in a complex workflow is considered to be good while debugging:

- The package you'll need to use comments inside a workflow needs to be installed from the Package Manager functionality that is available in the **Activities** panel (the **Manage Packages** icon). You can install **UiPath.Core.Activities** from the packages; inside you will find the **Comment** activity in the **Activities** panel as indicated by the arrow (in this case, it is installed):



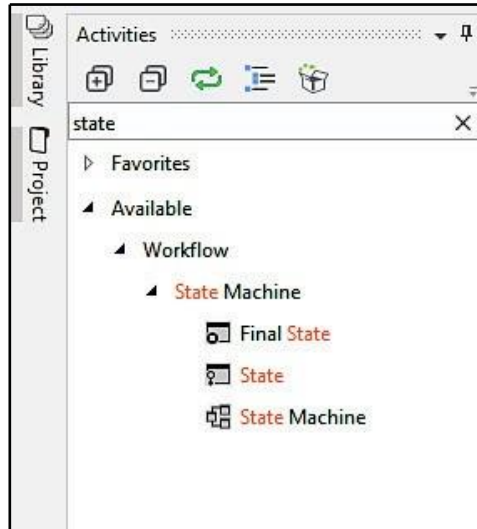
- Once the package is installed, just drag and drop the **Comment** activity from the **Activities** panel and add comments in between the workflows wherever you want:



State Machine

A State Machine uses a finite number of sets in its execution. It can go into a state when it is triggered by an activity; it exits that state when another activity is triggered. Another important aspect of State Machines is transactions. They enable you to add conditions based on which transactions jump from one state to another. These are represented by arrows or branches between states.

There are two activities specific to State Machines. They are **State** and **Final State**, and they are shown in the following screenshot:

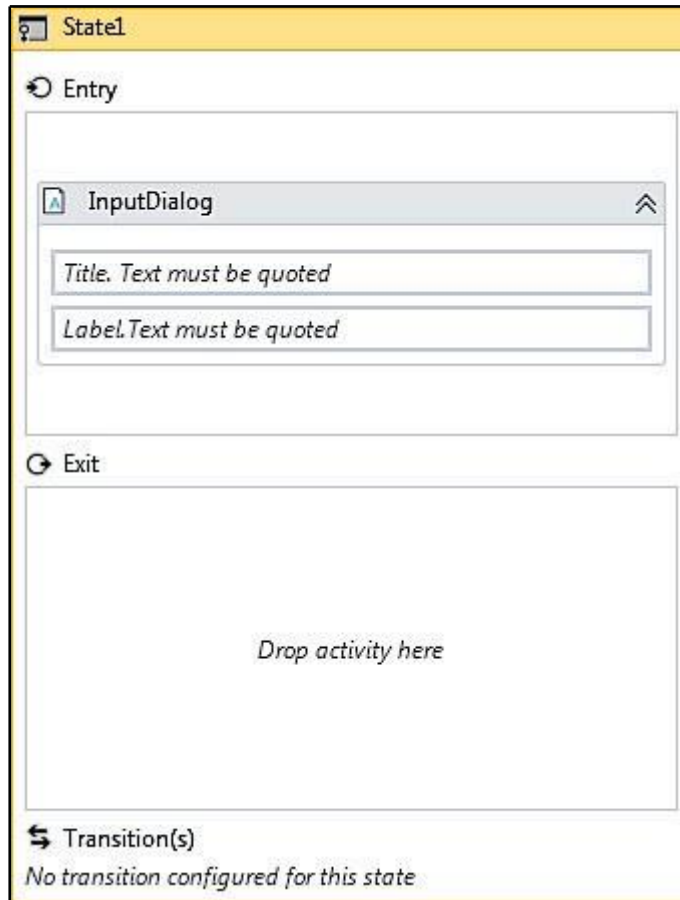


The **State** activity consists of three sections—**Entry**, **Exit**, and **Transitions**, while the **FinalState** only contains **Entry**. We can expand these activities by double-clicking them to view more information and to edit them:

- **FinalState** activity: This activity contains all those activities that need to be processed when the state is entered:

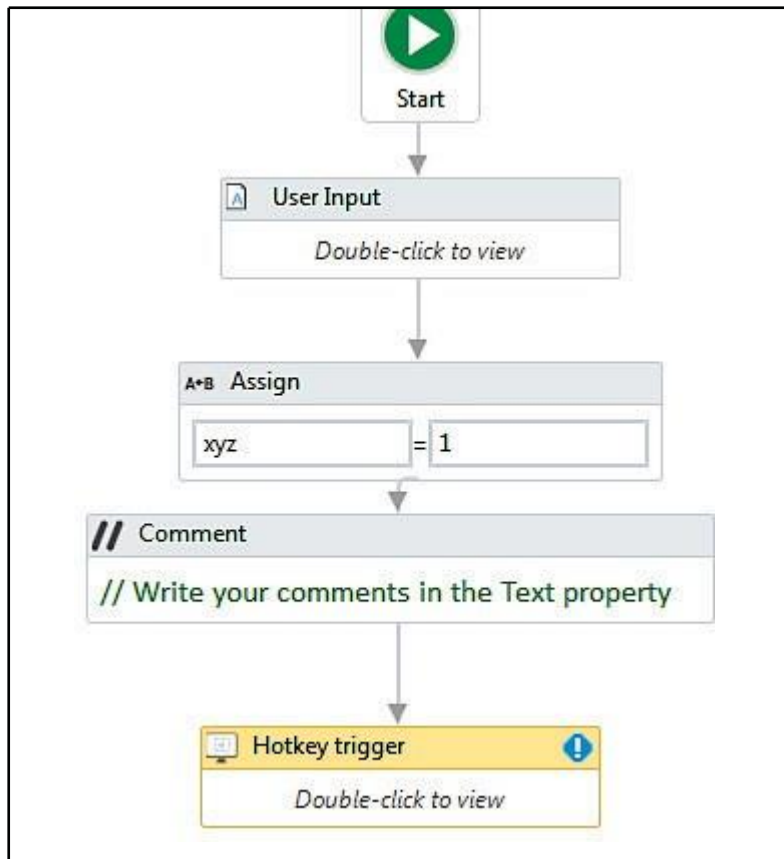


- **State** activity: Transitions contain three sections—**Trigger**, **Condition**, and **Action**, which enable you to add a trigger for the next state or a condition under which an activity is to be executed:



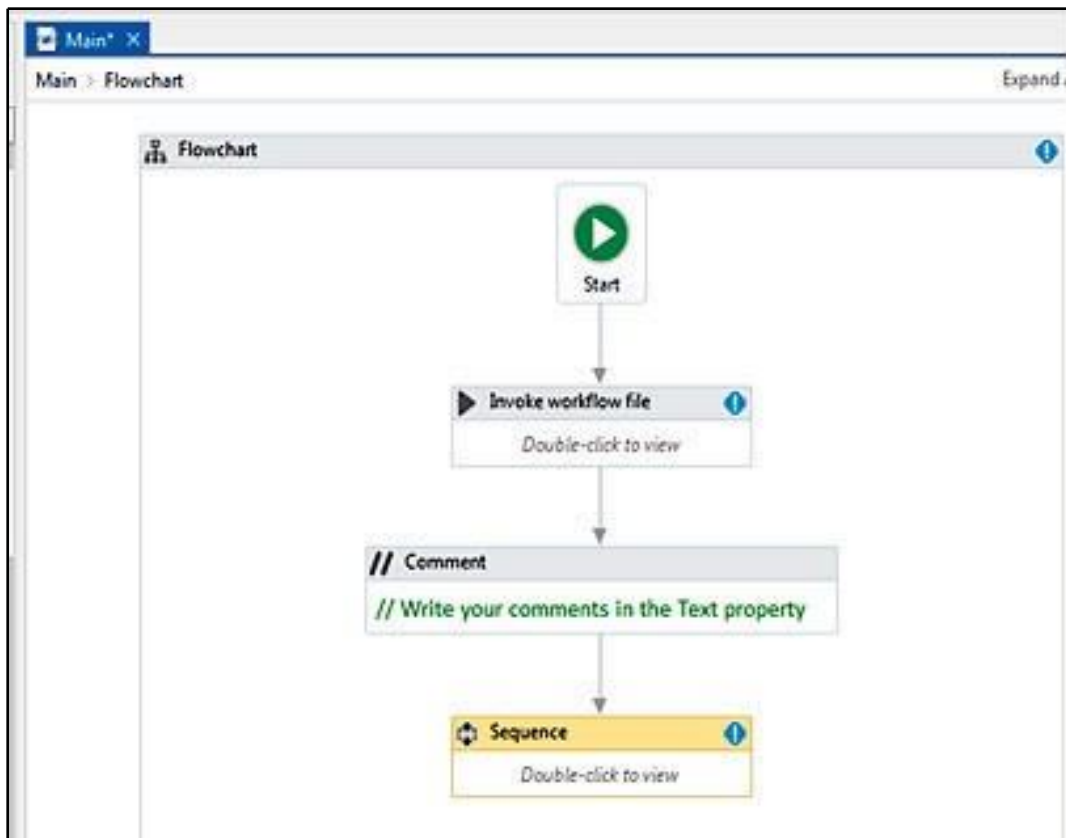
When to use Flowcharts, State Machines, or Sequences

A Sequence is used only when we have selected a straightforward set of instructions on how to create a workflow. That is, we do not have to make decisions. It is preferred when we are recording some steps in a sequential manner and we are creating a simple workflow. One such sequence is shown in the following screenshot:



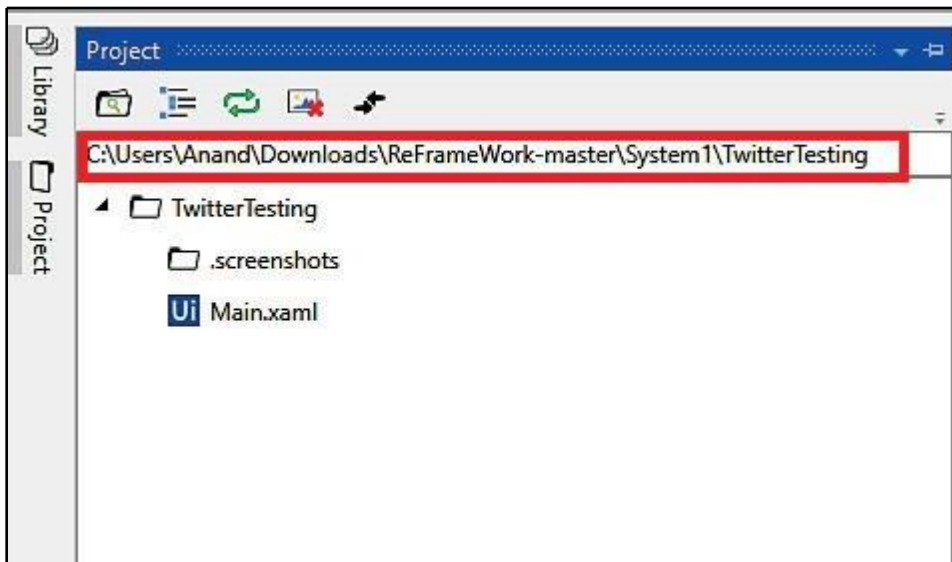
Now, when it comes to State Machines and Flowcharts, both are used for complex processes and both work well. They work in the same manner, but State Machines have some advantages over Flowcharts, which are listed as follows:

- Complex transitions are much clearer with State Machines as they have an inbuilt layout of the workflow.
- Flowcharts do not inherently have the concept of waiting for something to happen. State Machines do (a transition will not occur until a trigger completes and the condition evaluates to true).
- State Machines naturally encapsulate the action group:



Using config files and examples of a config file

When it comes to configuration, UiPath does not have any pre-built configuration file such as Visual Studio, but we can create one. It is considered to be one of the best practices to keep environment settings in a config file so that they can be easily changed by the user when required. Thus, when we create a project, the `project.json` file that holds all the activities is created automatically. `Project.json` can be found in the folder where the project is saved. To access the folder, we can just open the **Project**, then copy the path (as shown in the following screenshot), and paste it into File Explorer:



Then you can see a `project.json` file in File Explorer like the one shown in the following screenshot:

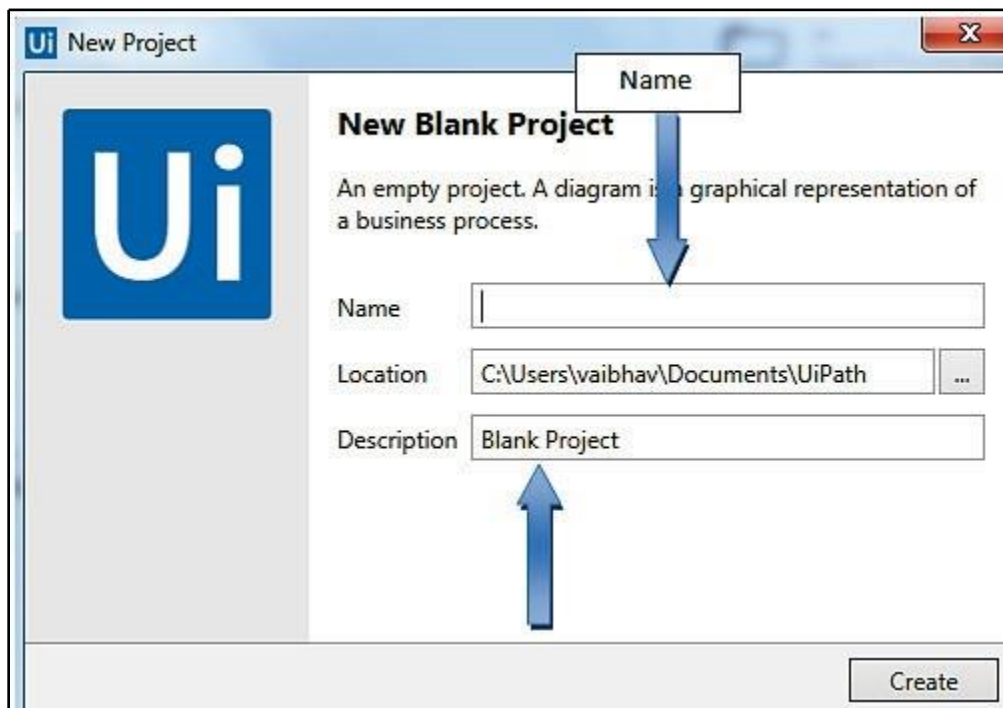
Name	Date modified	Type	Size
Initialization	11/14/2017 1:28 PM	XAML File	5 KB
Main	11/14/2017 1:28 PM	XAML File	36 KB
project	11/14/2017 1:28 PM	JSON File	1 KB
Transaction	11/14/2017 1:28 PM	XAML File	9 KB

The following screenshot displays the code inside the `project.json` file, when you open that file in Notepad:

```
{
  "name": "a",
  "description": "Transactional Business Process Project",
  "main": "Main.xaml",
  "dependencies": {},
  "excludedData": [
    "Private:*"
    "*password*"
  ],
  "toolVersion": "17.1.6522.14204",
  "projectVersion": "1.0.6527.24239",
  "packOptions": {},
  "runtimeOptions": {}
}
```

