FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Robotic Process Automation in practice
Student:	Bc. Tomáš Vahalík
Supervisor:	Mgr. Ondřej Dvořák
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2021/2022

Instructions

Even though the current technologies offer numerous ways to support employees in their daily business, many organizations still handle tedious administrative work manually. They often have to daily repeat similar tasks using office SW and using custom information systems. Although these repetitive tasks could be automated with the help of modern SW systems, typically, this innovation remains challenging for many organizations. Robotic Process Automation (RPA) could offer an answer to this challenge. RPA can automate user interactions with a range of applications and thus, it could tackle the tedious repetitive tasks. However, bigger case-studies of its implementation are still rarely available.

- Select RPA tool of your choice and review its capabilities in depth
- Analyze tedious administrative tasks in a specific environment (e.g., university, company, etc.)
- Automate the tasks using selected RPA
- Evaluate and comment on your results

Electronically approved by Ing. Michal Valenta, Ph.D. on 14 October 2020 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

RPA in practice

Bc. Tomáš Vahalík

Department of Software Engineering
Supervisor: Mgr. Ondřej Dvořák

May 3, 2021

Acknowledgements

I would like to thank to my supervisor, Mgr. Ondřej Dvořák, for his support and valuable advice throughout creating this thesis. I would also thank to the secretary and timetable scheduler of the Department of Software Engineering, Mgr. Alena Libánská, Ph.D., for consultations during the analysis and evaluation phases of this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 3, 2021 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Vahalík, Tomáš. *RPA in practice.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

RPA (robotická automatizace procesů) je technologie, která umožňuje simulovat lidskou interakci s počítačem. Simuluje aktivity jako klikání myší a psaní na klávesnici. Jejím cílem je automatizovat práci s počítačem, kdy uživatel vykonává mnoho takovýchto manuálních operací. Cílem této diplomové práce je prozkoumat možnosti této technologie a zjistit, jak je použitelná v reálném prostředí. Nejprve bude popsána RPA technologie jako taková a jeden vybraný dodavatel RPA řešení. Dále bude vybrán proces z reálného prostředí, na který by bylo vhodné RPA aplikovat. Následně bude implementován RPA robot, který tento proces automatizuje. Nakonec bude prozkoumána účinnost takového robota.

Klíčová slova RPA, proces, automatizace, podnikové procesy, UIPath.

Abstract

RPA (Robotic Process Automation) is a technology, that enables to simulate human interaction with a computer. It simulates actions like mouse clicks and key presses. Its goal is to automate tasks, which contain many manual operations. This thesis aims to explore the capabilities of this technology and its applicability in a real environment. Firstly, the RPA technology will be described as well as one chosen RPA vendor. Further, a business process from a real environment suitable for automation will be chosen. After that, an RPA robot will be implemented, which automates the given process. Finally, the applicability of the robot will be discussed.

Keywords RPA, process, automation, business process, UIPath

Contents

Introduction **1**
Motivation 1
Goals 1
Structure of the thesis 2

1 State-of-the-art **3**
1.1 Robotic process automation 3
1.2 Brief list of RPA tools 8
1.3 UI-Path in depth 9
1.4 Related work 21

2 Goals revisited **23**

3 Analysis **25**
3.1 Discovering a suitable process for automation 25
3.2 Understanding the process 27
3.3 Scoping the project 31
3.4 Conclusion 33

4 Realisation **35**
4.1 The structure of solution 35
4.2 Initializing the workflow 36
4.3 Filling the database 37
4.4 Generating documents 43
4.5 Sending the documents 49
4.6 Testing 49
4.7 Summary 51

5 Evaluation **53**
5.1 Impressions about UIPath 53

5.2 Applicability 54
5.3 Maintenance 55
5.4 Embedding in practice 56
5.5 The development effort evaluation 57
5.6 Future work 57
5.7 Evaluating goals 58

Conclusion **59**

Bibliography **61**

A Acronyms **65**

B Contents of enclosed CD **67**

List of Figures

1.1 Trending RPA vendors, image taken from [1] 9
1.2 Definition of an input argument 11
1.3 Calling a workflow with defined arguments 11
1.4 A simple sequence example 12
1.5 An example of a wrong use of a sequence 12
1.6 A flowchart example 13
1.7 A State machine example 14
1.8 Structure of a selector 15
1.9 Example of using anchors 16
1.10 Sending hotkey to notepad 18

3.1 Course schedule 30
3.2 Personal calendar 30
3.3 The use of mail merge feature in Word 30
3.4 The scope of the project 34

4.1 The structure of the folder with data 36
4.2 Scope of the first workflow 37
4.3 Trying to find the ”Log in” button 41
4.4 Code sample performing the login 41
4.5 Searching for person 41
4.6 Structure of the calendar 42
4.7 First workflow state machine 44
4.8 Scope of the second workflow 44
4.9 Screenshot of the template word file 46
4.10 Structure of the Work Report file 48
4.11 The test coverage, image taken from [2] 50

Introduction

Motivation

In many industries, employees are forced to do many repetitive and time-consuming tasks. Such tasks may include browsing a large amount of data, filling Excel tables, writing emails, or transferring data from one system to another. Robotic Process Automation (RPA) is a technology that aims to automate these types of manual tasks. This will let the employees to focus on actions that require more human thinking.

In this thesis, I will examine the RPA capabilities by applying it to an existing process in a real environment. I will take all the steps needed to create a prototype automating the process - choosing a process suitable for automation, analyzing it, and implementing an automation using RPA. Afterwards, I will analyze how usable the solution is and what benefits RPA could bring to organizations.

Goals

1. Get familiar with the concepts of RPA
2. Choose an RPA tool and explore its capabilities in depth
3. Choose a process from real environment suitable for automation
4. Automate the process using an RPA tool of your choice
5. Comment on your findings

Structure of the thesis

This thesis will consist of five chapters.

In **chapter 1 - State-of-the-art** I will provide an introduction to the RPA technology, comment on its strong and weak sides. I will then explore the UIPath tool in depth.

In **chapter 2 - Goals revisited** I will revisit my goals and define them more precisely.

In **chapter 3 - Analysis** I will describe how I chose a suitable process for automation. Then I will provide a detailed description of the process and I will set the scope for the realisation phase.

In **chapter 4 - Realisation** I will describe how I implemented the automation of the process using RPA. I will comment on the difficulties I encountered and the methods I used to overcome them.

In **chapter 5 - Evaluation** I will comment on embedding the solution in practice. I will describe how much time the automation would save and how it should be maintained. I will also try to estimate the cost of the solution.

Chapter 1

State-of-the-art

1.1 Robotic process automation

In this section, I will introduce the concept of RPA, its primary goal, and history. I will list the pros and cons of this technology and finally explain what RPA robots are.

Introduction

RPA (Robotic Process Automation), is a technology that enables computer software to emulate human interactions with a digital system. It is capable of launching and interacting with any application without any knowledge about its code base. RPA can emulate user interactions such as key presses and mouse clicks. In addition to simply mimicking human interactions, these robots perform complex calculations and take decisions based on predefined rules or data, and communicate with other systems.

The goal of RPA is to automate tedious tasks, which are performed manually by employees and consume a lot of their time. The targeted process should be repetitive, rule-based, with low exception rate and with well-defined inputs and outputs. An example could be filling registration forms for a large number of people. The same actions need to be taken for each person (filling in the required fields and, for example, click the confirmation button), the only thing that differs is the data that are filled into the form. Automating these tasks unburdens employees from repetitive work and allows them to focus on solving problems that require human thinking. [3]

History

According to Taulli [3], It is in the human's nature to think of ways how to have his job done for him. The first concept of automation comes from

Homer's *Iliad*, where Hephaestus (the god of blacksmiths) uses automatons (machines) to build weapons for the gods on the Mount Olympus.

In terms of software automation, Dilmegani in his article states that enterprises used automation tools even before RPA software existed if forms like scripts or Excel macros. These solutions were not scalable or sustainable, but they increase the productivity of individual employees. [4]

The real RPA technology emerged in the early 2000s. In this period, companies that lead today's trends in RPA were created. First, the robots were simple programmable bots that required specific inputs. Later, cognitive automation came in. Cognitive robots augment the capabilities of automation by incorporating functionality like NLP (natural language processing) and OCR (optical character recognition). These bots can be effective in decision making.[4]

Around 2012, the RPA market hit an inflection point. Companies were looking to lower their costs and lessen their paperwork. RPA technologies were starting to get more sophisticated, easier to use, and were already used by some large companies for mission critical applications. [3]

The future of RPA might be in self-learning robots. They watch employees in action, try to understand the processes, and take over when they are confident enough. However, these technologies are now confined to proofs of concept. [4]

Benefits of using RPA

These are the benefits of RPA as listed in a book by A.M.Tripathi [5]:

- **The process becomes optimized and well defined**
  To automate a process, I have to make sure I understand it well and I know which action should happen when. When analyzing a process, I can come across some inconsistencies (for example, one employee performing the task saves data to Excel, but the second employee saves data to a text file). When automating the process, these inconsistencies will be removed and it will be certain that the process does not have any ad hoc steps that are not precisely defined.

- **Greater accuracy**
  Humans are prone to errors such as making typing errors. Machines are immune to these kinds of errors. Furthermore, it is hard to trace where and when a human-made error occurred. All actions that RPA makes should be logged, so it is easy to recognize what was the cause of an error.

- **Speed**
  The process is executed much faster when handled by a robot. Computers can type and click much faster than a human being. The speed

difference is especially noticeable when there are many data to process or when the process is divided into phases, which are executed by multiple people. Employees have to wait to perform their role in the process until their forerunners finish their tasks. This may take even days, as some people may be ill or on vacation. This is no concern for RPA, each step is performed immediately after the last one.

- **Availability**
  Robots can work 24/7 without getting tired.

- **Scalability**
  It is easy to deploy another robot if there is a lot of work to be done or to shut one down if it is not used very much. If the process would be handled manually by employees, to scale the process would require to hire more employees or to fire them.

Downsides of RPA

RPA is definitely not a "silver bullet". I this section, I will list some issues that come up when using RPA.

- **Cost of license**
  Obtaining a commercial license can be quite costly. There are often free versions that are suitable for individuals or small projects, but are not enough to automate larger tasks. The vendors usually offer a personalized solution that can be scaled to our company´s needs. The prices start at $3990 per user every year (UI Path [6]) and can come up to $9000 USD Annually (Automation Anywhere [7])

- **Maintenance**
  The robots are made to interact with a specific UI. When the UI or process steps changes, the robot stops working. It is therefore necessary to revise the robot regularly to make sure it is up to date. There are ways to make sure that the robot will work even if the UI is changed slightly, but evolving UI is generally a problem. [3]

- **Enterprise scale**
  When RPA is used among the whole organization (which is a good thing and brings many benefits), it can be difficult to manage and maintain all robots. It is crucial to have a good cooperation between management and IT department. [3]

- **Fear of RPA taking employees jobs**
  Although RPA is meant to be a great help for the employees and save their time, some may feel threatened by this technology and fear to loose their jobs.

RPA robots

Robots are agents which run on client computers and execute assigned workflows. There are two types of robots - **attended** and **unattended**.

Attended robots

Attended robots are something like virtual assistants to human employees. The robot does not replace an employee, its goal is to help the employee to increase his or her effectiveness. Employees start the robot and communicate with it (for example, by providing input) while the robot is running.

Attended robots usually perform only repetitive, easy to define steps in a larger process (for example, copying data from one system to another or to put some data into an Excel table). The employee can then use the result of the robot and decide what to do next in the process.

Unattended robots

Unattended robots on the other hand are fully independent and don´t require any human interaction at all. They are started automatically, either by some event or at a predefined time. Unattended robots are used to fully replace human employees. They are available 24/7, not only if someone tells them to run.

Unattended robots perform the whole process end-to-end. It is important to keep logs of unattended robot activities, so any error that may appear can be later analyzed. (Mullakara, Asokan in [8])

Typical tasks to automate

As mentioned in section 1.1, a typical task that should be automated is

- rule based
- repetitive with high transaction values
- well defined

Such tasks can be found in various companies and environments. According to Tripathi [5], the following industries can largely benefit from using RPA:

1. **Insurance companies** - task such as managing policies and processing claims across multiple platforms.
2. **Utility companies (Gas, electricity, etc.)** - processing many monetary transactions, meter reading or processing custom payments.

3. **Healthcare** - RPA can help with patient scheduling, sending them reminders for appointment and filling patient records
4. **Business process outsourcing** - Business process outsourcing, is a practice in which one organization hires another company to perform a process task that the hiring organization requires for its own business to operate successfully.[9] RPA can replace this sector and make the company rely less on outsourced labor.

A book by Mullakara and Asokan [8] shows examples of real life scenarios that can be automated with RPA. I will list some of them here.

Help desk ticket generation

The goal is to ease the help desk agents to create tickets for incoming issues. In the scenario presented, the issues come in the form of a spreadsheet. The robot, once invoked, scans a folder containing these spreadsheets and creates a ticked in the *ZOHO desk application*. Zoho is a CRM (Customer relationship management) system providing features for sales, marketing, and customer support.[10]

CRM automation

There are scenarios, where people manually transfer data from one system to another. These are called "Swivel chair activities" (The term is derived from the practice of the user turning from one system to another using a swivel chair [11]). These activities are repetitive and may result in human error. The example presented in the book focuses on searching for customer information on an external website and enter it into the CRM system called *Apptivo*.

Challenges for RPA development

According to Van der Aalst et al. in [12], these are the most challenging aspects in developing RPA applications:

1. Identify a process suitable for automation
2. How to control RPA agents and avoid security, compliance, and economic risks?
3. Who is responsible when an RPA agent "misbehaves"?
4. Adapting to the changes of user interface

Summary

RPA is a technology that focuses on simulating human interactions with a computer. Its main goal is to automate well-defined, repetitive tasks with many interactions with a UI. Typical industries that benefit from RPA include insurance, utility companies and healthcare.

The programs that run on client computers and perform defined actions are called robots. Robots can be used to help individual employees with their work (attended robots) or to complete complex workflows on their own without any human interaction (unattended robots).

Using RPA in business brings improved speed and accuracy in the execution of processes. However, RPA software has to be maintained regularly because the UI to which the robots are used to may be changed from time to time.

1.2 Brief list of RPA tools

The trend of RPA is growing and there are many vendors offering their RPA tools. (According to Dilmegani in [4] more than fifty). Most of them support only Windows operating systems, but there are some exceptions (such as Argos Labs)[4]. However, according to Clair and O'Donnell in their Forrester research [13], there are vendors that lead the market and are more popular than others. These are

- UI-Path
- Automation Anywhere
- Blue Prism

Figure 1.1 shows, that UI-Path and Automation Anywhere are the leading tools with Blue Prism being sightly behind them in terms of popularity.

