

---

# Connecting RPA Development and Business

- A Tool for Process Definition, Agile RPA Development and Maintenance -

---

Master's Thesis in Technology  
University of Turku  
Department of Future Technologies  
Software Engineering  
2021  
Ahmed Abdulghani



UNIVERSITY OF TURKU  
Department of Future Technologies

AHMED ABDULGHANI: Connecting RPA Development and Business  
- A Tool for Process Definition, Agile RPA Development and  
Maintenance -

Master's Thesis in Technology, 124 p.  
Software Engineering  
June 2021

---

The world of automation is changing rapidly, especially in the Robotic Process Automation (RPA) field, as companies implement software robots to alleviate the repeatable manual work.

This thesis has two purposes. The first purpose is to find out if it is possible to improve Robotic Process Automation projects' kick-off, communication, documentation and maintenance with a PDD-SDD Tool. And the second purpose is to demonstrate how the PDD-SDD Tool was designed and developed.

The initial hypothesis for the project: A specialized Tool for automating documentation and project kick-off could streamline RPA projects and mitigate communication related misunderstandings.

The methods used in the thesis assisted with understanding the needs of the RPA professionals that are working with the whole RPA project life-cycle. For creating the starting point, the hypothesis was constructed by analyzing the current issues in RPA projects in the industry. Based on the hypothesis an Initial Design of the Tool was created and interview questions were produced. Next, three rounds of structured interviews were held for data gathering and demonstrating the developed prototypes. Lastly, a final presentation of the Tool was given to the same study group and improvement proposals were gathered.

The findings of this thesis align with the initial hypothesis based on the data gathered from the interviewees during the interviews, the Tool aids the professionals with process definition by providing guiding questions during the definition process. Automated documentation based on the developers' code alleviated with documentation burden, which lead to up-to-date documentation, which also lead to better communication. The automatically generated code base for the developers created from the process definition and the up-to-date documentation enhanced maintenance work dramatically.

The value of this thesis lies with the actualization of a custom Tool developed specifically for MOST Digital's RPA projects' life-cycle. Also, the methodologies used in this thesis for data extraction could be further used to collect new features for the developed Tool or to develop a new tool altogether. This project was initiated in Fall 2019 and finished in Summer of 2021, and resulted with a fully functional StandAlone version of the designed PDD-SDD Tool.

Keywords: Robotic Process Automation, Software Development, Business Processes,  
Process Flow Visualization

# Contents

List of Figures

List of Tables

Listings

Acronyms

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Divided Masters Thesis . . . . .	2
1.2	The Structure . . . . .	2
1.3	The Relational Setup . . . . .	3
1.4	The Central Question . . . . .	4
<b>2</b>	<b>Overview of Robotic Process Automation</b>	<b>5</b>
2.1	Robotic Process Automation . . . . .	5
2.2	Potential . . . . .	8
2.3	Limitations . . . . .	10
2.4	Future . . . . .	11
<b>3</b>	<b>RPA Development</b>	<b>12</b>
3.1	Definitive Characteristics . . . . .	12
3.1.1	Process Definition . . . . .	13

3.1.2	Communication . . . . .	14
3.1.3	Documentation . . . . .	16
3.1.4	Development . . . . .	17
3.1.5	Testability . . . . .	19
3.1.6	Consistency and Maintenance . . . . .	21
3.2	RPA Examples . . . . .	22
3.2.1	Reading Data from Excel File . . . . .	24
3.2.2	Fetching Information from the Internet . . . . .	27
3.2.3	Saving Enriched Data to the Excel File . . . . .	32
3.2.4	Prerequisites and Settings for the Provided Examples . . . . .	32
3.2.5	Summary . . . . .	35
<b>4</b>	<b>Improving Development at MOST Digital</b>	<b>39</b>
4.1	Project Life Cycle . . . . .	39
4.2	Identifying Challenges . . . . .	42
4.3	Deriving Requirements for Improvement Proposal . . . . .	44
4.4	Research Questions . . . . .	47
<b>5</b>	<b>Improvement Proposal and Work Distribution</b>	<b>50</b>
5.1	PDD-SDD Tool . . . . .	50
5.2	Solution Compared with Other Mainstream RPA Platforms . . . . .	54
5.3	Work Distribution and Initial Roadmap . . . . .	55
<b>6</b>	<b>Research Questions and their Validation Method in this Thesis</b>	<b>57</b>
6.1	Validation Method . . . . .	58
6.2	The Professionals . . . . .	60
<b>7</b>	<b>The Interviews</b>	<b>64</b>
7.1	The First Interviews . . . . .	66

7.1.1	The Structure . . . . .	66
7.1.2	The Findings . . . . .	67
7.1.3	The Challenges . . . . .	68
7.1.4	The Improvement Proposals . . . . .	70
7.2	The Second Interviews . . . . .	73
7.2.1	The Structure . . . . .	74
7.2.2	The Findings . . . . .	75
7.2.3	The Challenges . . . . .	75
7.2.4	The Improvement Proposals . . . . .	76
7.3	The Final Interviews . . . . .	79
7.3.1	The Structure . . . . .	79
7.3.2	The Findings . . . . .	79
7.3.3	The Challenges . . . . .	80
7.3.4	The Improvement Proposals . . . . .	80
7.4	Summary . . . . .	81
<b>8</b>	<b>Tool's Development</b>	<b>86</b>
8.1	Ideation and Initial Design . . . . .	86
8.2	Challenges, Improvement Proposals and Tool's Functionalities . . . . .	88
8.2.1	Derived Functionalities and the Tool's Requirements . . . . .	91
8.3	Prototypes and development . . . . .	92
8.3.1	The Selected Technologies . . . . .	95
8.3.2	The First Prototype . . . . .	97
8.3.3	The Second Prototype . . . . .	100
8.4	Standalone tool . . . . .	105
8.4.1	The Definition View . . . . .	106
8.4.2	The Technical View . . . . .	110
8.5	Tool's Process and Validations . . . . .	115

<b>9 Conclusions</b>	<b>117</b>
<b>References</b>	<b>120</b>

# List of Figures

3.1	The Media Richness Theory (MRT) with communication effects of different communication media [1]. . . . .	16
3.2	The activities in a typical software development process [2]. . . . .	21
3.3	The input excel file for provided RPA examples. . . . .	24
3.4	The main workflow in UiPath process. . . . .	25
3.5	Reading data from the excel file with UiPath. . . . .	26
3.6	Fetching stock symbol data from nasdaq.com with UiPath. . . . .	30
3.7	Looping the DataTable in UiPath. . . . .	31
3.8	Searching for the stock symbol in UiPath. . . . .	33
3.9	Updating the DataTable in UiPath. . . . .	34
3.10	Saving enriched data to the excel file with UiPath. . . . .	34
3.11	The output excel file for provided RPA examples. . . . .	35
3.12	UiPath example's main XAML file, UiPath Studio Community Edition 2019. . . . .	38
4.1	Intertwined aspects in RPA development life cycle [3]. All the processes circle around communication and documentation. . . . .	48
5.1	Initial design of PDD-SDD Tool. . . . .	52
5.2	Initial rough design of VUI views. The VUI views are not presented all at once, rather they are accessed one level at a time from left to right. . . .	53



5.3	Initial roadmap for the PDD-SDD Tool’s development. . . . .	56
7.1	The structure of the first interviews, and how the Research Questions relate to the topics discussed. . . . .	67
7.2	Challenges and Improvement Proposals in each interview iteration. . . . .	84
8.1	Challenges and Improvement Proposals linked. . . . .	89
8.2	Additional functionalities for further development. . . . .	90
8.3	Improvement Proposals connected to the Tool’s requirements. . . . .	91
8.4	The PDD-SDD Tool’s architecture design. . . . .	95
8.5	A preview of the Behaviortrees application’s standard version. . . . .	96
8.6	First prototype’s definition view. . . . .	98
8.7	Second prototype’s definition view. . . . .	101
8.8	Adding a new Phase block, pop-up for data entry. . . . .	102
8.9	Fetch button appears after submitting the defined process. . . . .	104
8.10	The toggle of roles. Process Definer’s view activated. . . . .	106
8.11	Projects listing view with the Definer’s view activated. . . . .	106
8.12	The guiding questions for process definition . . . . .	107
8.13	Data collection and exceptions listing. . . . .	107
8.14	The Definition view. . . . .	108
8.15	The Step-level view. . . . .	108
8.16	The Details-level view next to the Step-level view. . . . .	109
8.17	The toggle of roles. Developer’s view activated. . . . .	110
8.18	The activated technical view’s button in project listing. . . . .	110
8.19	The flow of the defined process. . . . .	112
8.20	The updated definition view based on the updated code file. . . . .	114
8.21	The updated technical view based on the updated code file. . . . .	114

# List of Tables

2.1	RPA software market share by revenue 2017-2018, worldwide (millions of dollars) [4]. . . . .	9
4.1	Development processes and the main actors, marked by x, in RPA development at MOST Digital. Sometimes a process requires an additional actor, marked by (x). . . . .	40
4.2	Identified potential Challenges in RPA development grouped by development aspects. . . . .	45
4.3	Requirements for the Tool. Derived Requirements aim to solve identified root Challenges, which are linked to the sub Challenges. . . . .	47
4.4	The Research Questions to be investigated via the tool developed during this thesis. . . . .	49
7.1	Challenges gathered from the first interviews . . . . .	68
7.2	Improvement Proposals gathered from the first interviews . . . . .	71
7.3	Improvement Proposals gathered from the second interviews . . . . .	77
7.4	Improvement Proposals gathered from the final interviews . . . . .	80

# Listings

3.1	Python main logic. . . . .	24
3.2	Reading data from the excel file with Python. . . . .	27
3.3	Fetching stock symbol data from nasdaq.com with Python. . . . .	28
3.4	Saving enriched data to the excel file with Python. . . . .	32
3.5	Imports needed in Python example. . . . .	35
3.6	Configuration needed in Python example. . . . .	37
8.1	The generated code file of the first prototype. . . . .	99
8.2	The generated code file of the second prototype. . . . .	103
8.3	The generated code file of the StandAlone version. Phase 1 and Step 1. . .	111
8.4	The main function of the updated code file of the defined process. . . . .	113

# Acronyms

**RPA** - Robotic Process Automation

**PDD** - Process Design Documentation

**SDD** - Solution Design Documentation

**IPR** - Intellectual Property Rights

**MRT** - Media Richness Theory

**VUI** - Visual User Interface

**MVP** - Minimum Viable Product

# 1 Introduction

Robots are here to stay. The digital landscape of the 21st century is mostly shaped by systems, web-applications, data and people. What ties these themes together is the ever-increasing load of processes and data management posed on human knowledge-workers, and that is exactly where automation, either by robotics or integration, comes into play.

Automation by robots or Robotic Process Automation (RPA) has been around from the early 2010s and has gained much attention after the release of the major RPA platforms. This emergence and the market demand got Ailea Ltd and Most Digital Ltd to work in a companionship to satisfy the needs of Finnish companies.

Throughout the years working with RPA, Abdulghani and Vuorela have seen the bottlenecks and the misunderstandings happen during automation projects. Due to the projects' short cycled nature, volume gave deeper insight into the field. Although RPA development is in many areas comparable to the modern software development, there are still some characteristics that cause challenges to the whole project life-cycle. With this experience arsenal Abdulghani and Vuorela set out to revamp the ways of RPA projects.

To provide value, increase project success-rate and decrease the maintenance needs, a tool was designed. The upstream challenges were identified and a hypothesis was formed. The hypothesis was: If project documentation could be alleviated by automating parts of it,

and if communication related misunderstandings could be mitigated by facilitating clearer communication using a Tool, most of the identified bottlenecks could be diminished.

## **1.1 Divided Masters Thesis**

The hypothesis of this thesis, Tool's development and the introduction to the world of Robotic Process Automation is a broad academic work, thus the project was divided into two separate theses.

The division of the theses clarifies the whole and allows both researchers to gain better insight into the specifications and the demands of the research area. Sampsa Vuorela focuses on the developer team's viewpoint in his thesis, and Ahmed Abdulghani focuses on the definer and the maintenance teams' viewpoints.

Both theses had the same background material which were written together, Chapters 2-5. Also, the methodology of interviewing and gathering the data was similar in both theses. And lastly, the Tool that was designed and implemented, was also accomplished together.

## **1.2 The Structure**

The thesis outlines the reason why a Tool was designed, how the data and feedback was gathered, the method of validation, and how the actual Tool was developed. The thesis begins by giving the reader a clear image of the current state of RPA in Chapters 2-5. Chapter 6 is where the project is split into two theses.

The roles beyond the shared thesis was based on the researchers' interests. For this thesis the scope was to understand how professionals with business backgrounds work in the

RPA project life-cycle, as well as understanding the maintenance team's bottlenecks and improvement needs.

The structure of the joint background Chapters (Chapters 2-5) was designed to give the reader a clear understanding of the Robotic Process Automation world. First, a general introduction to the topic is given with references to academic work and work experience. Second, the joint chapters dive into the RPA's development process with examples. Third, a general understanding of the project life-cycle at Most Digital Ltd. is given and the current challenges are discussed. Lastly, the Improvement Proposal, the Tool is introduced as a solution for the identified challenges, also the work distribution regarding the roles and the Research Questions are raised in the final joint chapter.

Chapters 4 and 5 are at the core of this thesis, which help understand what the goal is and how it is set to be achieved. The purpose of Chapters 2 and 3 is to help the reader understand the challenge at hand and also to support the core Chapters 4 and 5.

### **1.3 The Relational Setup**

Varying thesis topics were initially considered with MOST Digital Ltd before this specific topic was discussed and approved. The reason for choosing this topic is that it has functional value compared to theoretical analysis of how a new technology could benefit MOST Digital. This research topic and the Tool developed would help with the daily challenges definers, developers and maintainers face.

A written contract was signed between Abdulghani, Vuorela and MOST Digital Ltd. regarding the Intellectual Property Rights (IPR) and the usage of the Tool in its final stage.

## 1.4 The Central Question

The central question of this thesis is: Could a Tool help with the challenges business people and the maintenance team face in their daily work with RPA projects?

Based on the work experience in the RPA field, Abdulghani and Vuorela wrote the initial requirements needed to satisfy the initial challenges identified. And from these, three Research Questions were formed (Table 4.4). This thesis focuses on and answers the Research Questions Q1 and Q3, which are:

Q1 - Does the tool ease the definition, project kick-off and maintenance of RPA projects?

Q3 - Does the tool ease communication and documentation between stakeholders in RPA projects?

Chapter 2 presents an overview of the Robotic Process Automation, which gives a basic understanding of RPA by providing theoretical knowledge and also visual examples. The next chapter is the start of the background study that was conducted by Abdulghani and Vuorela, which encompasses four chapters in total.



## **2 Overview of Robotic Process**

### **Automation**

The time of robots and automation is upon us, but why now? Automation in general has been present for a long time, but what is the big fuss about it nowadays and why is it getting more attention now than ever before? To understand this, we must dive into Robotic Process Automation (RPA) first. Robotic process automation is a piece of software that executes processes in a computer's operating system, acting in many ways like a human would do.

This chapter provides an overview to RPA based on scientific publications, companies' white-papers, articles and the work done at MOST Digital [3]. The chapter is divided into four categories that examine RPA at a general level, discuss its potential and limitations, and provide current future views. More advanced examples of RPA software, work methodologies at MOST Digital and the goal of this thesis are introduced in Chapters 3, 4 and 5.

#### **2.1 Robotic Process Automation**

Robotic Process Automation (RPA) as a term is used to describe tools and processes that operate the user interface of other computer systems in a way a human would do [5]. RPA is especially useful in situations where redesigning and creating integration between

systems is considered to be too difficult or costly. RPA gives organizations the ability to automate on the frontend, for example to gather data from applications via user interface. This kind of 'outside-in' approach has the advantage of keeping existing information systems intact. This is one of the main reasons why RPA is currently seen as a way to quickly achieve high Return of Investment (RoI) and that is what the companies that provide Robots as a Service (RaaS) are aiming for when delivering robots to their clients [5, 6, 7] .

Software robots can help us in many different ways. They can completely automate time-consuming and tedious processes performed by humans, or provide help to an employee during the process. RPA software has often been referred to as 'macros on steroids' because it can replicate computer-based tasks as many times as necessary and at a very rapid rate [7]. One thriving idea behind RPA is to supplement human work in a meaningful way so that companies benefit from both automation and more value-added work from their employees.

RPA combines different automation techniques, including screen scraping software, workflow automation and management tools, and artificial intelligence [8]. Software robots can navigate different digital landscapes and also make use of web scraping tools and other scripting tasks [5, 9, 7]. Nowadays, there are different tools to develop RPA software. Some RPA behemoths like UiPath<sup>1</sup> and Blue Prism<sup>2</sup> offer development platforms that focus on visual presentation of the software logic [10, 11]. Visual programming could be easier to work with, especially for people who do not want or know how to program. Along with platforms focusing on visual workflows, RPA is developed with many programming languages which have numerous technologies and libraries for software au-

---

<sup>1</sup><https://www.uipath.com/>, 9.5.2020

<sup>2</sup><https://www.blueprism.com/>, 9.5.2020

tomation.

RPA platforms are built to ease automation development, especially for people who might not have a programming background. The easiest way to familiarize them with the platforms is by using tools and programs that they are already familiar with. Most of these platforms offer a visual interface in which the developer can drag and drop activities or functions right into the robot's logic, in a Microsoft Visio [12] like fashion. The platform itself generates code automatically as users drag and drop or link activities and icons together on the visual platform. This feature enables the process and subject matter experts to automate their tasks after a few weeks of training. Developing automation without these platforms requires significantly more knowledge in software programming in order to achieve similar automation [6].

At MOST Digital, a generic RPA task is developed in a close collaboration with the customer who is familiar with the process to be automated [3]. This is because a well documented and communicated process is paramount to developing a robust robot. Once the developer is familiar with the process and the systems used in it, the programming can be started. Although, the robot is programmed to do the process in the same way as a human would do, it is sometimes possible and encouraged to optimize and streamline the process, for example calculations can be done without opening additional software. The end result is a software robot 'living' in virtual machine in the cloud or in customer's internal network, working autonomously or collaborating with human workers.

Developers at Most Digital have mainly been using Python<sup>3</sup> and its open source libraries<sup>4</sup>. These allow often a very flexible approach to automation task at hand, and without extra costs from licenses that are often needed with platforms like UiPath and Blue Prism. RPA software programmed with Python can include similar techniques and solutions imple-

mented in these RPA Platforms, such as visual recognition, web scraping and operating system specific scripts to accomplish their goals. Visual recognition is not always the optimal way, but still often needed when the target software can only be accessed through remote desktop protocol (RDP) or programs like Citrix [13]. The robot is able to use scripts saved in the customer's environment and transfer files between machines. These files may be processed to get the necessary data for the automation's next step, or they could be reported directly to the customer, for example via email. Web scraping and browser automation tools are often used when customer's program can be accessed from the robot's own network. Occasionally, software robots may combine these tools, fetching some information via RDP and then forwarding it to another system directly from the robot's environment, or vice versa.

## 2.2 Potential

RPA is set to be the biggest game-changer for organizations from 2016 and beyond as it has now reached a point where it is mature enough to be easily and cheaply adopted [14]. It is non-invasive and can automate processes that utilize all of companies' line of business systems. These could be old legacy systems [15] or newer applications running in the cloud and desktop environments [7, 14].

As business process management (BPM) [16] has a legacy of long implementations and unclear business cases, RPA aims to achieve quick wins with little investment [5]. Users have gained multiple benefits when humans and robots have started to work together, including cost saving and faster processing with higher quality. Errors have decreased while regulatory compliance and output have improved. Also growth, productivity and satisfaction for both customers and employees have improved. Humans can now focus on the meaningful tasks that require innovation, creation and human interaction while robots

handle repetitive and uninspiring tasks.

The 2019 market leader was UiPath according to Gartner's report from June 2019, illustrated in Table 2.1, as UiPath has captured 13,6% of the whole RPA market segment [4]. This is mostly due to their strategy of sharing the platform for free for community use, and by providing free RPA courses and certifications for interested developers on their RPA academy. By offering the knowledge for free, UiPath has gained a lot of interest within the RPA community.

2017 Rank	2018 Rank	Company	2017 Revenue	2018 Revenue	2017-2018 Growth (%)	2018 Market Share (%)
5	1	UiPath	15.7	114.8	629.5	13.6
1	2	Automation Anywhere	74.0	108.4	46.5	12.8
3	3	Blue Prism	34.6	71.0	105.0	8.4
2	4	NICE	36.0	61.5	70.6	7.3
4	5	Pegasystems	28.9	41.0	41.9	4.8
8	6	Kofax	10.4	37.0	256.6	4.4
11	7	NTT-AT	4.9	28.5	480.9	3.4
6	8	EdgeVerve Systems	15.7	20.5	30.1	2.4
7	9	OpenConnect	15.2	16.0	5.3	1.9
9	10	HelpSystems	10.2	13.7	34.3	1.6
		Others	273.0	333.8	22.2	39.4
		<b>Total</b>	<b>518.8</b>	<b>846.2</b>	<b>63.1</b>	<b>100.0</b>

Due to rounding, numbers may not add up precisely to the totals shown

Table 2.1: RPA software market share by revenue 2017-2018, worldwide (millions of dollars) [4].

The reason these RPA platform companies are gaining much traction is because of their

marketing, and especially by marketing the fact that you do not have to be a programmer to create robots that automate tedious front-office tasks [17]. This marketing strategy was intentionally aimed at desk workers who might not have any programming skills but have a lot of repetitive tasks, thus making them a great target.

## 2.3 Limitations

While users can achieve many improvements with RPA, there are still areas where robots do not excel. Research shows that they still must be guided by human intelligence. Limits and possibilities for software robots are set by human will and imagination which are yet far too complex for artificial intelligence to replicate [14].

If a customer's process contains many unknown factors and a human worker has to continuously process things differently, the task might not be suitable for automation. Usually these kinds of processes do not have clearly defined input data or they include frequently misbehaving or changing programs, which makes a robust automation much harder to accomplish. However, there might still be a chance for improvement by further optimizing, defining and analyzing customer's processes. This way the problematic issues can be recognized and defined more clearly. The result might be an automated sub process of the original goal which still improves the total process when compared to the initial situation.

As with software development and project management, the project triangle [18] holds true in the world of robots. The quality of the developed robots are tightly linked to allocated time and financial resources. If these are insufficient, it may lead to a robot solution that demands more maintenance later on, thus creating technical debt. This is because the tight time frame imposes restrictions on defining sufficient error handling cases, which

help the robot solution to cope with predefined error scenarios. Also, the robots are accessing software that might get updates, which may lead to runtime failures.

Robots can also make incorrect decisions when applied to human interface because of contextual changes. Some errors could be hard to notice and lead to disastrous situations when discovered. Also ethical and security issues arise when software robots impersonate people.

## **2.4 Future**

Technologies that involve learning on the part of the computer are expected to have great potential for the future. When these are combined with RPA solutions the result could be an automated process which could apply a whole new level of learning, judgment and creativity to their work.

Continuously evolving technologies like cloud computing [19] and Internet of Things [20] create unprecedented explosion of data. As workloads keep rising, RPA software will increase productivity and help organisations to cope. At the same time it will be evolving towards truly cognitive automation where software will be able to improve its own performance and make complex decisions.

## **3 RPA Development**

This chapter provides more detailed examples and discusses common characteristics of RPA development. Developing RPA is similar to common software development projects, but emphasizing the value of communication with the customer, who has an important role in the RPA life-cycle. This is particularly true after the initial process documentation has been made, as undocumented exceptions may arise during the development phase.

The main aspects of RPA development are divided and discussed related to academic resources and the concrete work that is done at MOST Digital [3]. A comprehensive example of a simple RPA process is provided at the end of this chapter.

### **3.1 Definitive Characteristics**

The most important aspects throughout the agile RPA development are undoubtedly communication skills and constant communication. Any problematic encounters with the customer's system must be dealt with swiftly by contacting the customer and coming up with solutions together, because they are the owners of the process and usually have the best knowledge of the system.

The aforementioned aspects have great impact on the process definition, thus it is imperative that the participants speak the same language not only on a technical but also on a business level. In order for the project to succeed, communication must be active, as the



process definition might change during the development phase.

The following subsections cater a detailed understanding of the RPA characteristics. These contain process definition, communication, documentation, development, testing and maintenance.