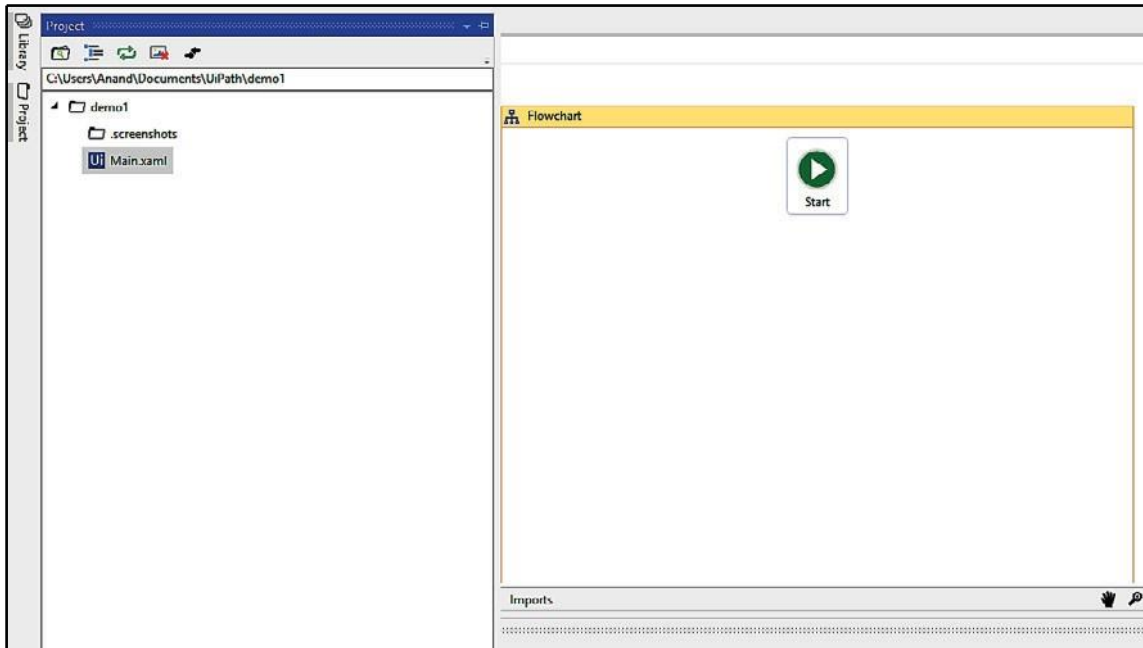
You can also store your settings with the help of a spreadsheet or credentials. There are various parameters contained in the `project.json` file. They are:

- **Name:** This is the title of the project that is provided when creating a project in the create New Project window:



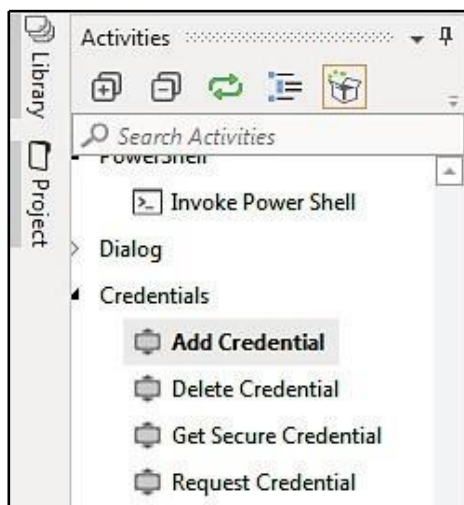
- **Description:** When creating a project, a description is also required. You can add the description in the Create New Project window, as shown in the preceding screenshot.

- **Main:** This is the entry point for the project. It is saved as `main.xml` by default, but you can change its name from the **Project** panel. Also, you have multiple workflows for a project, it is necessary to attach all these files to the main file with the **Invoke Workflow File** activity. Otherwise, those files will not be executed:

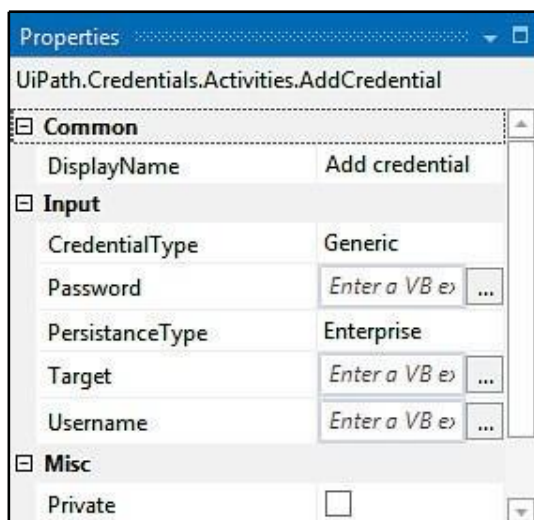


- **Dependencies:** These are the activities packages that are used in a project and their versions.
- **Excluded data:** Contains keyword that can be added to the name of an activity to prevent the variable and argument values from being logged at the verbose level.
- **Tool version:** The version of Studio used to create a project.

- **Adding Credential:** We can add particular settings that can be used further. For example, we can save the username and password to be used further, so this can be done with the help of the **Add credential** activity that can be found in the **Activities** panel, as shown in the following screenshot:

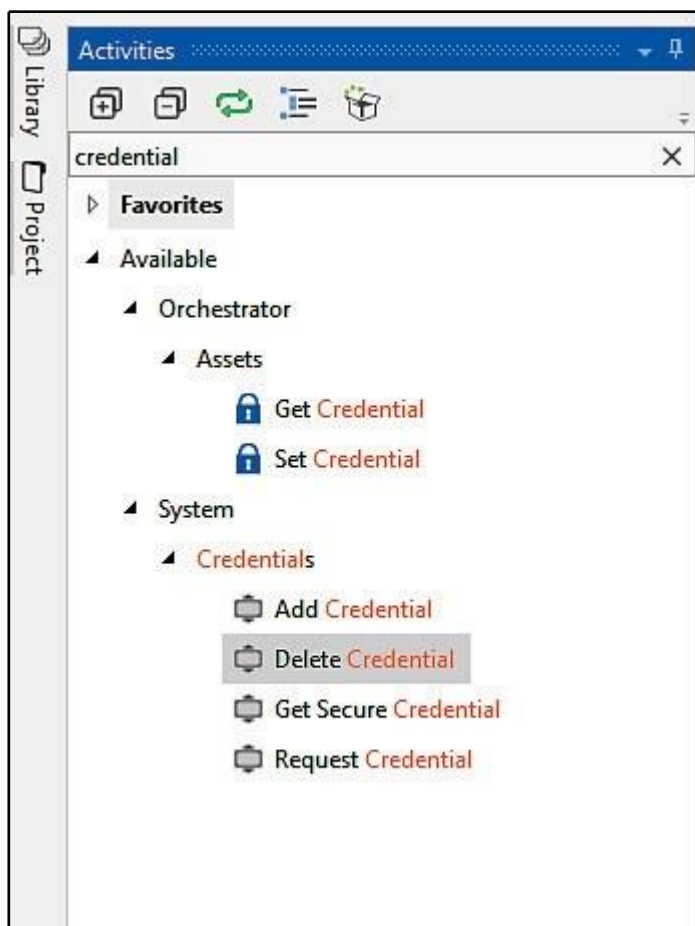


After adding credentials, type the required values in the **Properties** panel, as shown in the following screenshot:

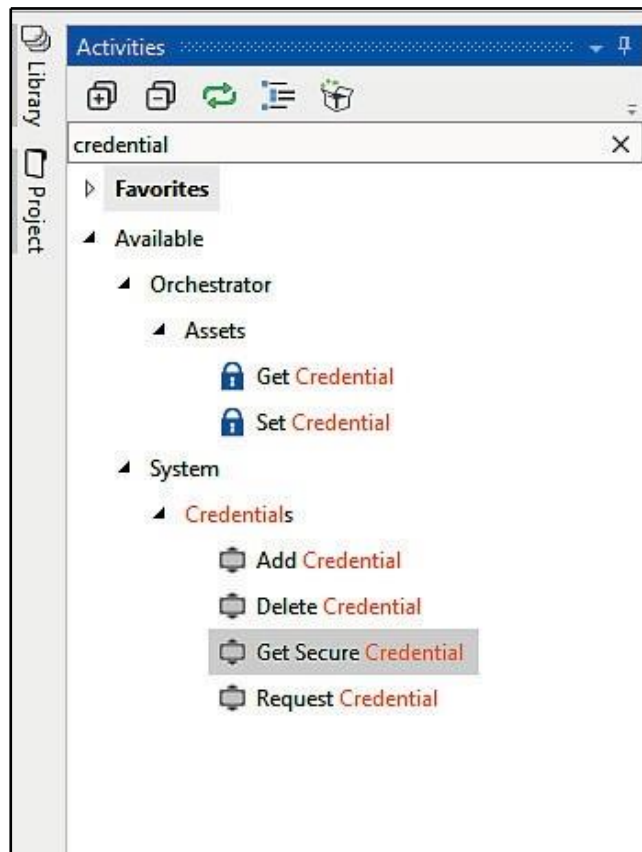


So, when the credentials are set, we can delete, secure, or request credentials, as shown in the following steps:

1. **Delete Credentials:** If we want to delete a credential then we can just drag and drop the **Delete Credentials** activity and then define the target for the credential:



2. **Get Secure Credential:** This is used to get the values, that is, the username and password, that were set during the addition of a credential. We have to set the target the same as before; the output will be the username and password:



3. **Request Credential:** This is a property in which the robot displays a message dialog asking the user for username and password and stores this information as a string. This can then be used in further processes. The user can select OK to provide credentials or even cancel it if they do not want to provide credentials.

Integrating a TFS server

UiPath integrates a series of actions that allow us to have better collaboration on the project. Inside the **Project** panel, by right-clicking on the file we can see a list of properties that are included in it:

- By clicking on the **Get Latest Version** option, we can get the latest version of the selected file from the TFS server
- You can also rename or delete an existing file
- To edit a Read Only workflow, you can select; **Check Out** for edit To
- check in changes, select **Check In** from the menu

Summary

This chapter covered the organization of projects, modularity techniques, workflow nesting, and using the TFS server to maintain a version of source code. In the final chapter, you will gain insight into how you deploy and manage your bots using the Orchestrator.

10 Deploying and Maintaining the Bot

On completion of an automation project design, we make use of the Orchestrator to manage our bots. Before that, we publish our workflow first using the publish utility. Once we publish the project, the package is uploaded to the server. Then we use Orchestrator to manage any number of bots for any number of tasks. The Orchestrator Server also provides the facility for scheduling bots and specifying the time intervals that they work in according to the user's needs.

To understand more, let's see an overview of the topics to be covered in this chapter:

- Publishing using publish utility
- Overview of Orchestration Server
- Using Orchestration Server to control bots
- Using Orchestration Server to deploy bots
- License management
- Publishing and managing updates

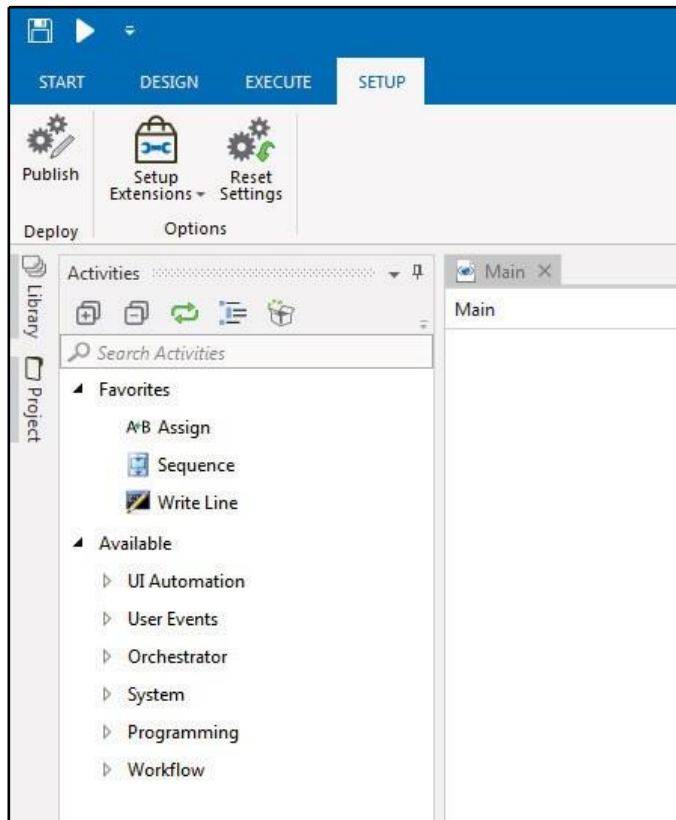
Publishing using publish utility

We design a workflow for some functionality so that it reduces our effort and time. When the workflow is successfully completed, we cannot afford to open UiPath time and again to run our workflow. So to use a workflow directly from the UiPath Robot, we have to publish our workflow first and then schedule it through Orchestrator. As soon as our workflow is published, we can directly run our workflow using a UiPath Robot from Orchestrator.

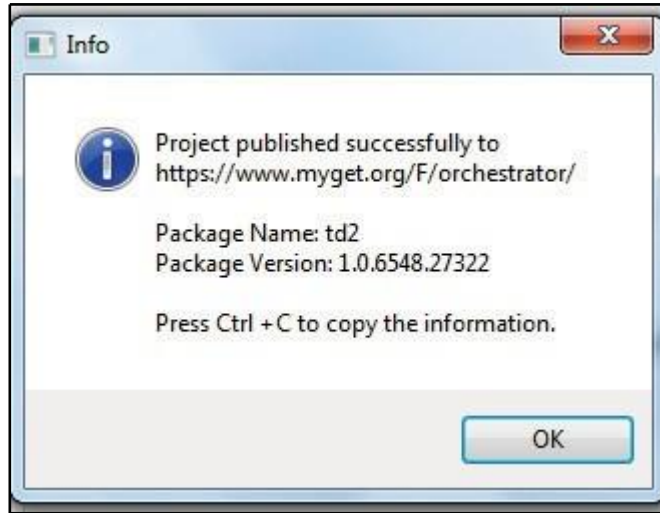
How to publish a workflow in UiPath

The following are the steps to be followed to publish a workflow in UiPath:

1. First, open UiPath Studio, create a new project, and give it an appropriate name.
2. Go to the **SETUP** Ribbon and click on the **Publish** button. Now check whether the project has been published successfully or not:



If the workflow has been published successfully, then a dialog box will appear containing all the necessary data required to run that workflow from Orchestrator:

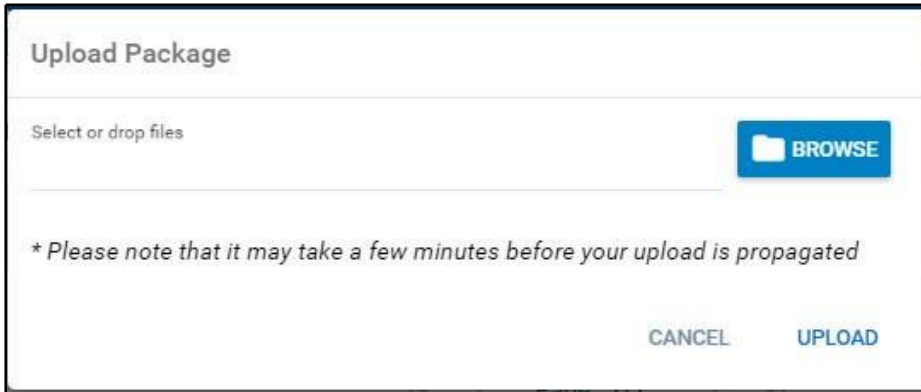


The **Info dialog** box displays all the information as mentioned in the following list:

1. The URL of the Orchestrator where the project is published.
2. The name of the package that has been published by you from UiPath Studio.
3. The version of the package that is published to Orchestrator. When we create any project in UiPath Studio then by default the path is saved in `C:\Users\username\Documents\UiPath` or it can be changed manually by the user. On creating a project, a folder is created that contains the following types of file:
 - The `.screenshot` folder
 - The `.xaml` file that is automatically created during automation
 - `project.json`; this project file holds information about the project

When we publish the package to Orchestrator Server then it automatically reaches the server and can be viewed on the **Packages** page. However, if somehow the package is not found on the Orchestrator Server then we can add a package manually as well, as shown in the following steps:

1. Click on the **Upload Package** option:

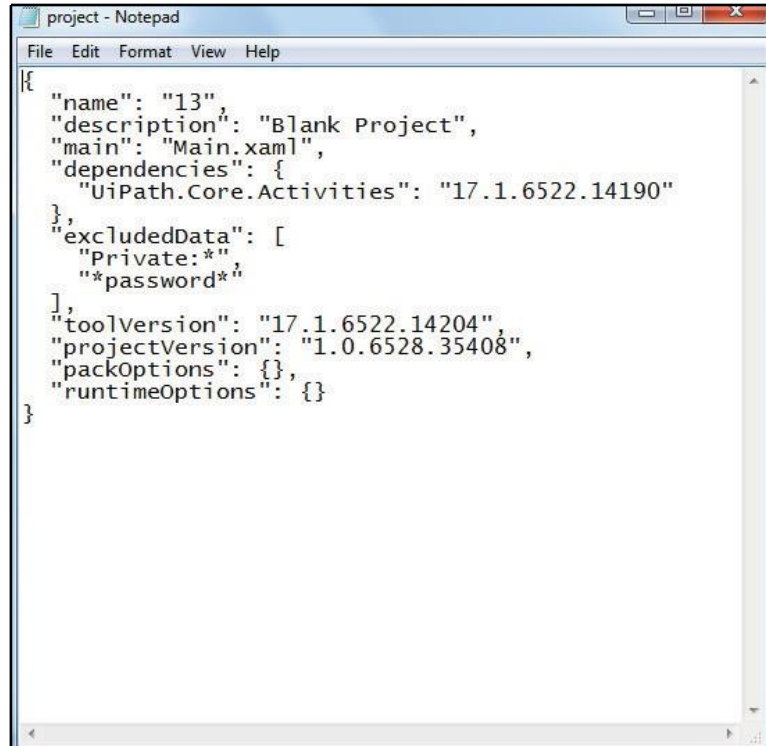
The screenshot shows a web-based dialog box titled "Upload Package". At the top, there is a text input field with the placeholder "Select or drop files". To the right of this field is a blue button with a folder icon and the text "BROWSE". Below the input field, there is a line of italicized text that reads: "* Please note that it may take a few minutes before your upload is propagated". At the bottom right of the dialog, there are two buttons: "CANCEL" and "UPLOAD".

2. Click on **BROWSE** and navigate to the location to which your package was published in the first step.
3. Click on **UPLOAD**.
4. Now search for your package by clicking on **Processes** and then typing your package name in the empty Search box on the **Packages** page.

Writing/editing the published package info into the .json file

The `.json` file can be found inside the project. To edit the `.json` file in the project, we should follow the given steps:

1. Go to the project directory
2. Open the `.json` file in any editor, such as Notepad++
3. Now edit the main parameter of the automation project that you want to execute when the Orchestrator starts:



```
{
  "name": "13",
  "description": "Blank Project",
  "main": "Main.xaml",
  "dependencies": {
    "UiPath.Core.Activities": "17.1.6522.14190"
  },
  "excludedData": [
    "Private:*",
    "*password*"
  ],
  "toolVersion": "17.1.6522.14204",
  "projectVersion": "1.0.6528.35408",
  "packOptions": {},
  "runtimeOptions": {}
}
```

4. Now save the .json file
5. Go to Orchestrator's **Processes** page and click on the **Packages** button

Overview of Orchestration Server

Using of Robots was not as popular as it is now. In other words, we can say that Robots worked within a limited environment. But today, due to **Robotic Process**

Automation (RPA), Robots can work in different environments. Nowadays, their performance is not limited. They are now playing a big role in terms of automation, working as assistant bots to fully potential Robots. They can work 24/7 and their operations can be managed and scheduled through the Orchestrator Server. UiPath Orchestrator is a web server that provides you with an environment for maintaining and scheduling your bots. Orchestrator is a highly accessible web server platform for fast deployment from one Robot to many Robots.

In autonomous automation techniques, one Robot can automate another Robot. This means a Robot can manage all the activities of another Robot-like process; scheduling, and so on.

Robots are of two types:

- Front office Robot (assistant Robot)
 - Back office Robot
1. **Front office Robot (assistant Robot):** Front office Robots act as a helping hand to its user. These are the Robots that require user interactions during the process. A Front office Robot is an agent assistant, which means the user is required to interact with the process. For example, the Robot requires the user to provide credentials or displays some message or dialog to which the user is required to respond otherwise further processes will not work. Some business processes are required to be performed by the trigger activity, in which once a task is triggered, the Robot is capable of running automation processes behind the lock screen.
 2. **Back office Robot:** Back office Robots can log in to window sessions and run the automated process in unattended mode. They can be started with the help of Orchestrator. We can schedule these Robots or we can also run them manually using UiPath Robot or UiPath Studio.

UiPath Orchestrator has some logical components, which are given as follows:

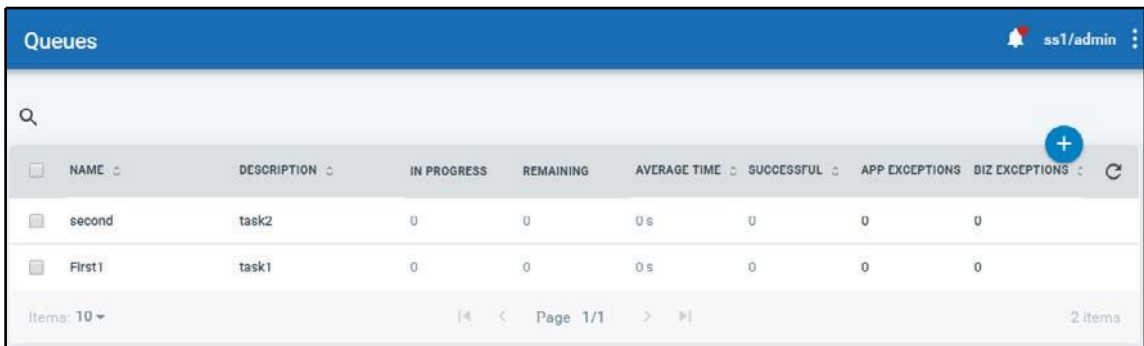
1. User Interface Layer:
 - Web Application
2. Web Services Layer:
 - Monitoring Service
 - Logging Service
 - Deployment Service
 - Configuration Service
 - Queues Service
3. Persistence Layer:
 - SQL Server
 - ElasticSearch

Queues

Queues work as a container that stores tasks that need to be implemented. Simply imagine a group of boys standing in a queue in front of a ticketing counter. The logic is that the person who goes in first gets out first. **First In First Out (FIFO)**.

Similarly, in the case of Robots, when we have a number of operations that are to be performed and when the server is busy, then tasks are moved in a queue and they are implemented on the same logic **First In First Out (FIFO)**.

To create a new queues, search for the Queue option in the Orchestrator Server listed on the left-hand side and then inside the Queue page, you can add one. It also allows you to access all those **Queues** that have already been created. It contains some information about the task such as the remaining time, progress time, average time, description, and so on, as listed in the following screenshot:



NAME	DESCRIPTION	IN PROGRESS	REMAINING	AVERAGE TIME	SUCCESSFUL	APP EXCEPTIONS	BIZ EXCEPTIONS
second	task2	0	0	0 s	0	0	0
First1	task1	0	0	0 s	0	0	0

We can also add queue items from UiPath Studio and there are various activities that support this feature, which are listed as follows:

- **Add Queue Item:** This activity is used to add a new item to the queue in Orchestrator. The status of the item will be New.

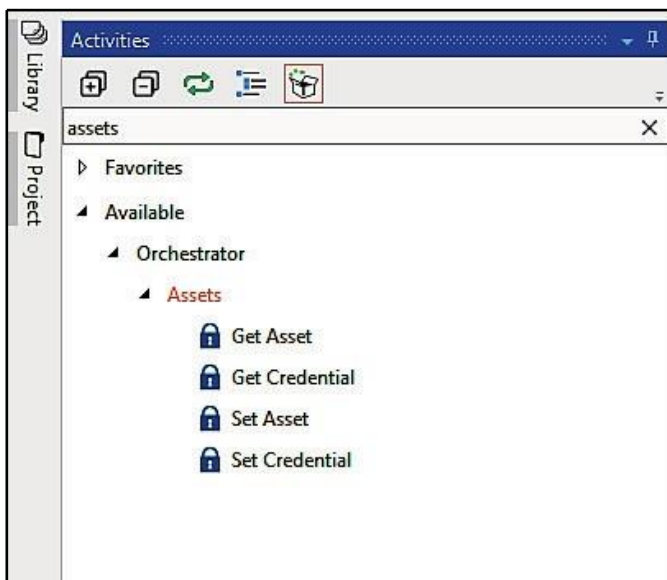
Add Transaction Item: This activity is used to add an item to the queue to begin transaction and set the status as In Progress. Here we can add custom reference for each respective transaction.

Get Transaction Item: This activity is used to get an item from the queue to process it and set its status as In Progress.

- **Postpone Transaction Item:** This activity is used to define time parameters between which transaction should be processed. Here, basically, we will specify the time interval after which a process will start.
- **Set Transaction Progress:** Used to assist and create custom progress statuses for In Progress transactions. To notify its progress if the process crashes. This activity plays a significant role in tackling troubleshooting processes.
- **Set Transaction Status:** Used to modify the status of the transaction item; whether it fails or is successful.

Assets

Assets work as variables or credentials and can be used in distinct automation projects. Assets provides the opportunity to hold specific information. This information can be easily accessed by the Robot. The Assets activity can be found from the **Activities** panel, as given in the following screenshot:



In addition, assets can also be used for security purposes to store credentials. As we know, all credentials are stored in encrypted format by the AES 256 algorithm. When an RPA developer is designing a process, it can be invoked by the developer, but its value is still hidden.

To create a new asset in Orchestrator, we are required to open the **Assets** page. It also shows all the previously created assets that can be removed or edited:



NAME	TYPE	VALUE
app	Bool	False
cond	Bool	False
str	Credential	username: krishna.dev@rpatech.in

Assets are of two types:

1. Get Asset
2. Get Credential

The **Get Asset** and **Get Credential** activities are used in Studio to request information from Orchestrator about a specific asset, according to a provided `AssetName`.

The `AssetName` is required for an already stored asset in the Orchestrator database so that the Robot can access the information stored in Asset. To do so, the Robot needs permission to retrieve information from that particular asset to be used in the automation project. We can get assets using the **Get Asset** activity in the **Activities** panel of UiPath Studio as explained earlier.

There are four types of asset values:

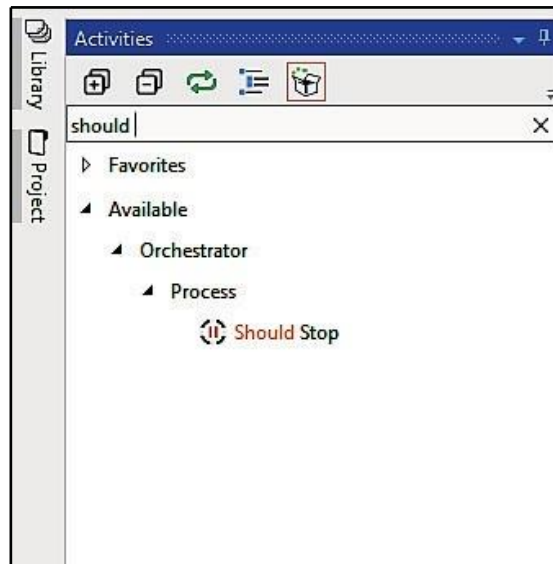
1. **Text:** This holds string values.
2. **Boolean:** This supports only true or false values.
3. **Integer:** This stores integer values.
4. **Credential:** This holds usernames and passwords that are needed by the Robot to execute specific processes, such as login details.

Furthermore, we also have the following types of Asset:

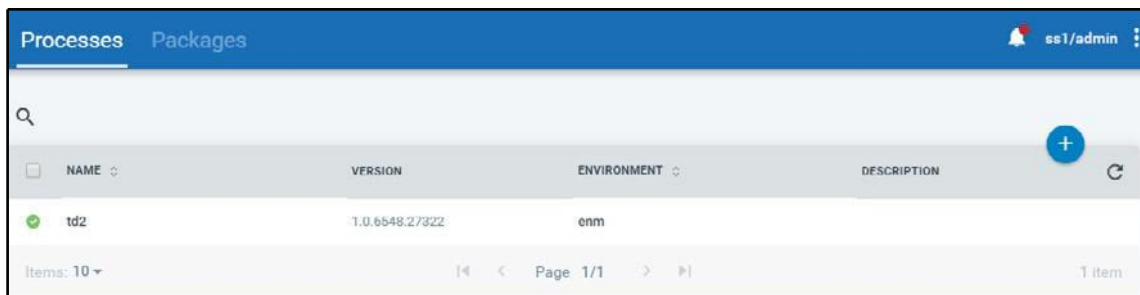
- **Global:** This can be accessed and used by all available Robots
- **Per Robot:** This can be accessed only by a specified Robot

Process

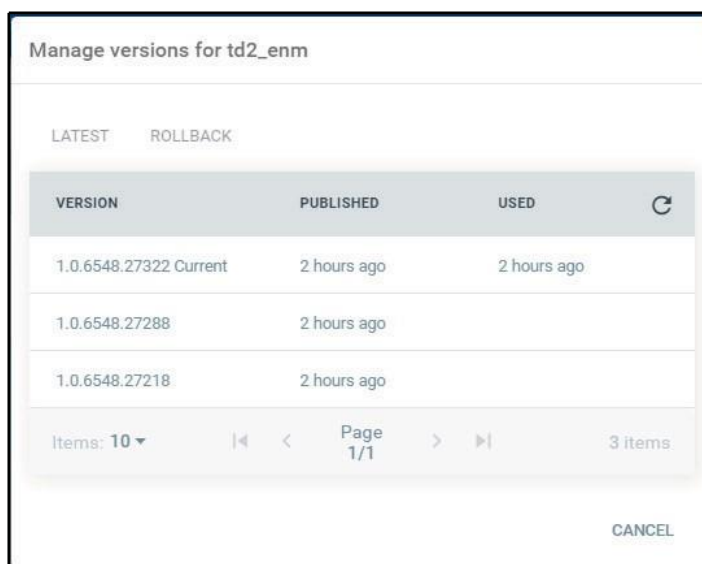
The process is responsible for deploying and uploading the package to the Orchestrator environment, and for deploying already created packages. In UiPath Studio, we can search for **Process** inside the **Orchestrator** option available in the **Activities** panel. It contains a **Should Stop** activity that can be used to stop a process whenever required:



Processes assist in distributing all **Packages** over the Robot machine, which makes execution faster. We can assign jobs to these processes from the jobs panel available in the left corner in Orchestrator:



At every stage a package is linked to the environment and it is automatically distributed with each Robot machine that belongs to a particular environment. Whenever you make changes in the package you created earlier and upload those changes, it creates a new version of that package. Thus, in order to update your package, you can go to the **Manage versions** option available on that package and select the version of that package that you want to use:



If a new update to a certain package is available then an icon is displayed on that package.