I have examined these three mentioned tools and found them to be very similar in usage and architecture. They all consist of three main components:

1. An application used to develop robots (UI-Path **studio**, Automation Anywhere **bot creator**, Blue Prism **object studio**)
2. A representation of the robot performing given tasks (UI-Path **robot**, Automation Anywhere **Bot** - task bot, meta bot, or IQ bot, Blue Prism **Work force**)
3. An application used to manage and monitor robots and assign tasks to them (UI-Path **Orchestrator**, Automation Anywhere **Control Room** for running and **Bot Insights** for monitoring, Blue Prism **Control Room**)

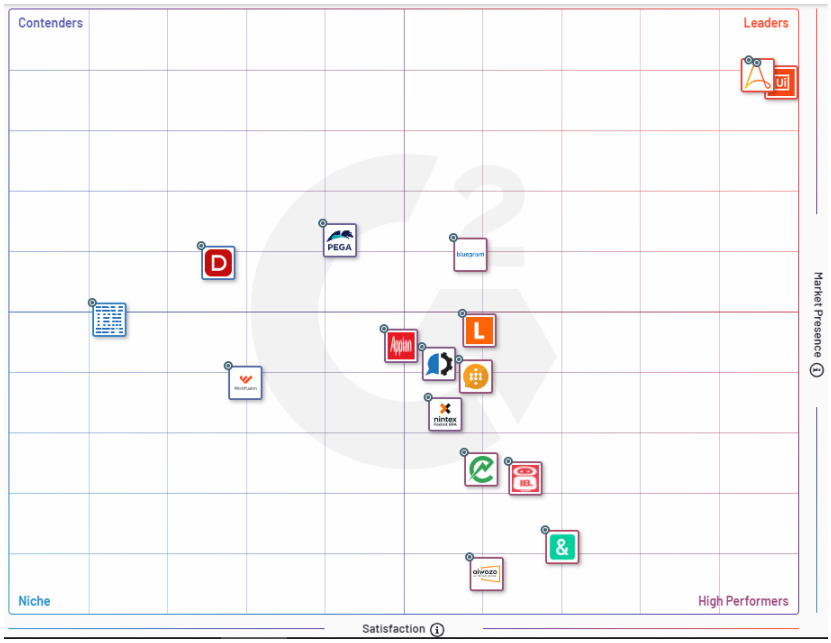

Figure 1.1: Trending RPA vendors, image taken from [1]

For the purpose of this thesis, I have decided to use UI-Path, because it is one of the most used platforms on the market, it has a live community and forums where people discuss any issues, and I have already worked with UI-Path in the past.

1.3 UI-Path in depth

In this section, I will explore UI-Path platform in depth. First, I will write about how robots are developed with UI-Path Studio, then I will focus on UI-Path Orchestrator.

UI-Path studio

UI-Path Studio is an application for developing workflows. It is a low-code environment, where developers drag and drop prebuilt components [8]. It allows the developer to run and debug the program. While debugging, the developer may make the robot pause after every step in the workflow and check the contents of all variables. Studio is also used to publish finished workflows to the Orchestrator (More about that in section 1.3), downloading and managing dependencies.

Development - idea

UI-Path development is realized by using existing components - "activities" and combining them into "workflows". Activities are plug-and-play segments that represent single actions that are taken in the process. They can be actions that simulate the behaviour of a user, for example "open the browser at a specified URL address", "type into a text field", "click a button", but also data-gathering and transforming actions, like "read the value of a cell in an Excel table and store it in a variable" or "sort the provided data table".

Developers combine these activities into workflows, which are more sophisticated components which take care of multiple steps in the automated process. Such a workflow could be, for example a "log-in" scenario - launch a target application or web page, fill in user credentials, press "log-in" and check whether the log-in attempt was successful. These workflows can then be used as any other activity. This allows us to develop on multiple layers of abstraction. We break down complex processes to smaller, more manageable activities. We automate them independently and then combine them to get the desired result. That way, we can also cut down the development time of the project, because we can reuse the functionalities on different places. Workflows also have input and output parameters so that data can be passed from one workflow to another. This approach also makes collaboration easier, because we can use other developer´s workflows without knowledge about its implementation. We only have to know what purpose the workflow has and what are its I/O arguments. Figure 1.3 shows how workflows can be invoked and how parameters can be passed between them.

UI-Path is meant to be usable even for people with low experience in programming. This is why the majority of tasks may be done only by dragging and dropping existing activities. These activities are then modified by providing arguments. For example, even cycles and decisions aren´t done by writing code. There is a prebuilt if-else activity, in which we then specify the condition. Any expression in UI-Path (for example, when assigning to variables or providing conditions to if-else activities) can be written in VB.Net or the C# language (only one language in the same project). VB.Net is the default language, as it has better runtime performance.[14]

Workflow layouts

In this section I will describe the layouts that a workflow can have. There are three types of layouts we can use while creating a workflow - sequence, flowchart and a state machine. Choosing an appropriate layout is crucial to make the project reliable, efficient, and extensible.[15]

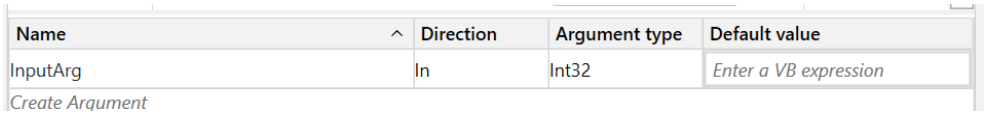

Figure 1.2: Definition of an input argument

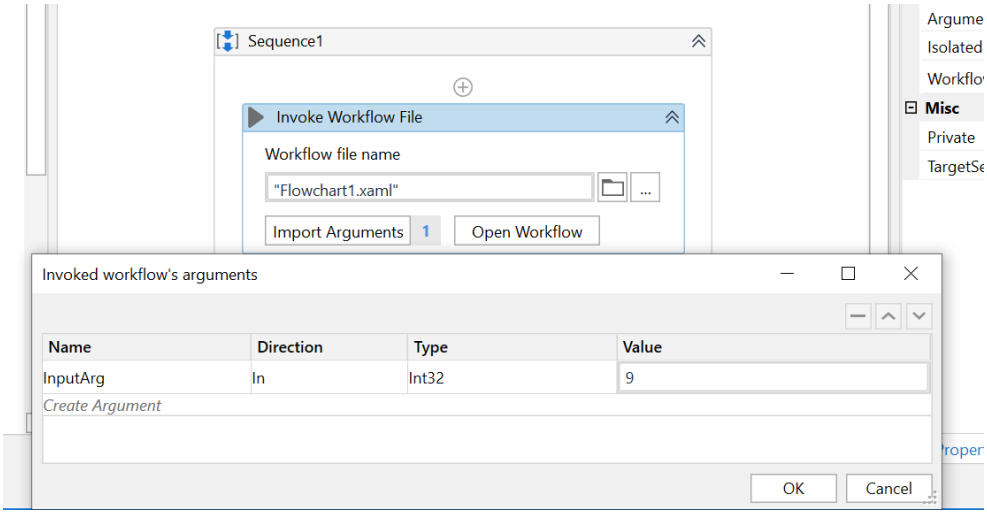

Figure 1.3: Calling a workflow with defined arguments

Sequence

A sequence is the most straight-forward layout. They simply execute each activity one after another. Sequences are suitable for simple linear workflows, like UI interactions or searching on the web. They are easy to understand, as they have a top-bottom approach.[16]

On the other hand, they are not good for workflows that need to check certain conditions and decide which steps to take next. It is possible to achieve this behaviour with a sequence, but it then becomes unreadable and confusing. Figure 1.4 shows a simple sequence interacting with a browser - navigating to ”google.com”, typing something in the search field, pressing ”find” button, and closing the window after a while. Figure 1.5 shows a bad example of using a sequence with multiple if-else blocks.[15]

Usually, sequences are used to automate low-level activities such as UI interactions, and the high-level logic is handled through flowcharts or state machines.

Flowchart

Flowcharts are similar to logic diagrams in software engineering. They are suitable for more complex workflows, where we have to decide which action should be performed next. Activities are not executed in a predefined or-

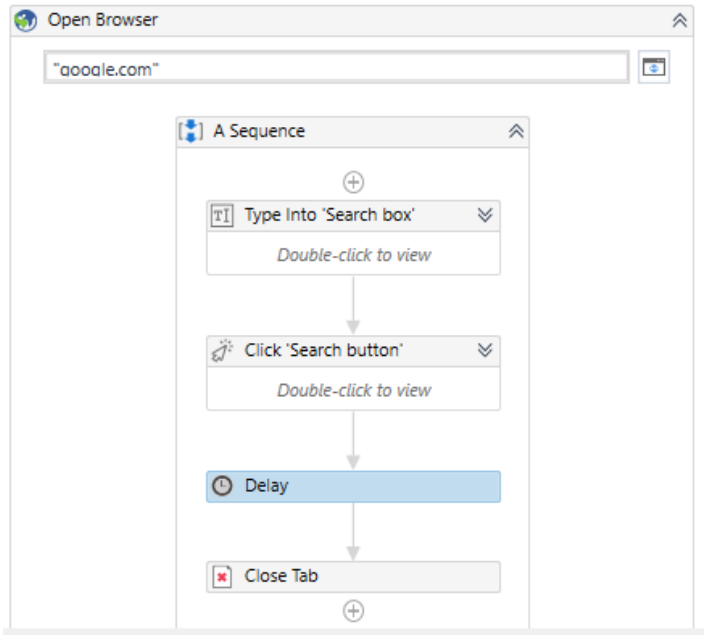

Figure 1.4: A simple sequence example

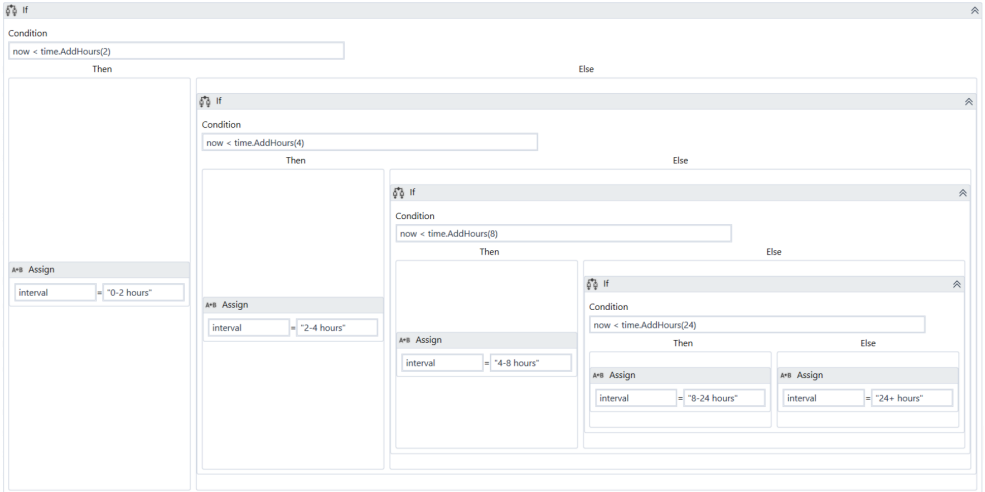

Figure 1.5: An example of a wrong use of a sequence

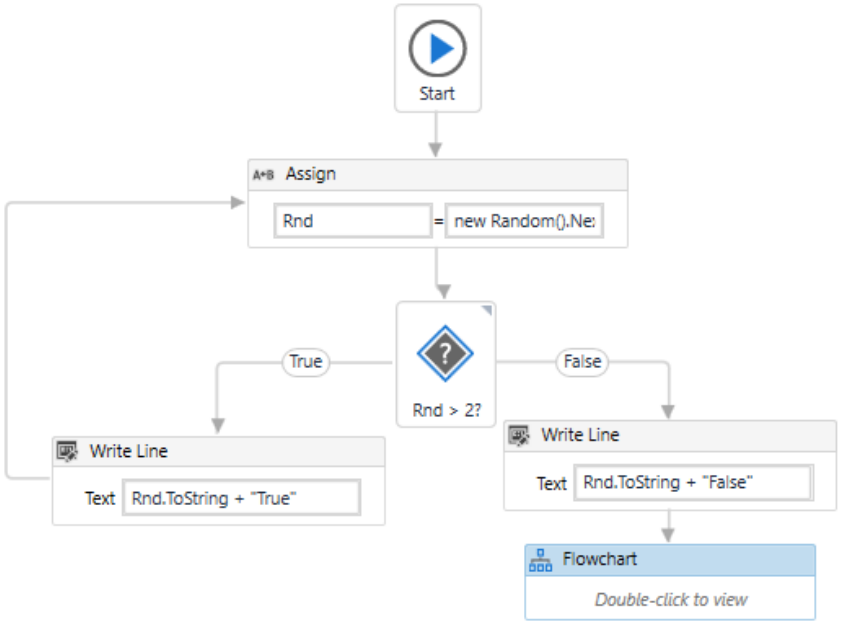

Figure 1.6: A flowchart example

der, but we manually link them in the order we wish them to be executed. Flowcharts provide tools for branching like decisions or switch blocks.

Another advantage of the flowchart is that it makes it very easy to test individual activities/subworkflows. We simply create a new block and connect it to the start node. That way, only the desired block will be executed.

At UIPath Academy in lecture on project organization [15] is also written, that *"Flowcharts can be used only as the general workflow (with sequences nested inside), not for individual parts of projects (nested inside other workflows)"*. I find this information to be false, because as shown at Figure 1.6, I can put a flowchart into another flowchart and it works without any issue. Figure 1.6 shows a flowchart which generates a random number and decides what to do next based on the number. It also shows that another flowchart can be embodied into the main one.

State machine

A state machine in UIPath is similar to the Finite-state machine computational model in mathematics. *It is an abstract machine consisting of a finite number of predefined states and transitions between these states. At any point, based on the external inputs and conditions verified, it can be in only one of the states*, as stated in UIPath Academy lecture on project organization [15]

State Machines can be used with a finite number of clear and stable states to go through. Some examples from our daily life include vending machines,

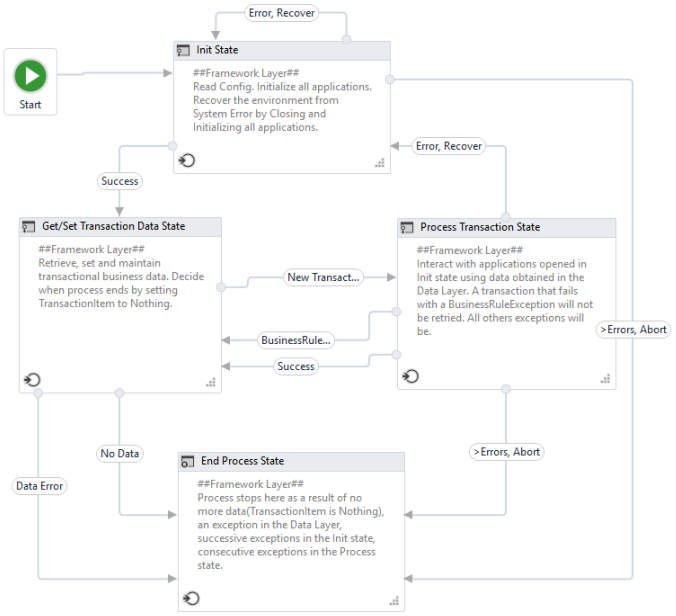

Figure 1.7: A State machine example

elevators, or traffic lights.

This layout is mostly used in processes that cannot be described with simple if-else decisions and cycles. It is also Excellent for reusing existing logic. For example, we can create an "init" state that makes sure that all applications that we are going to use are opened. When during the execution of a workflow some of the applications close, we can simply transition to the "init" state and then continue our workflow. Figure 1.7 shows states used by Robotic Enterprise Framework. That is a project template designed to automate large-scale enterprise applications.[17]

Interacting with UI elements

In this section I will describe how UIPath sees the UI and interacts with it. Every interaction with an application UI consists of 2 steps: recognizing UI elements and then performing the desired operation – clicking, inputting (or extracting) data, etc. In this section, I will first demonstrate how the elements which the robot should interact with can be described. Then I will describe interactions which the robot can perform.