### **3.1.1 Process Definition**

Developing automation relies on business process knowledge, which means that a process must exist before it can be automated. Some process optimization might occur before diving into the development part. The starting point of any RPA project is the process definition phase, which usually happens by co-operating with the customer. Initially, the definition process is about transferring the process knowledge, and the way the customer uses their systems, to the developer.

The first time any RPA process is defined, discussions with the customer are initiated to find potential processes to automate. Oftentimes, the process is defined in iterations, as information about the process becomes clearer later on, or the developer might have more technical questions about the initial process documentation, which leads to another definition iteration.

The definition process should be documented in various ways for later use, and it also helps to keep the defined process in scope. Definition documents should enable the use of images, shapes and written text to convey the defined process step by step to the developer. Additionally, a video is advised to be captured of the definition session, for later inspection, as some of the minor details might not find themselves to the process definition document (PDD) [3].

Defined processes are rarely similar, yet it is important to standardize the definition process in order to generate congruent process definition documents. This ensures that the most critical process information is gathered the same way from every process. Challenges may rise if the customer prefers to document the process on their own, without utilizing the standardized approach. Oftentimes, in these cases the customer generated documents lack the essential information for the developer, which in turn demand redefinition iterations. These result in unnecessary costs in both time and resources. [21]

### **3.1.2 Communication**

One major cause of problems in software development is a lack of clarity, not a lack of data [22]. RPA projects tend to be agile, fast paced and rely heavily on customer's instructions about the automated process. Usually in software development, the customer defines the desired final product. In RPA development, the customer also instructs how the process should be done. This does not mean that the developer cannot optimize the process, but the changes made to a customer's original instructions must be based on a clear understanding of how the customer's system is used and what the desired end result should be.

This special situation for many RPA projects means that the importance of efficient communication becomes one of the key factors for successful projects [1]. People perceive and understand things differently, for example customers may give instructions they consider to be obvious but contain a presumption that the developers know the system which is to be automated. Likewise, customers might get confused when developers explain things in too technical manner, with the presumption that the customer knows programming and its vocabulary.

Developers have to communicate with each other and with the customer to form a correct

idea of the project, therefore the information flow in software projects has a direct impact on productivity and quality [23]. There are two major forces in software development communication, uncertainty and equivocality, which may occur if the given information is ambiguous and confusing [22]. Especially troublesome issues may arise in projects, where communication's importance is not understood. Project managers will not be able to estimate realistic costs for communication and cannot recognize and solve communication problems when they occur. Communication cannot be improved on the basis of experience if the causes of its problems cannot be identified.

To ensure effective communication, the possibility of misunderstandings, cross-interpretations and lack of information should be minimized [1]. Especially in Agile environments, communication processes should be continuously updated and optimized to the project at hand. In addition, synchronized working hours, unified communication tools and official process verification policies can have a positive impact on the quality of communication [24]. Last but not least, documenting should be done at different levels of formality.

The Media Richness Theory (MRT, Figure 3.1) is a good guide to help organize communication methods according to their media richness and it can be used when choosing the best method for different situations [1]. The theory suggests the usage of high communication media, such as face-to-face conversation, in situations where things tend to be ambiguous and low communication media, such as emails, when it comes to reaching a long-term common understanding. Notes should also be taken from meetings which use high communication media to document decisions made.



should be informed with updates and modifications to the original business logic of the process, while developers and maintainers also need code-level documentation.

Since documentation has a crucial role in RPA development, an extra care should be paid when processes are defined and business logic is documented for the first time, prior to development. RPA development contains a lot of visual elements, which can form the base for high-level documentation. This kind of documentation is also more valuable for the customer, because it is more suitable for possibly non-technical people.

Usually high-level RPA documentation should model manual processes previously done by the customer's employees. A process should be documented by the customer in a way that the developers understand it. This is not always an easy task if the customer's employees have a lot of tacit knowledge about the usage of the tools to be automated. This kind of information is not always directly provided, and it needs to be purposefully searched during process definition. Additionally, documentation should be updated and maintained by the developers in a way that customers and maintainers understand it. High-level documentation should contain updates to the original business logic while technical documentation should clarify code-level decisions and structure.

### **3.1.4 Development**

This section is based on authors' work experience at MOST Digital [3].

The development processes can differ depending on the developer and the project. Some recurring aspects can still be recognized that may cause challenges during development processes. Often the cause for delays and misunderstanding results from aspects outside the actual development, from unclear definitions and insufficient initial documentation. The main reason for this is poor communication between the developer, the project's de-

finer and the customer. Therefore, the value of communication and documentation as a part of good development practices should be properly endorsed throughout the company.

At the beginning of project development, the developer studies the initial documentation in order to know what should be developed. The clearer and more precise the initial documentation, the easier it is for the developer to start working with the new project. Delays occur if the developer needs to clarify things he or she did not understand. Another cause for delays and extra costs could be misunderstood documentation caused by tacit knowledge that is not documented clearly.

From the company's point of view the developers should follow good development conventions. Best practices should be regularly instructed, discussed and reported to all developers so that everybody are at the same line. In order to accomplish this, the company's internal communication should work properly.

Changes made to the initial definition during the project development should be informed to both the project manager and the customer. Possible misunderstandings occur more likely at any stage of the development process, if the process definition is not actively updated. Both the developer and the definer should also ensure that outdated versions of the process definition are not left available to cause possible confusion in the future.

Oftentimes, the need for changes occur in older projects due to updates in software, environment or definition. In these situations the developer must study the old documentation and code base in order to become familiar with the project. This step is essential before any changes, and could cause errors in the developed code if not done properly. This kind of further development benefits tremendously from an up-to-date documentation of both high-level business logic and technical level resulting in faster development and less

errors.

### 3.1.5 Testability

Testing is as important in RPA development as in other software development projects. RPA processes often contain visual elements and follow predetermined execution paths, therefore being particularly suitable for black box testing methods [27], where outputs are validated against process specifications without inspecting internal code-level structure. In this section we combine our work experience [3] with testing related studies [6, 27, 28, 29, 2, 30] to discuss testing in RPA development.

Testing helps to find the areas where the program does not meet its specifications and it is therefore a necessary area for software validation. Testing can be done in many different ways and to all kinds of activities, therefore any action that helps to find a violation of specification can be called testing. Usually, software testing includes three activities: testing, debugging and verification. The first can be further separated to static testing, testing during development and testing by execution. Figure 3.2 illustrates the usual phases and testing activities in general software development flow, also present in RPA development. Phases from system requirements to coding can be inspected and tested with static testing methods, which can involve design reviews, code inspections and static analysis of source code.

Verification is needed in design and coding activities to ensure that the program satisfies the original specifications. Verifying can be done by modeling the program in a formal language or using the program itself. Formal specifications derived from system requirements help to verify programs. Specifications can be tested on usage level and functional level to show that the code implements the specifications.

Debugging is usually done during the coding process, later stages of testing and in production or deployment. A bug means a defect, a part of code logic that fails to meet a specification. Debugging is performed to locate and fix faulty code, usually by analyzing and modifying the program's logic. The goal is to diagnose and the precise nature of a known error and correct it. Especially during later stages of development, the cause of bugs may rather come from incorrect specifications than faulty code logic. Locating errors can be a hard process involving double checking known inputs and code logic. Function level test can be done with unit tests to recognize function-level defects. Good coding and logging practices can help tremendously by reducing the need to make extra prints and logging while debugging.

Program usage can also be tested by execution, usually in the last activities: testing, production or development. This could be done with white or black box testing. White box testing inspects and validates the internal inputs and outputs of the program logic, while black box methods use a program without code level inspection [27].

Testing can be a difficult process to be done thoroughly, as Dijkstra points out: "Program testing can be used to show the presence of bugs, but never to show their absence" [30]. This means that any amount of testing represents only a small sample of all possible computations. Test planning techniques based on partitioning of the functionality, data, end-user operation, et cetera, are very useful and popular and they form the base for many testing technologies.

Testing in RPA development follows the same patterns and practices as in general software development with some emphasised aspects. RPA testing relies heavily on output verification because it is usually performed on the terms of the customer's systems.



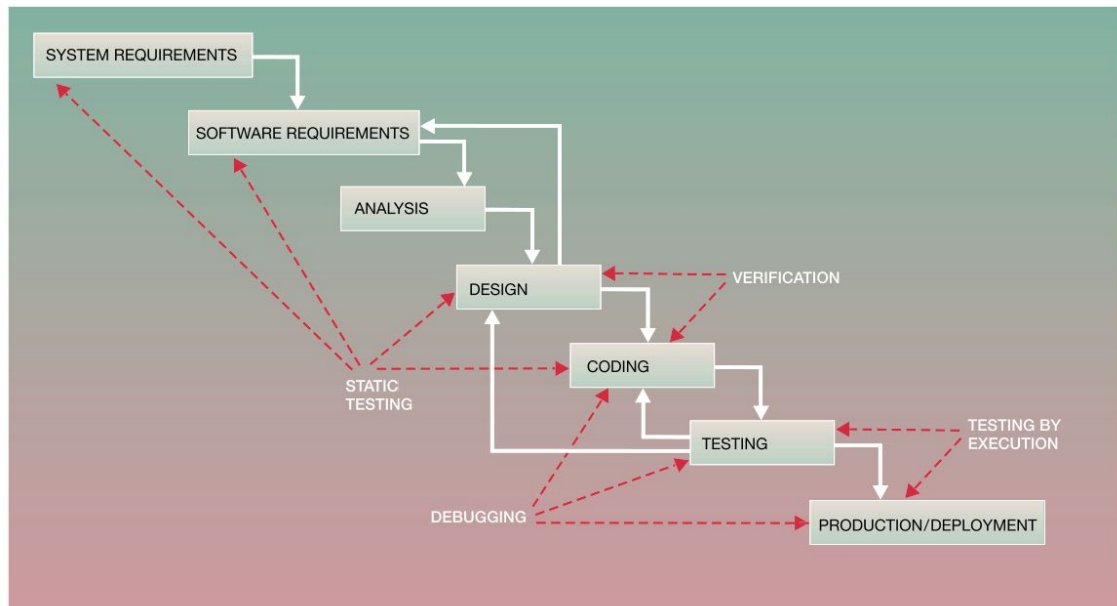


Figure 3.2: The activities in a typical software development process [2].

Due to restrictions in customer's systems or resources, test environments are not always granted, and the systems to be automated may be flawed. Testing should be done in a way that production data is not submitted forward, which means that testing needs should be taken into account from the very beginning, when defining the RPA process. The better the definition, the better the implementation. Good PDD allows for better and smarter implementation that can be programmed to recover from possible errors and continue execution even in an unstable environments. If the PDD lacks essential information, it may lead to insufficient testing as well. Therefore, it is paramount to document all known errors and special situations in the definition phase.

### 3.1.6 Consistency and Maintenance

RPA software is often sensitive to visual changes and crashes may occur more often if the automated software is not stable. In these situations the employee who is responsible for maintaining the tasks has to check the situation and restart or potentially update the

software. To do this effectively, some level of understanding about the task logic and the circumstances that caused the error is needed.

Studies have shown that a notable amount of resources used in maintenance is spent in understanding the code [29]. More complex systems are harder to maintain and tend to become even more complicated as time goes on. RPA solutions can differ a lot from each other and completely similar logic between tasks is relatively rare. But when inspected closely, there are similar base functionalities and structures that could be same between different projects. This means that the basic components that are used in different RPA tasks can be unified and maintained more easily.

The maintenance of RPA tasks requires knowledge about the business case in addition to the actual logic, thus an updated Solution Design Document (SDD) is imperative. If RPA tasks follow the same predefined basic guidelines with their high level code structure and basic behaviour implementations as well as unified version control practices, maintenance becomes significantly easier.

## **3.2 RPA Examples**

Automation has already been around before the rise of the RPA platforms. These solutions and tools have been based on pure programmed code without visual representations. These include combinations of scripts and techniques capable of navigating through the visual user interface. Many of these techniques originate from UI automation testing, which are found suitable also for the needs of RPA.

Lately, RPA platforms have included the visual interface to RPA developing, making it

easier to use and develop for people without programming background. This has greatly accelerated RPA's fame. In order to understand the differences between tools that require actual programming skills and the platforms based on visual user interface, a simple example is provided with both techniques.

The programmed example is represented with Python which is highly capable in automation with its wide scope of libraries suitable for RPA, thus making it optimal for comparison. It is also used as the main development language at MOST Digital. The visually developed example is provided with UiPath, the market leader which is already discussed earlier in this thesis. The reason these tools are represented in this thesis is that both techniques are valid and can be used for RPA developing, selecting the correct one mostly depends on the needs and preferences of both the customer and the developer.

Data is the starting point in most automation projects. In other cases the robot starts the automated task by acquiring the needed data. In this example the robot reads control data from input excel file, which is represented in Figure 3.3 and used with both Python and UiPath. In this case, it is simple and ready to use, but oftentimes cleaning up and fixing the initial data is also part of RPA development. Data may not always be in a coherent state and ready to be utilized as it is.

The example process reads stock symbols from the input excel file, fetches latest stock price from [nasdaq.com](https://www.nasdaq.com/)<sup>1</sup> with Google Chrome<sup>2</sup> and updates the original excel file with the acquired data. Although, the example is rather simple, it aims to demonstrate differences between the selected tools. Also, these steps reflect the basic procedures involved in common RPA solutions. The example's main logic is represented with Python in Listing 3.1

---

<sup>1</sup><https://www.nasdaq.com/>, 9.5.2020

<sup>2</sup><https://cloud.google.com/chrome-enterprise>, 9.5.2020

while UiPath's main workflow is displayed in Figure 3.4.

Stock Symbol	Company	Latest Price
AAPL	Apple inc common stock	
MSFT	Microsoft Corporation Common Stock	
AMZN	Amazon.com, Inc. Common Stock	
GOOGL	Alphabet Inc. Class A Common Stock	
IDCBY	Industrial and Commercial Bank of China Ltd ADR	
FB	Facebook, Inc. Class A Common Stock	
BABA	Alibaba Group Holding Limited American Depositary Shares each representing eight Ordinary share	
BRK/A	Berkshire Hathaway Inc.	
VOD	Vodafone Group Plc American Depositary Shares	
TCEHY	Tencent Holdings Ltd. ADR	

Figure 3.3: The input excel file for provided RPA examples.

Listing 3.1: Python main logic.

```

1 def main():
2     # 1 - read stock symbols from excel
3     excel_df = read_excel_data()
4     # 2 - search latest values from nasdaq.com with Google
5         Chrome
6     updated_df = fetch_stock_values(excel_df)
7     # 3 - updated stock symbol values in excel
8     save_updated_data(updated_df)

```

### 3.2.1 Reading Data from Excel File

An excel file can be processed multiple ways using different Python libraries. This example uses Pandas, an open source data analysis and manipulation tool [31]. The code, presented in Listing 3.2, reads the excel file's content into a DataFrame object, which is updated in later stages of the process.

In Figure 3.5, reading an excel file in UiPath generates a DataTable and a workbook object, named in this example as WbSymbols. The Read Range activity used inside the Read Excel and Create a DataTable activity is using the workbook object to read the whole of

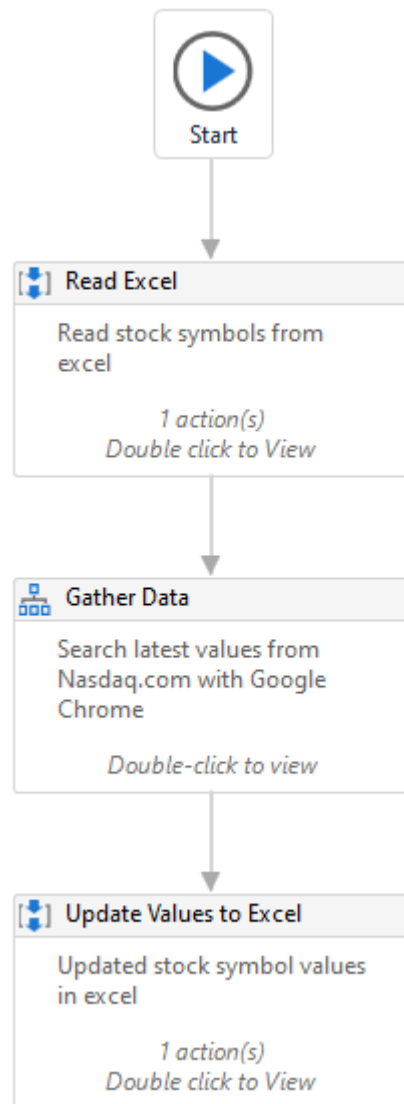


Figure 3.4: The main workflow in UiPath process.

the first sheet in the excel file.

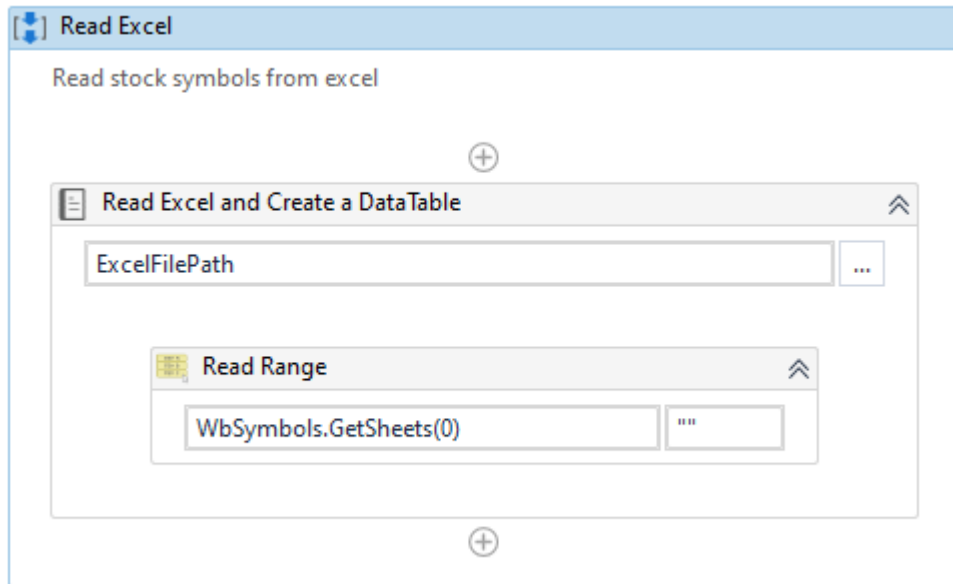


Figure 3.5: Reading data from the excel file with UiPath.

In both cases the code generated, either by coding with Python or by dragging and dropping activities in UiPath, has enabled the user to read the excel file's data into a structured object form — In Python's Pandas library it is called a DataFrame, and in UiPath it is called a DataTable.

Listing 3.2: Reading data from the excel file with Python.

```
1 def read_excel_data():
2     EXCEL_VARIABLES = config["EXCEL_VARIABLES"]
3
4     excel_df = pd.read_excel(
5         EXCEL_VARIABLES["BLANK_EXCEL_PATH"],
6         sheet_name = EXCEL_VARIABLES["SHEET_NAME"],
7         encoding = EXCEL_VARIABLES["ENCODING"],
8         dtype = object
9     )
10
11     print(f"excel_dataframe: \n {excel_df}")
12
13     return excel_df
```

### 3.2.2 Fetching Information from the Internet

The next step is to fetch the latest stock price from nasdaq.com. In this example, Python is used with Selenium library<sup>3</sup>, originally a framework for testing web applications, but widely used for automation and RPA processes as well [9]. Selenium is able to access Google Chrome via chromedriver.exe, a separate driver that needs to be available in order to get Selenium working.

Python logic in Listing 3.3 shows that querying the stock specific view in nasdaq.com is fairly simple. The correct url address can be constructed from the base url joined with the stock specific symbol at the end. The price can be extracted from the website's HTML structure. The parent element is fetched first and after it has emerged, the price can be read from its inner HTML structure. Finally the fetched price is saved to the symbol's row in the DataFrame object created in the previous step. This procedure is repeated for

<sup>3</sup><https://www.selenium.dev/>, 9.5.2020

every stock symbol found in the original input excel file.

Listing 3.3: Fetching stock symbol data from nasdaq.com with Python.

```
1 def fetch_stock_values ( excel_df ):
2
3     SELENIUM.VARIABLES = config [ "SELENIUM.VARIABLES" ]
4     EXCEL.VARIABLES = config [ "EXCEL.VARIABLES" ]
5
6     selenium_driver = webdriver.Chrome(
7         SELENIUM.VARIABLES [ "SELENIUM_DRIVER_PATH" ]
8     )
9     selenium_driver.implicitly_wait (20)
10
11     for stock_symbol in excel_df [
12         EXCEL.VARIABLES [ "COLUMNS" ] [ "SYMBOL_HEADER" ]
13     ]:
14         print ( f"fetching price for symbol: {stock_symbol}" )
15
16         nasdaq_url = "{}{}".format (
17             SELENIUM.VARIABLES [ 'NASDAQ_URL_BASE' ],
18             stock_symbol
19         )
20         print ( f"google_finance_url: {nasdaq_url}" )
21
22         selenium_driver.get ( nasdaq_url )
23
24         # getting the specific price element from nasdaq site
25         # by its HTML class name
26         price_element = selenium_driver.
            find_element_by_css_selector (
                SELENIUM.VARIABLES [ "
                    SYMBOL_PRICE_PARENT_ELEMENT_CSS" ]
```



```
27     ).find_element_by_class_name(
28         SELENIUM_VARIABLES["SYMBOL_PRICE_ELEMENT_CLASS"]
29     )
30     # value can be found inside element
31     symbol_price = price_element.get_attribute("innerHTML")
32     print(f"found symbol_price: {symbol_price}")
33
34     # updating dataframe read from excel
35     excel_df.loc[
36         (excel_df[
37             EXCEL_VARIABLES["COLUMNS"][ "SYMBOL_HEADER" ]
38             ] == stock_symbol
39         ),
40         EXCEL_VARIABLES["COLUMNS"][ "PRICE" ]
41         ] = symbol_price
42
43     print(f"updated excel_df: \n{excel_df}")
44
45     print("All stock prices fetched.")
46     return excel_df
```

The UiPath logic of searching the nasdaq.com website for the correct stock price is illustrated in the Figure 3.6. The first step of the logic is to read the stock symbol that has not yet been updated, as seen in Figure 3.7. In the referenced figure the datatable is looped and a couple of checks are asserted in order to get the needed data.

Next, the logic navigates to the Nasdaq website and retrieves the current stock price, by concatenating the Nasdaq website url and the stock symbol's name, as is used in the Python example. This is presented in Figure 3.8. The Get TStock Price Text activity reads the website's HTML structure and identifies the wanted text element.

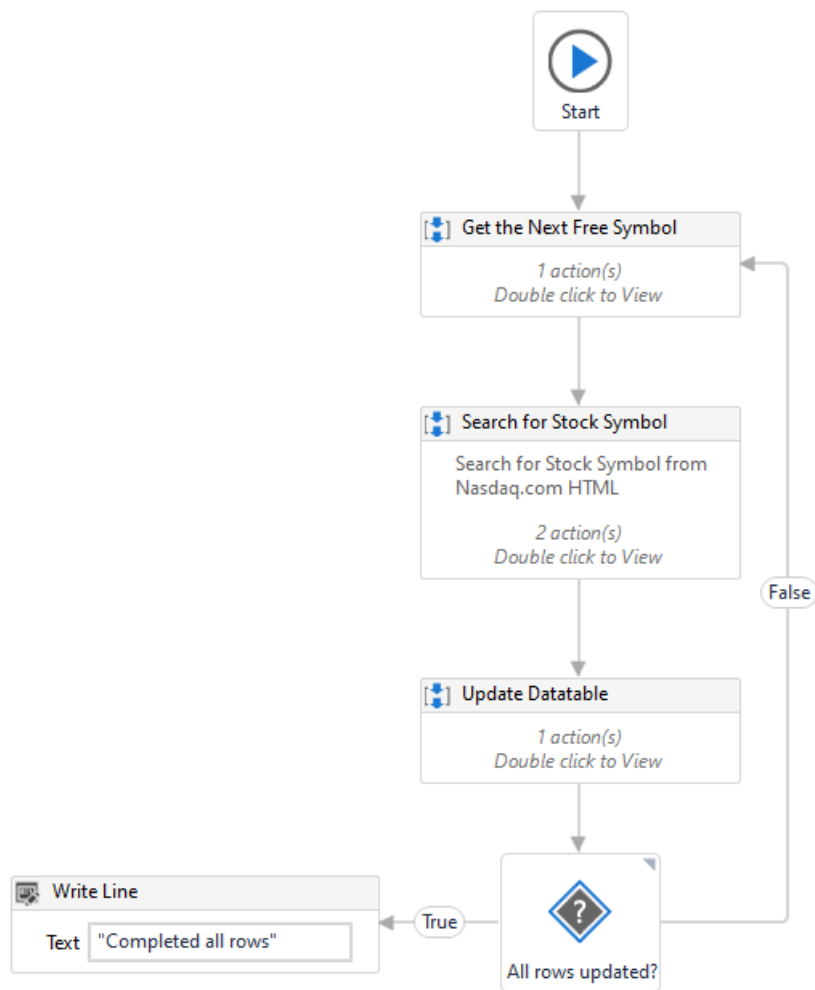


Figure 3.6: Fetching stock symbol data from nasdaq.com with UiPath.