When we are using the latest available version of a package with a specific process, then the icon  will show next to the process. All activities that we are using in Studio are stored in the NuGet feed that Orchestrator has access to. While we add a new process, the name of the environment should be the same as that of the Robot. This allows the execution of processes using respective Robots.

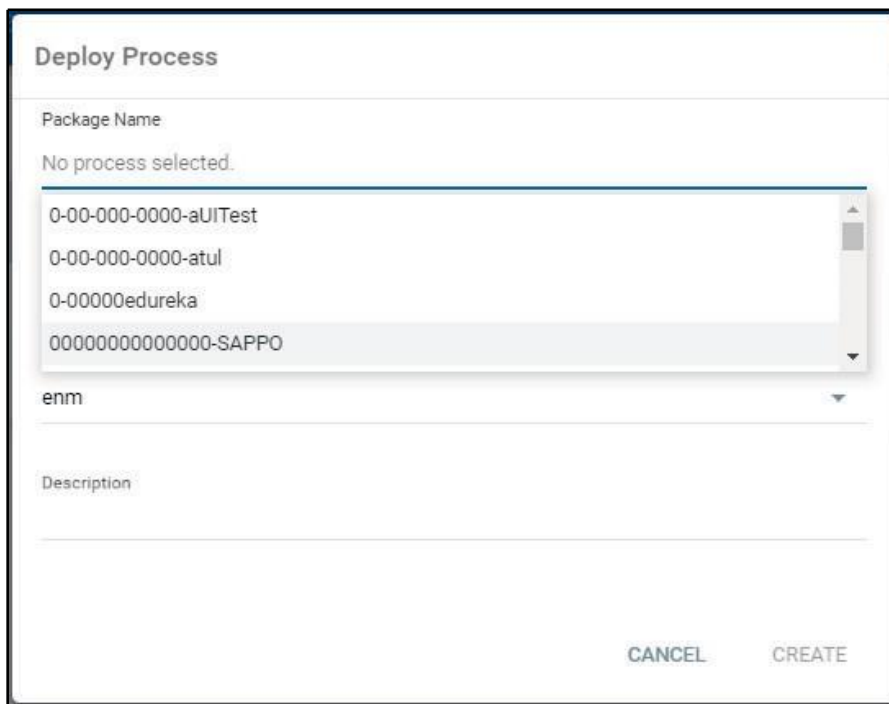
Deploying a process

Deploying a process basically refers to the distribution of the packages to the Robots available.

After successfully publishing your project from UiPath Studio, as explained earlier, we can follow these following steps to deploy a process:

1. Open the Orchestrator web page.
2. Click the **Processes** option on the left side.
3. **Processes** windows will appear on the web page.
4. Now click the + button to add a package. The **Deploy Process** windows will be displayed.
5. Now choose the desired package name from the drop-down list (package here corresponds to the project you have published from UiPath Studio).
6. Here, the description option is optional.

7. Finally, click the **CREATE** button to deploy the process:

A screenshot of a 'Deploy Process' dialog box. The title bar says 'Deploy Process'. Below the title, there is a 'Package Name' label followed by a text input field containing 'No process selected.'. Below this is a list box with four items: '0-00-000-0000-aUITest', '0-00-000-0000-atul', '0-00000edureka', and '00000000000000-SAPPO'. The last item is highlighted. Below the list box is a dropdown menu currently showing 'enm'. Below the dropdown is a 'Description' label followed by a text input field. At the bottom right of the dialog are two buttons: 'CANCEL' and 'CREATE'.

Using Orchestration Server to control bots

When it comes to controlling a Robot Orchestrator is the best option. Orchestration Server can be used to schedule bots so that Robots can execute their jobs within the time

interval specified on the server. Orchestrator can control an unlimited number of Robots and we can very easily assign various tasks to the Robots as per the user's needs. Also, a specific task can be assigned to multiple Robots. Orchestrator also gives us the facility to maintain all logs generated by the Robot.

Robot statuses

The status of a Robot tells us about its availability and connectivity. We get to know whether the Robot is available, busy, or disconnected. The following are the statuses that a Robot can have:

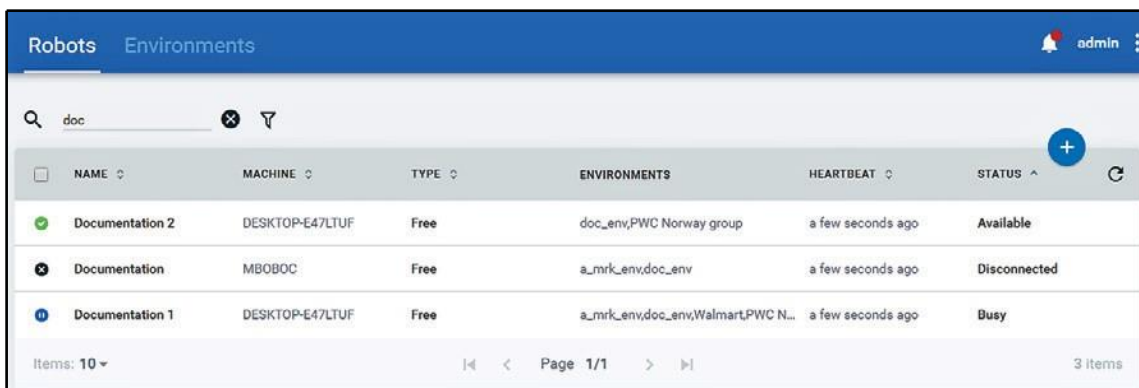
- **Available:** This status of the Robot shows that the Robot is not working on any other task and is freely available to take tasks
- **Busy:** This type of status will be shown when a Robot is currently executing some task and is not available
- **Disconnected:** This status of Robot shows that your Robot is no longer connected to Orchestrator Server

Editing the Robot

Sometimes we may have to edit the Robot in Orchestrator in case the Robot is either not functioning properly or we want to assign some other tasks to the Robot.

To edit the Robot, click on the Edit button from the Edit window and change the name or the necessary fields as given:

- Name
- Username
- Password
- Type
- Description:
-



	NAME	MACHINE	TYPE	ENVIRONMENTS	HEARTBEAT	STATUS
✓	Documentation 2	DESKTOP-E47LTUF	Free	doc_env,PWC Norway group	a few seconds ago	Available
✗	Documentation	MBOBOC	Free	a_mrk_env,doc_env	a few seconds ago	Disconnected
ⓘ	Documentation 1	DESKTOP-E47LTUF	Free	a_mrk_env,doc_env,Walmart,PWC N...	a few seconds ago	Busy

Items: 10 Page 1/1 3 items

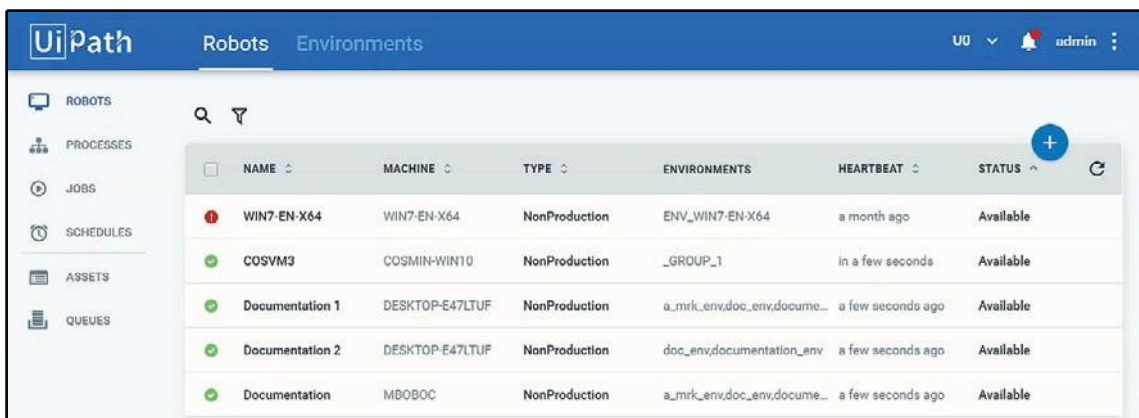
Deleting the Robot

When a Robot is not working at all, it is necessary to remove it, and we can use either of the following:

- To delete the desired Robot, select that Robot and go to the admin screen by clicking on the Admin tag on the top of the Orchestrator Server window. Then select the **More Action** button and delete the Robot.
- If you want to delete one or more Robots from the page, we can select them or remove them from the page.

Displaying logs for a Robot

To view logs of a Robot, go to the Robots page and search for your desired Robot, click on **More Actions**, and then click on **View logs** to view log messages from your Robot:



	NAME	MACHINE	TYPE	ENVIRONMENTS	HEARTBEAT	STATUS
❗	WIN7-EN-X64	WIN7-EN-X64	NonProduction	ENV_WIN7-EN-X64	a month ago	Available
✅	COSVM3	COSMIN-WIN10	NonProduction	_GROUP_1	in a few seconds	Available
✅	Documentation 1	DESKTOP-E47LTUF	NonProduction	a_mrk_env,doc_env,docume...	a few seconds ago	Available
✅	Documentation 2	DESKTOP-E47LTUF	NonProduction	doc_env,documentation_env	a few seconds ago	Available
✅	Documentation	MBOBOC	NonProduction	a_mrk_env,doc_env,docume...	a few seconds ago	Available

Using Orchestration Server to deploy bots

To deploy Robots to Orchestrator, we need to configure the machine with Orchestrator. To do this, we first have to create a provisioning Robot from the Orchestrator URL: <http://platform.uipath.com>.

Creating a provision Robot from the Orchestrator

The user needs; permissions to register the new Robot and they must have the following information:

- The name of the machine and the key required to connect to Orchestrator. They can be found through the **Control Panel | System**, for the key, log on to the Orchestrator URL, click on the **ROBOTS** page, then click on the + button. A pop up window appears where your key is visible. **Security | System** and **User | Settings | Deployment**, which provides us the Robot API Key.
- The username and password of the specified machine to access it.
- To create the provision Robot, go to the Orchestrator. Click on the Robot option on the left side of the Orchestrator page.
- After displaying the Robot page, click on the + button. It will display a small window to create a provision Robot:



- After the window is displayed, fill in all the mandatory information that is required to connect the Robot—machine name, Robot name, username/domain name, password, type, and description, as given in the following screenshot:

The screenshot shows a configuration window with two tabs: CONFIGURATION and RUNTIME. The CONFIGURATION tab is active. It contains several input fields and a checkbox. The fields are labeled: Key *, Machine *, Name *, Domain\Username *, Password, Type, and Description. The Key field contains the value 1a354725-847d-4e8b-92c3-898574f0f9ee. The Type field contains the value NonProduction. There is a checkbox labeled Create another. At the bottom, there are two buttons: CANCEL and PROVISION.

CONFIGURATION	RUNTIME
Key *	
1a354725-847d-4e8b-92c3-898574f0f9ee	
Machine *	
Name *	
Domain\Username *	
Password	
Type	
NonProduction	
Description	
☐ Create another	
CANCEL	PROVISION

- In the **Machine *** field section, enter the **machine** name that is required to connect to Orchestrator.
- From the **Name *** field enter the desired name of the Robot.
- Now in the **Domain\Username *** field, type the name that will be used to log in to the defined machine. Here, we need to define the format **Domain/Username *** if the user is in the domain. It is compulsory to choose a short domain name.
- **Password** field is optional; it can be skipped.
- We can choose the type of Robot from the drop-down list.
- **The description** field is also optional. You can give a short summary describing your Robot.
- Now copy the Robot **Key *** and paste the key in UiPath Robot while configuring your Robot.
- Click the **PROVISION** button and the Robot will be displayed on the Robot page.

Connecting a Robot to Orchestrator

When we deploy a Robot to Orchestrator, we must have the machine name and the key for each Robot. To keep the value of these fields, we can create another Robot from the **Provisional Robot** windows.

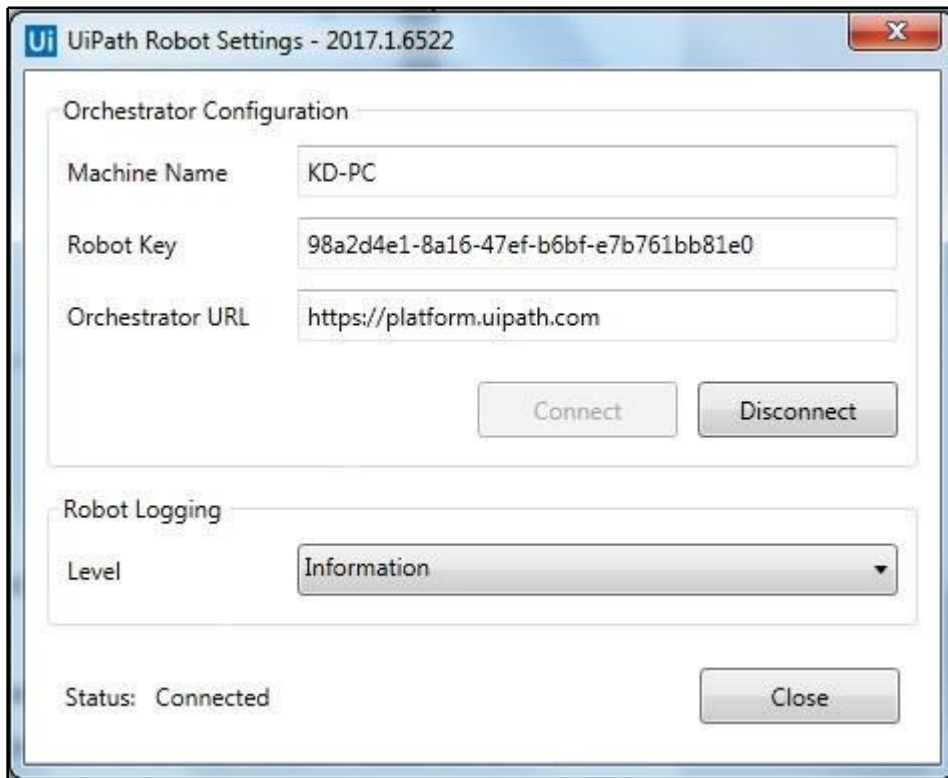
If we want to create a new Robot, we have to take authentication from administrator. By default, the administrator has the right to register a new Robot.