Describing a UI element

There are 2 ways to find a UI element: Using special constructs called "selectors" or providing a screenshot and using computer vision to match the image in the application. [18]

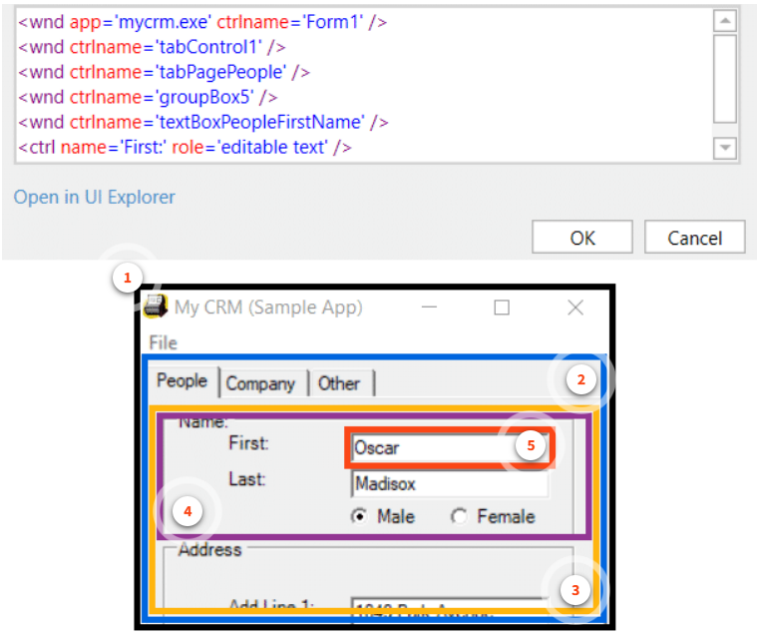

Figure 1.8: Structure of a selector

Selectors are constructs that store UI element attributes as XML fragments. UI is built using containers nested into each other, which makes XML great for describing it. Selectors also contain information about the element ´s parents [19]. The selector consists of tags and their attributes. Figure 1.8 shows how the selectors are structured.

Selectors are usually automatically generated by UIPath studio, but it is advisable to check them and manually modify them to make them more reliable. According to UIPath Academy lecture on selectors [19] the situations when the selectors need to be manually modified are:

- When the robot interacts with web pages, which generate dynamic IDs for the elements they comprise. When we reload the page, the ID is different, so we cannot use it reliably.

- Selectors are often too specific. They include, for example, the title of the web page the robot is interacting with. If the title changes, the selector stops working. It might be better to remove the title attribute from the selector. That way it will work even if the title changes.

- It is not recommended to use the attribute IDX. This attribute is the index that the current element has within its container. When the order of the elements changes, the selector might point to an unexpected element.

The strong side of selectors is that they are capable of differentiating between UI elements that look the same, which cannot be achieved by using computer vision.

We might encounter situations, when an element cannot be described by the selector reliably. We can use "Anchor base" activity [19]. Firstly, an element that can be described reliably (anchor) is identified, and then the position of the target element is described relatively to this anchor. To demonstrate this, I will use an example at http://www.rpachallenge.com. The UI changes constantly and the input fields are in different places every time a form is submitted. To correctly fill in the last name, I first detect the "Last name" label and use it as an anchor to tell UIPath that the correct text field is below this label. This is demonstrated in Figure 1.9

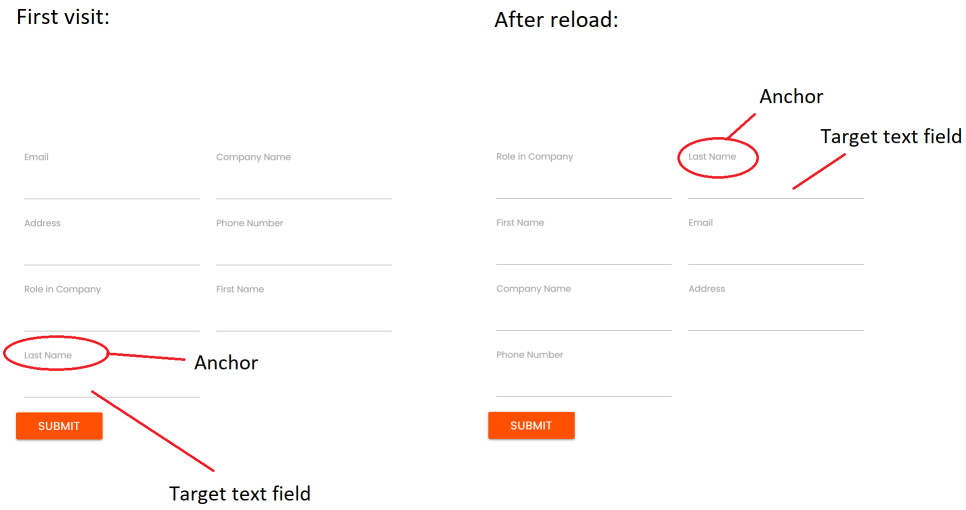

Figure 1.9: Example of using anchors

Some activities that use selectors are "Click" and "Type into".

The other approach is to use computer vision and scan the whole screen to match the provided screenshot. This approach is the only way to interact with virtual or remote environments. That is because these utilities only stream an image of the desktop to the user, which means normal UI selectors are impossible to find. The drawback of this method is that it is impossible to differentiate between two identical elements. Image recognition activities have an "Accuracy" parameter, which states whether the images must match 100% or less to be registered as found. This can compensate for possible small UI changes. [20]

Some activities that use computer vision are "Click image" and "Image exists". These activities may also be useful for pausing the workflow until a certain image appears. For example, we want to pause the execution until some website is fully loaded. This can be done by trying to find an image

which appears only after the page is fully loaded. [20]

Input methods

After the robot has successfully identified a UI element, it can interact with it in certain ways. Most common interactions are clicks, hovers, and text inputs. Each of these interactions has corresponding activities in UIPath studio (activities called "Click", "Type into" and "Hover"). Each of these activities needs to know which input method to use when interacting with the UI element. As stated in the official UIPath documentation on input methods [21] there are three available methods: **Default**, **Send window messages** and **Simulate click/type/hover**

- **Default** method uses mouse and keyboard drivers to simulate user behaviour. That means that the mouse physically moves across the screen and individual characters are typed. The strong side of this method is that it is 100% compatible with every application, as it uses only hardware drivers. It also supports entering special characters like enter, tab, and other hotkeys. On the other hand, this method is the slowest of the three methods and cannot work on background (the user cannot touch the mouse or keyboard while the robot is running, otherwise the results wouldn´t be precise).

- **Send window messages** method communicates with the target application via dll (Dynamic link library)[22] and replays the message that occurs when a key is pressed or the mouse clicked. It is slightly faster than the default method and supports special hotkeys. It can also work in the background, that means the user can perform other tasks even while the robot is currently running. This method works only with applications that respond to window messages.

- **Simulate** method uses API of the target technology to communicate with the application. It is the fastest of the three methods at the cost of compatibility - it cannot be used every time. It also works in the background, but does not support hotkeys. [21]

Figure 1.10 demonstrates how sending hotkeys to applications works. The activity simply types the text into an opened notepad. If the "Default" or "Send window messages" methods are used, the output will be

```
hello
there
```

If the "Simulate" method is used, the text will be typed much faster, but the output will be

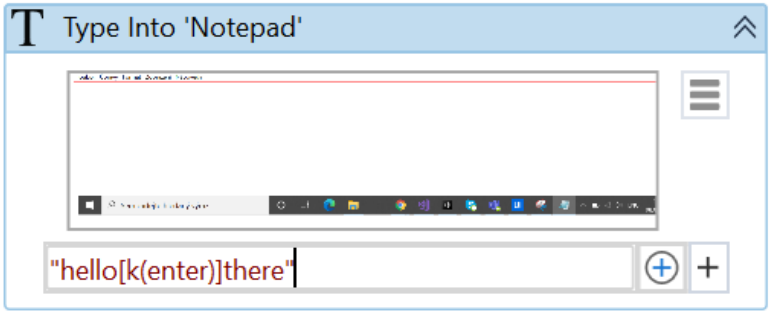

Figure 1.10: Sending hotkey to notepad

```
hello[k(enter)]there
```

I have done a test comparing these methods in terms of speed. I have generated 250 words of lorem ipsum and let the robot type them into the notepad:

Method	Execution time (minutes)
Default	1:13
Send window messages	1:03
Simulate type	0:04

The "Default" and "Send windows messages" are comparable in terms of speed, while the "Simulate" method outranks them by a lot. The execution of the robot took only 4 seconds, most of that was opening the notepad window, the text appeared almost instantly. The result of the experiment confirms the information stated in the documentation [21].

Exception handling

In this section I will describe the exceptions in UIPath. Firstly, I will list the types of exceptions and then describe how to deal with them.

Exception types

There are two types of exceptions that may occur during the execution of a robot. **System exceptions** and **Business exceptions**

System exceptions are errors that prevent the robot from continuing the workflow. These are exceptions like *NullReferenceException*, *IndexOutOfRange-Exception* or *SelectorNotFoundException*. They all derive from a base class *System.Exception* and are automatically thrown by the system.

Business exceptions, on the other hand don´t necessarily prevent the robot from running, but they represent errors in the data that the robot is working with - the data may be incomplete or malformed, or do not pass other validation criteria. These exceptions need to be thrown manually by the developer. [23]

Dealing with exceptions

There are several ways how to handle errors, I will mention **Try - catch** activity, **Retry scope** activity and **Continue on error** property.

The **Try-catch** activity works similarly to try-catch blocks in other programming languages. It is used to recognize which type of exception occurred and act accordingly. I can specify what actions should be taken for every exception that is caught.

The **Retry scope** activity is similar to try-catch. The difference is that this activity simply retries to execute the activity from which an exception was thrown. It doesn´t provide a more complex handling mechanism like the Try-cach activity. I can specify how many times an action can be retried and the time interval between retrying.

Every activity has a **Continue on error** property. If it is set to be *true*, any errors coming from this activity will be ignored and the execution will continue. It is not recommended to use this for every activity, but there are some use cases when it can be useful - for example, when I need to gather information from a table that is spread across multiple pages linked by the "next" button. The last page will not contain such a button, therefore an error would occur when the robot would try to click it. In this case, it could be safely ignored. [23]

Recording

In this section I will describe the recording feature of UIPath studio. UI-Path provides us with the option to record our actions, save them as a workflow, and replicate them later. The resulting workflow is always saved as a sequence. It is suitable for automating simple linear actions, which do not need any decision making (similar use cases as using a sequence as a workflow layout).

When the recording is started, I can see that every UI element I hover over gets highlighted, letting me know how UI-Path "sees" the environment. Some user interactions can be recorded, while others can not. **Recordable** actions are left clicks on UI elements like buttons, checkboxes, etc. and text typing. **Non-recordable** actions are right clicks, keyboard shortcuts (like Ctrl + A) and mouse hovers. These actions must be automated in a different way. The generated workflow should be fine-tuned to make it more reliable and maintainable. Selectors should be modified and variables should be used instead of hard-coded values generated in the workflow.

There are four recording modes in UI-Path. **Basic, Desktop, Web** and **Citrix**. Basic, Desktop and Web are very similar. The only difference is that the desktop mode wraps all actions inside an "Attach window" activity, the web mode in "Attach browser" activity, and the basic mode does not wrap the actions in any activity at all. The difference is that the "Attach browser" and "Attach window" activities help us distinguish between two instances of an identical application. For example, if we had two notepad windows opened, the workflow generated using the desktop mode will type in the right window. If we used the basic mode, the robot would use the top-most window. The Citrix mode is used to record actions taken in virtual environments. [24][25]

Orchestrator

In this section I will describe the functionalities of the UIPath Orchestrator. As stated in UIPath Academy lecture on the Orchestrator [26], *Orchestrator is the component of UiPath Suite through which the automation workflows developed in Studio are published, assigned to robots and executed. It comes in the form of a web application that enables the management of robots, activity packages, data to be processed, execution schedules, as well as other assets.*

Features of the Orchestrator:

- **Conducting execution of robots**
- **Version control** - workflows published to the Orchestrator remember their past versions, and we can specify which version should be used when robots execute this workflow
- **Transactional processing**
- **Storing assets**
- **Statistics**

I will comment on these features in the sections below.

Process execution

To use the Orchestrator for running our robot, I must first take care of some configurations. I must first register my machine to the Orchestrator, then create an **environment** for the robot and finally register a robot. An environment is a group of robots. When I call a process, I specify the environment and it will be called on all robots in the environment (unless specified otherwise).

Next, I have to publish my workflow to the Orchestrator as a **package**. This is done directly from the UI-Path studio. After I have successfully published a workflow, I can create a **process**. There I specify the package and its

version to be run and I specify the environment in which the process should be executed. Now I can run the process manually from the Orchestrator.

To run some process automatically, I can create **triggers**. There I specify which process should be run at which time (for example, every Monday and Wednesday at 13:00). This way the **unattended automation** can be achieved by using the Orchestrator [26].

Assets

Assets are shared variables or credentials that are stored in the Orchestrator and used by the robots in different automation projects.

Assets can be used when I need to pass data from the Orchestrator to robots and the data does not change very often. I can specify which robots can use which assets. Assets are imported into workflows using "get asset" activity. They are especially useful for storing credentials such as a username and a password. It is safe to store credentials this way, as they are encrypted using the AES 256 algorithm.[26]

Queues

Queues enable dividing work between multiple robots and implement the producer-consumer model. Items that are stored in queues are treated like transactions. Robots get individual items from the queue, perform the desired logic, and then set the transaction status. We can check in the Orchestrator the status of every queue item and see if the processing of all items went well. The queue makes sure that the items are distributed in a FIFO (First-in-first-out) way.

Usually, some robots serve as producers – they fill the queue with items (for example, some gathered data from an Excel sheet), and other robots serve as consumers – they get items from the queue, perform some business logic and set their state to be successful (or failed, if any error occurs). This behaviour can be achieved even with one robot. Firstly, the workflow for the producer is run, and when it is finished, the workflow for the consumer is performed.[26]

1.4 Related work

In this section, I will list the previous work related to this thesis. Examples of related work can be found among research papers that focus on finding a suitable use case for applying RPA or implementing a proof of concept.

Article "Impact of RPA Technologies on Accounting Systems" by Kaya et al. [27] analyzes the applicability of RPA in the financial and accounting sector and the effect such automation would have on the processes. It comes to a conclusion, that "*RPA will disruptively change the processes in accounting.*

Inefficient accounting operations will be handled by RPA and accounting professionals will concentrate more strategic operations in the context of Strategic Accounting Management."

Study "Applying robotic process automation (RPA) in auditing: A framework" by Huang and Vasarhelyi [28] implements an RPA framework that frees auditors from doing repetitive and low-judgment audit tasks. The result was that the created framework performed the task correctly and needed only 3 minutes to do so , compared to approximately 30 minutes for a human employee.

In the book "Robotic Process Automation Projects" by Mullakara and Asokan [8] are step by step solutions to individual use cases such as Help desk ticket generation or CRM automation.