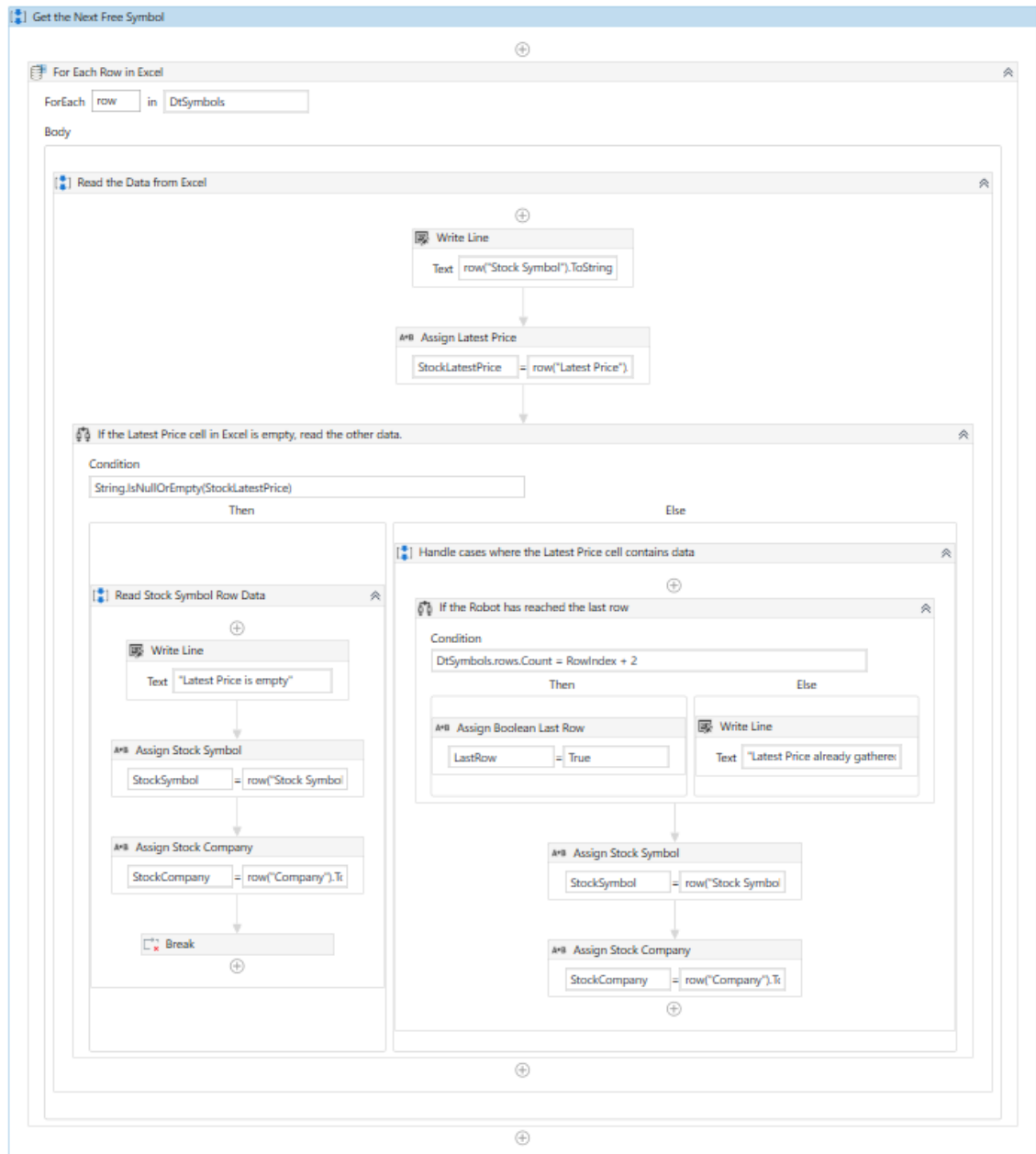


Figure 3.7: Looping the DataTable in UiPath.

Lastly, the logic updates the datatable with the gathered data to the stock symbol specific row. The logic is depicted in Figure 3.9. A flow decision is placed at the end of the logic to check if the robot has iterated all of the datatable rows. If not, the robot starts the "Gather Data" process again until all of the rows are updated, as represented in Figure 3.6.

### 3.2.3 Saving Enriched Data to the Excel File

The final step is to save updated stock prices to the output excel file. Python does this again with Pandas, in Listing 3.4. UiPath writes the updated DataTable to output excel file in Figure 3.10. The resulted output excel file for both examples is represented in Figure 3.11.

Listing 3.4: Saving enriched data to the excel file with Python.

```
1 def save_updated_data(excel_df):
2     EXCEL_SETTINGS = config["EXCEL_SETTINGS"]
3
4     excel_df.to_excel(
5         EXCEL_SETTINGS["RESULT_EXCEL_PATH"],
6         sheet_name = EXCEL_SETTINGS["SHEET_NAME"],
7         encoding = EXCEL_SETTINGS["ENCODING"],
8         index=False
9     )
10    print("excel saved")
```

### 3.2.4 Prerequisites and Settings for the Provided Examples

To be able to use Pandas, Selenium and other libraries with Python, they must be installed in Python environment and imported in the beginning of the script, as presented in Listing 3.5. UiPath requires only the UiPath Studio application as it contains all the necessary dependencies.

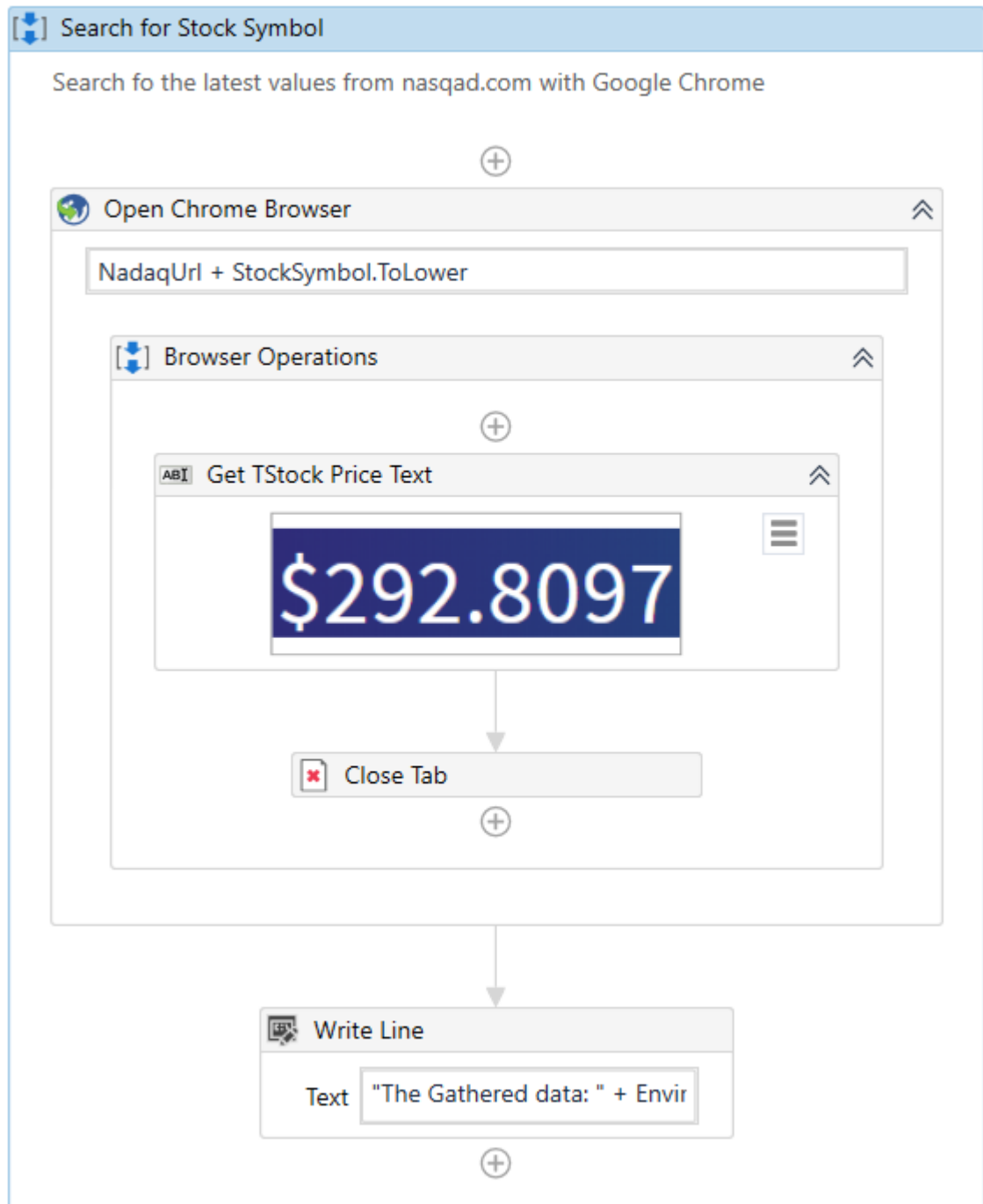


Figure 3.8: Searching for the stock symbol in UiPath.

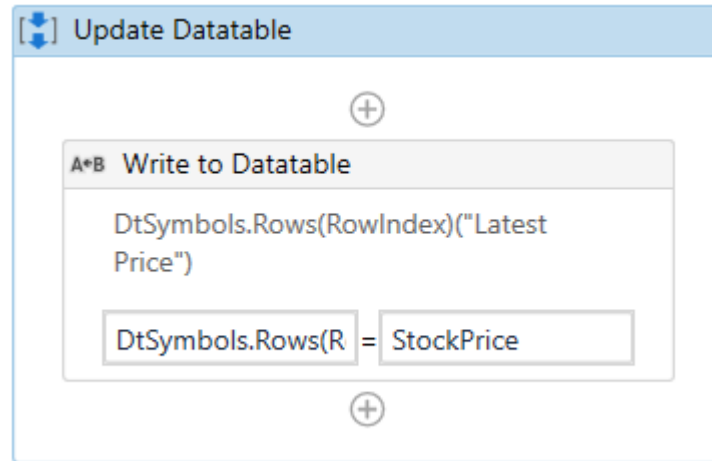


Figure 3.9: Updating the DataTable in UiPath.

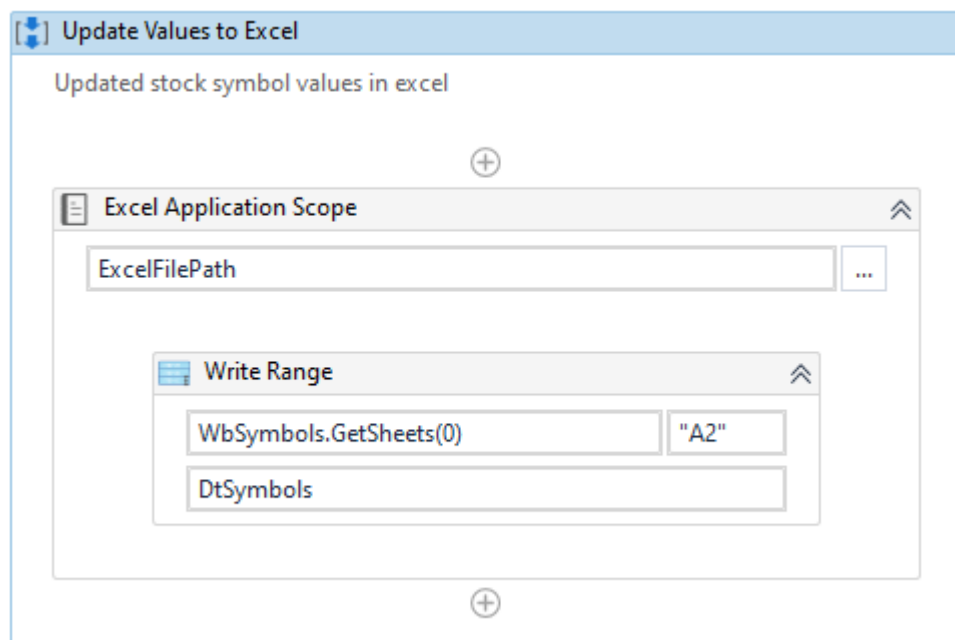


Figure 3.10: Saving enriched data to the excel file with UiPath.

Stock Symbol	Company	Latest Price
AAPL	Apple inc common stock	\$289.32
MSFT	Microsoft Corporation Common Stock	\$164.51
AMZN	Amazon.com, Inc. Common Stock	\$1908.99
GOOGL	Alphabet Inc. Class A Common Stock	\$1337.72
IDCBY	Industrial and Commercial Bank of China Ltd ADR	\$13.925
FB	Facebook, Inc. Class A Common Stock	\$185.89
BABA	Alibaba Group Holding Limited American Depository Shares each representing eight Ordinary share	\$207.41
BRK/A	Berkshire Hathaway Inc.	\$214748.3647
VOD	Vodafone Group Plc American Depository Shares	\$16.88
TCEHY	Tencent Holdings Ltd. ADR	\$50.19

Figure 3.11: The output excel file for provided RPA examples.

Listing 3.5: Imports needed in Python example.

```

1 import os
2 import pandas as pd
3 from selenium import webdriver
4 import openpyxl

```

Both techniques can use a separate configuration file. UiPath recommends using a separate excel file, whereas Python can read the configurations from various file types. The provided Python example declares settings in dictionary object as illustrated in Listing 3.6, which could also be read from a separate file.

### 3.2.5 Summary

Both Python and UiPath provide a powerful way to develop automation processes. Choosing the right tool depends on the user's development skills, as UiPath demands less technical knowledge compared to Python. An experienced developer used to programming may find Python more convenient compared to UiPath's drag and drop user interface. Similarly for more business oriented user, UiPath provides faster results without the need to write code.

Programming languages such as Python give the developer more freedom to use custom approaches for the project. This enables more flexibility, but also demands more technical skills and responsibility concerning error handling and security issues. Most of these are already handled with UiPath components, which run in the background. As a closed source application [32], UiPath hides the explicit code logic and links it to the drag and drop user interface with XAML structure, illustrated in Figure 3.12. The structure holds the visual formation instructions for the UiPath Studio and also, function calls with parameters. This way UiPath can provide more ready to use functions. However, it could limit the user's options for wanted, customized functionalities, as the user is unable to access and modify the executable code itself.



Listing 3.6: Configuration needed in Python example.

```
1 config = {
2     "EXCEL_SETTINGS":{
3         "BLANK_EXCEL_PATH":os.path.join(
4             "..","blank_stock_symbol_excel.xlsx"
5         ),
6         "RESULT_EXCEL_PATH":os.path.join(
7             "..","stock_symbol_excel_output_python.xlsx"
8         ),
9         "SHEET_NAME":"symbols",
10        "ENCODING":"utf-8",
11        "COLUMNS" : {
12            "SYMBOL_HEADER" : "Stock Symbol",
13            "DESCRIPTION" : "Company",
14            "PRICE" : "Latest Price"
15        }
16    },
17    "SELENIUM_SETTINGS":{
18        "SELENIUM_DRIVER_PATH":"chromedriver.exe",
19        "NASDAQ_URL_BASE":"https://www.nasdaq.com/market-
20            activity/stocks/",
21        "SYMBOL_PRICE_ELEMENT_CLASS":"symbol-page-
22            header__pricing-price"
23    }
24 }
```

```

Main.xaml
<Flowchart.StartNode>
  <x:Reference>__ReferenceID7</x:Reference>
</Flowchart.StartNode>
<FlowStep x:Name="__ReferenceID7">
  <sap:WorkflowViewStateService.ViewState>
    <scg:Dictionary x:TypeArguments="x:String, x:Object">
      <av:Point x:Key="ShapeLocation">200,127.5</av:Point>
      <av:Size x:Key="ShapeSize">200,112</av:Size>
      <av:PointCollection x:Key="ConnectorLocation">300,239.5 300,285.5</av:
      PointCollection>
    </scg:Dictionary>
  </sap:WorkflowViewStateService.ViewState>
  <Sequence sap2010:Annotation.AnnotationText="Read stock symbols from excel"
  DisplayName="Read Excel" sap:VirtualizedContainerService.HintSize="200,112"
  sap2010:WorkflowViewState.IdRef="Sequence_1">
    <sap:WorkflowViewStateService.ViewState>
      <scg:Dictionary x:TypeArguments="x:String, x:Object">
        <x:Boolean x:Key="IsExpanded">True</x:Boolean>
        <x:Boolean x:Key="IsAnnotationDocked">True</x:Boolean>
      </scg:Dictionary>
    </sap:WorkflowViewStateService.ViewState>
    <ui:ExcelApplicationScope Password="{x:Null}" AutoSave="False" CreateNewFile="
    False" DisplayName="Read Excel and Create a DataTable" sap:
    VirtualizedContainerService.HintSize="434,168" sap2010:WorkflowViewState.IdRef=
    "ExcelApplicationScope_1" Visible="False" Workbook="[WbSymbols]" WorkbookPath="
    [ExcelFilePath]">
      <ui:ExcelApplicationScope.Body>
        <ActivityAction x:TypeArguments="ui:WorkbookApplication">
          <ActivityAction.Argument>
            <DelegateInArgument x:TypeArguments="ui:WorkbookApplication" Name="
            ExcelWorkbookScope" />
          </ActivityAction.Argument>
          <ui:ExcelReadRange AddHeaders="True" DataTable="[DtSymbols]" DisplayName=
          "Read Range" sap:VirtualizedContainerService.HintSize="334,59" sap2010:
          WorkflowViewState.IdRef="ExcelReadRange_2" SheetName="[
          WbSymbols.GetSheets(0)]">
            <ui:ExcelReadRange.Range>
              <InArgument x:TypeArguments="x:String">
                <Literal x:TypeArguments="x:String" Value="" />
              </InArgument>
            </ui:ExcelReadRange.Range>
          </ui:ExcelReadRange>
        </ActivityAction>
      </ui:ExcelApplicationScope.Body>
    </ui:ExcelApplicationScope>
  </Sequence>
</FlowStep>
</Flowchart>

```

Figure 3.12: UiPath example's main XAML file, UiPath Studio Community Edition 2019.

# **4 Improving Development at MOST Digital**

The main motive for this thesis is to design and create a tool that produces more value to MOST Digital, making definition and documentation of RPA projects easier, which should lead to improved communication. These main aspects should lay foundation to ease development, testing and maintaining of RPA projects. At best, the use of the tool would standardize good practices and structures inside the company, which could streamline and accelerate the development phase.

This chapter introduces the life-cycle of RPA projects carried out at MOST Digital. Later, the usual Challenges of RPA development are identified from both MOST Digital's life-cycle and common aspects of RPA development discussed in Chapter 3. The last two sections use these identified Challenges to derive Requirements and to form Research Questions, from which the design for the wanted tool is created and introduced in Chapter 5.

## **4.1 Project Life Cycle**

General RPA project at MOST Digital has four different development phases containing several sub-phases that make up the whole process. As described in Table 4.1 the main phases include definition, development, testing and maintenance. The table is further

		Development Processes			
		Definition	Development	Testing	Maintenance
Actors	Customer	x	(x)	(x)	
	Definer	x	(x)	(x)	
	Developer	(x)	x	x	(x)
	Maintainer			(x)	x

Table 4.1: Development processes and the main actors, marked by x, in RPA development at MOST Digital. Sometimes a process requires an additional actor, marked by (x).

separated between four main actors consisting of one or more individuals, which are the customer, definer, developer and maintainer. The quality of communication and documentation between the actors throughout the development process is paramount in order to achieve the best possible outcome.

At MOST Digital everything starts with the customer. The definition phase consists of meetings between the management of the customer's side and the project managers on MOST Digital's side. This phase involves understanding the customer's process and their needs. Once a process is found to be a good match for automation, the definition of the task begins. In some occasions developers are also asked to provide technical input during the definition process, as seen in Table 4.1.

Once the process is defined and documented, a Process Design Document (PDD) is produced, which should contain all the necessary information about the process in order to start the technical development. Before the RPA developer can begin automating any process, the needed development environments must be set up and additional software installations and access rights must be granted.

Testing and developing RPA solutions go hand in hand. Oftentimes, developing RPA

solutions demand continuous process logic testing, as customer's systems may require data to be inputted in order to progress in the process' next step. More comprehensive testing is done at the later stages of development, where the process is validated with the customer. During development and testing, unexpected situations may emerge that were not mentioned in the initial PDD. For these situations the developer usually asks help from the project manager or directly from the customer, in order to solve the task at hand swiftly, as seen in development and testing columns in Table 4.1. The identified changes should be documented and the PDD updated. The solution is validated with the customer by arranging demonstration meetings, where the solution is inspected step by step.

Documentation of the project and the changes that took place during the project are essential for the customer, other developers and for the maintenance team. First of all, the technical documentation of the actual solution must be up to date. Second, a Solution Design Document (SDD) should be created by either updating the original PDD or creating a new and a final version from it, by adding all the changes made to the initial document. Finally, as sharing knowledge helps everyone in the long run, these documents must be shared internally with the support team and with other developers. Additionally, SDDs should be available to the customer.

The last phase of the life-cycle of a general RPA project is ensuring that the solution works consistently on its own, and for this reason MOST Digital offers its customers an alerted support for a fixed time period, assuring that the functional continuity is delivered to the customer. Support is done by the maintenance team, which monitors the solution's status and investigates possible errors, also working with the developers to ensure that the encountered errors are fixed properly. The maintenance team aims to solve problems first by themselves, with the help of documented error scenarios. If they are unable to solve the problem, the error is raised to the developer. This means that the maintenance

team is involved with testing and the developer may also take part of the maintenance process as seen in Table 4.1. When the solution's functionality is confirmed, maintaining is continued on a default level. The default level offers customers a continuous support based on the Service-Level Agreements (SLA) [33].

## 4.2 Identifying Challenges

In order to derive Requirements for the tool to be developed during this thesis, potential Challenges in RPA development are identified from project life cycle discussed in previous section 4.1 and aspects introduced in Chapter 3. Most of the Challenges (C1-C25) presented in Table 4.2 circle around communication and documentation, as they lay foundations to all other parts of RPA project's life-cycle.

From the very beginning of the definition phase, the quality of communication is essential. Most common issues relate to lack of common language, when vocabulary is either too technical or high level for all participants to understand (C4), resulting in vague instructions (C5). Initial challenges in definition phase are associated with missing detailed instructions (C1) and tacit knowledge (C2, C13). More severe consequences may arise if the defined process is not optimized for automation (C3).

Unrealistic customer expectations (C7) are commonly the result of insufficient or inadequate communication (C4), and as the project is usually quick paced and intensive, the communication must flow effortlessly throughout the development, from the developer to the project manager or to the customer. One major obstacle for communication quality is possible availability issues between stakeholders (C9). Without knowing the progression challenges (C6), the customer might think that the initial process definition holds place and development is on schedule (C8). This might be true in some projects, where the

process definition is correct throughout the process, but that is rarely the case.

The RPA process documentation should be comprehensive, detailed, precise and formal in order to provide the best known documentation of the process to be automated. Poorly defined PDD, which lacks detailed instructions (C10) may cause inefficient decision making for the developers and raises plethora of questions that end up wasting everyone's time. And, if possible, the PDD should also include known errors and the possible ways of handling them (C24).

Poor technical documentation is another issue RPA projects face (C11), as developers might not have time allocated (C16), or they might not have set the technical documentation to a high priority, which might be caused of overlapping projects. As projects are short and agile, the developers must take care of the technical documentation during and after development. It is vital for the maintenance team, as well as for the other developers to avoid misunderstandings (C18, C19). Additionally, the documents should be unified in both format and file location to ensure the ease of use and availability (C14, C15).