To connect the Robot machine to Orchestrator, follow the given steps:

1. From the system taskbar, click on UiPath Robot. The **Robots** windows will be displayed:



2. Now go to **Options** and choose **Settings...**; the Robot Settings will be displayed:

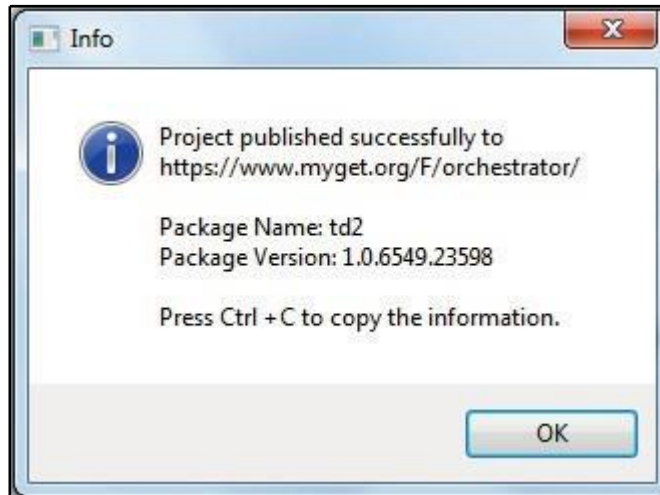


3. In the **Robot Key** field, paste the received key of provision Robot to Orchestrator.
4. In the **Orchestrator URL** field, enter the address of the Orchestrator.
5. Now click the **Connect** button. After clicking the **Connect** button, the Robot will connect to the Orchestrator.

Deploy the Robot to Orchestrator

To deploy our Robot, first of all, it must be connected to Orchestrator. Ensure that our bot is connected to Orchestrator then follow the given steps to deploy it:

1. First of all, install UiPath on the machine.
2. Provision the Robot machine and take the Robot key from Orchestrator.
3. After receiving the key go to the Robot configuration panel and enter the key here.
4. Also, you need to enter the Robot key into the configuration URL, which can be found from the admin section of Orchestrator.
5. Publish the project with publish utility from UiPath. When it is published successfully, it will display the information shown in the following screenshot:



6. The project has been published in the Orchestrator.

7. To create the environment, go to the home page, click on the **ROBOTS** option, and click on the **Environments** Tab. Then click on the **+** button:



8. Once the details are filled in, click on **Create**:

Create Environment

Name *

Type

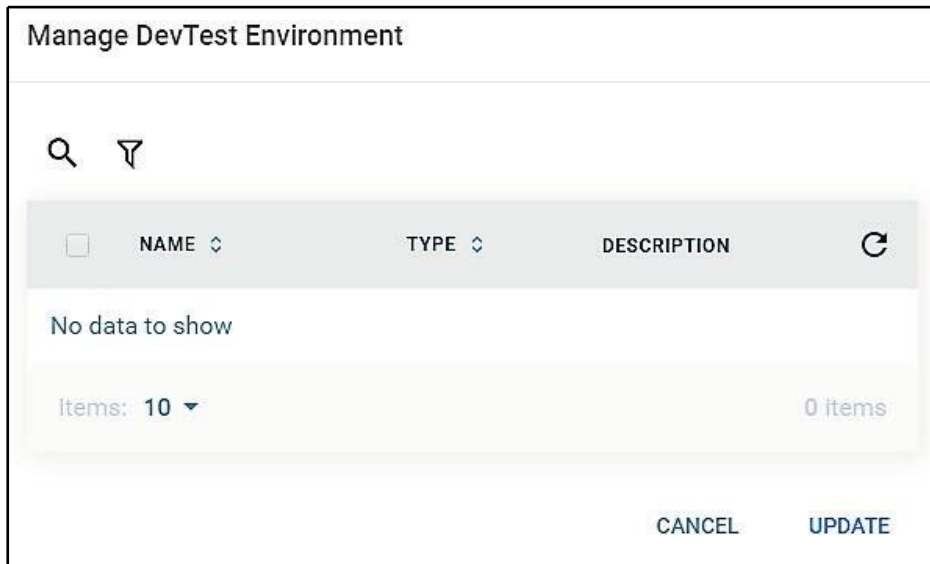
Dev

Description

CANCEL

CREATE

9. After creating the environment, a small window will appear as shown in the following screenshot, where we can manage the Robot within the environment:



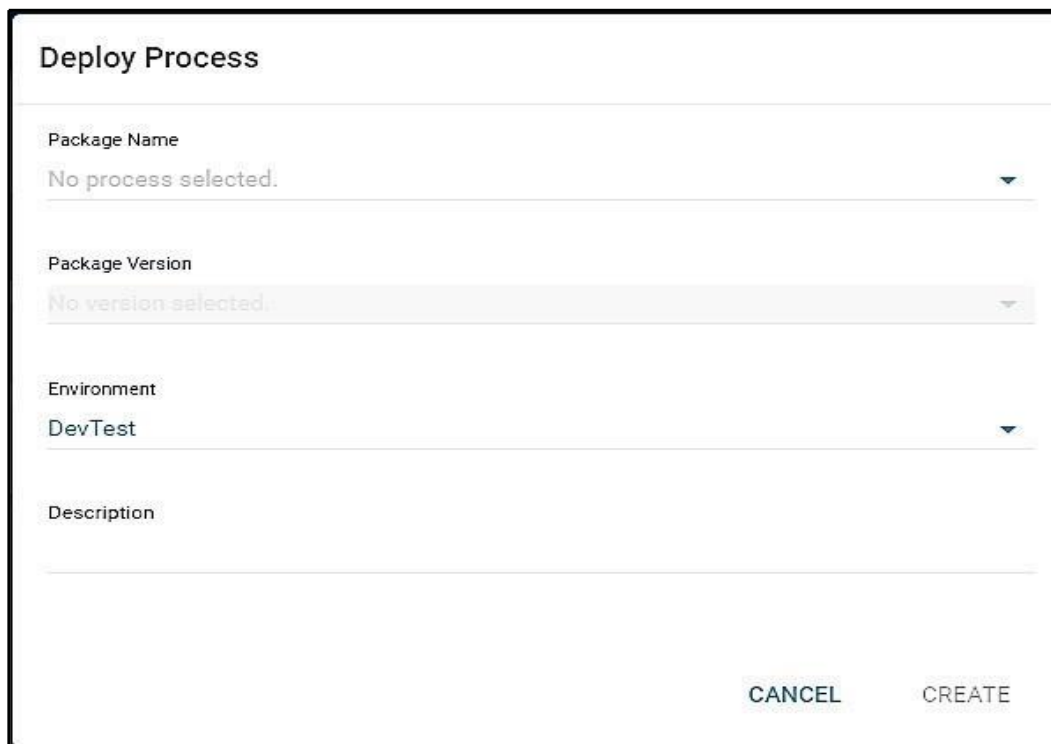
10. After clicking on the + button, a will window pop-up where we can choose the Published package, as shown in the following screenshot, and then click on the **CREATE** button:



11. After clicking the **Deploy Process** button, a window popup will appear where we can choose the published package, as shown in the following screenshot, and then click on the **CREATE** button.

OR

12. Packages can be manually uploaded from the local directory after clicking the **View Packages** option and then clicking the **Upload** button as given in the following screenshot: **PROCESS | View Packages | Upload Packages**:



The screenshot shows a 'Deploy Process' dialog box with the following fields and options:

- Package Name:** A dropdown menu with the text 'No process selected.' and a downward arrow.
- Package Version:** A dropdown menu with the text 'No version selected.' and a downward arrow.
- Environment:** A dropdown menu with the text 'DevTest' and a downward arrow.
- Description:** A text input field.
- Buttons:** 'CANCEL' and 'CREATE' buttons at the bottom right.

13. Now the package has been deployed to the Orchestrator and is ready to be executed through the web.
14. Next, click the **JOBS** option for execution and click on the **Start** icon as shown:

The screenshot shows the 'Start Job' dialog box. It features a 'Process' dropdown menu, search and filter icons, and a table with columns for 'ROBOT', 'MACHINE', and 'STATUS'. The table is currently empty, displaying 'No data to show'. At the bottom, there are 'Items: 10' and '0 items' labels, and 'CANCEL' and 'START' buttons.

15. After clicking the **Start Job** button, the Robot will execute over Orchestrator.

License management

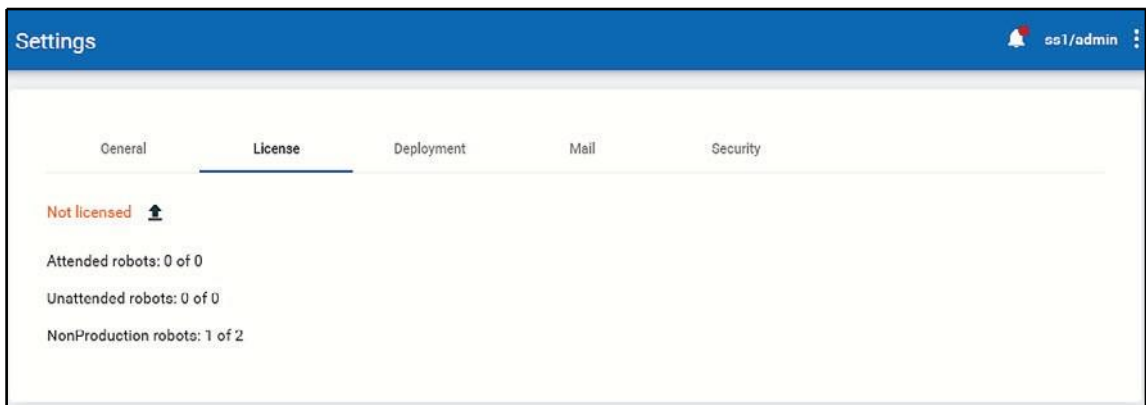
To manage and deploy bots, we are required to register the license on their server. Once you have received the license, the deployment and maintenance processes become faster.

Activating and uploading a license to Orchestrator

After getting the license code from the sales support team or any centralized location, we need to follow the given steps to activate the license and upload it to Orchestrator:

1. It is mandatory to have the UiPath Platform installed on the local machine.
2. If the UiPath Platform is not available then install it.
3. Now open the Command Prompt as an administrator account from the local machine.

4. Here we can change the directory for installing the path manually with the command `cd C:\Program Files (x86)\UiPath Platform\UiPath`.
5. To activate a license, we need the Regutil tool. If this tool is available, then type the following command to activate it through the command line: `regutil activate /email=emailaddress /code=licensecode`.
6. Now export the license information to the file using the command `regutil export-info /out_file=D:\license.txt`.
7. Go to Orchestrator and click the **Admin** option. Then select the **Settings** page from the drop-down list:



8. As the **Settings** page becomes visible, we can see the **License** tab, as shown in the preceding screenshot. Click on **License**. Then under the **License** portion of the page, we can see our available license with an option for uploading it. Click on the **Upload** option.

After successfully uploading the license, navigate to the license information, which is created by using the Regutil tool and the uploading methodology. Here, the license expiry date of each Robot can be checked. Whenever we connect a new Robot to Orchestrator, it consumes a new license.

Publishing and managing updates

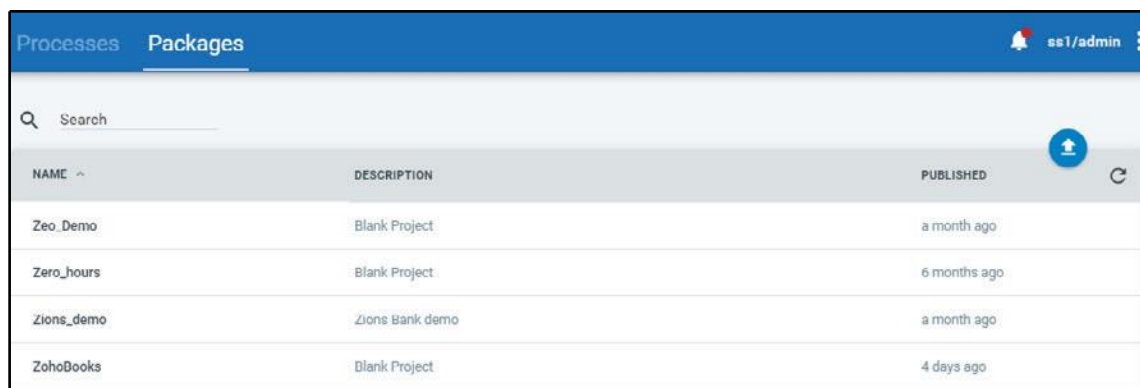
When we successfully create a workflow that can be used to perform certain automation, it should be published. This is necessary because, if we open UiPath Studio and run the workflow each time we require it, then it will consume a lot of time and also require human involvement to do it. This is not the right approach while automating. Hence, we publish our workflow so that it can run very easily using UiPath Robot or Orchestrator. Sometimes, we may want to make some changes to the workflow that we published earlier. For that, we have to make changes and publish it again so that the latest workflow is available on Orchestrator. This workflow is also updated to the latest version.

In this section, we will understand how to publish a project and how to update it.

Packages

When projects are published to Orchestrator from UiPath Studio, they become packages. These packages can be found on the **Process** page after clicking on the **Packages** option.

Published packages are shown in the following screenshot:



NAME ^	DESCRIPTION	PUBLISHED
Zeo_Demo	Blank Project	a month ago
Zero_hours	Blank Project	6 months ago
Zions_demo	Zions Bank demo	a month ago
ZohoBooks	Blank Project	4 days ago

Orchestrator provides the facility to update, view, or delete your packages from the **Packages** page. Every package in Orchestrator contains a version, its published date, and its description. Suppose we have a package and we want to make changes to that package, say, we have added some new functionality to that package from UiPath Studio and publish it again. In order to use the newest version of that uploaded package, we can view that package from the **Packages** page, as shown in the preceding screenshot, and then it will display all the available package versions for that package. In order to update the version, either select the version of your choice or click on **Get the latest version** to use the latest uploaded version of that package:

Manage versions for td2_enm

LATEST ROLLBACK

VERSION	PUBLISHED	USED	
1.0.6549.23598	3 hours ago		
1.0.6549.23513	3 hours ago		
1.0.6548.27322 Current	a day ago	a day ago	

Items: 10 ▾

|◀ < Page 1/1 > ▶|

3 items

CANCEL

In the preceding screenshot, the user can view and delete all published packages that were published to Orchestrator.

Package versions can have two statuses:

- **Active:** When a package version is in active mode then it means that the version is currently in use
- **Inactive:** If a package version is inactive then it means that this version is not in use

Managing packages

After a package has been created on the Orchestrator Server, we can easily view them in the

Packages tab on the **Processes** page. Here, we can upload or remove packages.

Uploading packages

When we publish our project to the Orchestration server, it will automatically send the package to the server. If we want, we can upload the package manually as well. For that, Orchestrator provides the facility to upload a project manually from the local machine. To upload packages, we need to follow the given steps:

1. Navigate to the **Processes** page, select the **Packages** option, and then click on Upload packages. A dialog will appear in the window like in the following screenshot:

Upload Package

Select or drop files

 **BROWSE**

** Please note that it may take a few minutes before your upload is propagated*

CANCEL UPLOAD

2. Now click the **BROWSE** button and select the packages from the local machine with a name for the uploaded file.
3. After selecting the proper package click on the **UPLOAD** button so that the package becomes available with the currently existing packages on the **Packages** page.