The paper "Toward robotic process automation implementation: an end-to-end perspective" by Santos et al. [29] examines how the key concepts of RPA are applied in selected past RPA case studies. It concludes, that "*RPA main concepts gathered in the literature review are not reported in the selected RPA case studies*".

There are also studies that explore the capabilities and challenges of RPA as a whole. Article "Robotic Process Automation" by Van der Aalst et al. [12] which talks about the difference between RPA and STP (Straight Through Processing), provides an example case in the university environment and lists the challenges that occur when developing RPA.

Article "The Future Digital Work Force: Robotic Process Automation (RPA)" by Madakam et al. [30] focuses on defining the key concepts of RPA, finding suitable applications, and listing some RPA vendors. It predicts that RPA will completely replace the manpower in some industries and that its use will become compulsory to perform business operations.

Chapter 2

Goals revisited

In chapter 2, I got an insight into the capabilities of the RPA. I listed its benefits and downsides. I also provided a list of top trending RPA vendors and explored the UIPath tool in depth. I have decided to search for a suitable process for automation at the Faulty of Information Technology of CTU.

After this research, I gained enough information to revisit my goals and define them more precisely.

1. **G1** Get familiar with the UIPath tool to the level which allows creating a proof of concept in practice
2. **G2** Select a process at the Faculty of Information Technology of CTU and analyze, how could it be automated using RPA.
3. **G3** Implement a solution, that will automate selected process.
4. **G4** Discuss, how useful would the solution be in practice and what implications would it have to the current personnel.

Chapter 3

Analysis

This chapter will consist of three main sections. The first section will focus on finding a suitable process for automation. When the process is chosen, I will describe its structure in the second section. Finally, I will go through each step of the process and discuss whether it can be automated with RPA and will be in the scope of the realisation phase.

3.1 Discovering a suitable process for automation

In this section, I will describe how I searched for a process to use as a target for the implementation phase.

To find a suitable process for automation at the Faculty of Information Technology, I contacted Mgr. Alena Libánská, Ph.D., who is the secretary and timetable scheduler of the Department of Software Engineering. We discussed the administrative tasks that are done at the department. I will provide examples of some activities and explain why they are good or bad candidates for automation.

Approval of the final thesis topic

First candidate process is the "Approval of the final thesis topic". Choosing a topic for the final thesis needs several administrative tasks to be done. First, the student contacts the topic supervisor, who creates the topic in the system. Then, the topic needs to be approved by the head of the department. Finally, the student himself confirms the topic.

This process is not very good candidate for automation, because the decision whether approve or reject a topic needs to be done by a human. I think it would be impossible to transfer this task to a robot. Additionally, the robot would need to have access permissions to the system where topics for the final thesis are stored. This is generally an issue with RPA – a special account for the robot has to be established. Such account must have sufficient permissions

for the task the robot is assigned with. It would be very dangerous to let the robot interact with the production system during development, so a testing environment of the target system should be made (if it doesn´t exist already). If a task that requires access permissions to some system is being automated using RPA, it is necessary to have the support of the top management of the organization. Otherwise, acquiring the needed permissions will be very difficult.

Oracle certificates for students

Another example of an academical process is processing Oracle certifications for students.

Students of the BI-DBS course have the opportunity to receive certificates from Oracle Academy. There are two major tests in the course and a series of lectures, which are all followed by a minor test. Students with grades A-C in the BI-DBS course will need to pass only the major tests to receive the certificate. Students with grades D or E will need to pass all minor tests as well.

It is necessary to get information from students who would like to enroll in this program and create an account at the Oracle Academy portal for them. When the deadline for passing the tests ends, it is needed to find the accounts of all successful students, create and sign a certificate for them, and invite them to receive it.

This process is a good candidate for automation. The tasks are rather simple, monotonous, and need to be performed for a large number of inputs (students). There isn´t any decision making that would require a human being – everything that is needed is information about the students and their grade. Only the signing of the certificate has to be done manually and cannot be automated.

However, it would be very difficult to create a production ready solution. Firstly, the tasks are divided to multiple people. To analyse and understand the process, it would be necessary to contact each of them individually, analyse their parts, and then combine them in the final solution. This would be very time consuming. Next, the tasks would require permissions to interact with Oracle Academy system. This is the same issue described in section 3.1

Creating a work performance agreement for teachers

Another example is a process that occurs in almost every company – creating work performance agreements for employees.

Before every semester, work performance agreements have to be created for teachers, that do not have working at the faculty as their main employment. This task includes gathering personal information about these teachers,

creating an agreement and a work report for each of them, and sending these documents to the personal department.

This activity is a good candidate for automation. It is done only by one employee (at other departments each step may be done by a different person, but at the Department of Software Engineering, this whole process is done by Mrs. Libánská). That means I can get all the required information about the process from one person. It also does not require any permissions to external systems, information about teachers are stored in an Excel file, and the agreements are written in Word. The workflow is also monotonous, doesn´t require many decisions, and only the input parameters vary (the teachers).

After considering the pros and cons of every process, I have decided to use the process of creating work performance agreement for teachers as the target for further analysis and development phase.

3.2 Understanding the process

In this section, I will describe my chosen process in detail.

To set up the big picture, I will first list all phases that occur before, during, and after the process. Then I will provide a detailed description of each phase.

1. Schedule the courses that will be taught next semester
2. Send questionnaire to teachers about their personal information
3. Fill in the information into an Excel file
4. Create agreement for each teacher in evidence
5. Create "Command for payment" documents and "Work report" documents
6. Send all the documents to personal department
7. Invite all the teachers to come and sign the agreements

Creating the schedule

Firstly, it is necessary to list the courses that will be scheduled the following semester. Usually this list is only copied from the past years. However, still some communication with the department leader and the teachers is needed. The teachers need to confirm that they will be teaching the course next semester and should provide information about their time capabilities (how many study groups should be assigned to them). There also has to be some number of students who have an interest in attending the course for the course to be listed. This is why preliminary registrations exist, where students state the courses they wish to attend.

Gathering personal information

A questionnaire is sent to all teachers that will be teaching the following semester. The questionnaire is a Word file, where teachers fill in required information like name, address, phone number, account number, etc. There is one field that asks whether the teacher has a working relation with parts of CTU other than the Faculty of Information Technologies. If so, the teacher is processed in a different way and the responsibility falls under the personal department. The questionnaire template will be provided in the appended medium of this thesis.

Filling Excel table

Information about all teachers is contained in one Excel file, which serves as a database. There are three categories of information that needs to be filled in.

Personal information

First, there are personal information gathered from questionnaires. Whenever a reply containing the filled questionnaire comes from a teacher, the information are copied to the Excel file. If the teacher has already taught at the department in the past, there already exists an entry for him which will be updated. New teachers will be appended to the table.

Hourly rates

Next, the hourly rates for leading lectures, tutorials, and exams have to be gathered. These depend on the title of the teacher - person with the title Ing. will have higher rates than a person with the title Bc. Additionally, only people with title Ing. / Mgr. or higher may lead lectures, lesser titles may only lead tutorials. Hourly rates for each title are stored in a second Excel file. For every teacher in evidence, we need to find their hourly rates by matching their title to the second Excel table. Then this information needs to be inserted to the main database.

Number of hours

After the hourly rates are determined, we need to determine which courses the teacher will teach, how many hours of tutorials and lectures will he lead, and how many students will he examine. This is the most problematic part of the process, because there is no standardized way of getting this information. It would be ideal if there was a system which would provide this information. For each teacher, it would show how many hours for each course will he be teaching. The information system KOS is capable of doing this. However,

this feature is not used at our faculty. Therefore, to get this information, the "timetable" application and calendar have to be used. I can gather the required data either by browsing the teacher´s **personal calendar**, or the **schedule of the course** that he is teaching. Both approaches have some drawbacks.

First option is to look at the personal calendar of the teacher. The benefit of this approach is that I can see all his activities at one place. However, there can be different schedules for odd and even weeks, therefore I have to browse the whole semester to find the exact number of hours. Another drawback is that a course can be taught by multiple teachers. The hours then have to be split between the teachers, but from the personal calendar it appears that the teacher is alone in the course.

The second option is to look at the schedule of the course. This partially solves the problem with multiple teachers – when there are two teachers leading the course, I can see it. However, it works only for two teachers. If there are three or more, the information is lost. The drawback of this option is that I have to examine all courses the teacher will teach. Figures 3.2 shows how these two options look like.

In essence, the problem occurs when there are multiple teachers teaching the same study group. In these cases, some additional communication with the teachers is usually needed to gather this data.

The number of examined students will be the capacity of the study group the teacher is leading.

Finally, the total salary is computed using the hourly rates, taught hours, and examined students. At this point, I have all information needed to create the work performance agreement and related documents in an Excel database. An example of the database is also provided in the appended medium.

Creating agreements

Agreements are created using the mail merging feature of the Word application. This feature allows me to define a template with variable sections. I can then map the variables to columns in an Excel file (this is the reason Excel is used to store the information about teachers). For each row in the table, a document is automatically generated. Figure 3.3 shows how the Word template cooperates with Excel table. After all documents are generated, it is necessary to go through each one, check it, and manually save it. Mail merging feature provides only the option to save all files in one large document, which is not something I would want.

Creating reports

One additional document has to be created for each employee. If the salary does not exceed 10 000 CZK per month, it is not necessary to pay social and

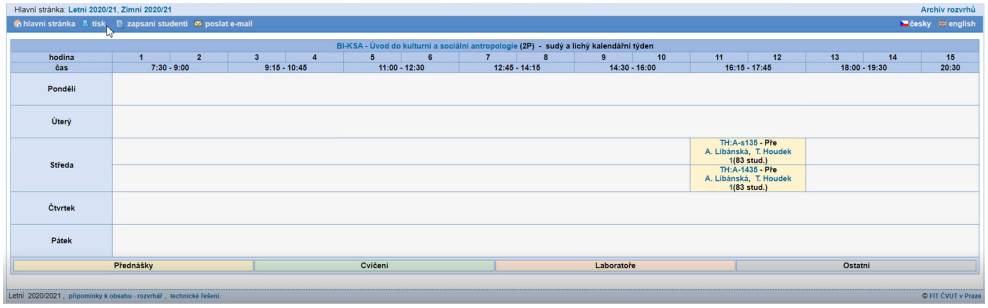

Figure 3.1: Course schedule

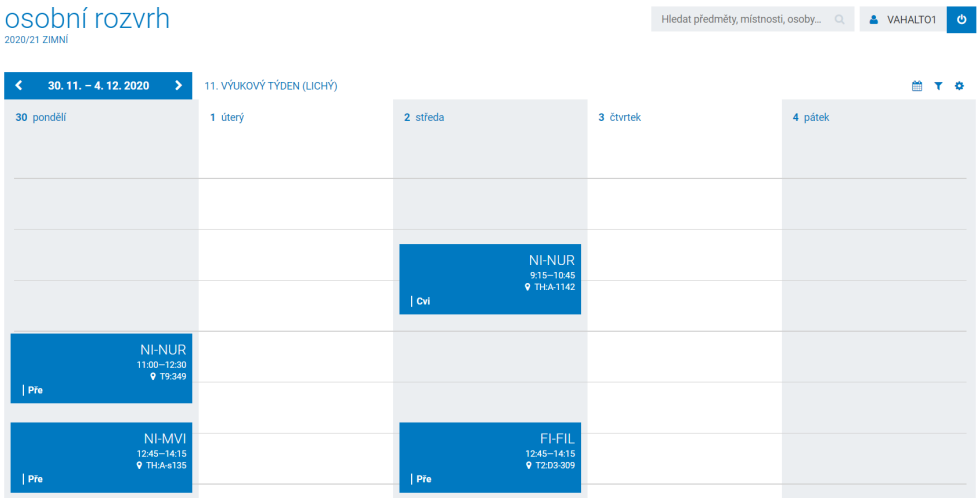

Figure 3.2: Personal calendar

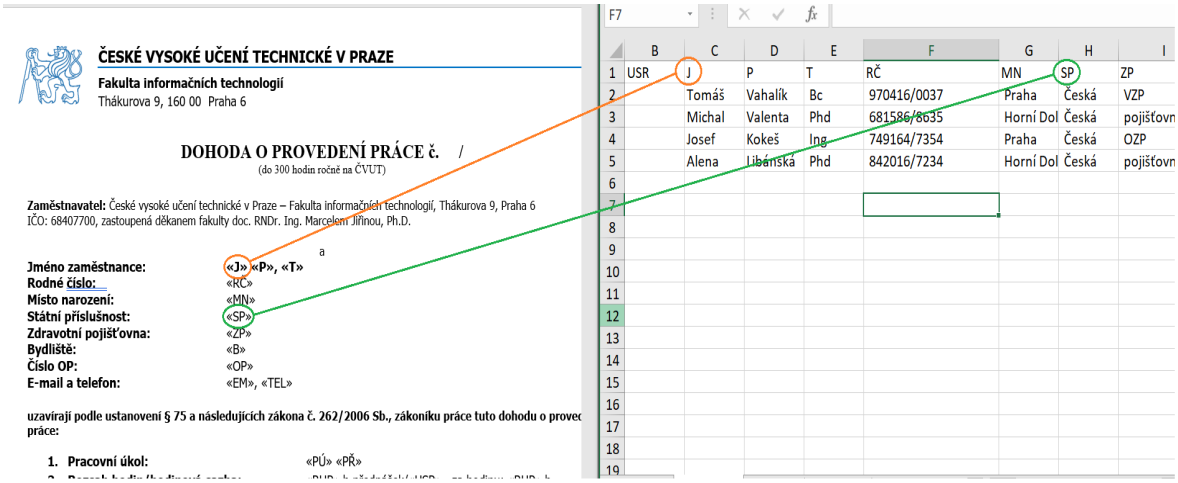

Figure 3.3: The use of mail merge feature in Word

health insurance. Therefore, if the total salary is lesser than 10 000 CZK, a simple Word document is created, which only tells the amount of money that should be payed to the teacher. This document is called "**Command for payment**".

If the total salary is higher than 10 000 CZK, it is needed to create a report describing teacher´s activities and provide information how much money he should get each month. This document is called "**Work report**" and will be created using Excel. There will be multiple sheets in the file (one for each month). In every sheet will be a list of activities (leading lectures/tutorials), name of the courses, hourly rates, and number of hours for each activity. Additionally, there will be a payment that should be sent to the teacher that month. The sum of payments has to correspond to the total salary stated in the work performance agreement.

Sending the created documents to the personal department

At this point, two documents for each employee have been successfully created. Now, all documents have to be sent to the personal department via e-mail. There they will be printed and signed by the dean of the faculty. After the documents have been signed, they are returned to me and I have to invite the employees to come and sign them as well.

3.3 Scoping the project

In this section, I will go through each part of the process and discuss whether it will be part of the automation. I will also mention how the steps could be improved by using other methods than RPA.

Creating the schedule

According to my opinion, this activity is impossible to automate with the use of RPA nor in any other way. It contains a lot of communication and decision making that only a human being is capable of (at least at the time of writing this thesis). It will therefore be out of the scope of this project.

Gathering personal information

This activity consists of two related steps - sending every teacher an email with the questionnaire and then wait for replies and copy information from each questionnaire into the Excel table.