Undefined exceptions are what developers encounter during most RPA development projects, as it is hard to tackle every situation beforehand, thus communication is an integral part of the development. This situation leads to updating the process definition during development based on decisions made. Changes should be also documented in PDD, as it is essential to keep the process documentation up to date (C12). Undocumented changes greatly complicate both further development and maintenance, as well as result in an incomplete SDD (C17, C25). Furthermore, without applying a company wide coding and testing standards, more maintenance is expected (C20).

Immature IT infrastructure is one of the challenges RPA development faces (C21), al-

though the robot does not usually need large systems to work. Missing test environment and data cause additional work to the developer, as they are not able to freely test the system while connected to customer's production systems (C22, C23).

### **4.3 Deriving Requirements for Improvement Proposal**

The identified Challenges listed in Table 4.2 pave the path for the Requirements needed for developing the tool during this thesis. Challenges are spread over the different phases of RPA development and therefore are not solvable with a single solution. However, root and sub Challenges, which could be eased by a single tool, can be identified. The root Challenges are the tip of an iceberg which, when solved, could potentially ease the sub Challenges as well. A Challenge is considered to be a root Challenge when it can not be directly solved with an another Challenge. The derived Requirements (R1-R3) with their root and sub Challenges are listed in Table 4.3.

Seven root Challenges were identified from the definition, documentation and testing aspects while the Challenges in communication, development and maintenance are all considered to be sub Challenges. The first two root Challenges are missing detailed instructions and tacit knowledge during definition in C1 and C2. When the definition is done in high detail, it prevents unoptimized processes in C3, misunderstandings due to communication and documentation in C5, C18 and C19, lack of detailed and tacit knowledge in C10 and C13, poorly documented business logic exceptions in C24, as well as unclear customer expectations due to vague PDD in C7. Furthermore, solving C1 and C2 can prevent immature IT architecture in C21 because the architect can plan a better infrastructure when the process needs are documented thoroughly.

C14, C15 and C16 were identified as root Challenges in the documentation aspect. Solv-



<b>Aspect</b>	<b>Challenge</b>	<b>Description</b>
<b>Definition</b>	C1	Missing detailed instructions during definition
	C2	Missing tacit knowledge during definition
	C3	Failing to optimize the process during definition
<b>Communication</b>	C4	Using too technical or high vocabulary in communication
	C5	Instructions are too vague, leaving space for misunderstandings
	C6	Stakeholders not up to date of the current level of progress
	C7	Unclear customer expectations
	C8	Unclear timetable
	C9	Availability issues between the stakeholders
<b>Documentation</b>	C10	Lacking detailed instructions in PDD
	C11	Lacking detailed technical documentation
	C12	Decisions made during meetings are not properly documented
	C13	Tacit knowledge not documented
	C14	Documentation not in unified location
	C15	Documentation not in unified format
	C16	No time to update documentation
<b>Development</b>	C17	Outdated PDD
	C18	Misunderstandings due to communication
	C19	Misunderstandings due to documentation
	C20	Divided development conventions inside the company
	C21	Immature IT architecture
<b>Testing</b>	C22	Lack of test material
	C23	Lack of test environments
	C24	Poorly documented exception situations with automated systems
<b>Maintenance</b>	C25	Outdated SDD

Table 4.2: Identified potential Challenges in RPA development grouped by development aspects.

ing the issue of decentralized project documentation in C14 could assure that all the updated information is in the correct location. This directly helps with the stakeholders not being up to date with the progress in C6, as well as unclear customer expectations and timetables in C7 and C8. Also, when both C14 and divided documentation format in C15 are fixed, the misunderstandings due to communication and documentation in C18 and C19 are eased as well. Moreover, no time to update documentation in C16 is directly linked to the lack of detailed technical documentation in C11, and outdated PDD and SDD in C17 and C25.

Lack of test material and test environments in C22 and C23 are the last root Challenges, as they are not solvable by other challenges but should be discussed and planned at the early stages of the definition phase and are more related to the customer's ability to provide them. The rest of the Challenges C4, C9, C12 and C20 cannot be solved with a single tool as they are more linked to the company's or the customer's internal processes and best practices.

To ease the identified root Challenges, three Requirements were derived from them. The first Requirement (R1) is to provide guiding questions during the definition phase, in order to answer the most common pitfalls of RPA process' data gathering and testing needs. R1 aims to ensure that the process instructions and tacit knowledge are properly documented, solving C1 and C2, as well as discussing the need for test material and test environments, thus providing ease to C22 and C23.

The second Requirement (R2) states that accessible, unified and up to date documentation should be provided, solving C14. The third Requirement (R3) answers to the rest of the root Challenges with automatic documentation, which should ease C15 and diminish the issue of insufficient time resources in C16.

Requirement	Description	Root Challenges	Sub Challenges
R1	Providing guiding questions during the definition phase	C1, C2, C22, C23	C3, C5, C7, C10, C13, C18, C19, C21, C24
R2	Providing accessible, unified and up to date documentation	C14	C6, C7, C8, C18, C19
R3	Providing automatic documentation	C15, C16	C11, C17, C18, C19, C25

Table 4.3: Requirements for the Tool. Derived Requirements aim to solve identified root Challenges, which are linked to the sub Challenges.

## 4.4 Research Questions

To validate the value provided by the tool, the effects must be assessed against the Challenges listed in Table 4.2 after the Requirements in Table 4.3 are met. As previous sections have addressed thus far, communication and documentation are the key factors in the RPA life-cycle. The Figure 4.1 illustrates that every process and actor in RPA development listed in the Table 4.1, are leaning on the communication and the documentation. Documentation can be seen as a long term communication, an utility ensuring that tacit knowledge and made decisions would not be forgotten.

The validation is done through three main Research Questions (Q1-Q3) listed in the Table 4.4. The Research Questions are divided between the company's internal actors of the RPA process, containing the definer, developer and maintainer. The tool should provide value for all of them and their viewpoints. The first Research Question (Q1) focuses on the definition and maintenance phases, where PDD is initially defined or the finalized

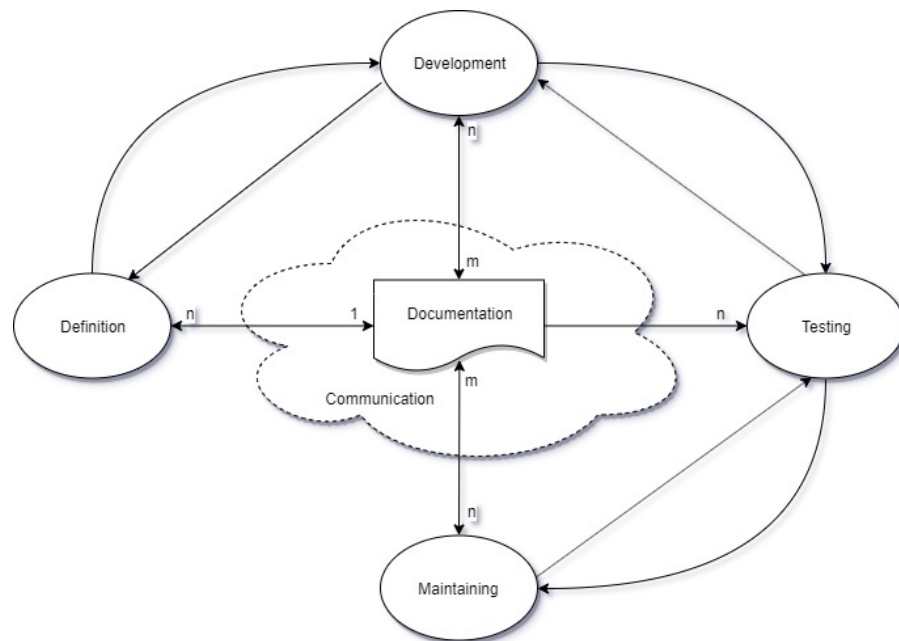


Figure 4.1: Intertwined aspects in RPA development life cycle [3]. All the processes circle around communication and documentation.

SDD is inspected. The second Research Question (Q2) investigates the value provided in a more technical level, when developing and testing the RPA process. Finally, the third question (Q3) addresses the overall communication and documentation between project actors.

<b>Research Question</b>	<b>Description</b>
Q1	Does the tool ease the definition, project kick-off and maintenance of RPA projects?
Q2	Does the tool ease the development and testing of RPA projects?
Q3	Does the tool ease communication and documentation between stakeholders in RPA projects?

Table 4.4: The Research Questions to be investigated via the tool developed during this thesis.

# 5 Improvement Proposal and Work Distribution

## 5.1 PDD-SDD Tool

In order to improve the RPA development aspects, and to address the Requirements listed in Table 4.3, a tool was designed, to automate the documentation flow for the whole project life-cycle, from process definition to maintenance. The tool would form the initial Process Design Document (PDD) and through the project advancement it would automatically be updated to reflect the code logic changes, finally representing the Solution Design Document (SDD).

PDD is first generated in Visual User Interface (VUI) and then processed to the initial code file for the developer. Automated PDD to SDD process is considered to minimize the human effort to maintain process level documentation which in turn supports communication. This should lower confusion and misunderstanding caused by outdated documentation, and address R3. In order to meet R1, guiding questions are added in the VUI's definition phase. After both R1 and R3 are accomplished, the tool needs to be validated and published inside MOST Digital, thus satisfying R2.

The initial design for the tool consists of five main steps (Step 1–Step 5), visualized in

Figure 5.1. In this thesis a standalone version of the tool is developed, ignoring cloud infrastructure, as the tool can be customized for the wanted deploying environment. The user authentication is also ignored to be potentially implemented during further development.

The initial PDD is created by the customer and the project manager at the very beginning of RPA development process. A high level process workflow is crafted in VUI, where parameters and additional information can be inserted and saved (Step 1). The base for code logic is generated automatically, after PDD is ready and submitted in VUI. Additional cloud infrastructure logic would also take place here, if implemented (Step 2).

The development begins after the definition is ready and the initial code base is generated (Step 3). The PDD-SDD Tool reads the code base and updates VUI during the development process, reflecting the latest changes (Step 4). Updating could be done by triggering it manually, or it could be bound to version control system's actions.

The VUI updates itself during the development phase, eventually transforming from the initial PDD towards a continuously more accurate SDD (Step 5). The customer and the project manager cannot make changes to the VUI after the initial definition phase. After that, the changes should be discussed directly with the developer. This feature prevents possibly problematic code logic changes which are not done by the developer.

The VUI should have different levels of views, depending on the user. The customer and the project manager should view the process from a high level, to confirm that the business logic is correct. Initial design dictates this level including three types of views, represented in Figure 5.2, which are divided depending on their information content and amount of detail. The VUI views are not presented all at once, rather they are accessed

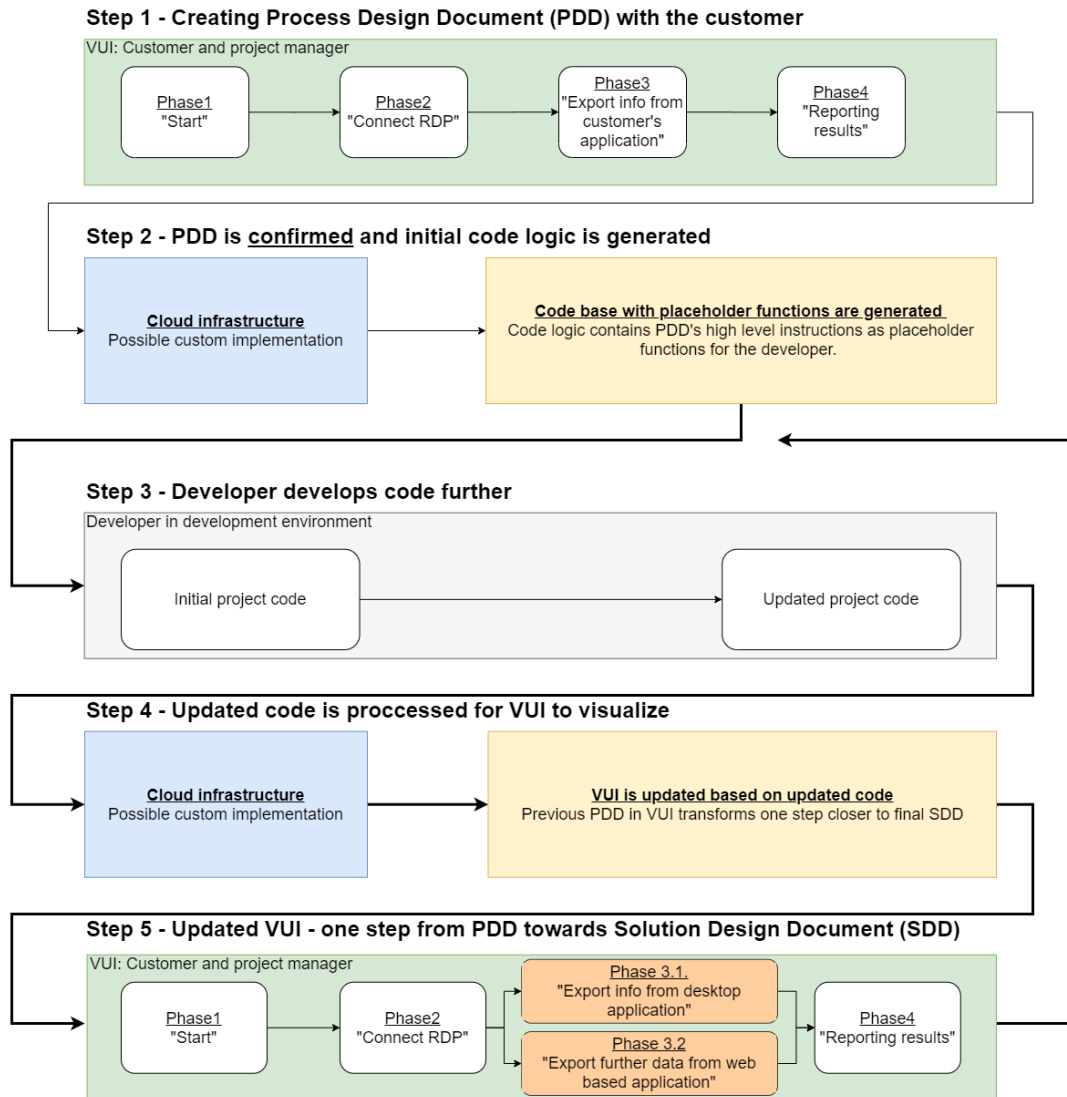


Figure 5.1: Initial design of PDD-SDD Tool.



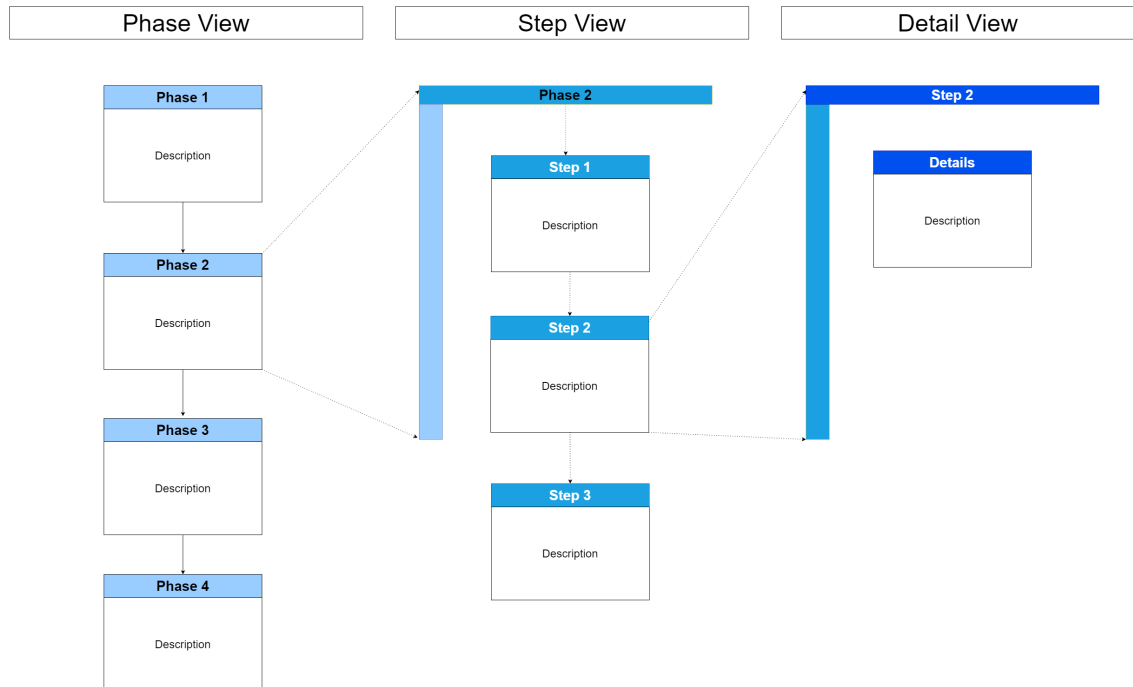


Figure 5.2: Initial rough design of VUI views. The VUI views are not presented all at once, rather they are accessed one level at a time from left to right.

one level at a time from left to right.

The Phase level is the first view aiming to define the big picture of the process. The process' high level Phases are separated to be further described in Step and Detail levels. Therefore, the Phase level should not contain more than headers and overall descriptions of the Phase elements. The Step level divides every Phase element into more detailed Steps. This level is meant to behave in a similar way as the Phase level, but in more detailed manner, focusing only on the defined Steps of one specific Phase. The accurate details and descriptions are added into the final view, the Detail level. The Detail level should contain explicit instructions in order to automate the described Step. Instructions may include text and pictures of the automated systems. Additionally, the Detailed view should provide known business errors of the Step. For every element in the Phase, the Step and the Detail levels a placeholder function is generated into the code file, to be later

modified by the developer.

Additionally, developers, testers and maintainers should be able to view the process in more detailed manner. For this purpose, a technical view should be implemented, to present the actual code logic and its documentation. The details of the technical view were left to be discussed in more detail in the interviews.

## **5.2 Solution Compared with Other Mainstream RPA Platforms**

Unlike the mainstream RPA platforms such as UiPath, the tool developed during this thesis does not aim to generate finished software. The main motive is to create a tool that unifies communication processes and provides up to date documentation for all stakeholders involved in the development processes. This is considered to result in more unified and straight-forward development practices, creating more value for both the RPA provider and the customer.

The developer has more freedom to use whatever techniques and tools they want, as the PDD-SDD Tool does not create the actual logic. This enables more flexibility and better-suited solutions to different RPA tasks under development. The developer has also the ability to change the project structure, if needed. In these situations, the change should also be validated by the customer and the project manager via updated PDD.

The tool allows custom solutions in the cloud infrastructure (Step 2, Step 4) allowing it to be integrated into the desired infrastructure as a part of other company-specific technical solutions. It should be possible to integrate PDD-SDD Tool into different environments with both RPA and more traditional development projects.

### 5.3 Work Distribution and Initial Roadmap

As this project is divided into two separate theses developing and researching the provided value of the created tool, the research questions listed in Table 4.4 are divided between Ahmed Abdulghani and Sampsa Vuorela. Abdulghani focuses on Q1, covering the definition and maintenance phases, while Vuorela concentrates on Q2 which includes development and testing. Communication and documentation intertwine with all the other phases, therefore Q3 is investigated in both theses.

The PDD-SDD Tool's development was conducted from Fall of 2019 to Spring of 2020 by both Abdulghani and Vuorela. Two groups of interviewees were gathered from MOST Digital, to provide input for the design and the progress of the tool. With the gathered intel, the tool's design and development was adjusted accordingly. Project's initial roadmap consists of four main development iterations which are illustrated in Figure 5.3.

The Chapters 4 and 5 are part of the first iteration in the roadmap, which were conducted in the beginning of fall 2019. These chapters identified potential Challenges in RPA development, derived Requirements for the PDD-SDD Tool and introduced its initial design. The rest of the iterations in the roadmap are discussed in more detail and validated through interviews in both theses from their separate viewpoints.

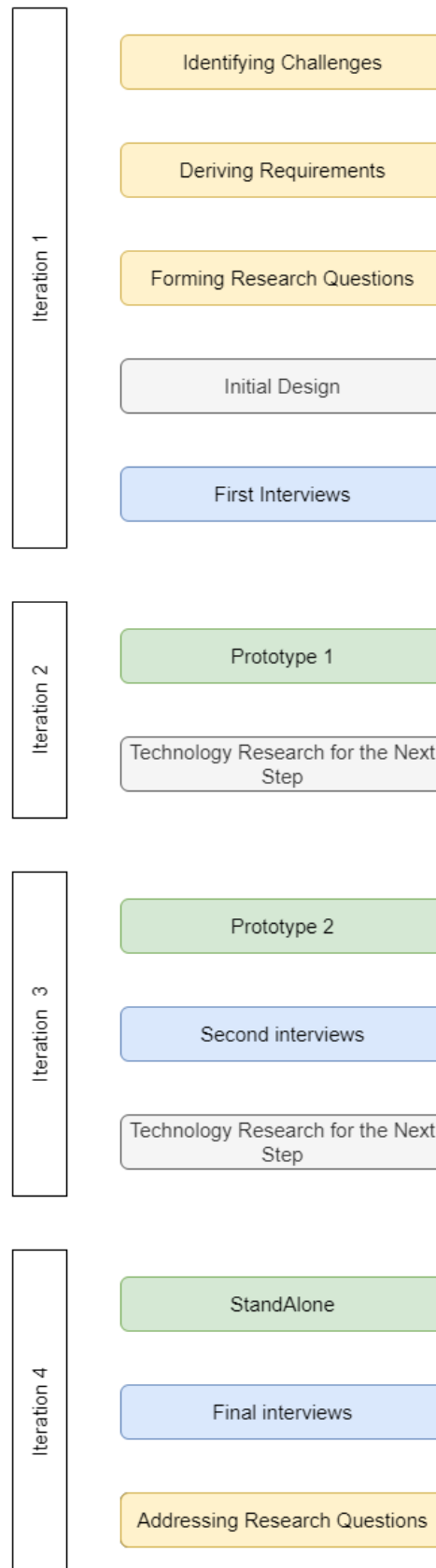


Figure 5.3: Initial roadmap for the PDD-SDD Tool's development.

# **6 Research Questions and their Validation Method in this Thesis**

This chapter provides the understanding of the structure of this thesis from this chapter onward. This chapter also addresses the methods used to validate the Research Questions that were presented in Table 4.4, and how the interviews were conducted. Chapters 2 - 5 were written in co-operation with Sampsa Vuorela, who continued his thesis with a different angle to the subject matter. He is focusing on the developer's viewpoint for the implemented Tool, and how it will or might affect the developer's work at MOST Digital.

This thesis focuses on Research Questions 1 and 3 (Q1 & Q3), which were listed in the Table 4.4. In other words, how the business, or the process definers, and the maintenance team can benefit from the implemented Tool? And does it ease communication and process documentation? To arrive at the conclusion of the benefits of the Tool, multiple interviews were held to guide the development iterations and to validate the Research Questions.

After the validation of the Research Questions done in the interviews, new proposals and challenges are introduced to render the outcomes of the Tool's iterative development phases. The Tool's development is also covered throughout the initial design phase to the added must have functionalities and lastly to the final working StandAlone version of the

Tool.

Chapter 7 inspects the conducted interviews thoroughly, one interview at the time. The interview chapter follows a specific structure in order to extract the gathered information from the interviews in a coherent manner. As the chapter focuses solely on the interviews, the structure and the data gathered, technicalities are saved for Chapter 8, which dives in to how the Tool was developed. The final chapter is the Conclusions Chapter (9), which compiles the findings and the implemented Tool's potential effects.

## 6.1 Validation Method

Addressing the Tool's Requirements, from the business and maintenance perspective, demanded three interview rounds with the process definers, the project managers, the developers and the maintenance team. These people represent the users of the Tool throughout the Robotic Process Automation's (RPA) project life-cycle, and thus were selected to be interviewed. The knowledge gained from the work experience aided with identifying the correct professionals for the interviews, as Abdulghani and Vuorela have worked with every phase of RPA's life-cycle (Section 4.1) [3].