Deleting packages

When we no longer require a package then we can easily delete it. Make sure that the process is not in active mode. We can just select our package and click on the **Remove** button. We may also remove multiple packages at the same time by checking one or more packages from the list or by clicking on the **REMOVE ALL INACTIVE** tab to remove all packages that are inactive:

Package td2 Versions

REMOVE ALL INACTIVE

☐	STATUS	VERSION	PUBLISHED	↻
☐	Inactive	1.0.6549.23598	3 hours ago	
☐	Inactive	1.0.6549.23513	3 hours ago	
☐	Active	1.0.6548.27322	a day ago	

Items: 10 ▾ ⏪ < Page 1/1 > ⏩ 3 items

CLOSE

Summary

As we approach towards the end of the book, let us quickly go through what we learnt. We started off by learning about RPA, its scope, and some tools available for RPA. Then we delved into more details about UiPath, its components, and setting up UiPath Studio to train your own bot. You also got to know about the user interface for UiPath Studio before beginning to design your first Robot. Once comfortable enough, we explored a fascinating aspect of UiPath, that is, recording. In *Chapter 3, Sequence, Flowchart, and Control Flow*, we saw how a workflow was structured and the different types of project available in UiPath—when they are used and how to use them. *Chapter 3, Sequence, Flowchart, and Control Flow*, also introduced you to activities and how to manually drag and drop activities to make our workflow, all this in the user-friendly interface provided by UiPath Studio!

In Chapters 4 through 7, we went deeper into UiPath: we learned more about recording, about data manipulation, the various controls in UiPath, about extraction, selectors, OCR data scraping, and screen scraping. We also read about the various plugins available and about assistant bots.

That's not all; an important aspect of your automation journey is properly organizing your project, as well as being prepared for handling exceptions. All of these were dealt with in detail in *Chapter 8, Exception Handling, Debugging, and Logging*, and *Chapter 9, Managing and Maintaining the Code*.

Finally, you were taught how to deploy your bot.

From being a novice at the beginning of the book, you are now skilled enough to develop and deploy a bot! Your automation journey has begun!

8. Conclusions

Chapter 7

Conclusions

Abstract This chapter presents the final conclusions regarding disassembly automation in general and the cognitive robotic disassembly system developed in this research. The technical aspects of this research, the results and the lessons learned are summarised. In light of this, our conclusions and direction for future work are also presented.

7.1 Conclusion in Technical Perspectives

According to the principles of disassembly automation presented in Chap. 3, the system consists of three primary elements: the disassembly operation unit, vision system, and intelligent agent. In this research, these take the form of the disassembly operation modules (DOM), vision system module (VSM), and cognitive robotic module (CRM), respectively. The steps taken, and lessons learned in the implementation of these modules are summarised in the following sections. Finally, the method in which the system addresses the problem of various uncertainties, which has thus far hindered the implementation of disassembly automation, is addressed.

7.1.1 Disassembly Operation Module

The disassembly operation module deals with physical operations conducted in the disassembly process. The development involves three primary parts, including product analysis, hardware design, and operation plans.

Product analysis

In general, product structure information of specific models is required to select disassembly operations and an optimal disassembly sequence plan. Unfortunately, it is often the case that EOL products are returned where the specific information,

e.g. CAD model and assembly sequence, is unavailable. Such information can be generated from the examination of select models; however the effort to research and implement for a specific model tends to be unjustifiable in industry practice, where the products returned are in great variety and the lot sizes are unpredictable. In this research, the product family is analysed in order to identify the primary variations between models. The CRA is programmed to address these uncertainties and variations, eliminating the need for a priori model-specific information. A broad idea of the main product structure is given to guide the process; the concepts used are general enough to cover the variations between models but not too broad such to limit the size of the search space. In this case, the order of the main components in LCD screens is quite consistent; only two types of main structures are defined. The types and characteristics of the components—main and connective—also need to be identified in order to design the operation plans and required disassembly tools.

Mechanical system

The mechanical system as the disassembly operation unit is designed according to these requirements. In general, the current research trend focuses on two perspectives which are (a) the development of the entire disassembly system and (b) the development of specific disassembly tools for removing types of connectors as presented in Chap. 3. The primary components of the automatic disassembly workstation are robot arms equipped with disassembly tools, grippers, and the fixture system.

Automatic disestablishment of fasteners generally requires specifically-designed tools and accuracy of actions. Greater accuracy can be attained with force-torque control and/or closed-loop visual feedback. Detachment of main components is generally achieved using grippers. Complexity usually arises according to the variable geometry of the components and the requirements to create a firm grip. However, physical uncertainties are found in both cases resulting in difficulty in the removal process. If the failure of an operation can be identified, the system may be able to recover by appropriately executing an alternate control sequence.

In this research, the (semi-) destructive approach is employed for a higher success rate in the midst of uncertainties, requiring lower accuracy in force-torque and position control to achieve the goal of component separation. In addition, the complexity of the gripping and fixture system is eliminated by using the flipping table, which is able to remove detached components without any knowledge of their geometry. The proposed design simplifies the process and reduces the setup cost of the system.

Operation plans and process parameters

Where this information of a particular product is known in advance, the operation plans and parameters can be predefined. However, this is often infeasible as discussed earlier. The method of formulating and utilising operation plans and parameters is one of the primary contributions of this research. Only the broad scheme of operation plans and parameters are supplied, as choice points in the space that

the CRA searches while carrying out the disassembly process in a trial-and-error manner. As a result, the specific information of the product does not need to be supplied a priori.

In this research, the general operation plans for each type of component is formulated from statistical information of possible locations of the corresponding connective components. Given that the partially damaged disassembly outcome is acceptable, instead of destroying the connective component (the semi-destructive approach), the most effective operations consist of cutting near the border of the main component (the destructive approach) to detach the majority of its material. A major benefit is to compensate the overhead vision system's inability to detect fasteners such as hidden snap-fits and screws located around the sides. This method is also able to identify the process parameters, e.g. the critical depth of cut for successful removal of a section, which is also unobservable from the outside.

7.1.2 ***Vision System Module***

Vision is generally the primary sensing method in disassembly automation. Its main function is to detect—recognise and localise—the components in the disassembly process. Sensing capabilities are crucial for the disassembly process since certain information is only disclosed during the disassembly process. Therefore, the current condition of the process needs to be updated for appropriate decisions to be made by the planner. In this research, the vision system module is able to address the uncertainties of physical appearance, quantity, and location of the component. The vision system needs to be considered in two perspectives: hardware capability and algorithms.

Hardware capability

A number of techniques with different advantages and limitation are currently used for sensing the information in 2D, 2.5D, and 3D, as discussed

in Chap. 4. The techniques must be selected according to the requirements of the disassembly process. In this research, the colour and the depth camera is used for generating a 2.5D image map. This selected option is more cost effective and consumes less computational resources in comparison to other techniques. Data loss occurred in the infrared-based depth sensor at reflective surfaces perpendicular to the infrared emitting direction. Additionally, accuracy is reduced at the edges of the object. These problems are partly eliminated by a filtering algorithm that disregards the irrelevant data. The inaccuracy at the edges is taken into account by setting appropriate cutting offsets in the operation plans.

Detection algorithms

There is no general solution that is effective for detecting every type of component. A detector needs to be developed targeting particular types of components, in most cases, by combining a number of machine vision and pattern recognition techniques. The problem of detection becomes more difficult due to the

unpredictable EOL condition, e.g. damage, partial occlusion and missing parts. The detector must also be able to address these issues.

In this research, the rule-based recognition of the components using the concept of *common features* addresses these variations in appearance. This recognition technique is one contribution of this research. Predefined rules are developed according to the common physical appearances of each type of component observed from the samples. The developed algorithms are able to accurately recognise the type of component and the location. This is flexible enough to deal with the variations between different product models of products, however shows varying performance according to the degree of damage to the component.

Another important function of the vision system in this research is to assess the success of the operations. This assessment is part of the execution monitoring which is done by checking for a change of disassembly state. This measurement is designed to support destructive disassembly, in which success is defined as the removal of a significant part of the component, rather than its entirety. The detector is developed based on similarity measurements based on both the colour and depth images. This has been shown to achieve 95 % accuracy in determining state change in this application.

7.1.3 *Cognitive Robotics Module*

The intelligent agent controls the automatic disassembly system to perform the disassembly process. To make the system robust and flexible, the agent needs to take existing knowledge and plans and adapt it according to the currently-sensed information. A number of examples can be seen from the existing research work associated with adaptive planning in Chap. 2 and the disassembly systems described in Chap. 3.

In this research, the cognitive robotic agent controls the system using four cognitive functions: (a) reasoning, (b) execution monitoring, (c) learning, and (d) revision. The uncertainties regarding product structure

and process are addressed by this agent. The CRM is composed of the CRA and the KB. The CRA controls the behaviour of the system and the KB contains the model-specific knowledge that is obtained from the previous disassembly processes. A key benefit of using cognitive robotics in this research is the ability to make decisions according to actual execution outcomes. The system improves from previous experience by learning new instructions given by the human operator. These features give the system flexibility to deal with various models of products, by addressing the uncertainties at both the planning and operational levels.

Architecture and language platform

The cognitive robotic architecture is based on the closed perception-action loop which expresses the key features of the behaviour, i.e. perception, action, reasoning, learning, planning, behaviour control, and human interaction. The CRA is

programmed with action programming language *IndiGolog*, which is based on Situation Calculus. The key benefit to this research is the online execution which supports sensing and exogenous actions which allow the CRA to effectively respond to the external dynamic world. The language of the *Golog* series also benefits the development process, as the behaviour of the system can be clearly described by actions, preconditions, and effects.

Reasoning and execution monitoring

The CRA schedules actions by reasoning about the current condition of the disassembly state, the disassembly domain, and the execution outcome. The existing model-specific knowledge is also taken into account for disassembling the known models. In addition, the CRA can decide to switch to user assistance when the autonomous operations have failed too many times.

For the *unknown model*, the key feature of reasoning is to select the operation plans and parameters according to the current main component. They are considered choice points to be pruned along with the two main structure definitions. The input is obtained by the component detectors and the execution outcome which is determined by the execution monitoring that examines the change of disassembly state. This input is used in the trial-and-error process in order to find the critical plan and parameters that lead to the state change. As a result, this can eliminate the need of the disassembly sequence plan (DSP) and disassembly process plan (DPP) supplied a priori. The CRA will also learn these generated DPP and DSP. This also addresses the uncertainties due to the variation in the quantity of the components, e.g. of PCBs. In the case of the *known model*, reasoning is used to execute the operation according to the knowledge previously learned. The sensing input in regard to the component type and location is less significant in this process since the information is already known.

The CRA performs the disassembly according to the order of the states defined by the given main structure. These predefined structures benefit the reliability of the process by reducing the effect of the misclassification of main components, which is more frequent when the components are damaged. Effects of misclassification include infinite execution loops and

redundant physical damage, leading to increased time consumption and the learning of incorrect information. However, a major drawback of using given broad structures lies in the inability to handle a product whose structure differs significantly from the given definition. This may be the case in other product families where the structure is more complex. However, this has not been found to be a problem in the observed case-study products.

Learning and revision

In this research, learning occurs in two forms. First, in *learning by reasoning*, the CRA learns the parameters for the predefined general operation plans that have been executed prior to the successful component removal. The critical values of all executed operations need to be recorded even if the state change does not immediately occur, since some cuts may passively contribute to the detachment. Second, in *learning by demonstration*, the CRA learns from the assistance

given to overcome unresolved problems. Assistance may be given to change original beliefs caused by inaccurate visual detection, e.g. regarding the existence of a main component or the occurrence of state change. In addition, assistance is given in the form of additional sequences of primitive cutting operations (custom plans), which may disestablish the remaining connections that are non-detectable or require deeper cuts.

A major benefit of learning is to reduce the need for assistance for disassembling known models. The time consumption is also marginally reduced by skipping redundant steps, e.g. flipping the table and visual sensing. A limitation of the described strategy is that knowledge cannot be adapted between different models. Therefore, specific information is generally supplied by the human user during the first disassembly of every unknown model. However, the experimental results prove that the CRA requires significantly less human intervention in subsequent appearances of the model, due to learning and executing taught steps autonomously.

In the *revision* process, the disassembly process of known models can be optimised by retracting the redundant general operation plans that have been learned previously. Redundant operations, which do not contribute to the removal of the main component, can be found by executing the operation plans in an alternative order and checking for the detachment of the component before all actions in the plan have been executed. A simplified form of this concept is implemented in this research, by reversing the order of operations in the plan.

Experiments show a significant improvement in process efficiency after a small number of appearances of the same model. The time consumption reduced by more than 50 % from the first appearance and the process was able to be carried out without assistance after the first few revisions. However, small fluctuations occurred due to inaccuracies in the visual localisation and physical operation.

In conclusion, the performance of the system significantly improves in every aspect after the first few tries to disassemble a particular model. This is due to the learning and revision strategy which is able to obtain the necessary specific model information during online operation. Subsequently, the process can be conducted largely autonomously and in a robust manner. This strategy has not previously been trialed in other research work.

7.1.4 Flexibility to Deal with Uncertainties

In this research, a system has been built to deal with the uncertainties that have prevented the implementation of disassembly automation for the treatment of EOL electrical/electronic waste. The uncertainties can typically be addressed autonomously by the integration of the operation modules. Principles that have been applied in this system include:

- identifying broad, abstract product structures that describe the entire range of known models;
- detecting components by type;

- supplying and pruning the search space of a range of possible operations, created in consideration of the entire product range; and
- learning from assistance, which is only provided in cases that are unresolvable by the autonomous system.

The primary uncertainties listed in Table 3.1 are discussed as follows.

First, the *uncertainties in EOL condition* that cannot be observed by the vision system are expected to be addressed by the CRM and DOM. As a result, the cutting location cannot be determined accurately. The general operation plans, which are part of DOM and used by CRM, cut the main component at the estimated location that expects to disestablish the connection. The force-torque feedback is used to acknowledge collisions and find alternative cutting method to achieve the operation. Second, the *diversity of the supplied products* concerns the variation in component appearance, quantity and location between models of the same product. This is addressed by detecting components by type, as opposed to using a specific template. The broad structure category that each sample falls under is identified by the CRA according observations during the disassembly process. The success of addressing these uncertainties depends on the performance of the vision system.

Third, uncertainties in the required *process and operations* plan is typically addressed by the CRA, which uses reasoning and execution monitoring to obtain effective sequence plans, operation plans, and process parameters through trial-and-error. Errors due to sensor or actuator limitations are addressed in the same way as uncertainties in the EOL condition, compensated by the trial-and-error strategy or assistance. The success of addressing these uncertainties depends on the accuracy of the predefined structure and operations, and the constraints of the process parameters. The ability of the system in dealing with these uncertainties has been validated experimentally on numerous models of LCD screens. The success rate of the automatic operation can be increased if less strict constraints are assigned e.g. deeper maximum cutting depth. This will lead to a greater potential for the trial-and-error approach to complete the task, at the expense of increased time consumption. The amount of uncertainty and the need for assistance reduces significantly after the

model-specific knowledge has been learned.

The integration of these principles is a starting point in bridging a gap in existing research work, where previously only known models can be disassembled by an autonomous system.

7.2 Economic Feasibility

In practice, one of the major problems of disassembly in industrial scenario is the unpredictable quality and quantity of the products returned. The fully automatic disassembly system can be economically feasible if:

- The system supports a wide range of products;
- Enough uncertainties can be addressed autonomously;

- The process is reasonably fast;
- Human operators' direct exposure to hazards can be avoided; and,
- High value can be recovered from the disassembly outcome.

The system developed in this research addresses some of these issues. Economic feasibility is preliminarily evaluated in two perspectives, including the cost of the automation platform and the operating cost.