Both steps could be automated by using RPA. However, I have decided to exclude the first step from the scope of this project for the following reasons. It could be automated by using other technologies such as mailing lists, which allow the user to send identical emails to a group of people. The effort the

user would make to send the emails would be similar to run the robot that would perform the task for him. I also think that each message should be personalized, which would not be easy to automate.

On the other hand, searching the incoming emails for specific replies, downloading the attached document, and copying the information is a tedious manual task which is an ideal target for using RPA. It will be in the scope of the project.

Filling the Excel table

This is according to Mrs. Libánská the most time consuming part of the process and therefore automating this activity would be a substantial improvement to the current state. It will therefore be in the scope of the project.

As mentioned, there are two additional categories that need to be filled in the table – hourly rates for the teachers and the hours they will actually be teaching. Filling the hourly rates is straight forward – simply copying data from one Excel file to another.

The second part is the problematic one, as there is no standardized way of getting this information. One way of improving the process would be to use the feature of KOS system, which is capable of providing information about a person along with the number of hours he teaches each course. I have stated that currently there are two ways of getting the desired information manually. Scraping through the course´s schedule or through the personal calendar of the teacher. For this project, I have decided to gather this information from the personal calendar by using RPA. This method would give incorrect output for courses, where more teachers teach a single group of students simultaneously, but these situations will not be accounted for.

Generating work performance agreements

This step is done by applying the mail merging feature of Word to a given template. Using mail merge feature is already a large time saving optimization opposing to filling in all values manually. However, there still remains a lot of manual work in saving the outgoing documents. And according to Mrs. Libánská, other departments don´t use this feature at all and fill in the information manually.

One way to improve this step would be to download the add-on to Word, which allows us to save each document to a separate file (found at [31]).

However, this would require an experienced user capable of running macros, so this approach would not be usable for ordinary users with zero programming experience.

Generating and manually saving each document is a good use case for RPA and will be in scope of the project.

Generating work reports

This step requires some decision making (checking if the total salary of the teacher is lower or greater than 10 000 CZK and either creating the "Command for payment" in Word or "Work report" in Excel). This can easily be done by a machine and doesn´t require a human being. Both activities are therefore suitable for RPA and will be in the scope of the project.

The work report file requires to separate working hours between individual months. After consulting with Mrs. Libánská, I have decided to separate the working hours equally between months.

Sending documents to the personal department

Opposing to contacting each teacher, this message will have only one recipient and therefore doesn´t have to be personalized. This is why I have decided to include it in the project scope to demonstrate how RPA can be used for sending emails.

Exceptions in the process

There are some exceptions in the process that are processed in a different way than the main line. These include teachers that teach combined courses or have a working relation with other parts of CTU. This project will focus only on performing the main line.

3.4 Conclusion

I have explored several processes that occur at the Department of Software Engineering at my faculty. I found the process of creating work performance agreements for teachers to be the most suitable for automation using RPA.

I have then analyzed this process carefully, understood its goals and actions that need to be taken to achieve these goals.

Finally, I set the scope for the implementation phase of the project by deciding which tasks are the most suitable for RPA and automating them would save the most time for the person performing them. Figure 3.4 shows a mind map representing the scope of the project.

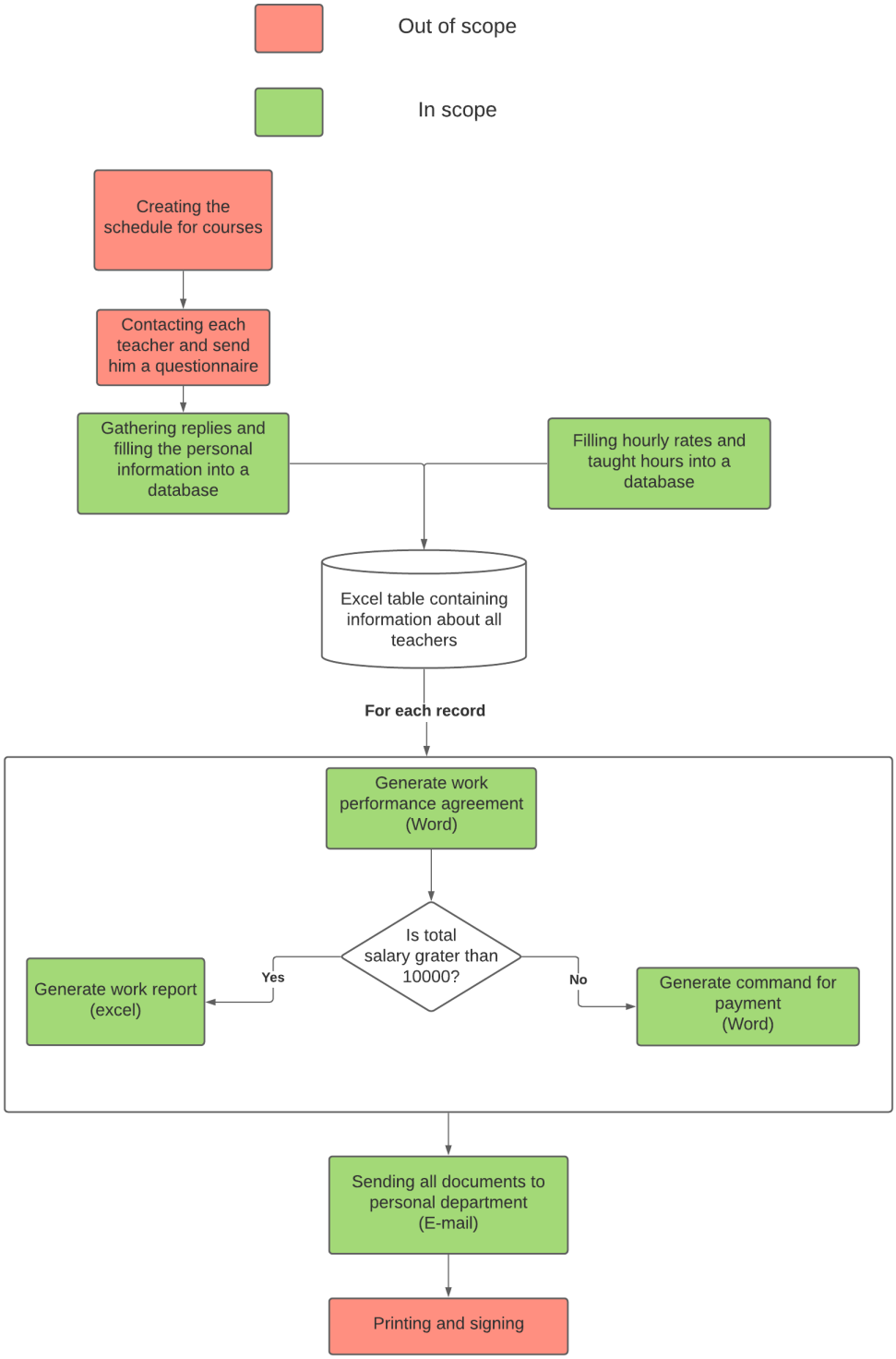

Figure 3.4: The scope of the project

Chapter 4

Realisation

4.1 The structure of solution

In this section, I will explain how I designed the solution to be used and the folder structure of the data used in the process.

Attended or unattended robot

First step was to decide whether the solution should be implemented as an attended or an unattended robot. I chose the attended robot version. The robot may encounter some problems during execution that cannot be automatically resolved and may require human intervention. For example, the questionnaires may have missing or incorrect values, the name of the teacher is not found in the timetable, etc.

This is also why I decided to split the solution into three executable workflows. Fist workflow will fill in all information required to generate the documents into the Excel database. The human employee can then check if any errors occurred and the database is filled correctly. Then he/she may run the second workflow, which will create all documents (two per database record). Then there is another space for human intervention to check if all documents have been generated correctly. The third workflow will compress the generated documents into a zip file and send it via email to the personal department.

Folder structure

Figure 4.1 shows the structure of the folder, where all data necessary for the process will be stored. In the **Database** folder will be the Excel file containing all information about the teachers. The **Documents** folder will serve as the output folder for the generated documents. All work performance agreements, commands for payments and work reports will be stored here. In the **Questionnaires** folder will be all questionnaires. The folder will have

```
C:.
|   Config.xlsx
|   tree.txt
|
+---Database
|       Database.xlsx
|
+---Documents
+---Questionnaires
|   +---Done
|   +---Failed
|   \---New
+---Rates
|       Rates.xlsx
|
\---Templates
        AgreementTemplate.docx
        CommandForPaymentTemplate.docx
        QuestionnaireTemplate.doc
        WorkReportTemplate.xlsx
```

Figure 4.1: The structure of the folder with data

three subfolders - **New, Failed** and **Done**. In the *New* subfolder will be questionnaires that have been downloaded, but haven´t been processed yet. In the *Failed* subfolder will be questionnaires, that haven´t been filled correctly and an error occurred during their processing. And in the *Done* subfolder will be successfully processed questionnaires. **Rates** folder will contain an Excel file with defined hourly rates for all titles (none, Bc., Ing., Mgr. etc.). The templates for work performance agreement, command for payment and work report are stored in the **Templates** folder. I will describe the functionality of the **Config** file in the next section.

4.2 Initializing the workflow

In this section, I will describe the first step that is made after running the robot. This initialization will be used for all three workflows I will create.

Reading Configuration file

During the execution, the robot will need information about the folder structure discussed above. It will need to know where the database and templates are located, and where to store the newly created documents. It is considered best practice not to have these values hard-coded in the program, but to use a file from which the robot will read them. The path to this file is the only

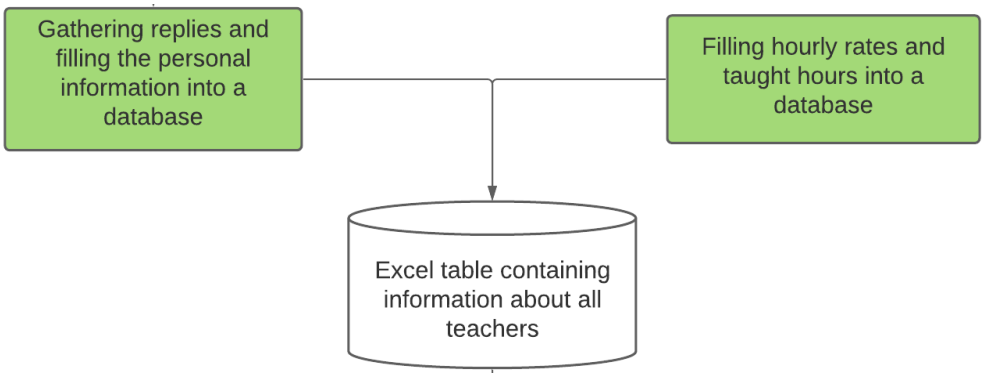

Figure 4.2: Scope of the first workflow

hard-coded value that will be in the program, every other parameter may be changed only in the configuration file without changing the code.

In my implementation, the configuration file is an Excel table where in the first column are the names of the parameters that should be passed to the robot, in the second column are the values. The robot parses this file and creates a global dictionary variable that can be used throughout the whole workflow.

Closing and reopening applications

Before interacting with any application, I have to make sure that it is in the state I expect it to be. In my case, the robot will interact with Word and Excel applications. Therefore, it will first terminate all instances of Word and Excel processes if there are any, and then relaunch them to make sure they have not been interacted with before the execution started.

4.3 Filling the database

In this section, I will describe how I implemented the first executable workflow, which fills all necessary information into the Excel database. The scope of this workflow is shown at Figure 4.2.

I have implemented this workflow as a state machine with six states. I will now describe each state, its goals and functionalities.

"Init" state

This state is the entry point of the workflow. It performs the initialization steps as described above – reading configuration file, closing and reopening applications. In this case, two Excel files are opened. One is the database where I will store the information from questionnaires and the second is a file containing hourly rates for different titles.

"Read emails" state

The goal of this state is to go through the mail inbox, find all emails that contain filled questionnaires, and store the attachments into a specified folder.

There are multiple ways that UIPath can manipulate emails. First option would be to use activities that download incoming mail from a server using POP3 or IMAP protocols. These activities require to configure the server URL, port, username, password, and a mail folder from which the mail should be downloaded.

If a user has an Microsoft Outlook application installed on his device, their is another option - an activity designed to interact with the desktop Outlook application. This activity requires less configuration – only thing it needs is the account and name of the mail folder. It also has a feature, which allows me to filter the messages I want to download. This is why I used this approach instead of the first one. I provided the following code to filter the mail messages I want to process – incoming messages having the subject specified in the configuration file.

```
"[Subject] = '" + Config("QuestionnaireSubject") + "'"
```

The POP3 and IMAP variants download all messages and I would have to go through each one of them.

Attachment of every email (filled questionnaire) is saved to the *New* subfolder of the *Questionnaires* folder.

"Process all questionnaires" state

The goal of this state is to go through all unprocessed questionnaires, enter the information contained in them into the Excel database, and move the questionnaire into the *Done* folder if it was processed correctly, or the *Failed* folder, if an error occurs (for example some field was not filled).

This state loops through all the files in the *New* subfolder of the *Questionnaires* folder. For each file, two activities are performed. First activity parses the questionnaire and returns the information as a variable of type Dictionary (set of key-value pairs). Keys are the fields of the questionnaire, values are the answers (for example, *First name - John*). The second activity then takes this dictionary and enters the values into the Excel database. Below I will describe these two activities.

Parsing word file

There is no way how to extract structured data from a text file using UIPath. It only provides a function to read the whole text file and store it in a String variable. Therefore, I had to use string manipulation functions to extract the information in a structured manner. Here are the steps I followed:

1. Load the whole questionnaire into a String variable
2. Split the variable by newline character to get an array of lines
3. Split each line by the : character. The first half of the line will be the key, the second half will be the value.
4. Add the key-value pair into the dictionary

When the whole file is processed, the dictionary is returned and the second activity is invoked.

Entering information into database

There are two ways how to insert data into an Excel table. First option is to enter the data cell by cell using "Write cell" activity. The second and more efficient way is to create a variable of type DataTable. This is a table consisting of rows and columns. Then I need to only specify the cell in an Excel file, where the top-left corner of the DataTable should be and the whole table is written into the Excel file at once.

I used the second option and transformed the Dictionary containing information from the questionnaire into a DataTable. This was not an easy step. First, I used the "Build data table" activity to define the structure of the table (number and names of columns). Then I needed to get all values of the dictionary as a List. I created a List variable and then used a series of "Invoke method" activities to fill the list with every value of the dictionary in the correct order (the invoked method was the *Add* method of the list). Finally, I used the "Add Data Row" activity to store the List as a row in the DataTable.

After that, I need to decide if the information should be appended to the database, or an existing entry should be modified (if the teacher has already taught at the faculty in the past, there should be a record for him in the database.) To do so, I try to match the name and surname from the questionnaire with some entries in the database using. I used "For each" and "if" activities to determine whether an entry is new. If a match is found, the existing entry will be replaced, otherwise a new entry will be appended at the bottom of the database.

"Add rates" state

The goal of this state is to fill the hourly rates for leading lectures and tutorials for each teacher in the database.

The implementation of this state is straight forward. Firstly, I create two DataTable variables. The first will be the table containing data from the teacher database, the other will contain data from the file containing information about hourly rates for different titles. UIPath has an activity

that reads a whole Excel file and stores it as a DataTable – "Read Range" activity.