In order to validate the Research Questions (Q1-Q3), which were listed in Table 4.4, interview rounds were tied to the Tool's development phases. The features of the Tool helped in clarifying not only the Requirements but also the Research Questions. At the last phase of the Tool's development, last rounds of the final interview were held to demonstrate the StandAlone Tool and its capabilities. During the demonstration the Research Questions were presented for the final time and were fully validated. The validation metrics were tied to the Challenges (C1-C25) and the Requirements (R1-R3) presented in Table 4.3. If the interviewees acknowledged that the PDD-SDD Tool satisfied the gathered Challenges

and the Requirements, the Research Questions of this thesis would also be validated and acknowledged.

The reason for three developmental and interview iterations was decided in the Initial Design Figure 5.1, which was to give the development work structure and deadlines for the deliverables for Most Digital. Additionally, to develop in respect to Agile methodologies<sup>1</sup>, iterative and value driven delivery cycles were set high in the priority list for the methods of working. With each iteration and requirement resolving, the interviewees identified through the demonstrations that the Research Questions were indeed validated, thus tying them to the Tool's requirements (Table 4.3).

The methodology of acquiring the feedback for the Tool's specific phase was accomplished by organizing multiple 45 to 60 minutes of interview sessions with the professionals. These sessions were recorded for later inspection, and comprehensive notes were also taken by Abdulghani and Vuorela, the interviewers. From the interview material they gathered the feedback data in order to improve on the technical solution and to determine if our technical solution was heading towards the right track, whilst keeping the scope in mind. That way they were able to keep the Tool's development iterations short and agile. This was the constructive method for gathering data in a consistent manner throughout the different stages of the Tool's development. The gathered data is also compared to the previously identified Challenges in Section 4.2.

The research method used with the gathered data was based on unfolding the findings into a visual table format, from which it is notably easier to link different Challenges and Improvement Proposals together. The visually linked data eases the filtering of the Tool's "nice-to-have" and "must-have" functionalities, as well as eases the managing of

---

<sup>1</sup><https://www.agilealliance.org/agile101/the-agile-manifesto/>

the project scope.

There were initial thoughts for the Requirements and the features for the Tool. Based on these thoughts and the work experience in the RPA field (Chapter 4), designing and drafting the first iteration of the prototype was initiated, resulting in the creation of the Initial Design of the PDD-SDD Tool, which is presented in Figure 5.1. From these initial drafts, supporting material was created for demonstration purposes for the interviews, which were held with the professionals presented in the next section.

## **6.2 The Professionals**

The selected professionals have a specific knowledge in their domain, as they work in that part of the RPA life-cycle daily. Projecting their thoughts and needs to the designed Tool was paramount, as the aim of the developed Tool is to be used by these professionals to aid their daily work.

As this thesis is focused on the business side of the workflow, the chosen interviewees are indulged in process definition and solution maintenance. In order to understand their viewpoints, specific interview questions were generated. Also, a couple of developers and a technical architect were chosen to be interviewed to give their viewpoints on how does the definition and the maintenance processes affect their work.

Initial Design's (Figure 5.1) Steps 1 and 5 are the only Steps that affect the target groups mentioned earlier (Process Definers and Solution Maintainers). These Steps encompass the front-end of the PDD-SDD Tool, and these Steps are used by two different user groups. The first group is the Business users or the process definers, who are defining a new process with or without the customer (Step 1), while using the PDD-SDD Tool. And the



second group is the solution maintenance team, who use the PDD-SDD Tool's visual process flow and the automatically generated code documentation to solve issues and update the processes as needed (Step 2).

From MOST Digital Ltd, six professionals were chosen to represent the business, process workflow definition, project management and maintenance. From Ailea Ltd two professionals were chosen to represent the developer's maintenance duties at Ailea to MOST Digital.

### **Maria Vuontisvaara**

Maria Vuontisvaara is a seasoned professional with over 20 years of experience in financial and human resource management. Maria is also one of the co-founders of MOST digital. One of Maria's tasks is defining the processes with the clients from a business logic perspective.

### **Katri Tapio**

Katri Tapio is an experienced project and service manager at MOST Digital, and she has been working with RPA since 2017. Being the service manager, Katri handles MOST Digital's maintenance processes.

### **Essi Nevalainen**

Essi Nevalainen works also at MOST Digital as a consultant and project manager. Essi has been also been actively defining processes to be developed.

### **Heikki Liukkonen**

Heikki Liukkonen is Ailea's CTO and senior RPA and integration developer. Heikki has been working with RPA projects for the past 3 years, and in the IT sector for the past 9

years. Regarding RPA projects, Heikki has been on the both sides of the project. He has been defining the processes, implementing the architecture, while also developing and maintaining RPA projects.

### **Juuso Jaakola**

Juuso Jaakola has been working as a developer at Ailea on RPA development projects for 3 years. Most of the projects Juuso developed were delivered to MOST Digital. As a developer Juuso also maintains and further develops his solutions, thus it is important to have his point of view as a maintainer.

### **Niina Mäkelä**

Niina Mäkelä has been working at MOST Digital from 2019 as a RPA developer. Similarly to Juuso, Niina also maintains RPA projects when the need arises, thus it is also important to understand her perspective on the matter.

### **Elias Koskinen**

At MOST Digital Elias Koskinen is known as the robot's well-being personal trainer, that is to say that Elias is the maintenance mastermind. Working tirelessly on keeping customer satisfaction above average. His knowledge in the maintenance field is imperative for understanding the needs of the maintenance team. Elias has been working at MOST Digital for almost 3 years.

### **Esa Poutanen**

Esa Poutanen is an experienced C-sharp developer who focuses on developing support-aiding functionalities into the cloud. Esa works part of the maintenance team as well and has had his fair share of maintenance work. He's been also actively developing solutions

to ease and automate the work done at MOST Digital's maintenance team.

Some of the professionals were interviewed in both theses, they were:

1. Heikki Liukkonen
2. Juuso Jaakola
3. Niina Mäkelä

The reason these professionals were chosen to both interview sessions is based on their expertise working with different aspects of the RPA project life-cycle, from defining the process to developing and maintaining the project, thus it was important to have their views included on the topics discussed. All of the interviews were held separately from Vuorela's interviews, because of our different research topics.

The next chapter dives into the findings of the interviews and how the surfaced data is used to guide the Tool's developmental phases. Also, each interview round is broken down structurally to give a better understanding of the data collection methods used.

## 7 The Interviews

A clear structure was designed for the held interviews in order to present and collect the data in a coherent manner. The initial interview design was divided into two interview questionnaire structures, one for Abdulghani's Research Question Q1 and the other for Vuorela's Research Question Q2. The structure and questions dedicated for Research Question Q3 were the same in both interview designs.

The interview questions discussed in this thesis centered around three categories:

1. Process definition and documentation
2. Solution maintenance
3. General communication during the project

In each category there are multiple structured questions that enable a methodical way for querying the information from each interviewee.

The questions in each category are:

### **Process definition and documentation:**

1. What are the top 3 steps necessary for defining a process?
2. Are the above mentioned steps established as a system?
3. What are the challenges in process definition?

4. What could ease the process definition procedure?
5. What are the top 3 challenges in process documentation?
6. How often the document needs to be updated?
7. Does an outdated process documentation cause issues?
8. What could help with creating and updating the process document?

**Solution maintenance:**

1. What are the challenges regarding project maintenance?
2. Do you use process documentation to aid in maintenance work?
3. What do you wish would be easier regarding project maintenance?

**General communication during the project:**

1. Has there been issues with the communication with the developers? If so, what were the top 3 issues?
2. Has there been issues with the communication with the customers? If so, what were the top 3 issues?
3. What do you wish would could be easier regarding the communication during the beginning of the project?
4. How often you need the latest status update about the project?
5. How and with what tools you figure out or query the project status?

A demonstration was also held at each interview iteration, to present visually what was achieved so far, and to validate the development scope. Ideas, Challenges, feedback and Improvement Proposals were also encouraged during the interviews. The specific questions were not static throughout the iterations. The interviews' three different structures

are presented in detail in the next three sections, with each interview's findings, new Challenges and new Improvement Proposals.

The gathered Challenges and the Improvement Proposals from the interviews, and how they affected the Tool's developmental scope is presented in the next chapter, Chapter 8.

## **7.1 The First Interviews**

The first interview began with a short introduction of the Thesis and a short background presentation on how Abdulghani and Vuorela got to the point of designing a tool to help MOST Digital's RPA project life cycle. The first round of interviews were held in September 2019, where the initial design, which is depicted in Figure 5.3, was presented to the interviewees.

To make sure the interviewees were not biased with their answers regarding the RPA Challenges and the Tool's requirements, the Initial Design was not mentioned, nor the material presented at the end of the interview was at their disposal beforehand.

### **7.1.1 The Structure**

The first set of questions were revolving around the current working methods regarding the three main question categories, which were Process definition and documentation, Solution maintenance and General communication during the project. The interviewees were also asked about the current general RPA Challenges and Improvement Proposals, which were documented.

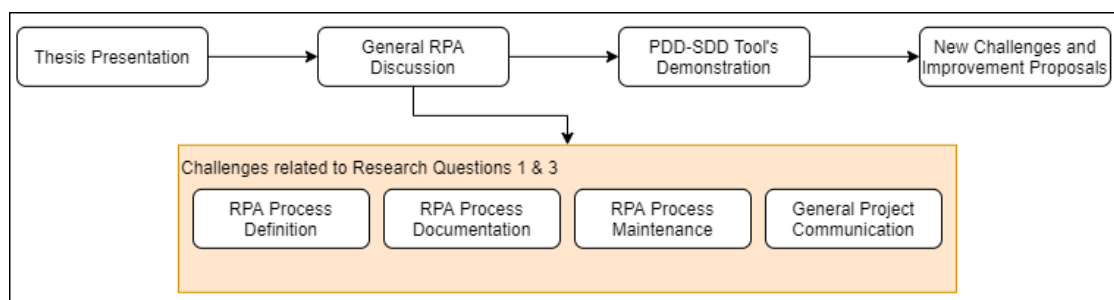


Figure 7.1: The structure of the first interviews, and how the Research Questions relate to the topics discussed.

### 7.1.2 The Findings

The initial thoughts of the interviewees were positive but not everyone saw that it would be technically possible to tackle some of the communication challenges discussed. The initial criticism didn't diverge Abdulghani and Vuorela from the idea of the Tool, the criticism was taken as new Challenges and Improvement Proposals to further develop the Tool.

To some extent the tools and methodologies used to define, document and communicate were not standardized, as the professionals use what is convenient and natural to themselves with each project. This somehow non-standardized way of functioning exposes the human nature of working with high paced projects, and how one Tool cannot accommodate every need - as it is not a "one size fits all" solution.

However, MOST Digital agrees that a standardized way of working and using a dedicated Tool could benefit not only the RPA projects' life cycle phases but also the professionals' work satisfaction.

### 7.1.3 The Challenges

Multiple Challenges were presented in the discussed categories and some of them corresponded with the "Identified Potential Challenges" presented in Table 4.2. Below are listed the new Challenges, gathered from the interviews, in a visual manner grouped by category.

Phase	Challenge	Description
<b>Process Definition</b>	C-Def1	Customer defines the process alone
	C-Def2	Non-standard definition process
	C-Def3	Lack of technical professionals during the definition process
<b>Process Documentation</b>	C-Doc1	Unclear responsibilities regarding the documentation updates
	C-Doc2	Documentation language
<b>Solution Maintenance</b>	C-Mai1	Unclear and non-standardized logging
	C-Mai2	Definition of done
	C-Mai3	The use of personal notes causes scattered information
<b>General Communication</b>	C-Com1	Lack of centralized project communication channel
	C-Com2	Using too technical or high vocabulary in communication (C4 - Table 4.2)
	C-Com3	Manual status updates

Table 7.1: Challenges gathered from the first interviews

#### Process Definition

The current process definition utilizes a PowerPoint template for documenting the "as is process" to be automated. The challenges related to process definition with a flexible template is it allows non-standard usage seen fit by the process definer (C-Def2). Also, an experienced customer, who has co-defined a process or two, could define the process alone. This fact adds more complexity and variance to the definition process (C-Def1). Letting the customer define the process alone is not necessarily a challenge for process definition per se, it might cause side effects that might demand clarification later [3]. The last challenge identified from the gathered data was not utilizing the technical professionals in the definition process (C-Def3) from both sides. In most cases C-Def3 challenge



leads to some of the Challenges depicted in Table 4.2, more specifically; C1, C2, C3, C5, C21, C22 and C23.

### **Process Documentation**

The challenge C-Doc1 "Unclear responsibilities regarding the documentation updates" relates usually to process definition, where the responsibilities are set. This same challenge could be linked to the Challenges C13 and C17 mentioned in the Background Study's Table 4.2. The second Challenge mentioned in the Table 7.1 is C-Doc2 "Documentation language" indicates the two languages used at MOST Digital. Although the main working language is Finnish, RPA development and technical documentation is done in English, but the process documentation language is in most cases in Finnish. This causes additional work for the maintenance team as the needed information can be dispersed in multiple documents.

### **Solution Maintenance**

Exceptions' handling is developed to the solution when they have been indicated beforehand in the PDD, or when they arise during the testing phase. However, when exceptions happen, after the alerted support phase and usually unsupervised during the production runs, the maintenance team relies heavily on logging generated by the software robots, as it is the only gateway to understand what happened during the run and what caused the issue. Having unclear and varying logging principles prolongs the issue solving time, hence C-Mai1.

Another challenge mentioned in the interviews was the unclear "definition of done" of the projects, C-Mai2, which is caused, in many occasions, by the leaking scope of the project. This is mostly due to the Challenges C1-C5, C7 and C8 depicted in the Table 4.2. Lastly, Challenge C-Mai3, the usage of personal notes during maintenance that are not shared

with the rest cause scattered information and outdated documentation.

### **General Communication**

Communication affects immensely all phases of the project life-cycle, and the quality of the project is directly correlated to the quality of the communication, as can be seen from the Challenges listed in Table 4.2. In the first interviews communication was, unsurprisingly, one of the central topics, and regarding the communication during the project, Challenge C-Com1 was raised. Project communication tend to span over multiple services and channels, thus retrieving or combining information get complex.

Regarding the information retrieval from multiple sources, Challenge (C-Com3) was also raised, as it is easier to ask for the needed information from others than trying to find it. Hence, manual status updates were raised as a Challenge in general communication. The last Challenge that was derived from the collected data was related to the Challenge C4 from Table 4.2; Using too technical or high vocabulary in communication. This Challenge punishes both sides of the project, the customer and the service provider i.e. MOST Digital.

### **7.1.4 The Improvement Proposals**

During the first interviews the initial design, which is illustrated in Figure 5.1, was presented after the the General RPA Discussions, depicted in the Figure 7.1. The demonstration of the initial design ignited the discussions in a way that led to more feedback and Improvement Proposals, as the interviewees felt that there might be a way to affect the current Challenges.

Below is presented the table of Improved Proposals to the corresponding Challenges discussed. Not all Challenges received an Improvement Proposal nor all of the Improvement

Proposals were connected to the Challenges, as some of them were general proposals. The gathered data is grouped by category and examined accordingly.

<b>Phase</b>	<b>Proposal</b>	<b>Description</b>
<b>Process Definition</b>	P-Def1	Standard definition process
	P-Def2	Technical professionals take part in the definition process
<b>Process Documentation</b>	P-Doc1	Clear responsibilities regarding updating the documentation
	P-Doc2	Multiple process definition templates
	P-Doc3	Project status visualized in the PDD
	P-Doc4	Latest Changes -functionality to the PDD
	P-Doc5	Listing project members in the PDD
<b>Solution Maintenance</b>	P-Mai1	Clear and standardized logging
	P-Mai2	Solution's status indicator: WIP & Done
	P-Mai3	Standardized documentation
<b>General Communication</b>	P-Com1	Centralized project communication channel
	P-Com2	Automatic status updates

Table 7.2: Improvement Proposals gathered from the first interviews

### **Process Definition**

A more standardized approach to the process definition phase was indeed one of the most sought after Improvement Proposals (P-Def1) as it could solve multiple challenges down the line, and this was directly connected to C-Def2 Challenge. Another proposal was linked to the lack of technical professionals taking part in the definition process, as they could inquire minute technical details about the process or the used systems (P-Def2). This Improvement Proposal is a direct consequence to Challenge C-Def3.

### **Process Documentation**

During the development of the solution various kinds of exceptions and challenges arise which might not have a description in the PDD. In these cases the new defined extensions

should be added to the PDD (P-Doc1), but there are occasionally no clear responsibilities regarding who should update the documentation as declared previously as Challenge C-Doc1. Another Improvement Proposal that could help with assigning responsibilities is listing all of the project members with their contact information into the PDD (P-Doc5).

If the Tool would be used for process definition, it should provide the ability to use multiple process definition templates to satisfy the most common forms of processes (P-Doc2), as it would mitigate time needed for preparation.

A visual way to present the project status was also proposed, as it would centralize the project status to one Tool (P-Doc3). Also, the interviewees wanted a way to observe the latest changes in the documentation (P-Doc4).

### **Solution Maintenance**

The Improvement Proposals gathered from the maintenance team circulated around standardization of the solution's accessible documentation (P-Mai3), the Solution Design Document (SDD), and the logging created by the solutions (P-Mai1). This way the maintenance team can shuffle between different RPA projects and maintain them without spending time figuring out how or why the logging is incoherent between solutions or how to interpret the SDD. Both of these Improvement Challenges relate to Challenges C-Mai1 and C-Mai3.

The maintenance team also receives multiple emails and notifications during the day from the robots, to which the team proposed a way to recognize those that truly matter. Some of the notifications are sent by solutions that are still under development or further development is in progress. In these cases the developer is usually in control, thus no maintenance is needed, hence solution's status indicator would ease the load on the maintenance team

(P-Mai2). The Challenge C-Mai2 is directly associated with the aforementioned Improvement Proposal.

### **General Communication**

The less the professionals have to search for the correct data or quest for the correct channel from which the needed information should be mined, a centralized communication channel for each project (P-Com1) was introduced as an Improvement Proposal, which is directly connected to the Challenge C-Com1.

Related to the Challenge C-Com3, the opposite was proposed; automatic status updates (P-Com2), which would decrease the need to ask someone else about the solution's status. This proposal is in a way connected to what the maintenance team proposed: solution's status indicator (P-Mai2). However, in this case, the automatic status updates are proposed to be more informative as in Git Commit Message<sup>1</sup> would be used.

## **7.2 The Second Interviews**

The second interviews were held in March 2020, during which Abdulghani and Vuorela have been developing two prototypes of the Tool based on the feedback from the first interviews, as illustrated in the Initial Roadmap for the PDD-SDD Tool's development (Figure 5.3). The second interviews' goals were to assess the direction and the functionalities of the Tool against the identified Challenges and Improvement Proposals. Other goals were to gather user and usage experiences and other feedback by allowing the interviewees to experiment with the Tool.

At this point in development, the second interview brought up less new challenges and

---

<sup>1</sup><https://git-scm.com/docs/git-commit>

more user experience related Improvement Proposals. This section explores the impressions the interviewees had on the developed prototype.

### **7.2.1 The Structure**

The structure of the second interviews was straightforward compared to the first interview's structure, as there was nothing that needed concealing or revealing. The structure consisted of four parts:

1. Validating the challenges and the derived requirements.
2. Demonstration of the second prototype.
3. Prototype experimentation.
4. Prototype feedback.

#### **Validating the challenges and the derived requirements**

The validation of the Challenges gathered from the first interviews was executed by reviewing the derived requirements, which are listed in Table 4.3. After the validation of the challenges, the interviewees were asked about the derived requirements, and to validate them as well. This provided the needed feedback for Abdulghani and Vuorela to continue the development on the right track.

#### **Demonstration of the second prototype**

The prototype was demonstrated through Microsoft Teams<sup>2</sup>. The functionalities that were demonstrated focused on process definition and the usability.

---

<sup>2</sup><https://teams.microsoft.com/>

### **Prototype experimentation**

The interviewees were able to test the prototype through MS Teams "Give Control" - functionality<sup>3</sup>. Once the control was given to the interviewee, they explored the prototype on their own in order to generate their own opinions about the design, the experience, the usability and the functionalities developed.

### **Prototype feedback**

The feedback was documented for later analysis. The feedback was positive as the general idea of how the Tool would be used was communicated clearly with an interactive demonstration. The interview yielded also new Improvement Proposals and new Challenges.

## **7.2.2 The Findings**

The Tool's functionalities are presented more thoroughly in Chapter 8, but the findings that received most praise were related to the simplicity, automatic documentation (R3, Table 4.3) and continuous project saving while using the Tool. Nonetheless, questions were raised regarding the data collection, ease of PDD updates and how should the functionality of the guiding questions (R1, Table 4.3) support flexible use.

## **7.2.3 The Challenges**

Some of the Challenges were converted into Improvement Proposals as they were out of scope but important nonetheless. The only Challenge raised in the second interview, which could be identified as a challenge, by the interviewees regarding process definition and solution maintenance was the *process definition language*.

---

<sup>3</sup><https://support.microsoft.com/en-us/office/share-content-in-a-meeting-in-teams-fcc2bf59-aecd-4481-8f99-ce55dd836ce8>

The language used to define the process is usually Finnish, and software development is usually done in English. As one of the Tool's functionalities is to generate a code structure based on the defined process for the developers, Abdulghani and Vuorela had to solve this Challenge in the final version, the StandAlone version. Not solving this particular challenge would cause unnecessary complexity, as the developers would have to use the Finnish function names. Or it would push the definers to use English for documenting the process.

This Challenge was raised by multiple interviewees as the prototype was developed to only handle and parse English language, thus this Challenge was present at every interview.

#### **7.2.4 The Improvement Proposals**

The second interviews leaned heavily towards Improvement Proposals, as the focus of the interviews was the demonstration of the prototype and its functionalities. This naturally generated discussions about how the current state of the Tool could be improved. The findings are listed in their categories below, and one new category was also added, as a couple of Improvement Proposals could not be directly appointed to the existing categories.

##### **Process Definition**

In most automation processes the robot is accessing a plethora of different types of data, and the customer is usually interested in gathering and analysing their process data. Often, data collation is done during the process definition, thus the ability to list the collectable data during the process definition (P-Def1) was proposed.



Phase	Proposal	Description
Process Definition	P-Def1	List of collectable data during the process
	P-Def2	Color tagging
	P-Def3	Visual indication of the process sections that are executed by the robot, the AI or with the help of humans
	P-Def4	Using existing process definitions as templates for new processes
Process Documentation	P-Doc1	Changes to the PDD should be easy after the project has started
Solution Maintenance	P-Mai1	The ability to add exception handling documentation afterwards
General Communication	P-Com1	Commenting the PDD
Other Proposals	P-Oth1	Access right control of customers to projects
	P-Oth2	Control project hierarchies

Table 7.3: Improvement Proposals gathered from the second interviews

Color tagging (P-Def2) and visual indications (P-Def3) help to dissect and understand the process as it is used by multiple people with different technical backgrounds. Oftentimes, the process encompasses more than one technology in the process, thus it is easier to acquire a holistic image of the whole by using visual indications and colors.

Generating templates from existing processes that have been defined (P-Def4), lessens the need to start from scratch with each process, especially if earlier processes have similar characteristics.

### Process Documentation

At times, the PDD is in constant change, even after the process has been defined. New exceptions and business rules arise during the development, thus it is important to be able to update and modify the PDD after the project has started (P-Doc1). It is preferable that the PDD should not change, but that would not represent the agile world of RPA, where changes are constant, hence the Improvement Proposal.

### Solution Maintenance

Linked to the previous Improvement Proposal (P-Doc1), the maintenance team faces new exceptions even after the process has been running stably in the production environment,

and for that the Improvement Proposal P-Mai1 was proposed. Having the ability to add new exception handling documentation to the defined process using the Tool would benefit the maintenance team immensely. The difference to P-Doc1 is instead of updating a certain documentation, basically replacing the existing content with the update, the Tool could provide a listing of existing exceptions and a functionality to append new exception handling documentation to the list.

### **General Communication**

Commenting the defined process in the Tool (P-Com1) would help with centralizing the communication regarding the process documentation.

### **Other Proposals**

Some of the proposals were disregarded as out of scope entirely, as they would focus on a different use case compared to the PDD-SDD Tool developed. Nonetheless, some of the other proposals were gathered that were deemed in scope.

Controlling who has access to which project (P-Oth1) is definitely one proposal that should be implemented. Security and access control by authenticated users eliminates the misuse of the Tool by the customers, as they would be granted access to the Tool as well. Without access control the customers would be able to access other customers' critical processes to the detail.