First, a low-cost platform that is flexible to deal with a wide range of models of LCD screens has been designed in this research. Using the destructive approach and specially designed tools, the system achieves a high success rate of disassembly. Regarding value recovery, the damaged condition of the disassembly outcome is suitable only for recycling. The system should aim for non-destructive or semi-destructive disassembly to acquire higher value returned.

Second, time consumption is a key concern which directly relates to the operating cost. In this, the current prototype system still needs further improvement. In comparison to a comparable manual process which takes 6.2 min/screen on average [1], the proposed system takes approximately 48 min for an unknown model sample, which reduces to around 24 min after a few revisions. Improvements in regard to physical operation and hardware are needed to overcome this limitation. It is a goal for the disassembly time to be reduced to less than 10 min.

7.3 Conclusions of the Research

7.3.1 *Conclusions*

In regard to the development of disassembly automation, the flexibility to deal with various models of products is crucial for the industrial application. This research shows the possibility of making disassembly automation economically feasible by using the concept of cognitive robotics together with the associated operating modules. In addition, learning and revision are key features that allow the system to improve the process performance

from previous experience. Even though the human needs to be involved in the first stages upon receiving an unknown product, the system learns from this to become more autonomous.

7.3.2 ***Future Work***

Regarding future work, the system should be further developed towards economic feasibility. A primary goal is to bring the physical system near the efficiency and flexibility of manual disassembly through the improvement of the hardware and the optimisation of operations. In addition, the following additions are foreseen for extending the system to support a wider range of products.

The basic behaviour of the cognitive robotic agent will become more complex, in order to address the uncertainties in a wider range of products. A limitation in the current learning and revision strategy is that newly-generated knowledge is specific to individual models. Therefore, a strategy that allows the robot to adapt the existing knowledge of one model to disassemble another model should be developed. This implementation can take the form of learning broad rules relating learned operations or parameters to observations, so that with experience, the system also increases in its ability to autonomously handle unknown models.

The mechanical system will be extended to include a variety of disassembly tools and grippers to facilitate the non-destructive disassembly of select components. This is desirable for increasing the economic gains from the disassembly outcome, since undamaged components can re-enter the product stream as spare parts or for remanufacturing. It is desirable for aspects of the system to be modular; system components such as the product fixture may need to be re-designed to hold products from a different family. Increased reliability is desirable for both the disassembly operation unit and the vision system.

Reference

1. Kernbaum S, Franke D, Seliger G (2009) Flat screen monitor disassembly and testing for remanufacturing. *Int J Sustain Manuf* 1(3):347–360

9. References

10. Absolute Vacuum. (2006). www.absolutevacuum.co.uk (14 November 2006)
- 11.
12. Blum-Evitts, Shemariah. (2000). Business Recycle-Link. WasteCap of Massachusetts. (Vol. 6, 2000)
- 13.
14. British Standards Institute. (2002). *Environmental Management Integrating Environmental Aspects into Product Design and Development*.
- 15.
16. Carwood. (2006). <http://www.carwood.co.uk/Remanufacturing.aspx> (12 December
17. 2006).
- 18.
19. The Centre for Sustainable Design. (2006). *Expert Remanufacturing Workshop*. London. (Workshop at The Brunei Gallery, 27 November 2006).
- 20.

21. Chalkley, A., Billett, E., Harrison, D. and Simpson, G. (2003). Professional Engineering Publishing. *Development of a method for calculating the environmentally optimum lifespan of electrical household products*, 217(11).
- 22.
23. Chartered Institute of Environmental Health. (2005). http://www.cieh.org/ehn/news_focus/2005/june/articles/something_disturbing_in_the_air.htm (17 February 2007).
- 24.
25. Cosgrove, Andy. (2007a). [Interview by Gray, C, 09 January 2007]

Cosgrove, Andy. (2007b). [Interview by Gray, C, 20 February 2007]

Chiodo, Joe. (2007). [Interview by Gray, C, 2 January 2007]
26. Clements, Vic. (2007). Electronics Weekly.com. <http://www.electronicsworld.com/ARTICLES/2007/02/08/40705/Business+needs+to+know+about+WEEE+Directive.H TM> (20 February 2007)
- 27.
- 28.
29. Crofts, Derek. (2006). [Interview by email, 30 October 2006]
- 30.
31. Design Council. (2005). The Business of Design – Design Industry Research 2005.
- 32.
33. Dowie-Bhamra, Tracy. (date unknown). *Design for Disassembly*. In: <http://www.co-design.co.uk/design.htm> (15 December 2006)
- 34.
35. DTI. (2006). <http://www.dti.gov.uk/consumers/fact-sheets/page24700.html> (13 December 2006)
- 36.
- 37.
38. Engineering Talk. (2006). <http://www.engineeringtalk.com> (15 December 2006)
- 39.
40. Essex, Jonathan. (2006). [Interview by email, 2 October 2006]
- 41.
42. First Sustainable. (2004). <http://www.firstsustainable.com/view.php?show=1569> (20/02/07)

43. Flextronics. (2006). <http://www.flextronics.com> (15 October 2006)
- 44.
45. Fraunhofer Institute. (2006). <http://www.iese.fraunhofer.de> (16 October 2006)
- 46.
47. Freeman, Steve. (2006). Resource Efficiency Network – Bulletin. C-Tech Innovation, 1 December 2006.
- 48.
49. Freiburger, Stefan. (2006). [Interview by Gray, C, 22 November 2006]
- Giuntini, Ron. (2006). [Interview by Gray, C, 8 December 2006] Gould,
- Ray. (2006). [Interview by Gray, C, 1 December 2006.
50. Hammond, Rick., Amezcua, Tony and Bras, Bert. (1998). *Issues in the Automotive Parts Remanufacturing Industry – A Discussion of Results from Surveys Performed among Remanufacturers*. The George W. Woodruff School of Mechanical Engineering Georgia Institute of Technology.
- 51.
52. Harrison, David. (2006). [Interview by Gray, C, 8 December 2006]
- Hanover. (2006). <http://www.hanover-co.com> (8 December 2006)
53. Harding, Adrian. (2006). *The Resource Efficiency KTN: WEEE Directive Compliance*. Webconference (Lecture at www.resource-efficiency.org, 15 December 2006).
- 54.
55. Hauser, William and Robert Lund. (Date unknown). *An American Resource.ppt*.
56. Boston University
- 57.
58. Haynes, Kevin. (2006). [Interview by email, 30 September 2006]
- 59.
60. King, Andrew and Ijomah, Winnie. (date unknown) Reducing End-of-life Waste: Repair, Recondition, Remanufacture or Recycle?
- 61.
62. King, Andrew. (2006a). [Interview by email, 11 December 2006]
- 63.
64. King, Andrew. (2006b). Comments on CfSD Remanufacturing Workshop Notes. (11 December 2006)
- 65.

66. Kurk, Fran and McNamara, Curt. (date unknown). *Better by Design – An Innovation Guide*.
- 67.
68. LaserXChange. (2006). <http://www.laserxchange.co.uk> (6 November 2006)
- 69.
70. Leppard, Guy. (2006). [Interview by email, 5 December 2006]
- 71.
72. Maddox, Peter. (2006). [Interview by email, 2 October 2006]
- 73.
74. MAS. (2006). [http://www.mas.dti.gov.uk/content/resources/categories/case-](http://www.mas.dti.gov.uk/content/resources/categories/case-study/cs_Lynwood_Engineering.html)
75. [study/cs_Lynwood_Engineering.html](http://www.mas.dti.gov.uk/content/resources/categories/case-study/cs_Lynwood_Engineering.html) (13 November 2006)
- Mayers, Kieren. (2007). [Interview by Gray, C, 15 February 2007]
76. Milliken Carpet. (2006). <http://www.millikencarpet.com> (2 November 2006)

77. Morley, Nick. (2006). Presentation of "The Potential of Remanufacturing to Increase Resource Efficiency", Japan (2006)
- 78.
79. Morely, Nick. (2007). [Interview by email, 10 January 2007]
- 80.
81. Morley, Nick and Parker, David. (2007). *Comments on Draft*. (4 January 2007) Morrell, John. (2006). [Interview by email, 6 October 2006]
82. Nasr, Nabil and Thurston, Michael. (2006). *Remanufacturing: A Key Enabler to Sustainable Product Systems*. Rochester Institute of Technology.
- 83.
84. Oakdene Hollins Ltd. (2004). *Remanufacturing in the UK: A Significant Contributor to Sustainable Development?* Skipton: The Resource Recovery Forum.
- 85.
86. Parker, David. (2006). [Interview by Gray, C, 18 September 2006]

Parker, David. (2007). [Interview by Gray, C, 8 January 2007] Perkins

Engines. (2006). <http://www.perkins.com> (30 October 2006)
87. Plumeyer, Martin. (2006). [Interview by email, 20 November 2006]

Price, Kati. (2006). [Interview by Gray, C, 16 November 2006]

Rayner, Eddie. (2006). [Interview by Gray, C, 14 November 2006]
88. Recharger Magazine. (2006). <http://www.rechargermag.com> (20 October 2006)
- 89.
90. SEEDA. (2006). *Review of Regional Economic Strategy for South East England 2006-2016 Consultation Document*.
- 91.
92. Seddon, Jackie. (2006). [Interview by email, 4 October 2006]
- 93.
94. Siemens. (2006). <http://www.medical.siemens.com> (20 November 2006)
- 95.
96. Simms, Sir Neville. (2006). *Procuring the Future*. Sustainable Procurement Taskforce.
- 97.

98. Statham, Steve. (2006). *Remanufacturing Towards a Sustainable Future*. Pera International, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University.
- 99.
100. Steinhilper, Rolf. (1998). *Remanufacturing; The Ultimate Form of Recycling*.
101. Germany: Druckerei Hoffman.
- 102.
103. Steinhilper, Rolf. (2006). [Interview by Gray, C, 22 November 2006]
- 104.
105. Sundin, Erik. (2004). Product and Process Design for Successful Remanufacturing. PhD Dissertation, Linköping's Universitet.
- 106.
107. Sustainable Carpet. (2006).
[http://www.sustainablecarpet.com/MillikenCarpet
Website/Americas/Contract/Corporate/Overview/default.htm](http://www.sustainablecarpet.com/MillikenCarpetWebsite/Americas/Contract/Corporate/Overview/default.htm) (2 November 2006)

-
108. The Manufacturer. (2004). http://www.themanufacturer.com/britishindustry/content_page.html?article_id=15 (12 December 2006)
- 109.
110. The Remanufacturing Institute. (2006). <http://www.reman.org> (1 December 2006)
- 111.
112. The White House. (2004). <http://www.whitehouse.gov/news/releases/2004/06/20040610-53.html> (16 November 2006).
- 113.
114. Turner Engineering. (2006). <http://turner-engineering.co.uk> (14 November 2006) Turner, Frida. (2006). [Interview by Gray, C, 14 November 2006]
115. UKCRA. (2006). <http://www.ukcra.com/> (13 November 2006)
- 116.
117. Webb, Roy., Kelly, John and Thomson, Derek. (1998). *Designing for Short Life: the Emerging Need for Packaged Reusable Building Services Components in the UK Healthcare Sector*. Department of Building Engineering and Surveying, Heriot-Watt University.
- 118.
119. Wikipedia. (2006a). *Remanufacture*. <http://www.wikipedia.com> (11 September 2006) Wikipedia. (2006b). *Reconditioning*. <http://www.wikipedia.com> (11 September 2006) Wikipedia. (2006c). *Reuse*. <http://www.wikipedia.com> (11 September 2006) Wikipedia. (2006d). *Repair*. <http://www.wikipedia.com> (11 September 2006) Wikipedia. (2006e). *Recycle*. <http://www.wikipedia.com> (11 September 2006) Wikipedia. (2006f). *Design Engineer*. <http://www.wikipedia.com> (1 November 2006) Wikipedia. (2006g). *Product Designer*. <http://www.wikipedia.com> (1 November 2006) Wikipedia. (2006h). *Engineer*. <http://www.wikipedia.com> (1 November 2006)

120. Xerox. (date unknown a). Film: Remanufacturing at

Xerox.mpeg. Xerox Xerox. (date unknown b). *Reman.ppt*.

Xerox
121. Xerox. (2004). Environment, Health & Safety Report 2004, in <http://www-rohan.sdsu.edu/faculty/dunnweb/rprnts.xerox2004report.pdf> (20 February 2007)
- 122.
123. Yellow Pages. (2006). www.yell.com (4 January 2007)
- 124.
- 125.
126. Allen, Colin, Varner, Gary, and Zinser, Jason (2000). "Prolegomena to Any Future Artificial Moral Agent", *Journal of Experimental and Theoretical Artificial Intelligence* 12.3:251–261.
- 127.
128. Anderson, Michael, and Anderson, Susan Leigh (2007). "Machine Ethics: Creating an Ethical Intelligent Agent", *AI Magazine* 28.4: 15–26.
- 129.
130. Arkin, Ronald C. (1998). *Behavior-Based Robotics*, Cambridge: MIT Press.
- 131.
132. Arkin, Ronald C. (2007). *Governing Lethal Behavior: Embedding Ethics in a Hybrid Deliberative/Hybrid Robot Architecture*, Report GIT-GVU-07-11, Atlanta, GA: Georgia Institute of Technology's GVU Center. Last accessed on September 15, 2008: <http://www.cc.gatech.edu/ai/robot-lab/online-publications/formalizationv35.pdf>
- 133.
134. Asaro, Peter (2007). "Robots and Responsibility from a Legal Perspective", Proceedings of the IEEE 2007 International Conference on Robotics and Automation, Workshop on RoboEthics, April 14, 2007, Rome, Italy. Last accessed on September 15, 2008: <http://www.peterasaro.org/writing/ASARO%20Legal%20Perspective.pdf>
- 135.
136. Asaro, Peter (2008). "How Just Could a Robot War Be?" in Adam Briggles, Katinka Waelbers, and Philip Brey (eds.) *Current Issues in Computing and Philosophy*, pp. 50-64, Amsterdam, The Netherlands: IOS Press.
- 137.

-
138. Asimov, Isaac (1950). *I, Robot* (2004 edition), New York, NY:

Bantam Dell. Asimov, Isaac (1957). *The Naked Sun*, New York, NY:

Doubleday.
139. Asimov, Isaac (1985). *Robots and Empire*, New York, NY: Doubleday.
- 140.
141. BBC (2005). "SLA Confirm Spy Plane Crash", *BBC.com*, October 19, 2005. Last accessed on September 15, 2008:
http://www.bbc.co.uk/sinhala/news/story/2005/10/051019_uav_vavunia.shtml
- 142.
143. BBC (2007). "Robotic Age Poses Ethical Dilemma", *BBC.com*, March 7, 2007. Last accessed on September 15, 2008: <http://news.bbc.co.uk/2/hi/technology/6425927.stm>

- 144.
- 145.
146. Bekey, George (2005). *Autonomous Robots: From Biological Inspiration to Implementation and Control*, Cambridge, MA: MIT Press.
- 147.
148. Brooks, Rodney (2002). *Flesh and Machines*. New York: Pantheon Books.
- 149.
150. Canning, John, Riggs, G.W., Holland, O. Thomas, Blakelock, Carolyn (2004). "A Concept for the Operation of Armed Autonomous Systems on the Battlefield", *Proceedings of Association for Unmanned Vehicle Systems International's (AUVSI) Unmanned Systems North America*, August 3-5, 2004, Anaheim, CA.
- 151.
152. Canning, John (2008). "Weaponized Unmanned Systems: A Transformational Warfighting Opportunity, Government Roles in Making it Happen", 2008 American Society of Naval Engineers' (ASNE) Proceedings of Engineering the Total Ship (ETS) Symposium, September 23-25, 2008, Falls Church, VA.
- 153.
154. Čapek, Karel (1921). *R.U.R.* (2004 edition, trans. Claudia Novack), New York, NY: Penguin Group.
- 155.
156. CBS (2007). "Robots Playing Larger Role in Iraq War", October 21, 2007 news report. Last accessed on September 15, 2008:
<http://cbs3.com/topstories/robots.iraq.army.2.410518.html>
- 157.
158. Churchland, Paul (1995). *The Engine of Reason, the Seat of the Soul: A Philosophical Journey into the Brain*, Cambridge, MA: MIT Press.
- 159.
160. Clarke, Roger (1994). "Asimov's Laws of Robotics: Implications for Information Technology," *IEEE Computer* (part 1: December 1993, pp. 53–61; part 2: January 1994, pp. 57–66).
- 161.
162. Coffee, Jr., John C. (1981). "'No Soul to Damn: No Body to Kick': An Unscandalized Inquiry into the Problem of Corporate Punishment," *Michigan Law Review*, Vol. 79, No. 3, pp. 386-459.