Then, I iterate through the rows of the first table (row = teacher) and I match their title with the second data table. When the title is matched, the corresponding hourly rates are copied.

Finally, the DataTable with updated rates for teachers is written to the database file (again by using "Write Range" activity).

"Get working hours" state

The goal of this state is to find which courses each teacher will be leading. For each course, it will find the number of hours the teacher will be leading tutorials and lectures. Finally, the total salary will be computed by multiplying the working hours with the corresponding hourly rates gathered in the previous state.

The first step of this state is the same as in the last state - read the database and store it as a DataTable variable.

Up to this point, the workflow didn´t simulate any UI interactions, it was mainly using string manipulation and inserting data into an Excel file. Now, it will have to search for personal calendars of teachers. This information will be gathered from the website *timetable.fit.cvut.cz* and many clicks and typings will be used in the process.

Logging in

The website is available only to registered users (people with the FIT CTU username and password). If no credentials are stored in the cookies, the user is prompted to log in. Otherwise, the user is presented with his personal calendar and may perform the desired operations.

This behaviour must be considered by the robot. After navigating to the site, the robot will check whether he needs to log in first. If so, it will fill in user credentials and click the "Log in" button. If no log in is required, this part is skipped.

I implemented this functionality using the "Element exists" activity. The robot will try to find a defined element on the screen. It will return *true* if it succeeds, *false* otherwise. Using this activity, the robot will try to find the "Log in with CTU credentials" button (see Figure 4.4 for demonstration). If it succeeds, it will know it has to take the steps necessary to log in.

When the robot is logged in, it will perform the following steps for every teacher in database

Finding the teacher

To view other people´s calendar, I have to type their name into the search field. A dropdown menu will appear with all people matching the filter as

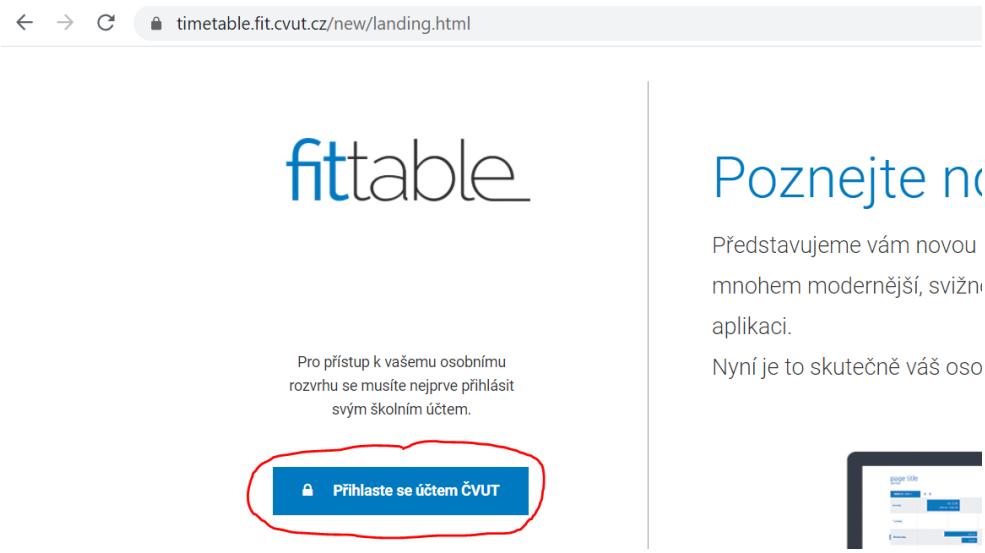

Figure 4.3: Trying to find the ”Log in” button

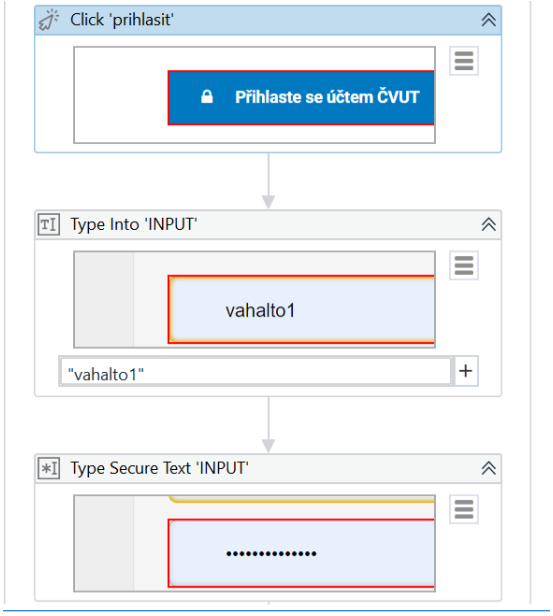

Figure 4.4: Code sample performing the login

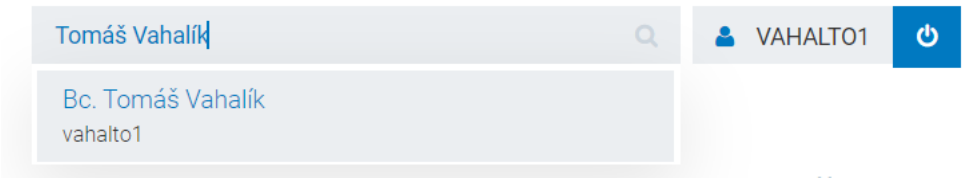

Figure 4.5: Searching for person

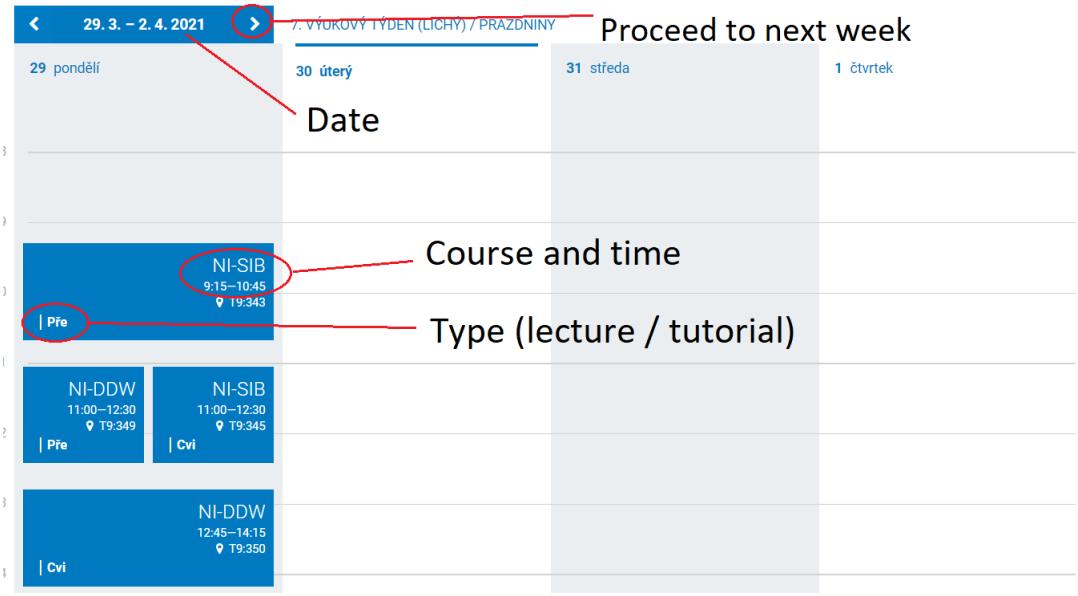

Figure 4.6: Structure of the calendar

shown on Figure 4.5. The robot will enter the name and surname of the teacher it is currently processing, and then click the element whose inner text contains the name and surname (It also contains the title of the person, so one-to-one match is not possible). Then the calendar for the searched person appears. For this project, I assume there are not two people with identical names.

Gathering information from page

Figure 4.6 shows, which information can be found in the calendar. The robot needs to detect all elements on the page that represent events. For each event, the course, type, and duration have to be determined.

This is not an easy task, as the number of these elements is not defined and they can appear anywhere on the screen. I used the source code of the page to determine that every event is contained in a *div* tag with the class *event regular*, *event half* or *event half-first*. Then I used the "Find children" activity, which allows me to specify a parent element (the whole page) and find all child elements that meet the given criteria (their class starts with the string "event").

To gather the desired information from each event element, I used the source code of the page again. I found out that each information is contained in a *div* with a defined class – the course has a class "head-name", time has a class "head-time" and type has a class "head-type". I then iteratad through all found event elements and for each used the "Find children" activity to get

all the information I need.

I store the gathered information in a Dictionary variable. The key in this dictionary is the name of the course, the value will be a pair of numbers – total number of hours leading lectures and the total number of hours of leading tutorials for this course (one hour equals 45 minutes).

Going through whole semester

Now, the robot is able to gather all information for one week. However, the number of hours of lectures and tutorials for the whole semester is needed. Unfortunately, this cannot be done by simply multiplying the weekly hours with the number of weeks in semester. It is because some events happen only every fourteen days and there can be many anomalies during the semester (Easter holiday, etc.) The robot therefore has to go through all weeks by clicking the *next week* button (see Figure 4.1). The computed hours are added to the dictionary.

Once all weeks are processed, the dictionary values are inserted in the DataTable. The robot then navigates to the first week of the semester(by using a specified date as the URL parameter) and continues with processing the next teacher.

Updating database

Once all teachers are processed, the DataTable is once again used to update the Excel file. There is one special column in the database, where the data will be stored in the following format:

```
subject hours_of_lectures hours_of_seminars; subject ....
```

This format will be useful during the implementation of generating work reports.

"Cleanup state"

This is the terminal state of this workflow. A cleanup is performed by closing all opened applications - both Excel files and the browser tab with calendars.

At this point, the Excel database is fully updated and I have all the required information to generate the necessary documents. The whole state machine of workflow one is shown at Figure 4.7

4.4 Generating documents

In this section, I will describe how I implemented the second executable workflow. Its goal is to create a work performance agreement and related documents for every teacher in the database, which was filled by the previous

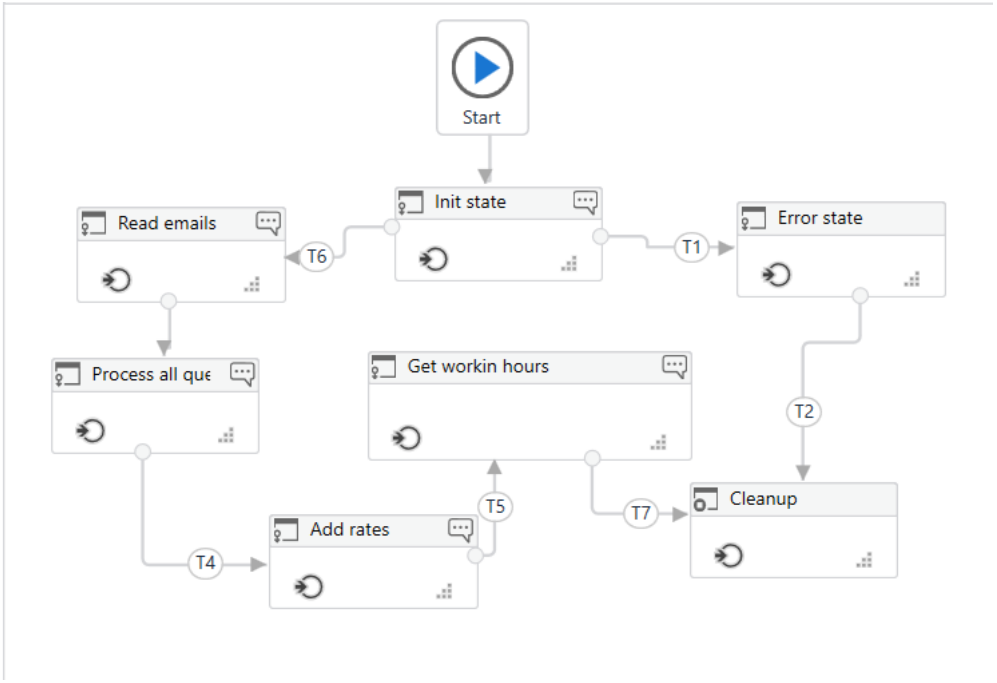

Figure 4.7: First workflow state machine

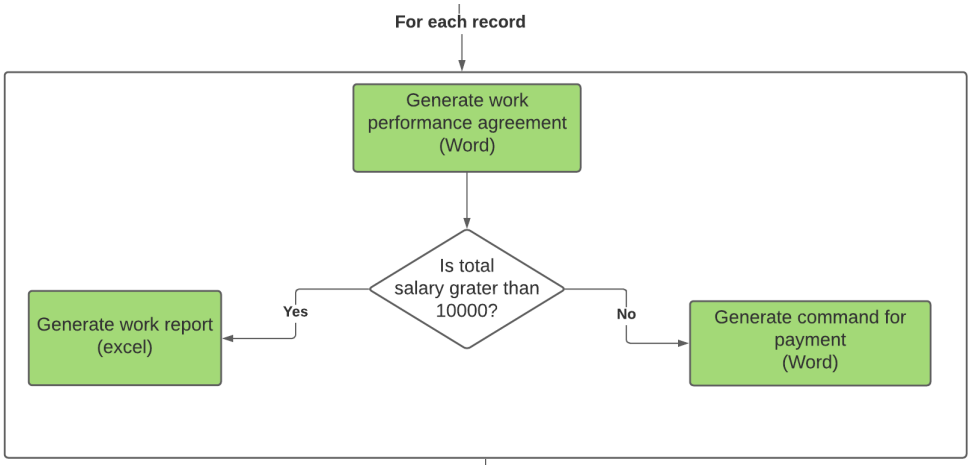

Figure 4.8: Scope of the second workflow

workflow.

There are four main parts of his workflow:

1. Initialization
2. Generate work performance agreements

3. Generate commands for payment
4. Generate work reports

The initialization step does the same work as in previous workflows - closes the applications and reads the configuration file.

I will describe how the other steps work in the following sections.

Creating work performance agreement

The goal of this activity is to generate work performance agreements using the provided Word template and the mail merging feature.

Opening the template

Firstly, the template has to be opened. This is done by using the "Open application" activity. Path to the executable file of the Word application has to be specified. Next, the robot opens the template in the same way as a human user would do it:

1. click the "Open" button
2. click the "Browse" button
3. provide a path to the template file in the popup window
4. click the "open" button in the popup window

The correct file is now opened and we can interact with it.

Performing mail merging

Once the template is opened, the mail merging has to be performed. The robot will do the following steps. For better understanding, figure 4.9 highlights the important elements in the window.