Organizing projects into folders and customers (P-Oth2) would help control the documents saved in the Tool. This Improvement Proposal would benefit all of MOST Digital's users, as it mitigates the time needed to find the correct project with clear hierarchical structure of the customers and their projects.

## **7.3 The Final Interviews**

The objective of the final interviews was to determine if the StandAlone, the final version of the agreed scope of the PDD-SDD Tool, accomplished what it was designed for.

The final interviews were held online in April 2020, a month after the second interviews. The Tool's implementation pace was more intense in order to meet the agreed deadline. The final interviews also marked the end of the PDD-SDD Tool's development, as the scope of the StandAlone version was complete. The goal of the final interview round was to officially conclude the Tool's development and to complete the validation of the Initial Requirements and the Research Questions.

### **7.3.1 The Structure**

The final interview's structure was divided into three sections. The first section was to remind the interviewees of the Initial Requirements of the Tool, which were presented in Table 4.3. This method would help the interviewees to focus on the Initial Requirements during the StandAlone's presentation. The second section of the interview focused on the demonstration of the finished StandAlone version of the PDD-SDD Tool. The third and final section of the interview was designed for analyzing the Research Questions with the interviewees.

### **7.3.2 The Findings**

The target group of this thesis were pleased with the usability of the Tool, as it provided a clear and precise structure for not only documenting the defined process but also for following the process path later during the project. The definers had contradicting thoughts of the actual usage of the Tool, as for some the Tool seemed to be aimed for technical users with its bells and whistles. For others the Tool's interactivity and the way it enabled

knowledge to flow was a colossal leap compared to the previous option.

The maintenance team addressed that the standardized project definition and documentation provides sturdier grounds for the maintenance work as they would not deplete time in searching the needed information regarding the project. The maintenance team also commented on the possibility of eased communication, as the client and project the maintainer could communicate using the Tool's Phases', Steps' or Details-levels' names.

### 7.3.3 The Challenges

The challenges yet again were minimal as the main technical issue from the previous interview was the *process definition language*. At this point of the project, the main technical challenges were fixed in the development. Nonetheless, two non-technical challenges were raised, which were:

1. How to get the customers to use the Tool?
2. Remembering to update the PDD (Non-developers)

Both raised challenges are linked to the working methods of the professionals and the company's best practices, but not directly to the Tool's usage.

### 7.3.4 The Improvement Proposals

<b>Improvement Proposal</b>	<b>Description</b>
<b>I-Pro1</b>	Navigating from current step to another specific step
<b>I-Pro2</b>	Tool's on-boarding
<b>I-Pro3</b>	Phase's WIP and completed statuses

Table 7.4: Improvement Proposals gathered from the final interviews

The gathered Improvement Proposals from the final interviews generated I-Pro1, I-Pro2 and I-Pro3. The interviewees suggested having an on-boarding session on how to use the Tool (I-Pro1), as there were a lot of features to use, even though not all of the features were directed to this thesis' target group. Some of the features are only for the development team.

One of the interviewees nominated the idea to have a visual cue over the phases that are currently "work in progress" (WIP) and also over the completed phases (I-Pro2). This would enable the project managers to get the status updates that are based on the developmental work, which means that the project managers do not have to ask for the project status from the developers, as they can see the status by themselves from the Tool.

The last Improvement Proposal which was proposed, is a technical functionality which would help navigating the project's Steps and Phases inside the Tool. The StandAlone enables the users to drill down from a specific Phase to the Step level and from there to the last level, the Detail level. The only way to navigate to a different Phase is to click through the previous view. The proposal was documented for further development proposals, as bigger projects demand better navigational features.

## **7.4 Summary**

This chapter answered the question of "How the Research Questions were validated?". Shortly, by interviewing the chosen professionals from both companies and by demonstrating the developed Tool's different phases, the Research Questions and the Tool's Requirements were approved and validated by the interviewees. The gathered and analyzed data was used to help guide the Tool's iterative development phases, and to guide the Tool's developers (Abdulghani and Vuorela) towards the correct functionalities, keeping

in mind the usability of the Tool and the scope of the project. The Tool's requirements were listed in Table 4.3, which were:

1. R1 - Providing guiding questions during the definition phase
2. R2 - Providing accessible, unified and up to date documentation
3. R3 - Providing automatic documentation

Based on the feedback, the answers answered in the interviews and the direct questions regarding the Research Questions, which are presented in Table 4.4, the Research Questions were validated orally. This validation was based on the demonstrations of prototypes and the StandAlone version of the Tool.

The gathered data from the interviews gave the Tool's developers the insight and the correct direction regarding what to implement and which proposals should be left for further development. The questions asked from the interviewees were based on the work experience of Abdulghani and Vuorela [3]. The questions were guiding the interviewees to give feedback on the developed Tool in every phase of the development. Also, different Challenges and Improvement Proposals were gathered, which gave Abdulghani and Vuorela multiple perspectives and thoughts that had not been considered in the Initial Design (Figure 5.1).

In order to sum up the gathered and the analyzed data from the interviews, an overview is presented below to understand how the Challenges and the Improvement Proposals emerged during the interview iterations. All in all there were 14 Challenges and 24 Improvement Proposals, which were presented and analyzed in this chapter. Naturally some of the Improvement Proposals were linked to the Challenges, and some of them were accepted into the Tool's development scope as they were linked to the Tool's main requirements, others were left for later inspection and later development. These findings

are presented in the next chapter (Chapter 8).

The interview iteration with the most Challenges and Improvement Proposals was the first interview (Challenges Table 7.1 and Improvement Proposal Table 7.2), as it contained 11 Challenges and 12 Improvement Proposals. The presented Challenges and the gathered Improvement Proposals were divided into separate Phases that were linked with the RPA project's life-cycle. The Phases are:

1. Process Definition
2. Process Documentation
3. Solution Maintenance
4. General Communication

From the first interview, the Challenges were evenly divided between the Phases, with 3 Challenges in each Phase except the Process Documentation Phase, which had only 2 Challenges. Interestingly, the Phase with the most Improvement Proposals was the Process Documentation Phase, which accumulated 5 Improvement Proposals, which are presented in Table 7.2. As for the second and the final interviews, the Challenges and the Improvement Proposals reduced in count, except for the second interview's Process Definition Phase's Improvement Proposals amount. The reduction in Challenges' and the Improvement Proposals' count is attributed to the development of the Tool in between the iterations. The Figure 7.2 below visualizes a summary of the Challenges and the Improvement Proposals, and their emergence in each interview iteration.

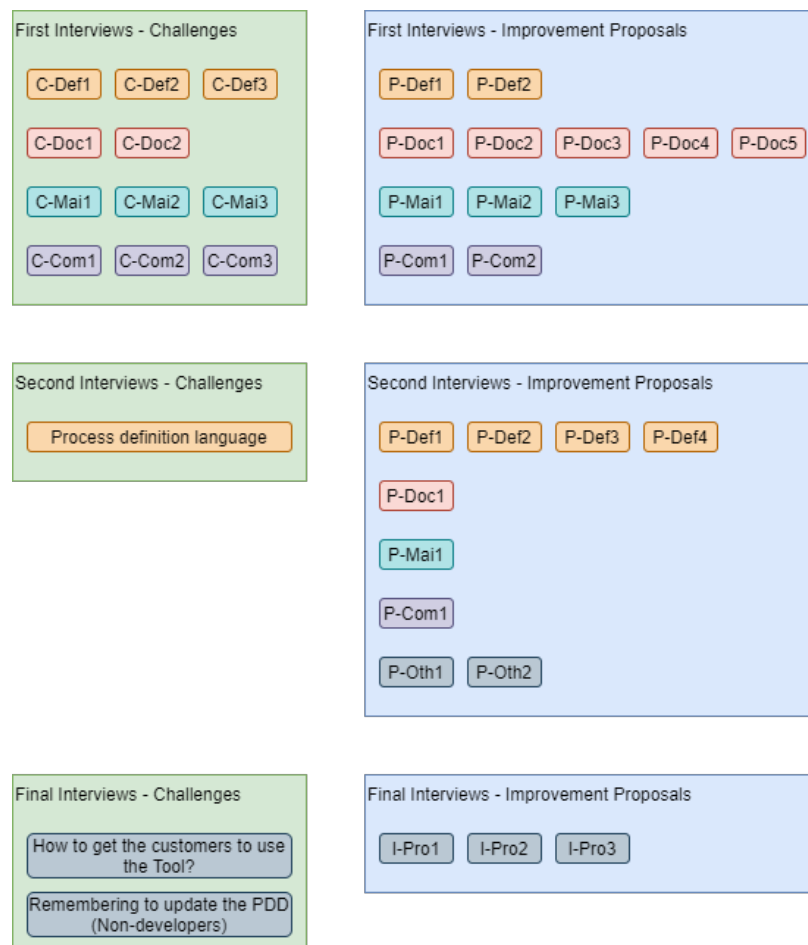


Figure 7.2: Challenges and Improvement Proposals in each interview iteration.



---

The three rounds of interviews were based on the three iterations of the Tool's development phases: Prototype 1, Prototype 2 and StandAlone. The interviews were video recorded and comprehensive notes were also taken. From the videos and the notes Abdulghani and Vuorela analyzed the gathered data and produced the final version of the analyzed data from each interview. The final version of the analyzed data was then used to construct the tables presented in this chapter.

The interviews gave insight to the actual bottlenecks in RPA project development and life-cycle. Not only the gathered information could be used for bettering the project definition and maintenance, but also project development and communication. This information could be expanded to other than RPA projects with modified scopes, as most of the work done in IT-projects contain similar if not exact challenges.

The next chapter examines the Tool's developmental phases from the technical point of view, as the basis of the Tool's Requirements, Challenges and Improvement Proposals were presented successively in this chapter.

# 8 Tool's Development

The Tool's initial design followed along the initial hypothesis created in the beginning of this project. The first and second prototypes were developed after the first interviews. With the gathered data from the interviews, the prototypes were modified accordingly in respect to the scope. Also, technology research was conducted during and after prototypes' development, as depicted in Figure 5.3.

This chapter goes into detail of how the Tool was designed and developed. The Intellectual Property Rights (IPR) are shared with MOST Digital Ltd., thus the source code will be kept out of this thesis, but screenshots and functionality explanations are presented.

This chapter's structure is divided into development phase sections, as the Tool's development consisted of two prototypes and one StandAlone version, which is the final working version of the Tool that fulfills the agreed scope. Also, an explanation is given on how the identified Challenges and the gathered Improvement Proposals affected the functionalities of the Tool. To understand how the Challenges and the Improvement Proposals were connected, a minimal visualization and explanation is given in this chapter as well.

## 8.1 Ideation and Initial Design

The initial ideation of the PDD-SDD Tool was based on the knowledge and work experience gained from working at Ailea Ltd. and at Most Digital Ltd., and the resulting Initial

Design was generated, Figure 5.1. The design was focused on usability and minimalism, and the technical development was designed for re-usability between the Tool's different views.

The central challenge that the Tool was initially designed to resolve was the misunderstandings regarding the outdated documentation. The lack of time (C16 - Table 4.2) was causing outdated documentations and hence it was identified to be one of the root challenges. For this reason one of the Tool's initial functionalities was aiming for providing automatic documentation (R3 - Table 4.3).

The starting point for the Tool's design was the ease of user experience, meaning that the Tool should not add any additional friction to the current process, nor should it be difficult to learn and use. As the team members have enough tools in their daily use. There was a high risk that this PDD-SDD Tool would not elevate to be one of the daily applications the professionals used. For this reason, the design followed closely the original PDD, which was in PowerPoint document format.

The VUI views presented in Figure 5.2 were designed using the PowerPoint template Most Digital was using for their process definition and documentation. The main difference was having a functionality to drill into the Phases, Steps and Details. The same information was presented in each slide in the original format, but the Tool would help focus on each area at once and to enable data collection from each VUI level. The collected data is then used for various functionalities mitigating the Challenges presented in the interviews, as visualized in Figure 7.2. For these views and the desired functionalities, an open source front-end tool was used and modified combined with a custom made back-end solution for the purposes of PDD-SDD Tool.

## 8.2 Challenges, Improvement Proposals and Tool's Functionalities

To understand the affects the interviews had on the functionalities developed, an overview is presented in Figure 8.1 below. The Figure 8.1 contains four sections visualizing the interconnections between the Challenges and the Improvement Proposals.

The top section of the Figure 8.1, which depicts how the Challenges C-Def1, C-Def2, C-Def3, C-Mai2, C-Doc2 and Process definition language and Improvement Proposal P-Def2 are all linked to the P-Def1 Improvement Proposal. This in essence means that by creating a standard definition process (P-Def1) that forces having a technical professional to take part in the process definition (P-Def2) would solve the following Challenges:

1. C-Def1: Customer defines the process alone.
2. C-Def2: Non-standard definition process.
3. C-Def3: Lack of technical professionals during the definition process.
4. C-Mai2: Definition of done.
5. C-Doc2: Documentation language:
  - The language could be set as a standard for all documentation, thus resolving the second interview's Process definition language Challenge.

With standardizing the definition process and having clear responsibilities (P-Doc1) set during the definition phase and listing the project members in the documentation (P-Doc5) helps with C-Doc1: Unclear responsibilities regarding the documentation updates. With this analysis it is clear that solving the upstream Challenge (C-Def2), for not having a standard way for process definition, with the proposed Improvement Proposal P-Def1, Most Digital is able to stop the ripple effect that C-Def2 Challenge causes. Moreover, by

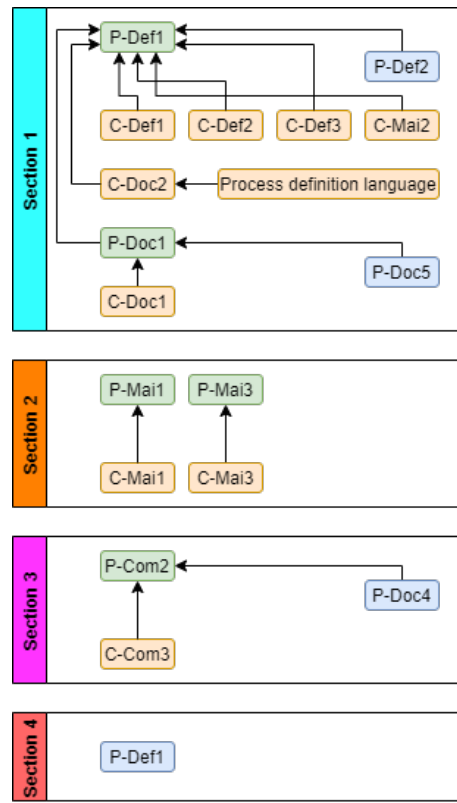


Figure 8.1: Challenges and Improvement Proposals linked.

incorporating the definition of done (C-Mai2) into the process definition from the beginning, it solves the issue of having unclear target.

The second section is more straightforward, as the Challenge C-Mai1 is linked to the corresponding Improvement Proposal P-Mai1. The maintenance proposal would alleviate the maintenance Challenge by having consistent and standardized logging (P-Mai1). The Challenge and the Improvement Proposal is listed in Table 7.1 and in Table 7.2. One of the most important Improvement Proposals is Standardized documentation (P-Mai3), which provides code documentation directly from the code file, and thus relieves the Challenge with having scattered documentation and personal notes not utilized in the final documentation (C-Mai3). Having an up to date automatic documentation aids with the documentation process, thus mitigating the aforementioned Challenge (C-Mai3).

The third section depicts how automating the project status updates (P-Com2) links visual latest changes -functionality in PDD (P-Doc4) into one solution, which solves the manual status updates (C-Com3) Challenge.

Lastly, the final section contains only one Improvement Proposal, which is list of collectable data during the process (P-Def1) from the second interview, found in Table 7.3. Most of the Improvement Proposals could be linked to the Challenges identified in the interviews, and by developing the proposed Improvement Proposals, some of the Challenges would also be resolved. However, not all Improvement Proposals could be linked to the actual Challenges that were gathered during the interviews nor all of them could fit the initially defined scope, thus a list of additional functionalities was created for project's later further development. Also, some of the Challenges were non-technical and difficult to solve with technology alone. The additional functionalities Figure 8.2 is presented below for clarification, grouped by interview iteration.

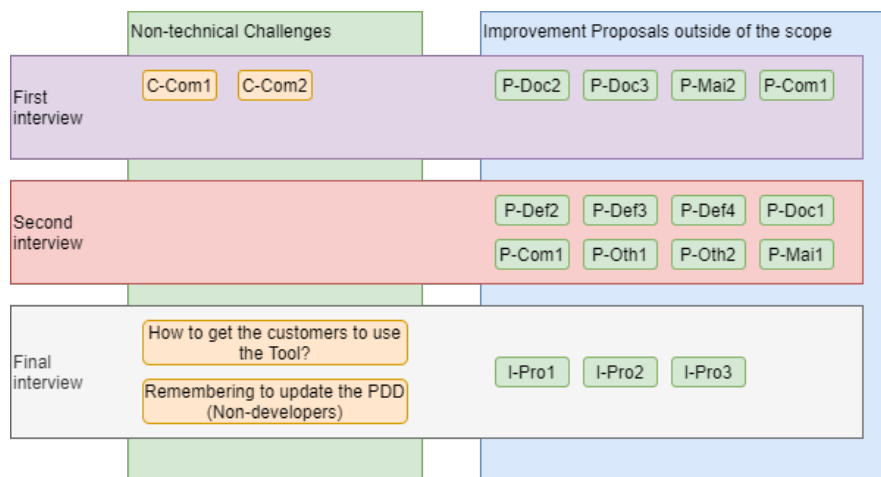


Figure 8.2: Additional functionalities for further development.

### 8.2.1 Derived Functionalities and the Tool's Requirements

To understand how the derived functionalities from the Improvement Proposals and the Challenges contributed to the Tool's requirements, a visualization of the components is presented in Figure 8.3.

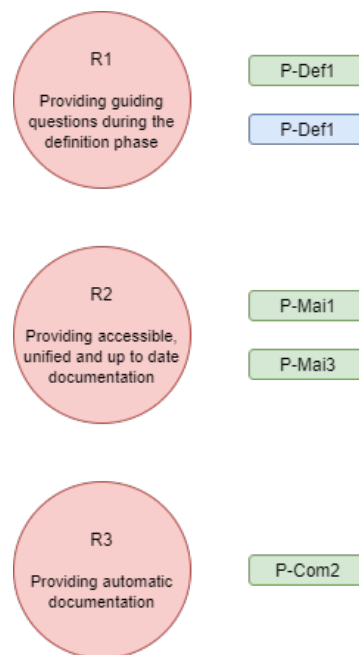


Figure 8.3: Improvement Proposals connected to the Tool's requirements.

The Improvement Proposal P-Def1 (Green) from the first interviews with the Improvement Proposal P-Def1 (Blue) from the second interview contribute to the first Tool's requirement, R1. By having a standardized procedure for the definition process (P-Def1 - Green) and a list of collectable data during the process (P-Def1 - Blue) direct the Tool's development towards having guiding questions for the process definer during the process definition phase (R1). As providing the guiding questions contribute to the Improvement Proposals' functional demands, the Tool is able to provide relief to the Challenges linked to both Improvement Proposals, which were presented in Figure 8.1.

Improvement Proposals P-Mai1 and P-Mai3 supply the second requirement R2 with func-

functionalities needed for achieving accessible, unified and up to date documentation. The standardization of documentation and the logging methodologies (P-Mai3 & P-Mai1) that are embedded into the Tool aid in generating the requirement's (R2) demanded functionalities.

Providing automatic status updates as an Improvement Proposal (P-Com2) mitigates the friction with getting the latest status of the project. This particular Improvement Proposal is connected to the R3 requirement as automatic documentation of the project, which is based on the code, generates the understanding of the developed code and understanding of how far the developer has developed the process, hence giving the status update to those who need it by reading the standardized documentation.

With this knowledge of how the gathered and analyzed data affected the decisions made regarding the functionalities of the Tool, approaching the development of the prototypes becomes clearer. The next section proceeds to unfurl the developmental methods and technologies.

### **8.3 Prototypes and development**

This section provides thorough inspection into what was developed and how it was developed. The subsections are presented in a similar structural manner based on the prototype phase. After each prototype's introduction, an analysis of the data is given based on the data that was at disposal at that particular moment.

The PDD-SDD Tool provides its users a visual user interface for creating a process definition, updating the available processes' visual flow and for understanding the process, without diving into the code, from the technical view's process flow. The interface con-



sists of two parts, one aimed for the business users: The definition view, and the second view: The technical view, is aimed for the technical users including the maintenance team. The Initial Design for the definition view is presented in Figure 5.2. Later, the technical view is presented alongside with the developed process definition view.

The Tool's requirements are presented in the Table 4.3 and the linked Improvement Proposals visualized in Figure 8.3 create the overall implementable functionalities for the Tool. This thesis focuses on Research Questions Q1 and Q3 which were presented in Table 4.4, thus the Tool's technical exploration will be limited to those scopes.

The developmental work of the Tool was divided between Abdulghani and Vuorela as mentioned in Chapter's 5 Section 5.3. And the Tool's development was divided into four main iterations, which are also presented in Figure 5.3:

1. Initial & Architecture Design
2. Prototype 1
3. Prototype 2
4. StandAlone

The Tool's name (PDD-SDD Tool) indicates that what the Tool should be able to provide for the users; A way to create a PDD and a way to automatically generate a SDD, based on the changes done to the PDD by the developers. In order to manage the Tool's developmental work, different goals were set for each prototype and for the final version, the StandAlone. The Tool was designed to have three levels of views as was depicted in Figure 5.2, which would help the process definers to add detailed information to each Phase, Step and Detail levels.

The first prototype's goal was to help with grounding the technology to be used and with the architecture design for the whole Tool. Moreover, the first prototype was defined to generate a simple code file based on the defined process in the definition view. The definition view was scoped to only have the top level of out of the three views, the Phase level of user interface and its functionalities.

The second prototype's goal was to add depth to the generated code file, as more information would be gathered by the Tool from the process definers. Additionally, front-end user experience enhancements were added to the Tool, such as adding a less technical Phase level view for the definers. The final iteration resulted in StandAlone version of the Tool, which was delivered as is to Most Digital. The StandAlone version focused on providing the technical view, the rest of the VUI levels and the rest of the predefined steps in the Initial Design presented in Figure 5.1, at the same time mitigating exceptions that were faced during the Tool's development.

### 8.3.1 The Selected Technologies

The architecture designed and generated during the first prototype is presented in Figure 8.4. The technologies selected to be used are covered in this subsection not to interrupt the prototype explanations flow below.



Figure 8.4: The PDD-SDD Tool's architecture design.

The architecture Figure 8.4 depicts the technologies implemented and how they were connected to provide the needed functionalities for the Tool. From the Figure 8.4 presented, the technologies can be inspected. The database technology was chosen to be PostgreSQL<sup>1</sup> as the powerhouse for handling the data, for its ease of use and its great support with Tool's main development language, Python<sup>2</sup> [34].

For the back-end, Django<sup>3</sup> a Python Web Framework was a perfect fit for handling the Tool's orchestration and functionalities. To be able to connect the front-end to the back-end, Django REST Framework<sup>4</sup> was chosen to complement the back-end, as Django's

<sup>1</sup><https://www.postgresql.org/>

<sup>2</sup><https://www.python.org/>

<sup>3</sup><https://www.djangoproject.com>

<sup>4</sup><https://www.django-rest-framework.org/>

REST Framework provides a clear and powerful toolkit for creating and maintaining Web APIs.

For the front-end technology, AngularJS<sup>5</sup> was the technology underneath the open-source project Behaviortrees<sup>6</sup> that was selected to be the main driver for the Tool's front-end. Essentially, Behavior trees is a tool for modeling agent behavior, such as character behavior in games or a robot's decision making algorithm [35]. Behavior trees enable a visual way for modelling complex systems [36], with this specific open-source tool with the MIT license<sup>7</sup>, prototypes and the StandAlone was developed for process definition purposes. The standard version of the Behaviortrees application is presented in Figure 8.5.