163.

164. Computer Professionals for Social Responsibility (2008). "Technology in Wartime" conference, January 26, 2008, Stanford, CA. Last accessed on September 15, 2008: <http://technologyinwartime.org/>

165.

166. Davis, Burke (1980). *Sherman's March: The First Full-Length Narrative of General William T. Sherman's Devastating March through Georgia and the Carolinas*, New York, NY: Random House.

167.

168. DeMoss, David (1998). "Aristotle, Connectionism, and the Morally Excellent Brain", *The Proceedings of the Twentieth World Congress of Philosophy*, August 10-15, 1998, Boston, MA.

- 169.
- 170.
171. DesJardins, Joseph (2003). *An Introduction to Business Ethics*, pp. 99-103, Columbus, OH: McGraw- Hill.
- 172.
173. Dilov, Lyuben (1974). *The Way of Icarus*. Дилов, Любен. Пътят на Икар. Захари Стоянов. ISBN 954- 739-338-3.
- 174.
175. Fikes, Richard and Nilsson, Nils (1971). "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", *Artificial Intelligence* 2(3-4): 189-208.
- 176.
177. Foot, Phillipa (1972). "Morality as a System of Hypothetical Imperatives", *The Philosophical Review*
178. 81.3: 305-316.
- 179.
180. Garreau, Joel (2007). "Bots on the Ground", *Washington Post*, May 6, 2007. Last accessed on September 15, 2008: http://www.washingtonpost.com/wp-dyn/content/article/2007/05/05/AR2007050501009_pf.html
- 181.
182. Harrison, Harry (1989). "The Fourth Law of Robotics", in Isaac Asimov and Martin Harry Greenberg (eds.) *Foundation's Friends: Stories in Honor of Isaac Asimov*, New York, NY: Tor Books.
- 183.
184. Hemingway, Ernest (1935). "Notes on the Next War: A Serious Topical Letter", *Esquire*, Vol. 4, No. 3: 19, 156.
- 185.
186. Hew, Patrick (2007). "Autonomous Situation Awareness: Implications for Future Warfighting",
187. *Australian Defence Force Journal* 174: 71-87. Last accessed on September 15, 2008: <http://www.defence.gov.au/publications/dfj/index.htm>
- 188.
189. Hobbes, Thomas (1651). *Leviathan* (1982 edition), New York, NY: Penguin Group.

- 190.
191. Institute of Electrical and Electronics Engineers (2008). "International Conference on Advanced Robotics and its Social Impact" conference, August 23-25, 2008, Taipei, Taiwan. Last accessed on September 15, 2008: <http://arso2008.ntu.edu.tw/>
- 192.
193. International Federation of Robotics (2008). "International Robot Standards" page from International Federation of Robotics website. Last accessed on September 15, 2008: <http://www.ifr.org/modules.php?name=News&file=article&sid=20>

- 194.
195. Iraq Coalition Casualty Count (2008). "Deaths Caused by IEDs" and "U.S. Deaths by Month" webpages. Last accessed on September 15, 2008:
<http://icasualties.org/oif/IED.aspx> and <http://icasualties.org/oif/USDeathByMonth.aspx>
- 196.
197. Johnson, Robert (2008). "Kant's Moral Philosophy", The Stanford Encyclopedia of Philosophy, Fall 2008 Edition. Last accessed on September 15, 2008:
<http://plato.stanford.edu/archives/fall2008/entries/kant-moral/>
- 198.
199. Jónsson, Ari, Morris, Robert, and Pedersen, Liam (2007). "Autonomy in Space: Current Capabilities and Future Challenges", *AI Magazine* 28.4: 27-42.
- 200.
201. Joy, Bill (2000). "Why the Future Doesn't Need Us", *Wired* 8.04: 238-262.
202. Kahn, Paul (2002). "The Paradox of Riskless War," *Philosophy & Public Policy Quarterly*, Vol. 22: 2–8. Kant, Immanuel (1785). *Grounding for the Metaphysics of Morals* (1993 edition, translated by James W. Ellington), Indianapolis, IN: Hackett Publishing Co.
- 203.
- 204.
205. Kurzweil, Ray (1999). *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*, New York, NY: Viking Penguin.
- 206.
207. Kurzweil, Ray (2005). *The Singularity is Near: When Humans Transcend Biology*, New York, NY: Viking Penguin.
- 208.
209. Lee, Steven (2004). "Double Effect, Double Intention, and Asymmetric Warfare", *Journal of Military Ethics*, 3.3: 233-251.
- 210.
211. Levy, David (2007). *Love and Sex with Robots: The Evolution of Human-Robot Relationships*, New York, NY: HarperCollins Publishers.
- 212.

213. McIntyre, Alison (2004). "Doctrine of Double Effect", The Stanford Encyclopedia of Philosophy (Fall 2008 Edition). Last accessed on September 15, 2008:
<http://plato.stanford.edu/archives/fall2008/entries/double-effect/>
- 214.
215. Murray, Mary Elizabeth (2008). "Moral Development and Moral Education: An Overview", University of Illinois at Chicago website. Last accessed on September 15, 2008:
<http://tigger.uic.edu/~Inucci/MoralEd/overview.html>

216.

217. National Defense Authorization Act (2000). Floyd D. Spence National Defense Authorization Act for Fiscal Year 2001, Public Law 106-398, Section 220. Last accessed on September 15, 2008: <http://www.dod.mil/dodgc/olc/docs/2001NDAA.pdf>

218.

219. National Transportation Safety Board (2007). "NTSB Cites Wide Range of Safety Issues in First

220. Investigation of Unmanned Aircraft Accident", NTSB press release, October 16, 2007. Last accessed on September 15, 2008: <http://www.nts.gov/Pressrel/2007/071016b.htm>

221.

222. North American Computing and Philosophy (2008). "The Limits of Computation" conference, July 10- 12, 2008, Bloomington, IN. Last accessed on September 15, 2008: <http://www.ia-cap.org/na-cap08/index.htm>

223.

224. O'Brien, William V. (1981). *The Conduct of Just and Limited War*, New York, NY: Praeger Publishers.

225.

226. Oh, Daniel (2008). "The Relevance of Virtue Ethics and Application to the Formation of Character Development in Warriors", *The Army Chaplaincy* online journal, Spring-Summer 2008. Last accessed on September 15, 2008: <http://www.usachcs.army.mil/TACarchive/tacss08/tacss08oh7.pdf>

227.

228. Orend, Brian (2001). *Michael Walzer on War and Justice*, Montreal, Quebec: McGill-Queen's University Press.

229.

230. Orend, Brian (2002). "Justice After War", *Ethics & International Affairs*

16.1: 43-56. Orend, Brian (2006). *The Morality of War*, Peterborough, Ontario:

Broadview Press.

231. Padgett, Tim (2008). "Florida's Blackout: A Warning Sign?", *Time.com*, February 27, 2008. Last accessed on September 15, 2008: <http://www.time.com/time/nation/article/0,8599,1717878,00.html>

232.

233. Page, Lewis (2008). "US War Robots 'Turned Guns' on Fleshy Comrades", *The Register* (UK), April 11, 2008. Last accessed on September 15, 2008:
http://www.theregister.co.uk/2008/04/11/us_war_robot_rebellion_iraq/
- 234.
235. Royal United Services Institute (RUSI) for Defence and Security Studies (2008). "The Ethics of Autonomous Military Systems" conference, February 27, 2008, London, UK. Last accessed on September 15, 2008:
<http://www.rusi.org/events/past/ref:E47385996DA7D3/>
- 236.
237. Rowe, Neil C. (2008). "Ethics of Cyber War Attacks", in Lech J. Janczewski and Andrew M. Colarik (eds.) *Cyber Warfare and Cyber Terrorism*, Hershey, PA: Information Science Reference

- 238.
- 239.
240. Russell, Stuart J., and Norvig, Peter (2003). *Artificial Intelligence: A Modern Approach* (2nd ed.), Upper Saddle River, NJ: Prentice Hall.
241. Searle, John (1980). "Minds, Brains and Programs", *Behavioral and Brain Sciences* 3.3: 417-457
- Shachtman, Noah (2007). "Robot Cannon Kills 9, Wounds 14", *Wired.com*, October 18, 2007. Last
242. accessed on September 15, 2008: <http://blog.wired.com/defense/2007/10/robot-cannon-ki.html>
- 243.
244. Sharkey, Noel (2007a). "Robot Wars are a Reality", *The Guardian* (UK), August 18, 2007, p. 29. Last accessed on September 15, 2008: <http://www.guardian.co.uk/commentisfree/2007/aug/18/comment.military>
- 245.
246. Sharkey, Noel (2007b). "Automated Killers and the Computing Profession", *Computer* 40: 122-124. Last accessed on September 15, 2008: [http://www.computer.org/portal/site/computer/menuitem.5d61c1d591162e4b0ef1bd108bcd45f3/index.jsp?&pName=computer_level1_article&TheCat=1015&path=computer/homepage/Nov07&file](http://www.computer.org/portal/site/computer/menuitem.5d61c1d591162e4b0ef1bd108bcd45f3/index.jsp?&pName=computer_level1_article&TheCat=1015&path=computer/homepage/Nov07&file=profession.xml&xsl=article.xsl&)
247. [=profession.xml&xsl=article.xsl&](http://www.computer.org/portal/site/computer/menuitem.5d61c1d591162e4b0ef1bd108bcd45f3/index.jsp?&pName=computer_level1_article&TheCat=1015&path=computer/homepage/Nov07&file=profession.xml&xsl=article.xsl&)
- 248.
249. Sharkey, Noel (2008a). "Cassandra or False Prophet of Doom: AI Robots and War", *IEEE Intelligent Systems*, July/August 2008, pp. 14-17. Last accessed on September 15, 2008: http://www.computer.org/portal/cms_docs_intelligent/intelligent/homepage/2008/X4-08/x4his.pdf
- 250.
251. Sharkey, Noel (2008b). "Grounds for Discrimination: Autonomous Robot Weapons", *RUSI Defence Systems*, 11.2: 86-89.
- 252.
253. Simon, Herbert (1947). *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organizations* (1997 fourth edition), New York, NY: Free Press.

254.

255. Sofge, Erik (2008). "The Inside Story of the SWORDS Armed Robot 'Pullout' in Iraq: Update", *PopularMechanics.com*, April 15, 2008. Last accessed on September 15, 2008: http://www.popularmechanics.com/blogs/technology_news/4258963.html

256.

257. Solomon, David (1988). "Internal Objections to Virtue Ethics", in Peter A. French, Theodore Uehling, Jr., and Howard Wettstein (eds.), *Midwest Studies in Philosophy Vol. XIII Ethical Theory: Character and Virtue*, Notre Dame, IN: University of Notre Dame Press.

258.

259. Solum, Lawrence (1992). "Legal Personhood for Artificial Intelligences," *North Carolina Law Review*, Vol. 70: 1231-1287.

- 260.
- 261.
262. Sparrow, Rob (2007). "Killer Robots", *Journal of Applied Philosophy*, Vol. 24, No. 1: 62-77
- 263.
264. Thompson, Paul B. (2007). *Food Biotechnology in Ethical Perspective, 2nd ed.*, Dordrecht, The Netherlands: Springer.
- 265.
266. Turing, Alan (1950). "Computing Machinery and Intelligence", *Mind*, Vol. 59, No. 236: 434-460.
- 267.
268. University of San Diego (2008). Ethics Updates webpage. Last accessed on September 15, 2008: <http://ethics.sandiego.edu/index.asp>
- 269.
270. US Army Surgeon General's Office (2006). *Mental Health Advisory Team (MHAT) IV: Operation Iraqi Freedom 05-07*, November 16, 2006. Last accessed on September 15, 2008: <http://www.globalpolicy.org/security/issues/iraq/attack/consequences/2006/1117mhatreport.pdf>
- 271.
272. US Army Surgeon General's Office (2008). *Mental Health Advisory Team (MHAT) V: Operation Iraqi Freedom 06-08*, February 14, 2008. Last accessed on September 15, 2008: http://www.armymedicine.army.mil/reports/mhat/mhat_v/Redacted1-MHATV-OIF-4-FEB-2008Report.pdf
- 273.
274. US Department of Defense (2007). *Unmanned Systems Roadmap 2007-2032*, Washington, DC: Government Printing Office. Last accessed on September 15, 2008: <http://www.acq.osd.mil/usd/Unmanned%20Systems%20Roadmap.2007-2032.pdf>
- 275.
276. US Department of Energy (2004). *Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations*, Washington, DC: Government Printing Office. Last accessed on September 15, 2008: <https://reports.energy.gov/BlackoutFinal-Web.pdf>
- 277.

-
278. US Department of the Navy (2004). *Navy Unmanned Undersea Vehicle (UUV) Master Plan*, Washington, DC: Government Printing Office. Last accessed on September 15, 2008: <http://www.navy.mil/navydata/technology/uuvmp.pdf>.
- 279.
280. Van der Loos, H.F. Machiel (2007). "Ethics by Design: A Conceptual Approach to Personal and Service Robot Systems", *Proceedings of the IEEE Conference on Robotics and Automation, Workshop on Roboethics*, April 14, 2007, Rome, Italy.
- 281.
282. Veruggio, Gianmarco (2007). *EURON Roboethics Roadmap*, Genova, Italy: European Robotics Research Network. Last accessed on September 15, 2008:

- 283.
284. <http://www.roboethics.org/icra07/contributions/VERUGGIO%20Roboethics%20Roadmap%20Rel.1.2>
285. [.pdf](#)
- 286.
287. Wallach, Wendell and Allen, Colin (2008). *Moral Machines: Teaching Robots Right from Wrong*, New York, NY: Oxford University Press.
- 288.
289. Walter, W.G. (1950). "An Imitation of Life", *Scientific American*, 182:42-45.
- 290.
291. Walzer, Michael (1977). *Just and Unjust Wars: A Moral Argument with Historical Illustrations*, New York, NY: Basic Books.
- 292.
293. Weckert, John, ed. (2007). *Computer Ethics*, Burlington, VT: Ashgate Publishing.