1. Click the "Mailings" tab on top of the window (Nr. 1 at the screenshot)
2. Click the "Select recipients" button (Nr. 2 at the screenshot)
3. Click the "Use an existing list" item in the dropdown (Nr. 3 at the screenshot)
4. Provide a path to an Excel file in the popup window. In this case, this is the "database" file.
5. selecting the correct sheet of the Excel file (Nr. 4 at the screenshot)
6. clicking the "Ok" button

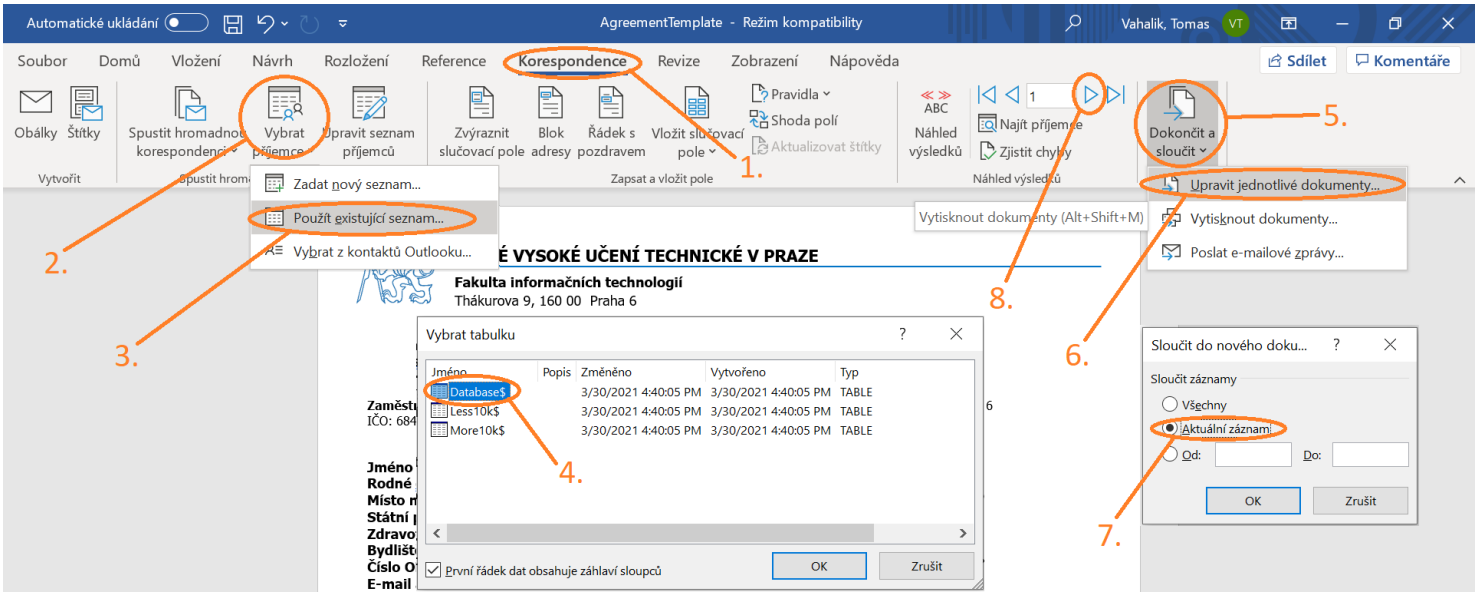

Figure 4.9: Screenshot of the template word file

At this point, the data from the "database" file are automatically filled in the template.

Saving each file

As mentioned before, every document has to be saved manually. To do so, the robot needs to know how many files he will create. This is done by reading the "database" file and counting its rows. Then, another sequence needs to be performed:

1. Click "Finish & Merge" button (Nr. 5 at the screenshot)
2. Click "Edit individual documents" (Nr. 6 at the screenshot)
3. Select the "Current file" radio button (Nr. 7 at the screenshot)

Then, a new word window will appear with a single finish work performance agreement. The robot will attach to this newly opened window and save it as a user would (Click "File" - "Save as" - "Browse" - provide path - "Save"). The file will be called "DPP_surname" and will be stored in the "Documents" folder.

When the agreement is saved, the robot will close its window, attach back to the template, and click the "next" button (Nr. 8 in the screenshot) to proceed to the next document. This sequence is done for each employee in the database.

Finally, the template window is closed as well without saving the changes made to it, so that it can be used again in the future.

*Comments on implementation I used the recording feature of UIPath a lot during the development of this part of the workflow. There are many clicks and typings that need to be done. The sequence of actions is linear and doesn´t change for different data (it is identical for all teachers). This is why recording is suitable for automating these actions.

I used the Desktop recording mode and performed each action manually once. Then, I added a "For each" activity to go through each document when saving. The next step was to parameterize the activities, which type into text fields. Finally, I changed the selectors for UI elements in a way that they are more reliable than the default ones. I used the "Simulate type" and "Simulate click" input methods wherever possible. Both the pros and cons of this method are applied - it is the fastest method, but not all UI elements support it. Where the method couldn´t be used, I used the default method instead.

Creating a command for payment and work report

The goal of this activity is to split the teachers into two groups - teachers with total salary greater than 10 000 CZK and those with total salary less than 10 000 CZK.

Splitting the teachers was a very easy task thanks to a UIPath activity "Filter data table". This activity can filter the provided data table according to the provided criteria and store the result into a new data table. I used this activity twice to get teachers with total payments greater and lower than 10 000 CZK. After splitting the teachers, the results are written in the database sheets called "More 10k" and "Less 10k".

Command for payment

Command for payment documents are created the same way as work performance agreements. Once again, the mail merging is executed in Word and every result is saved. The only difference is that the "Less 10k" sheet is selected instead of the "Database" one.

Work report

Work reports are stored in Excel files. There is a separate sheet for each month. Each month contains information about how many hours the teacher taught each subject and how much money has he earned. The last month is reserved for specifying how many students the teacher has examined. The structure of the report is shown in Figure 4.10 The teacher´s name, list of activities, and hourly rates only need to be filled in the first sheet (first month),

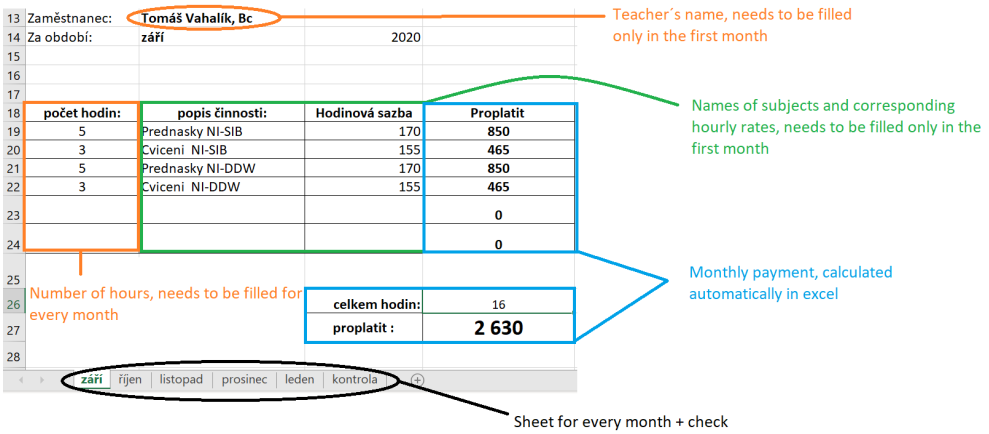

Figure 4.10: Structure of the Work Report file

while other months will copy them. For example, cell *B20* in the *October* sheet will have the following value:

```
=september!B20
```

This way, whatever appears in the first sheet will be copied to others. Monthly salary is also calculated in Excel and doe not need to be performed by the robot.

The robot needs to do the following tasks:

1. Write teacher´s name int the first month
2. Fill the number of examined students in the last month
3. Write the list of activities in the first month
4. Write the hourly rates in the first month
5. Write the number of hours for each activity each month
6. Save file

The data necessary for these tasks are read from the ”More 10k” sheet of the *database* Excel file and stored as a DataTable variable. Filling the teacher´s name and hourly rates in the work report is straight forward with the use of a series of ”Write cell” activities. To get the list of individual subjects and the number of hours for each of them, the format mentioned in section 4.3 will come in use. The robot splits the cell by the semicolon character to get information about each subject, and then each result by the space character to get the name of the subject, hours of lectures, and hours of seminars.

Next, it is necessary to split the hours between individual months. There are 5 months in the work report, but the last month contains information only about the examined students, not taught hours. Therefore, each remaining month should contain 1/4 of the taught hours. However, the number of hours doesn´t need to be divisible by four. Therefore, I used the following formula. First three months will have

$$\left\lfloor \frac{hours}{4} \right\rfloor$$

And the fourth will have

$$\left\lfloor \frac{hours}{4} \right\rfloor + (hours \mod 4)$$

The robot then loops for each month and fills in the information in the report using another series of "Write cell" activities.

Finally, the resulting file is saved in the same way as the work performance agreements - using clicks and "Type into" activities. There is one thing about the "Write cell" activity I would like to mention. The default behaviour is that the sheet where the activity writes the values is automatically saved. This is however not the behaviour I want in this workflow. This way, if I would fill the work report template for one teacher and then reopened it to start making another one, the values would remain filled from the last run. I therefore disabled the autosave feature for every "Write cell" activity I used.

4.5 Sending the documents

This workflow is very simple. Firstly, it once again reads the configuration file. Then, it uses the "Compress/Zip files" activity to compress all generated documents (the whole "Documents" folder 4.1") into a zip file. Then, it launches the Outlook application and uses the "Send Outlook mail message" activity to send the attachment. The receiver´s address is read from the configuration file.

4.6 Testing

In UIPath, developers can create test cases and test suites to test the workflows. A test case is also a workflow that consists of three sections – **Given, When,** and **Then**. In the **Given** section, everything that is necessary for initializing the test case is added (setting the environment, assigning variables etc.) The **When** section contains the test scope – the tested workflow is invoked. Finally, in the **Then** section, the results are verified – we check that the tested workflow ran correctly. [32] After the tests are run, the studio shows us how many activities in the workflow are covered by the tests.[2] This is shown at Figure 4.11.

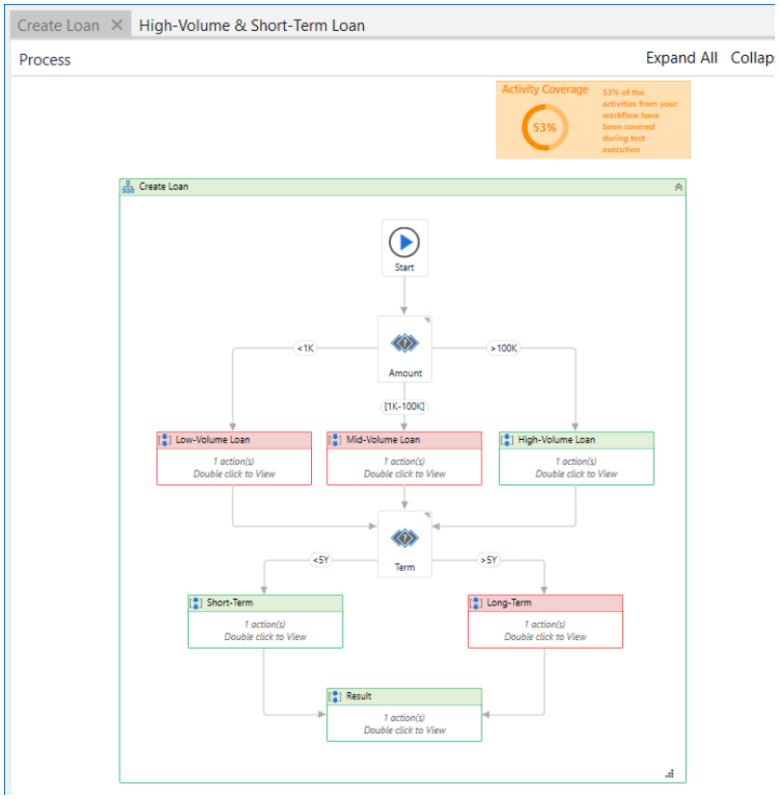

Figure 4.11: The test coverage, image taken from [2]

However, to use these testing features, the *UIPath Studio Pro* version is required. Because I do not have the pro version, I tested the implementation manually. I have created five questionnaires for testing purposes. The names in the questionnaires correspond to real people working at the Faculty of Information Technologies. All other personal information such as accounts or phone numbers are made up. I have sent these questionnaires to myself via email and they serve as the input data for the workflows.

Among the questionnaires, there is one containing missing values. The robot correctly notices this and moves the questionnaire into the "failed" folder. For every other person, the robot stores the information into the Excel database and finds their timetable. Then, all documents are successfully generated. There are three people with income larger than 10 000 CZK and one with income lower than 10 000 CZK. Therefore, three work reports and one command for payment are generated. Finally, I invoked the third workflow and the robot sent me all the created documents via email.

4.7 Summary

I have implemented the automation of the process using UIPath. The result is an attended robot, which performs all actions in the process for the employee. The solution consists of three executable workflows. The first is for gathering all the necessary information and storing it into an Excel database. The second creates documents for each teacher using Word and Excel application. Finally, the third compresses all generated documents and sends them via email. I have tested the solution manually by providing some dummy input data. Each workflow worked correctly.

Chapter 5

Evaluation

5.1 Impressions about UIPath

In this section, I will comment on my impressions about using UI-Path.

As stated in section 1.3, in UI-Path everything is done using prebuilt activities without much use of custom code. I appreciated this when automating any UI interactions like clicking, finding elements, and typing. UI-Path does its best in making these activities as easy to configure as possible. There is an activity almost for every possible action that the robot can make. The flow of activities is easily configured by dragging arrows between them (when using a flowchart or a state machine) or by switching their places in a sequence.

On the other hand, doing everything only by using activities without the possibility to use custom code brings some downsides. First of all, the developer is fully dependent on the scope of existing activities. If an activity doesn´t exist for a given action, it is hard to achieve the desired behaviour. I found this approach to be limiting when developing the logic of the robot like assigning to variables or manipulating strings. Every variable has to be declared in the studio interface by typing its name, setting the scope, and picking its data type. It is very time consuming, especially if the data type isn´t a common one (like "string") and has to be searched in the list of all types. Then, a special activity called "Assign" has to be used to save a value to the variable. Another example is adding an element to a collection. It is necessary to use an "Invoke method" activity, where we have to configure the method name, the target object, and method arguments (as mentioned in the section 4.3). However, on first sight it is impossible to see what exactly the activity does. Additionally, the sections of the workflow where many variables are used and modified (for example, performing string operations while processing questionnaires and saving elements into a dictionary), are confusing and hard to navigate, as the activities spread across a large area on the screen and due to the limited display space it is difficult to see the overall logic. It would be a lot easier for me to use several lines of VB.NET code to perform

the desired operations in one place. Another thing that I would appreciate would be the possibility to create custom data types (like "Teacher" class). In the current state, any object with some properties like "name" and "address" has to be implemented using a dictionary data type, which once again, I find a little confusing.

I like the debugging system in the UI-Path studio. It is easy to configure if I want to run the robot in the normal way or in debug mode. The debug mode runs considerably slower, but I can pause the execution at any time, check which activity is the robot currently performing and what are the values of any variables or arguments.

To summarize, I think that UI-Path´s main strength is the efficiency of creating a robot that simulates human interactions with a computer (which is, after all, the main goal of RPA). Solving problems that require some programming (like sorting data or string splitting) requires more time than simply writing the code, but after some time of getting used to the variables system and remembering the mostly used activities, it can be done without larger complications.

5.2 Applicability

In this section, I will discuss how the solution would be useful in a real environment. To determine how the robot would be applicable in the real environment, I showed the finished solution to Mgr. Alena Libánská, Ph.D. I presented how the robot is operated, what are the inputs and outputs.

We came to the following conclusions:

- The robot does a good job of simulating manual labor in the given process. The tasks are done in the same way as a human employee would, but much faster.