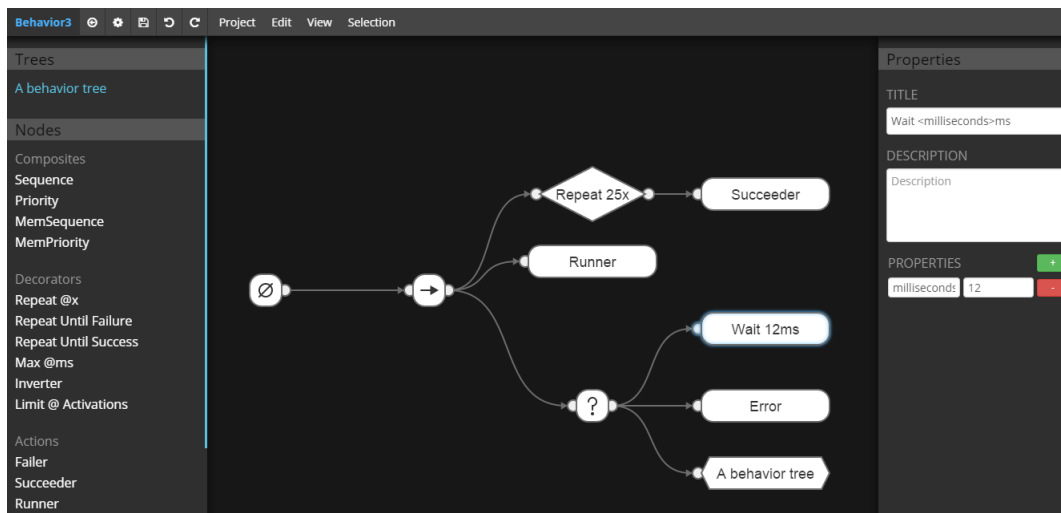


Figure 8.5: A preview of the Behaviortrees application's standard version.

<sup>5</sup><https://angularjs.org/>

<sup>6</sup><https://www.behaviortrees.com/>

<sup>7</sup><https://opensource.org/licenses/MIT>

### 8.3.2 The First Prototype

The goal of the first prototype was to implement the functionalities identified in the first round of the interviews, and the functions designed in the Initial Design (Figure 5.1). After the Initial Design was validated during the first interviews, the actual development started. The second iteration of the development roadmap (Figure 5.3) aimed to develop the first prototype, which implemented the basic functionality of the PDD-SDD Tool's Steps 1-3 (Figure 5.1), focusing only on the Phase level. The first prototype's output goal was to generate a code file from the visually defined process, in order for the developers to start developing the defined process using the generated code file as their starting point (Step 3, Figure 5.1). During the first prototype's development, a technology research was conducted for the second prototype.

The Table 4.2 visually explains the identified challenges before the first interviews, and from the same table the Tool's requirements were derived into Table 4.3, out of which the Initial Design (Figure 5.1) was generated. These three based the initial data for the Tool's features and functionalities. However, the first set of interviews added new challenges (Table 7.1) and added deeper explanations on top of the existing challenges. These Challenges and Improvement Proposals that surfaced during the first interview are presented in the Figure 8.1.

In order to achieve the first prototype's goals, the Behaviortrees' source code (AngularJS) was modified to enable the first version of definition view to be used. The plethora of functionalities were cut down to the mere necessities needed for the first prototype. The first version enabled the user to only add Phase Nodes, which were named as PhaseObjects in the first prototype, and to add a Node title and Node description. The first prototype's definition view is presented in Figure 8.6.



Figure 8.6: First prototype's definition view.

A "Submit project" button was added to enable the user to submit the process defined in the Tool, out of which the code file was generated based on the node structure, including the nodes' titles and descriptions. After submitting the defined process to the back-end, further modifications were not saved nor updated to the created code file. This was created as such by design to prevent overwriting the project's code file, which might have modifications done by the developer.

The generated code file respects the order of the nodes defined in the Tool by listing the created functions' execution order under the *main* function. Also, a *pass* command was added after each function, except the *main* function, to not raise any Python's `SyntaxError`<sup>8</sup> as there is no code to be executed inside the functions generated from the nodes. The generated code is presented in Listing 8.1.

---

<sup>8</sup><https://docs.python.org/3/library/exceptions.html#SyntaxError>

Listing 8.1: The generated code file of the first prototype.

```
1 def phase_1():
2     """
3     Node name: Sequence
4     Node description:
5     Description 1
6     """
7     pass
8
9
10 def phase_2():
11     """
12     Node name: Sequence
13     Node description:
14     Description 2
15     """
16     pass
17
18
19 def phase_3():
20     """
21     Node name: Sequence
22     Node description:
23     Description 3
24     """
25     pass
26
27
28 def main():
29     phase_1()
30     phase_2()
31     phase_3()
```

The first prototype's achieved the objectives that were designed and set for it, namely with the setup of the Tool and generating a simple code file from the process defined in the visual user interface (VUI). Additionally, the first prototype took partly on the first Research Question Q1 (Table 4.4.) by aiding in the process definition and project kick-off by generating the code file ready to be modified and further developed by the developers. However, the first prototype did not accomplish any requirements set out for the Tool, nor it did fully answer any of the Research Questions fully. But these were not the objectives that were set for the first prototype. The time invested in the first prototype including the design, technology research and the development was around 90 hours per developer (Abdulghani and Vuorela).

### **8.3.3 The Second Prototype**

The second prototype's implementation started with the third iteration of the development roadmap (Figure 5.3), which adds the PDD-SDD Tools' functionalities from Steps 3-5 (Figure 5.1) to the developed Tool. Separate views for technical and business level users were added. Second prototype was validated with second round of interviews. Furthermore, a technology research was done for the final iteration, in order to implement the remaining Step and Detail levels in VUI.

The objectives set for the second prototype included having a new definition view which was not based on the nodes, as it was in the first prototype. This is linked to the Improvement Proposal of having a standard definition process (P-Def1), which is associated to the Tool's R1 requirement, as presented in Figure 8.3. Moreover, the second prototype was set to implement P-Mai3 and P-Com2, which were associated to the Tool's R2 and R3 requirements respectively, thus closes the gap to the Research Questions 1-3, presented in Figure 4.4.



The second prototype would also accomplish the whole cycle of the Tool, all five Steps, where the code file is consumed and parsed by the Tool, and it would generate the process flow based on the functionalities of the process. Initially the second prototype would resemble the final product in a Minimum Viable Product (MVP) fashion, the whole cycle mentioned is presented in the Initial Design, Figure 5.1.

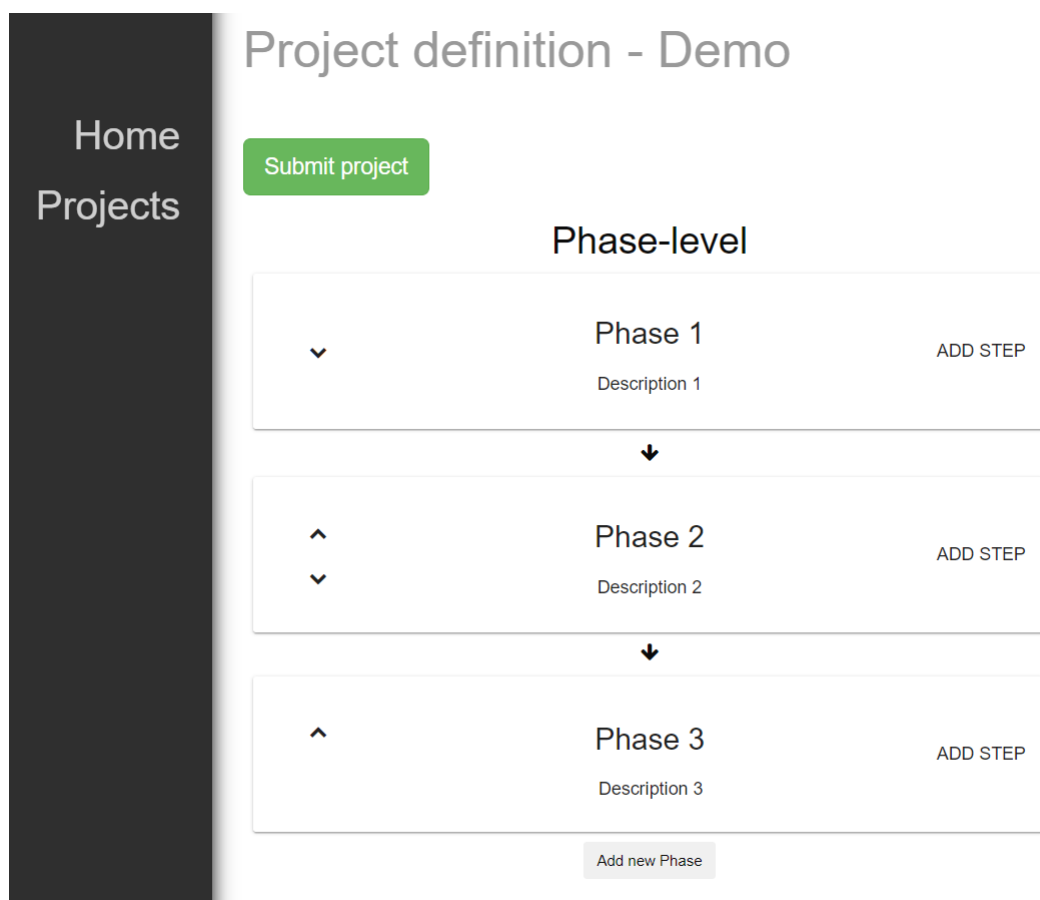
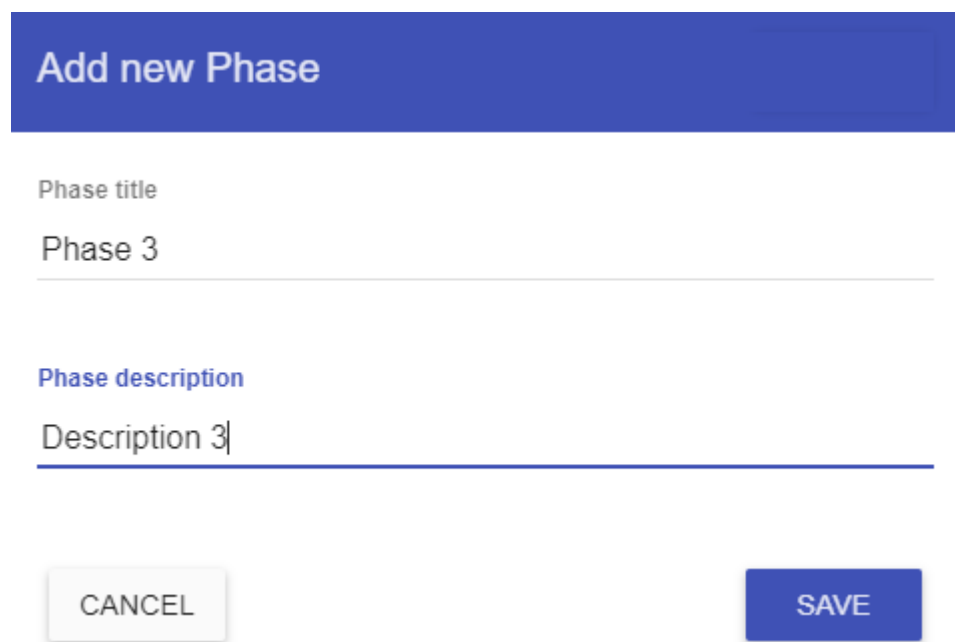


Figure 8.7: Second prototype's definition view.

The new definition view, which was designed to look and feel less technical compared to the first prototype's definition view, is illustrated in Figure 8.7. The definition view got implemented with the new color palette that was designed for the whole Tool, minimal aesthetic, where the dark sidebar is for navigating through different projects and the white center layout would help steer the focus of the process definer.

A top-down block approach for the process definition was deemed clearer than having nodes that could be moved anywhere in the canvas. Also, the rearrangement buttons were implemented to the side of the blocks. Adding new Phase blocks was implemented to exist at the end of the process, where naturally the process definer would go to, to add the next Phase block into the process. By clicking the "Add new Phase" button, a pop-up window appears for data entry, as show in Figure 8.8. Additionally, placeholder buttons for the next level, the Step level, was added to the right side of the Phase block.



**Add new Phase**

Phase title  
Phase 3

Phase description  
Description 3

CANCEL SAVE

Figure 8.8: Adding a new Phase block, pop-up for data entry.

In the back-end the code parser was developed further, being able to handle the rearrangements of the Phase level blocks. and to also generate a coherent data set that the Tool could parse back into the definition and technical view. For being able to parse each level's blocks, tags were utilized in the code file to help identify the different levels, and to distinguish normal functions from the Phase, Step, Detail level functions. The generated code file is presented in Listing 8.2.

Listing 8.2: The generated code file of the second prototype.

```
1 def phase_1():
2     """
3     TITLE: Phase 1
4     DESCRIPTION:
5     Description 1
6     @PHASE
7     """
8     pass
9
10 def phase_2():
11     """
12     TITLE: Phase 2
13     DESCRIPTION:
14     Description 2
15     @PHASE
16     """
17     pass
18
19 def phase_3():
20     """
21     TITLE: Phase 3
22     DESCRIPTION:
23     Description 3
24     @PHASE
25     """
26     pass
27
28 def main():
29     phase_1()
30     phase_2()
31     phase_3()
```

When the defined process is submitted and the code file is generated, the definition view removes the submit button and adds a new button, "Fetch latest status", which enables the user to fetch the latest changes to be visualized in the definition view, as is depicted in Figure 8.9.

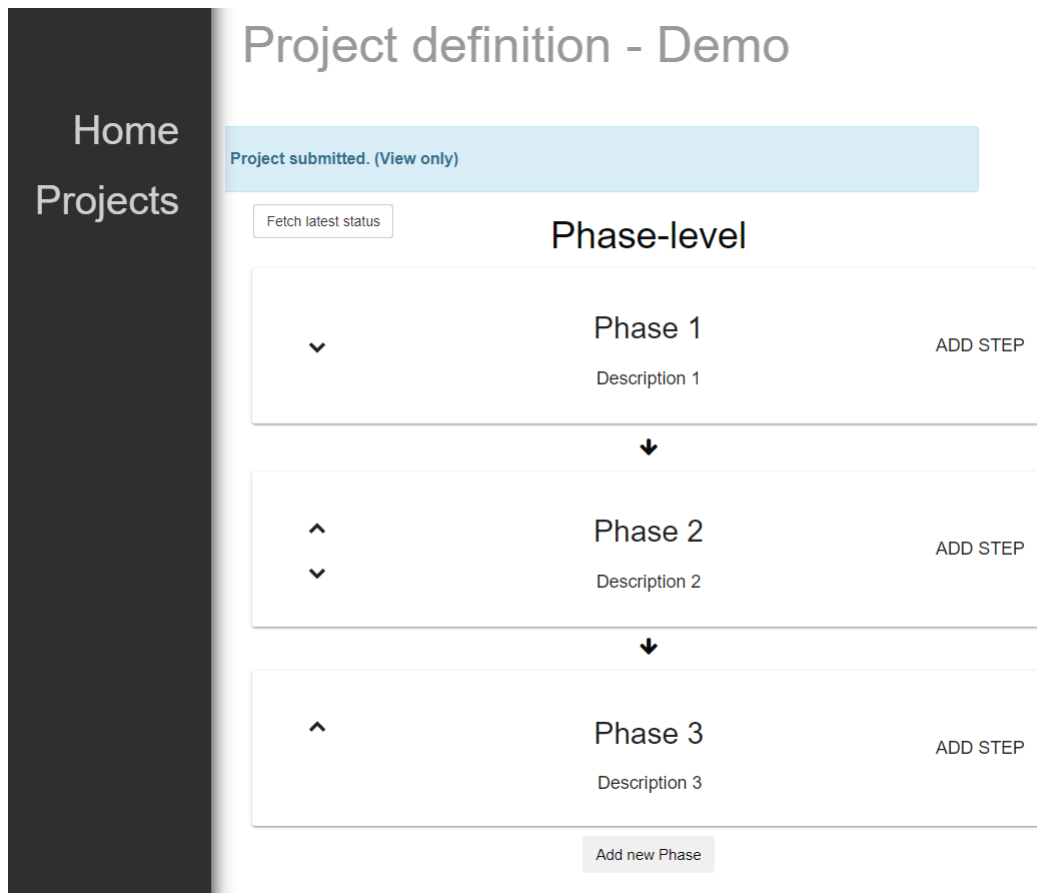


Figure 8.9: Fetch button appears after submitting the defined process.

The second prototype achieved the objectives that were designed and it implemented a better definition view and more functionalities in the back-end, which got the Tool closer to the StandAlone version. Additionally, the second prototype implemented the whole cycle (Steps 1-5) that was in the Initial Design 5.1. The Tool's requirement R3 was fulfilled in the second prototype, which provides automatic documentation based on the project's code file, as it is fetched. The time invested in the second prototype including the design, technology research and the development was around 75 hours per developer.

## 8.4 Standalone tool

The final development roadmap (Figure 5.3) iteration finalized the aspects implemented in previous prototypes, creating the first overall solution, a standalone version, for PDD-SDD Tool. Final interviews were conducted and research questions addressed.

The StandAlone version of the Tool was designed to accomplish the Initial Design 5.1 and fulfill the requirements set for the Tool, which are presented in Table 4.3, within the defined time and project scope. The StandAlone version completes the development iterations, depicted in Figure 5.3, and unlocks the Final Interviews, which is the last piece of the defined scope.

The final version updated the definition view, added the technical view, implemented the additional Step and Detail -levels, polished the generated code file and completed the guiding questions in the definition view. These were also the objectives set for the final version of the Tool, the StandAlone version. Also, based on the Final Interview with the professionals, the Tool's requirements are met and also the Thesis Research Questions are successfully accomplished. The objectives and the different views are presented and discussed in the next subsections.

### 8.4.1 The Definition View

The definition view is activated by choosing the user's role, as depicted in Figure 8.10. This affects the project listing showed in the StandAlone's home page, as is illustrated in Figure 8.11.

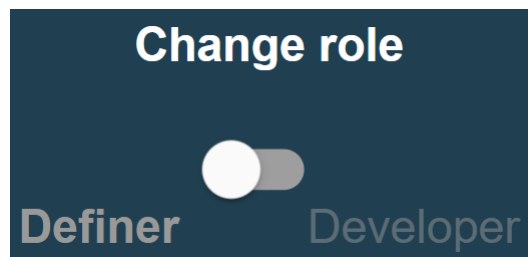


Figure 8.10: The toggle of roles. Process Definer's view activated.

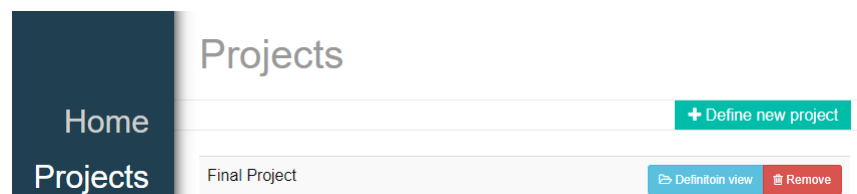
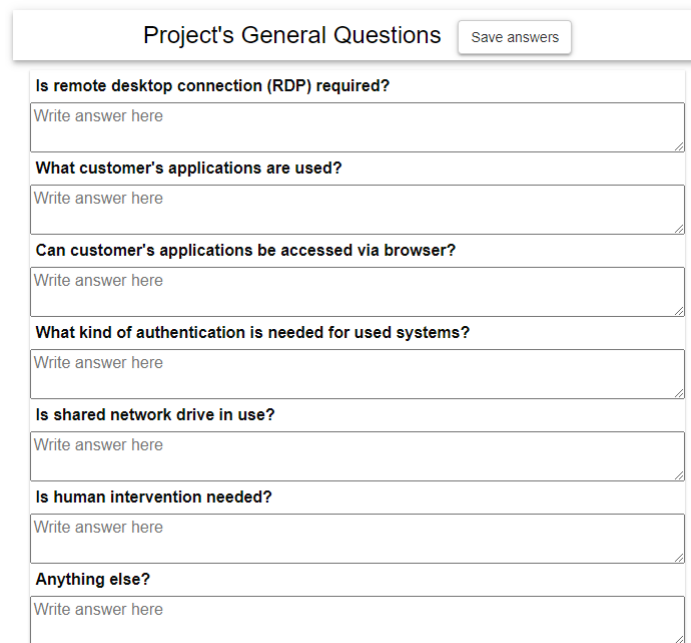


Figure 8.11: Projects listing view with the Definer's view activated.

The first interview's Improvement Proposal "Standard definition process" (P-Def1, Table 7.2) and the second interview's Improvement Proposal "List of collectable data during the process" (P-Def1, Table 7.3) are both linked to the first Requirement "Providing guiding questions during the definition phase" (R1, Figure 8.3). Both of these Improvement Proposals are implemented in the definition view.

In order to standardize the definition process, guiding questions were implemented and were made as a requirement for completing the process definition, Figure 8.12. The guiding questions are easily modifiable to suit different types of projects. Also, listing the collectable data was implemented in the Details level, which helps with documenting the

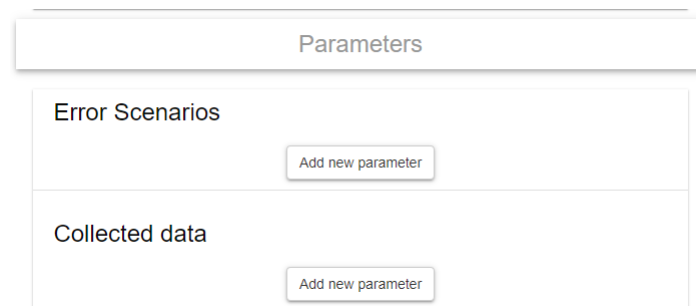


The screenshot shows a web form titled "Project's General Questions" with a "Save answers" button. The form contains seven questions, each followed by a text input field with the placeholder "Write answer here":

- Is remote desktop connection (RDP) required?
- What customer's applications are used?
- Can customer's applications be accessed via browser?
- What kind of authentication is needed for used systems?
- Is shared network drive in use?
- Is human intervention needed?
- Anything else?

Figure 8.12: The guiding questions for process definition

needed data to be collected during the process, as presented in Figure 8.13.



The screenshot shows a web form titled "Parameters". It has two main sections:

- Error Scenarios**: Contains an "Add new parameter" button.
- Collected data**: Contains an "Add new parameter" button.

Figure 8.13: Data collection and exceptions listing.

The StandAlone version's complete definition view with the updated color palette and guiding questions are presented in Figure 8.14. The Phase-level is the default level that is show first in the definition view. The other levels are hidden to reduce needless complexity during the process definition but the Step and Details -levels are easily accessible from the Phase-level's "View Steps" and "View Details" buttons.

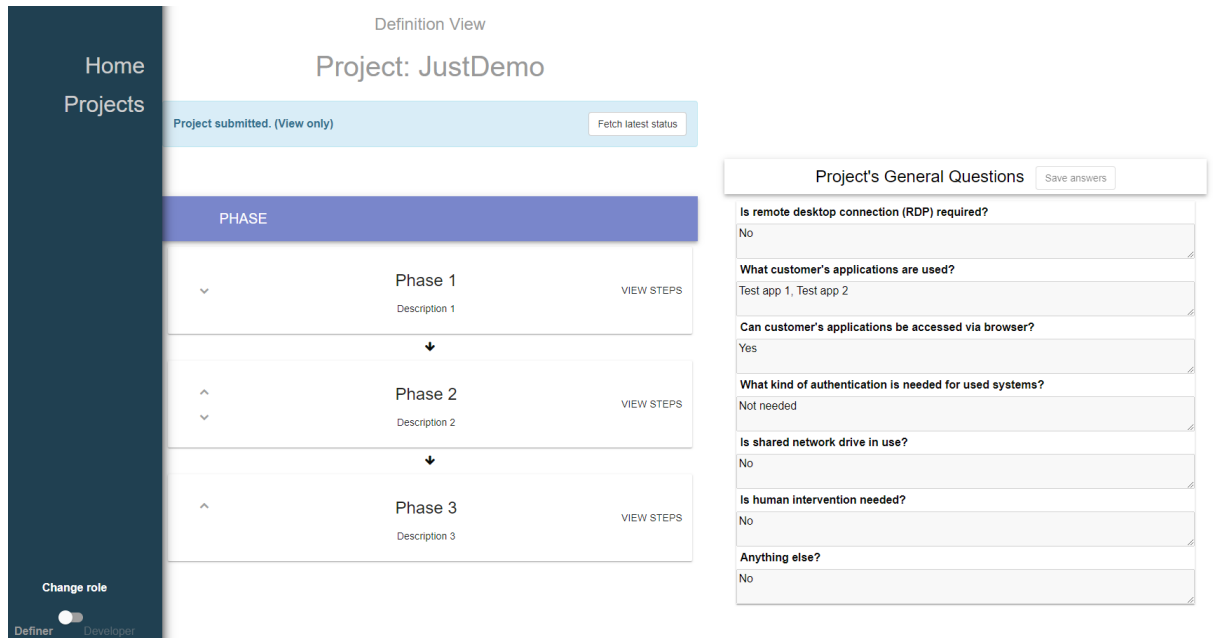


Figure 8.14: The Definition view.