- Workflows 2 (generating documents) and 3 (sending documents via email) would be applicable in a real environment. Workflow 1 (filling the Excel table) would be applicable only to situations, where two people don´t teach the same lecture simultaneously (as stated in section 3.2). However, even if the Excel table would be manually filled and only workflows 2 and 3 were used, the automation would still be a large improvement to the current state and would save employee´s time.

- Mrs. Libánská has to process circa 40 agreements each semester. The process of creating the documents alone (without filling the Excel table) takes about **10 hours** and it usually takes two working days to do so. I have performed an experiment to determine how quickly can the robot perform the same task. I filled some dummy data into the Excel database and had the robot create the necessary documents for them

(40 work performance agreements, 20 commands for payment and 20 work reports). The task was done in **25 minutes 36 seconds**.

In section 1.1 I stated, that some employees may feel threatened by RPA technology. I also talked about this topic with Mrs. Libánská. Her opinion on the RPA technology is overall positive. She sees the benefits that it brings, it can help workers with repetitive manual tasks while leaving space for people to focus on more challenging aspects of the processes. She doesn´t feel threatened by the technology. She thinks that there will always remain some work to be done by human employees, even after the repetitive parts will be automated.

On the other hand, she thinks that embedding RPA in practice wouldn´t be smooth at all. It would require to persuade the employees to abandon the current way of performing tasks (manually) and to adopt new technology. She thinks that employees would be at first skeptical and it would take time for them to get used to using RPA and appreciate the benefits it brings. This human behavior corresponds to the **SARAH** model. That is a model describing how people react to changes. The stages are **S**hock, **A**nger, **R**ejection, **A**cceptance, **H**ope. According to the model, the users must overcome the Shock, Anger and Rejection stages before they accept the change. Then they will see that the change brings new possibilities and can have a positive impact on their life. [33]

5.3 Maintenance

In this section, I will comment on how the program would have to be maintained. There are several scenarios that would require to interfere with the robot´s code.

1. **New version of used applications** – the solution should be revised after every Word/Excel update. Even a slight change of the UI may make the robot start working. It has to be assured, that the robot can still correctly identify all necessary UI elements. A large change would be needed, if the user had Excel/Word application using different language then Czech. The same applies for the website with personal calendars.

2. **Changing the template documents** – robot has to be updated to match the current version of template documents. It must have correct information about the structure of the questionnaire and work report documents. Documents that are generated by Mail merging (Work performance agreements and Commands for payment) can be changed freely, as long as they preserve the mapping mentioned in section 3.2.

3. **Updating the semester range** – while searching for working hours of teachers, the robot has to start at a specified date. This date needs to be changed every semester to correspond with the first week of lectures.

4. **Change of credentials** – the robot performs login to outlook application and the website with personal calendars. If the credentials changed, the robot´s code would have to be modified.

From this ensues that the code should be revised at least once per semester and whenever a new version of Word/Excel is installed. Most of the changes (changing credentials/updating the semester range) are minor. I estimate that they shouldn´t take more than an hour to complete. The changes related to the new Excel/Word/Calendar would be more complex. The effort depends on the amount of changes made to the UI.

5.4 Embedding in practice

Even though the solution is already implemented and tested, there are still many problems that stand in its way to be used in practice:

1. **Purchase a UIPath licence.** I have developed the solution under the UIPath Community edition, which is free. As stated in [34] for the community edition:

   Small Teams may use the Products for their internal business purposes. Large Teams may use the Products only to test and evaluate their suitability and for non-profit purposes (e.g. education, hackathons, individual or institutional research, internal trainings).

   Small teams are organizations with less than 250 physical or virtual machines or users and less than the equivalent of five million USD in annual revenues. Although this use case may fall under the *institutional research*, additional research would be needed to confirm, that the Community edition would suffice, or if a purchase of enterprise edition is necessary.

2. **Approval of the ICT department**. It would be necessary to install UIPath studio on the school computer, where the robot would be run. Therefore, it would be necessary to acquire permission from ICT department to do so.

3. **Assign responsibilities for the robot**. There would have to be a person responsible for the deployment and maintenance of the robot. Also, as stated in section 1.1, the question remains, who should be responsible if the robot misbehaves. Should it be the person performing the process, the person who developed the robot, or someone else?

5.5 The development effort evaluation

In this section, I will try to estimate the cost of this solution to be developed and maintained. I will describe the activities that I did and how much time I spent. Then I will compute the costs using the average salaries of the corresponding occupations.

Firstly, I had to discover a suitable process for automation and understand it. This phase includes on-site consultations with Mrs. Libánská and then going through the process, identifying risks and scoping the implementation phase. The consultations took approximately 5 hours in total. Then it is hard to estimate the effort spent in going through the process and thinking about it, as it is done continuously. I would estimate the time to 25 hours, which makes it **30 hours in total**. In practice, this phase is the job of **RPA Process Designer**. His/her salary ranges from 84,000 USD to 132,000 USD annually [35]. To compute the estimated cost, I will take the average (108,000 USD) and divide it by the number of working hours in a year ($8 \times 5 \times 52$). This gives us approximately 52 USD per hour. The estimated cost of this phase would be **1560 USD (33 462 CZK)**

Next, I had to implement the solution. This phase includes setting up the environment for implementation, discussing whether the robot will have access to all applications or if some mockups are necessary, and finally implementing the robot. This phase took me **Two weeks in total**. However, a real company would already have the environment set up and would charge only the implementation phase. I estimate, that an experienced developer could create my solution in 20 hours. The implementation of a robot is the responsibility of **RPA Developer (Automation architect)**. His/her salary ranges from 128,000 USD to 170,000 USD annually [35]. The average is 149,000 USD per year, which gives us approximately 72 USD per hour. The estimated cost of this phase would be **1,440 USD (31,147 CZK)**.

Finally, there is a person who is responsible for the robot once it has been deployed. This role is called **RPA Production Manager** and his/her annual salary ranges from 68,000 USD to 125,000 USD. However, I don´t think this role would be necessary for this project. Therefore, the total cost of this solution would be approximately 1,700 USD (31,407 CZK). All hourly rates were based on salaries in the US.

5.6 Future work

The result of this thesis is an unattended robot that needs to be activated manually. For future work, I suggest to focus on unattended automation. That would require to build a robot (even for a smaller task then for this thesis) that would be very robust – acquire the possibility to create UIPath test cases, and make sure that every activity is part of some test suite. Then, the

resulting workflow should be integrated to the Orchestrator, where the unattended automation would be configured. Furthermore, trying to resolve the problems with embedding in practice could be a potential scope of future work.

5.7 Evaluating goals

In this section, I will evaluate the goals presented in chapter 2.

G1 - Get familiar with the UIPath tool to the level which allows creating a proof of concept in practice

I have explored the capabilities of UIPath in depth in section 1.3. I described what UIPath Studio and Orchestrator are. I also provided a comprehensive description of the development of UIPath robots.

G2 - Select a process at the Faculty of Information Technology of CTU and analyze how could it be automated using UIPath.

I have described the selection of a suitable process for automation in section 3.1. I decided to automate the process of creating work performance agreements at the department of Software Engineering. In sections 3.2 and 3.3, I provided a full description of the chosen process. For each step, I discussed how it could be improved and if RPA is usable for it.

G3 - Implement a solution that will automate the selected process.

I have created a robot that automates the selected process. It consists of three executable workflows. The first is used for filling the Excel database. The second is used for generating documents in Word and Excel. The third workflow sends the generated documents via email. I described how I implemented each workflow in chapter 4.

G4 - Discuss, how useful would the solution be in practice and what implications would it have on the current personnel.

The finished prototype would be usable in a real environment and would save the time of the employees. In chapter 5, I described which parts of the solution would be most useful, how the solution could be embedded in practice, and tried to estimated the effort spent on implementation.

Conclusion

The scope of my thesis was to explore the usability of RPA in a real environment. Firstly, I listed the general concepts of RPA, its strong and weak sides, and typical areas where it could be used. The result was that businesses can benefit from RPA with greater speed and accuracy in the processes at the cost of having to maintain the solutions regularly. After that, I briefly listed some RPA vendors. Then I provided an in-depth description of the UIPath tool. I described how the robots are implemented and provided information about UIPath Studio and Orchestrator features.

Secondly, I searched for a suitable process for automation to make a proof of concept. I went through some processes at the Department of Software Engineering on the Faculty of Information Technologies. For automation, I chose the process of creating work performance agreements for teachers. I depicted the process in detail and described which steps are suitable for RPA automation.

Next, I created a robot that automates the chosen process. I used the RPA tool and described how I progressed during the implementation. The result is an attended robot with three executable workflows, which performs the activities in the same way as a human employee .

Finally, I discussed how the solution would be usable in practice, how it should be maintained, and estimated the cost of work spent on the implementation. It shows that although the robot couldn´t handle some exceptions in the process, it would still be an improvement to the current state and would save the employee´s time.

I concluded that RPA has a potential to be useful in practice. The development of RPA robots using UIPath is comfortable and the results are good. I also found out that the development of RPA applications has many more layers than just to learn a tool and use it to create a robot. It requires a lot of analysis of the processes and communication with people. For embedding RPA in practice, the whole organisation must be prepared and support the decision. Otherwise, it is very difficult to deploy a robot to production.

Bibliography

[1] Best Robotic Process Automation (RPA) Software. [cit. 2021-03-11]. Available from: https://www.g2.com/categories/robotic-process-automation-rpa

[2] UIPath documentation on testing. [cit. 2021-04-14]. Available from: https://docs.uipath.com/studio/docs/rpa-testing

[3] Taulli, T. *The Robotic Process Automation Handbook: A Guide to Implementing RPA Systems*. Apress, ISBN 9781484257289.

[4] Dilmegani, C. RPA Tools Vendors: In-depth vendor selection guide [2021]. January 2021, [cit. 2021-03-10]. Available from: https://research.aimultiple.com/robotic-process-automation-rpa-vendors-comparison/

[5] Tripathi, A. M. *Learning Robotic Process Automation: Create Software robots and automate business processes with the leading RPA tool – UiPath*. Packt Publishing Ltd, ISBN 9781788470940.

[6] UiPath-Robotic-Process-Automation. [cit. 2021-03-11]. Available from: https://www.capterra.com/p/135186/UiPath-Robotic-Process-Automation

[7] rpa-editions-comparison. [cit. 2021-03-11]. Available from: https://www.automationanywhere.com/lp/rpa-editions-comparison

[8] Mullakara, N.; Asokan, A. K. *Robotic Process Automation Projects: Build real-world RPA solutions using UiPath and Automation Anywhere*. Packt Publishing Ltd, ISBN 9781839217357.

[9] Pratt, M. K.; McLaughlin, E. Business Process Outsourcing (BPO). [cit. 2021-03-12]. Available from: https://searchcio.techtarget.com/definition/business-process-outsourcing

[10] What is ZOHO CRM? [cit. 2021-03-13]. Available from: https://www.zoho.com/crm/what-is-zoho-crm.html

[11] Beal, V. Swivel Chair Interface. [cit. 2021-03-13]. Available from: https://www.webopedia.com/definitions/swivel-chair-interface/

[12] Van der Aalst, W. M.; Bichler, M.; et al. Robotic process automation. 2018. Available from: https://link.springer.com/article/10.1007/s12599-018-0542-4

[13] Clair, C. L.; O'Donnell, G. *The Forrester Wave™: Robotic Process Automation, Q4 2019 The 15 Providers That Matter Most And How They Stack Up*. Technical report, Forrester 2019.

[14] About Automation Projects. [cit. 2021-03-12]. Available from: https://docs.uipath.com/studio/docs/about-automation-projects

[15] UI-Path academy, RPA Developer Foundation learning plan, Project Organization lecture. [cit. 2021-03-14].

[16] UI-Path documentation on sequences. [cit. 2021-03-14]. Available from: https://docs.uipath.com/studio/docs/sequences

[17] UI-Path documentation on Robotic Enterprise Framework. [cit. 2021-03-14]. Available from: https://docs.uipath.com/studio/docs/robotic-enterprise-framework

[18] UI-Path academy, RPA Developer Foundation learning plan, lecture on UI Automation. [cit. 2021-03-16].

[19] UI-Path academy, RPA Developer Foundation learning plan, lecture on Selectors. [cit. 2021-03-15].

[20] UI-Path documentation on Image activities. [cit. 2021-03-14]. Available from: https://docs.uipath.com/studio/docs/image-activities

[21] UI-Path documentation on Input Methods. [cit. 2021-03-16]. Available from: https://docs.uipath.com/studio/docs/input-methods

[22] Forum discussion about Send Window Messages method. [cit. 2021-03-16]. Available from: https://forum.uipath.com/t/what-are-the-differences-between-simulate-and-send-window-messages/51965/5

[23] UI-Path academy, RPA Developer Foundation learning plan, lecture on Error and Exception handling. [cit. 2021-03-20].

[24] UI-Path documentation on recording. [cit. 2021-03-20]. Available from: https://docs.uipath.com/studio/docs/about-recording

[25] UI-Path academy, RPA Developer Foundation learning plan, lecture on recording. [cit. 2021-03-20].

[26] UI-Path academy, RPA Developer Foundation learning plan, lecture on Orchestrator. [cit. 2021-03-21].

[27] Kaya, C. T.; Türkyılmaz, M.; et al. Impact of RPA technologies on accounting systems. *Muhasebe ve Finansman Dergisi*, , no. 82, 2019.

[28] Huang, F.; Vasarhelyi, M. A. Applying robotic process automation (RPA) in auditing: A framework. *International Journal of Accounting Information Systems*, volume 35, 2019: p. 100433.

[29] Santos, F.; Pereira, R.; et al. Toward robotic process automation implementation: an end-to-end perspective. *Business Process Management Journal*, 2019.

[30] Madakam, S.; Holmukhe, R. M.; et al. The future digital work force: robotic process automation (RPA). *JISTEM-Journal of Information Systems and Technology Management*, volume 16, 2019.

[31] Word add-in to merge letters to separate files. Available from: `http://www.gmayor.com/individual_merge_letters.htm`

[32] Mayer, C. RPA testing — Test suite — Containers Given, Then and When. [online]. Available from: `https://forum.uipath.com/t/rpa-testing-test-suite-containers-given-then-and-when/222226`

[33] Vahalík, T. Slovníček pro business analytiky. July 2019, [cit. 2021-04-15]. Available from: `https://www.linkedin.com/pulse/slovn%C3%AD%C4%8Dek-pro-business-analytiky-popit-sarah-co-napsala-tom%C3%A1%C5%A1-vahal%C3%ADk`

[34] UIPath Legal Terms. [cit. 2021-04-12]. Available from: `https://www.uipath.com/legal/trust-and-security/legal-terms`

[35] Samarpit. RPA Developer Roles and Responsibilities. [cit. 2021-04-13]. Available from: `https://www.edureka.co/blog/rpa-developer-roles-and-responsibilities/`

Appendix A

Acronyms

RPA Robotic Process Automation

UI User interface

NLP Natural language processing

OCR Optical character recognition

CRM Customer relationship management

DLL Dynamic link library

FIFO First-in-first-out

STP Straight through processing

POP3 Post office protocol

IMAP Internet message access protocol

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `Document_templates` .. the directory with all document templates used in the process
- `Project_source` ..... the directory of source codes
  - `data` ..... the directory where the robot stores all data
  - `src` ..... a UIPath project containing the source codes
- `Video` ..... folder containing video of the running robot
- `Text` ..... folder containing the thesis in .pdf format