The definition view incorporates all of the levels depicted in the initial VUI design, Figure 5.2. The Step-level respects the same visual design as the Phase-level did. From development's point of view, having similar views and control mechanisms aided in reusing the modularized components and functionalities used to implement the Phase-level. The Step-level is depicted in Figure 8.15.

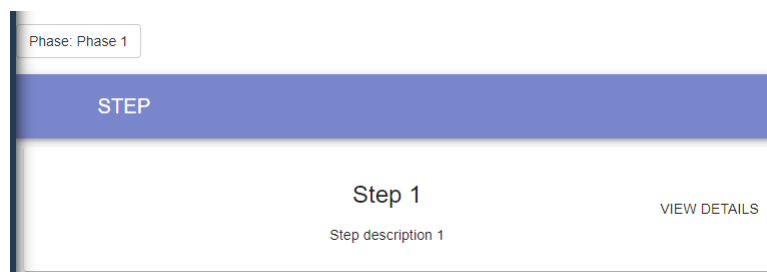


Figure 8.15: The Step-level view.

The final level, the Details-level is similarly hidden behind the "View Details" button,



where the minute details of each Step should be included. This is designed in a fashion that it would not distract the definer nor the definition process. The details-level is shown in Figure 8.16. The Details-level uses a similar panel implementation as the guiding questions (Figure 8.12). This solution aids in data collection, as the Details-level is only for collecting more in depth data about that specific Step-level that it is attached to, thus it was a better solution overall.

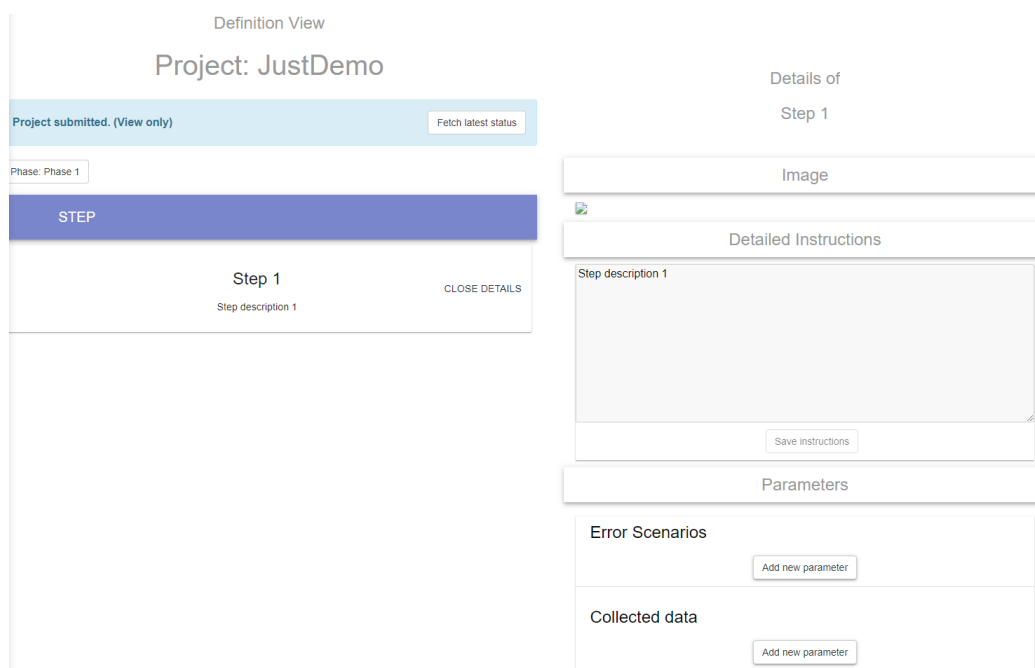


Figure 8.16: The Details-level view next to the Step-level view.

The definition view contains the Phase-level, the Step-level and the Details-level for process definition (Figure 8.14). The definition view also contains the guiding questions that are required for submitting the defined process (Figure 8.12). There are two different implementations used in the StandAlone's definition view: The first one is the middle panel, which embodies the Phase-level and Step-level views and implementations (Figure 8.15). The second is the right-side panel used for guiding questions and Details-level (Figures 8.12 and 8.13). The views and implementations were utilized as they were implemented in a modular fashion. Another view, which contains the projects is the Project listing

view, shown in Figure 8.11.

## 8.4.2 The Technical View

The technical view is activated by toggling the user's role to "Developer", as it is illustrated in Figure 8.17. By activating this role, a new button appears in the project listing, visible in Figure 8.18. The main idea of the technical view is to enable the maintenance team to inspect the project without diving into the code itself. This helps with accessing up to date documentation, as the latest code changes are projected into the technical view. This fulfills the second requirement of the Tool, R2, presented in Figure 8.3.

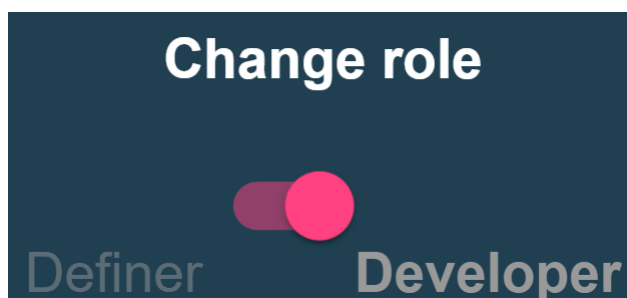


Figure 8.17: The toggle of roles. Developer's view activated.

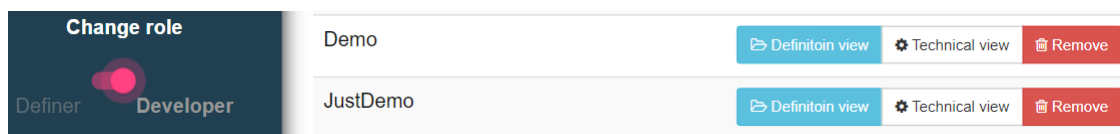


Figure 8.18: The activated technical view's button in project listing.

The connected Improvement Proposals to the second requirement (P-Mai1 and P-Mai3, Table 7.2) are implemented in the back-end and are shown in the technical view, which is based on the developers utilizing the logging and documentation conventions set by the PDD-SDD Tool. As the process definition is done in the StandAlone version of the Tool, a code file is generated, as was designed in the Initial Design (Figure 5.1). The generated

code file contains everything that was inputted into the definition view in a coherent manner, as show in Listing 8.3, under the correct tags, which are marked with a @-sign.

Listing 8.3: The generated code file of the StandAlone version. Phase 1 and Step 1.

```
1 def phase_1 () :
2     """
3     @TITLE: Phase 1
4     @DESCRIPTION:
5     Description 1
6     @PHASE
7     """
8     step_1 ()
9
10 def step_1 () :
11     """
12     @TITLE: Step 1
13     @DESCRIPTION:
14     Step description 1
15     @INSTRUCTIONS:
16     Step description 1
17     @PICTURE:
18     None
19     @ERRORS:
20     ----
21     @DATA_COLLATION:
22     ----
23     @STEP
24     """
25     pass
```

The guiding questions are left in the definition view. The Details-level's data is part of the Step-level's automatically generated function documentation, or in Python's case *docstring*<sup>9</sup>. The Tool creates the foundation for the developers to utilize the docstrings for adding their own documentation and other development related notes which tackles the Improvement Proposals P-Mai1 and P-Mai3, which were connected to the second development requirement set for the StandAlone Tool (Figure 8.3).

The technical view, which is generated from the parsed code file, is visible after clicking the "Technical view" button from the project listing, after activating the "Developer" toggle from the user's role, as depicted in Figure 8.18. Submitting the defined process generates not only the code file, but also the technical view's flow of the process, using the parsed code naturally. This initial flow is presented in Figure 8.19.

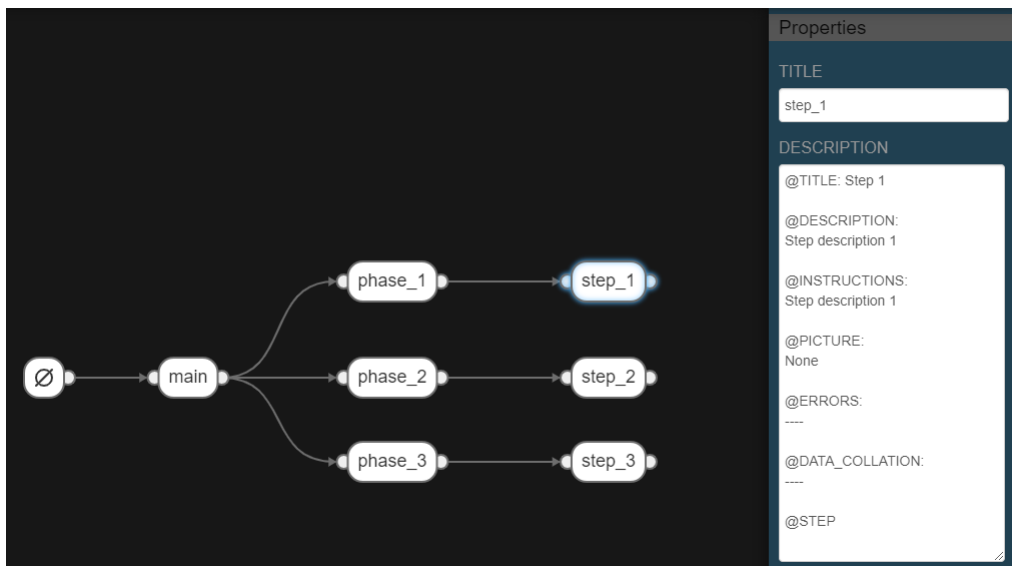


Figure 8.19: The flow of the defined process.

The technical view's flow visualization uses node representation of the defined levels (Phase and Step). And by clicking on a node in the technical view, node properties are

<sup>9</sup><https://www.python.org/dev/peps/pep-0257/#what-is-a-docstring>

shown. The node properties are the same that are visible in the generated code as docstring, hence automatic documentation, the third requirement R3 (Table 4.3) is achieved. This also tackles the Improvement Proposal P-Com2 which is connected to the R3 development requirement, as seen in Figure 8.3. Also, status updates are automatically updated to the visual representation of the code, the technical view's flow. The definition view is updated based on the developed code file, generating an updated definition view of the process.

A representation of how the updated code file (Listing 8.4) by the developer updates automatically the definition view (Figure 8.20) and the technical view (Figure 8.21). The code file from the example presented is focused only on the *main* function, as listing the whole code file would span multiple pages.

Listing 8.4: The main function of the updated code file of the defined process.

```
1 def main () :  
2     open_rdp ()  
3     open_sap ()  
4     searching_unrecieved_name_rows_from_sap ()  
5     suppliers_product_code_listing_in_excel ()  
6     data_unification_product_code_with_delivery_report ()  
7     send_emails_script ()  
8     sending_reminders ()  
9     end_procedures ()
```

PHASE		
v	<b>Open RDP</b> <small>Sign in to customer RDP</small>	ADD STEPS
↓		
^	<b>Open SAP</b> <small>Open SAP Application</small>	VIEW STEPS
↓		
^	<b>Supplier's product code listing in Excel</b> <small>List Supplier product code to excel file</small>	VIEW STEPS
↓		
^	<b>Data unification product code with delivery report</b> <small>Unify the data of product code with delivery report</small>	VIEW STEPS
↓		
^	<b>Send emails</b> <small>Send emails using scripts and SMPT</small>	VIEW STEPS
↓		
^	<b>Sending reminders</b> <small>Send emails for reminders</small>	VIEW STEPS
↓		
^	<b>End procedures</b> <small>Close the automation</small>	VIEW STEPS

Project's General Questions Save answers

**Is remote desktop connection (RDP) required?**  
Yes

**What customer's applications are used?**  
Excel, Outlook, Chrome, SAP

**Can customer's applications be accessed via browser?**  
Not all

**What kind of authentication is needed for used systems?**  
RPD & Application

**Is shared network drive in use?**  
Yes: \\123.1.12.108\SharedDrive

**Is human intervention needed?**  
No

**Anything else?**  
No

Figure 8.20: The updated definition view based on the updated code file.

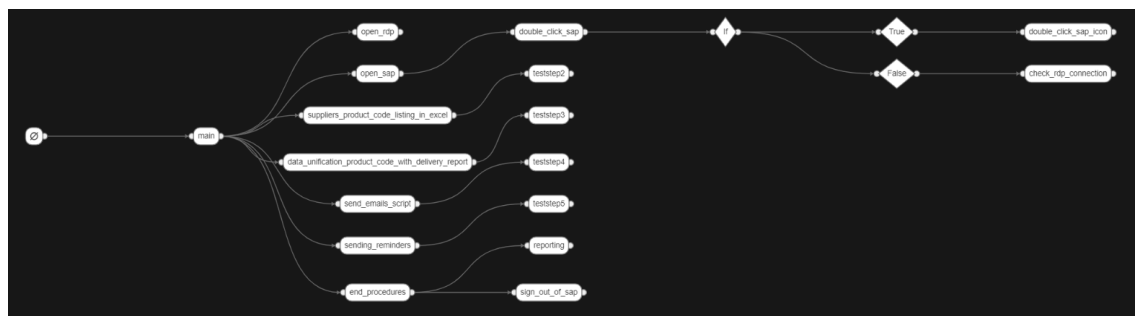


Figure 8.21: The updated technical view based on the updated code file.

## 8.5 Tool's Process and Validations

The StandAlone achieved the objectives that were agreed upon. The final version updated the front-end's color palette, implemented the fully functional code file parser in the back-end, completed the different definition levels and enabled the technical view to dive into the minute code functionalities and represent them in a process flow. Moreover, the StandAlone also added stability and exceptions handling. Additionally, the StandAlone fulfilled development requirements R1 and R3, the second prototype already fulfilled requirement R2, thus the final version of the Tool completes all of the development requirements designed. The time invested in finalizing the StandAlone was around 85 hours per developer, which added to the previous versions, the first and second prototypes, accumulates to approximately 250 hours in total per developer.

Finalizing the project, final interviews were held to demonstrate the StandAlone Tool and to gather impressions and validate the Research Questions presented in Table 4.4. The structure and the findings of the interviews help for the StandAlone version are discussed in Section 7.3, which is part of Chapter 7.

The StandAlone Tool works in a concise process, which is:

1. Process Definition, including project submitting.
  - (a) Phase-level
  - (b) Step-level
  - (c) Details-level
2. Generating code file from the defined process.
3. Back-end saving the code file into the PostgreSQL database utilizing the APIs.
4. Creating the process flow in the technical view.

The Tool is in stand-by mode waiting for the code file to be updated, which causes the Tool to:

1. Parse the code file and save the the updated version to PostgreSQL database.
2. Update the definition view, based on the updated code.
3. Update the technical view, based on the updated code.

Based on the functionalities successfully implemented and based on the feedback received from the interviewees, the Research Questions Q1-Q3, which are presented in Table 4.4, are generously validated by the interviewees and by the thesis researchers.



## 9 Conclusions

The implemented PDD-SDD Tool with three versions, the first and the second prototype, and the final version, the StandAlone, is remarked as a success by the professionals interviewed from both companies, Most Digital Ltd. and Ailea Ltd. With three interview rounds guiding and impacting the Tool's development, concrete functionalities were implemented which not only significantly affected the RPA projects' life-cycles, but also the professional's productivity when compared to the previous working methods of RPA projects. Without the developed Tool, the projects were decentralized from the project management's perspective, having multiple tools and applications to aid in controlling and maintaining the projects. Even though the PDD-SDD Tool helps with multiple aspects, there are challenges that are out of technological reach.

The Challenges related to RPA projects, which were listed in Table 4.2, are mostly solvable by methodologies and rules set by the company. Nonetheless, urgency and strict deadlines push professionals to cut corners, thus causing a ripple effect that impact the project in multiple areas. With the PDD-SDD Tool's guiding questions, automatic project code base generation and automatic documentation updates help the professionals to focus on their main work in RPA projects in their departments. The Tool's impact can also be measured against the broader RPA development world, where similar challenges arise on a daily bases. The same top-level challenges are caused by the lack of conventions related to the RPA development, as it is a quite new industry. The developed PDD-SDD

Tool helps with creating these conventions on a company basis, thus having similar impact as it did with Most Digital Ltd.

The Research Questions of this thesis were (Table 4.4):

- Q1 - Does the tool ease the definition, project kick-off and maintenance of RPA projects?
- Q3 - Does the tool ease communication and documentation between stakeholders in RPA projects?

and the Tool's Requirements (Table 4.3):

- R1 - Providing guiding questions during the definition phase.
- R2 - Providing accessible, unified and up to date documentation.
- R3 - Providing automatic documentation.

aided in focusing on the most important aspects of the Tool's development. The feedback and direct validations of the Research Questions and Tool's Requirements, presented in Chapter 7, provide the confidence to announce the project as a success and the Tool as valuable. The functionalities presented in Chapter 8 resolve the Requirements set for the Tool, and through the demonstrations to the target group and their validations, the research's Research Questions are validated.

By providing the guiding questions during the definition phase (R1) and accessible, unified, up to date and automatic documentation (R2 and R3), as presented in Subsections "Definition View" (8.4.1) and "Technical View" (8.4.2), the Research Questions Q1 and Q3 are answered positively and regarded as validated. Although, the Research Questions Q1 and Q3 are specific to this Thesis and to RPA development, they do ask the central question of how the human impact in any IT based projects can be boosted, thus it is safe

to assume that the Tool's impact can be, with modifications, used in different kinds of IT projects to aid with similar challenges. The Tool can be further developed by implementing the additional functionalities that were gathered from the interviews, which are presented in Figure 8.2.

Starting with the current state of the RPA world (Chapter 2), explaining the development methodologies with examples (Chapter 3), identifying the current challenges and bottlenecks of the RPA development framework (Chapter 4), brought the research to constitute the Initial Design (Figure 5.1) in Chapter 5, which would start the journey of developing the Tool to aid the professionals in their daily work.

The constant change in the IT field and the increasing demand for RPA solutions, communication and documentation are more important than ever before. With tools developed closely with the people who use them daily, we are **"Connecting RPA Development and Business"** for a better tomorrow.

# References

- [1] I. Jacobson. What they dont teach you about software at school: Be smart! In P. Abrahamsson, M. Marchesi, and F. Maurer, editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 1–4, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [2] B. Hailpern and P. Santhanam. Software debugging, testing, and verification. *IBM Systems Journal*, 41(1):4–12, 2002.
- [3] A. Abdulghani and S. Vuorela. Work experience from software and RPA development, 2016-2021.
- [4] S. Moore. Gartner Says Worldwide Robotic Process Automation Software Market Grew 63% in 2018. <https://www.gartner.com/en/newsroom/press-releases/2019-06-24-gartner-says-worldwide-robotic-process-automation-sof> (visited on 10.2.2020), 06 2019. Gartner Press Release.
- [5] W. Van der Aalst, M. Bichler, and A. Heinzl. Robotic Process Automation. *Business & Information Systems Engineering*, 60(4):269–272, Aug 2018.
- [6] L. Willcocks, M. Lacity, and C. Andrew. The it function and robotic process automation. *The Outsourcing Unit Working Research Paper Series*, (15/05):1–39, 10 2015.

- 
- [7] Automation Anywhere. The Complete Starter Guide to RPA. <https://www.automationanywhere.com/images/guides/rpa-starter-guide.pdf> (visited on 9.5.2020), Automation Anywhere.
- [8] N. Ostdick. The Evolution of Robotic Process Automation (RPA): Past, Present, and Future. <https://www.uipath.com/blog/the-evolution-of-rpa-past-present-and-future> (visited on 9.5.2020), UiPath.
- [9] C. Saxena S. Gupta, S. Kumar. Review Paper on Comparison of Automation Testing Tools Selenium and QTP. *MIT International Journal of Computer Science and Information Technology*, 5:55–57, 08 2015.
- [10] C. Le Clair. The Forrester Wave™: Robotic Process Automation, Q2 2018. Forrester Research, Inc., 06 2018.
- [11] O. Ylönen. Blue Prism- ja UiPath-vertailu ohjelmistorobotiikassa. Thesis, Metropolia University of Applied Sciences, 2018.
- [12] S. Helmers. *Microsoft Visio 2016 Step By Step: MS Visio 2016 Ste by Ste\_p1*. Step by Step. Pearson Education, 2015.
- [13] G. James. *Citrix XenDesktop Implementation: A Practical Guide for IT Professionals*. Elsevier Science, 2010.
- [14] L. Willcocks. Why Robots May Not Be Taking Your Job – at least, not in the next 10 years. <https://www.europeanbusinessreview.com/how-organisations-can-embrace-automation/> (visited on 9.5.2020), European Business Review, 2016.
- [15] N. Weiderman, J. Bergey, D. Smith, and S. Tilley. Approaches to Legacy System Evolution. Software Engineering Institute, 12 1997. Carnegie Mellon University, Pittsburgh.

- [16] W. van der Aalst, A. ter Hofstede, and M. Weske. Business Process Management: A Survey. In *Business Process Management*, pages 1–12, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [17] I. Chen and K. Popovich. Understanding customer relationship management (CRM): People, process and technology. *Business Process Management Journal*, 9:672–688, 01 2003.
- [18] R. Atkinson. Project management: Cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management*, 17(6):337–342, 12 1999.
- [19] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, J. Stoica, and M. Zaharia. A View of Cloud Computing. *Commun. ACM*, 53:50–58, 04 2010.
- [20] F. Xia, L. Yang, L. Wang, and A. Vinel. Internet of Things. *International Journal of Communication Systems*, 25, 09 2012.
- [21] I. Luukkonen, J. Mykkänen, T. Itälä, S. Savolainen, and M. Tamminen. *Toiminnan ja prosessien mallintaminen: Tasot, näkökulmat ja esimerkit*. Solea Project, University of Eastern Finland and Aalto University, 01 2012.
- [22] R. Daft and R. Lengel. Organizational information requirements, media richness and structural design. *Management Science*, 32:554–571, 05 1986.
- [23] C. Seaman and V. Basili. Communication and organization in software development: An empirical study. *IBM Systems Journal*, 36(4):550–563, 1997.
- [24] B. Ramesh, L. Cao, K. Mohan, and P. Xu. Can distributed software development be agile? *Commun. ACM*, 49:41–46, 10 2006.

- 
- [25] R. Hoda, J. Noble, and S. Marshall. Documentation strategies on agile software development projects. *International Journal of Agile and Extreme Software Development*, 1(1):23, 2012.
- [26] D. Spinellis. Code Documentation. *IEEE Software*, 27(4):18–19, 2010.
- [27] G. Myers, C. Sandler, and T. Badgett. *The Art of Software Testing*. ITPro collection. Wiley, 2011.
- [28] T. Heinonen. Ohjelmistorobotiikan testaus. Master’s thesis, University of Turku, 2019.
- [29] G. Gill and C. Kemerer. Cyclomatic complexity density and software maintenance productivity. *IEEE Transactions on Software Engineering*, 17(12):1284–1288, 1991.
- [30] E. Dijkstra. Notes on structured programming, 04 1969. Technological University Eindhoven.
- [31] W. McKinney. *Python for Data Analysis*. O’Reilly Media, Inc., 2 edition, 10 2017.
- [32] J. Joseph, S. Hissam, B. Fitzgerald, and K. Lakhani. Collaboration, conflict and control: the 4th workshop on open source software engineering. In *Proceedings. 26th International Conference on Software Engineering*, pages 764–765, 2004.
- [33] A. Correia, F. Brito e Abreu, and V. Amaral. SLALOM: a Language for SLA specification and monitoring. *ArXiv*, abs/1109.6740, 2011.
- [34] K. Douglas and S. Douglas. *PostgreSQL: A Comprehensive Guide to Building, Programming, and Administering PostgreSQL Databases*. Developer’s library. Sams, 2003.
- [35] M. Colledanchise and P. Ögren. Behavior trees in robotics and ai. Jul 2018.

- 
- [36] A.Marzinotto, M. Colledanchise, C. Smith, and P. Ogren. Towards a unified behavior trees framework for robot control. pages 5420–5427, 05 2014.