Utrecht University

Master Thesis Business Informatics

Aligning requirements and testing in RPA projects

Designing an RPA verification and validation method

Mees Mouwen
Department of Information and Computing Sciences

April 26, 2023

Supervisors:

Inge van de Weerd	Utrecht University, Associate Professor
Hajo Reijers	Utrecht University, Professor
Bo van den Oever	Ciphix, RPA team-lead

Utrecht University

CIPHIX

Abstract

Context: Robotic process automation (RPA) is a developing field of process automation, but academic research has not kept up with industry advancements creating an absence of a theoretical foundation for objective reasoning. There is a lack of research regarding the alignment between requirement management and testing practices within the RPA development life cycle (DLC).
Objectives: The objectives of the study are to investigate the current state of requirement management and testing practices in RPA, identify their limitations, and propose a novel RPA DLC method that can address the link between requirement management and testing for more effective RPA development.
Methods: To develop and evaluate the method, a design science study approach is employed, which integrates both theoretical and practical input. The theoretical foundation of the study is established by conducting a comprehensive multi-vocal literature review (MLR). The practical component of the research methodology entails utilizing problem-centred interviews, a multiple-case study, and a naturalistic project evaluation.
Results: The RPA verification and validation method (RPA-VV method) was designed as a result of the MLR and a multiple-case study. The RPA-VV method is based on the W-model and provides an effective way of integrating requirement management and testing practices in RPA projects, which can help provide guidance in during the RPA DLC and improve communication and collaboration between stakeholders. The validation results were promising with the method being deemed complete and very useful, with clear visualisation of the RPA DLC. However, the evaluation showed that executing the RPA-VV method in practice was difficult, causing problems with adherence to the method. Two main issues were identified which were difficult communication with the client about their input and deliverables for the project, and strict project planning. Despite providing clear guidelines and activities, deviations from the method occurred due to these issues.
Conclusion: The RPA-VV method is a promising approach to align requirements and testing activities in the RPA DLC, but integrating it into RPA projects is difficult. Future work should focus on how to improve and properly integrate the RPA-VV method in RPA projects to overcome the identified issues.

Keywords: Robotic process automation, development life cycle, testing, requirement management, W-model, RPA-VV method

Acknowledgement

This thesis marks the completion of my Master's degree, and I would like to express my gratitude to those who supported me throughout the journey. I am thankful to Ciphix, particularly Bo van den Oever, for the continuous support and valuable feedback on my thesis. I would also like to acknowledge Inge van de Weerd for her guidance, weekly meetings, and invaluable academic insights throughout the eight-month research period. Finally, a special thanks to my girlfriend for her unwavering support and encouragement during the long study sessions and for sharing this journey with me.

Contents

Abbreviations

DLC	Development life cycle
E2E	End-to-end
MBT	Model-based testing
MLR	Multi-vocal literature study
NFRs	Non-functional requirements
PCI	Problem-centred interview
PDD	Process definition document
RM	Requirement management
RPA	Robotic process automation
RT	Requirement traceability
SA	Solution architect
SME	Subject matter expert
SDD	Solution design document
TDD	Test-driven development
TPD	Test plan document
UAT	User acceptance test
RPA VV-method	RPA verification & validation method

Introduction & Research design

1. Introduction

Robotic Process Automation (RPA) is a novel field in process management and plays a leading role in digital transformation for companies looking to automate or optimize their processes (Syed et al., 2020). RPA projects involve building software agents or robots that can automate repetitive and rule-based tasks of a process, mimicking human interactions with digital systems through the UI or front-end of applications (Hofmann et al., 2020; Willcocks et al., 2015). It is anticipated that RPA will be adopted by organizations across different industries in the near future (Syed et al., 2020). This is due to the numerous benefits that RPA offers to businesses, such as the non-stop operation of robots, which makes them very efficient in their tasks, thereby saving time and cost (Syed et al., 2020). Furthermore, the quality of the process can be improved since human error or missteps, for example, in data inputs are removed. However, as RPA is still a new domain in automation, further research and improvement are necessary (Cernat et al., 2020; Syed et al., 2020). Currently, RPA employs techniques from generic software development, but their applicability for RPA has not always been extensively investigated (Cernat et al., 2020). In the following paragraph, we discuss two aspects of generic software development, namely requirement management and testing, and their usefulness in RPA development.

Requirement management (RM) is the transformation of customer needs into software specifications (Jiao & Chen, 2006). These needs need to be defined in a structured format. Accurate requirements are important for a satisfactory solution since it forms the basis for the further design, development, and testing of the application (Jiao & Chen, 2006; Song, 2017). RM is important to understand the scope and target of the project. Extensive studies have shown the relevance of careful assessment of requirements for the success of a project (Jiao & Chen, 2006). Often, a poor understanding of requirements and inaccurate assumptions have a big negative impact on design, development, and project time and cost (Jiao & Chen, 2006; Skoković & Rakić-Skoković, 2010).

Testing is an essential aspect of software development that validates system functionality and supports quality assurance (Mařík et al., 2000; Sawant et al., 2012). Its aim is to uncover defects, but it can only show their presence, not their absence (Ratilainen et al., 2019). Therefore, adequate time and effort should be dedicated to testing to uncover critical errors. Properly planned testing can significantly enhance product quality, while poorly executed testing can harm the product (Ratilainen et al., 2019). According to Mařík et al. (2000), testing is crucial to validate different phases of a development project and should be integrated within the various stages such as analysis, design, and development.

As shown in the previous paragraph, both RM and testing are crucial for the development of any project and have a big impact on the different phases of the development life cycle (DLC). However, RPA still differs from normal software projects (Cernat et al., 2020), and thus how these topics are performed or conducted in the RPA DLC is crucial for the maturity of the RPA domain.

1.1 Problem statement

The potential of the RPA field is significant, but its development lacks sound theoretical foundations for objective reasoning (Syed et al., 2020). This creates a need for additional research in several areas of RPA development. In terms of RM and testing in RPA, there are some observations to be made. RM in RPA involves a thorough analysis of the process to be automated, including all process steps and potential exceptions (Soybir & Schmidt, 2021). Mistakes in process analysis can lead to difficulties in other parts of the RPA DLC, making a strong understanding of essential requirements crucial (Soybir & Schmidt, 2021). On the other hand, testing is the least researched topic in RPA, with only three identified papers focusing only on automation testing (Enríquez et al., 2020). While RM and quality testing are typically part of the DLC, their link is missing in most cases (Skoković & Rakić-Skoković, 2010). This is also true for the most widely used RPA DLC method which is the waterfall method (Cewe et al., 2017; Tran & Ho Tran Minh, 2018). The waterfall method lacks proper testing integration in all development phases as suggested by Mařík et al. (2000). This is because the waterfall method has a sequential order of phases with testing being one of the last phases.

The problem-centred interviews with RPA experts further supported the findings regarding the importance of comprehensive requirement analysis and documentation in RPA projects and highlighted the absence of a standardized testing method among practitioners. These results indicate that RM and standardized testing practices are highly valued by both the academic literature and RPA practitioners and that there is a lack of research in these areas for establishing a solid theoretical foundation. Notably, the current most widely used RPA DLC does not address the connection between RM and testing. As such, the development of a new RPA DLC method that can incorporate this linkage would be highly beneficial for both academic research on RPA and RPA practitioners.

1.2 Research objective

The research is performed in collaboration between the Utrecht University[1] and the RPA software company Ciphix[2]. The problem statement in section 1.1 stressed the importance of combining RM and testing practices in a novel RPA DLC method. A combination of theoretical foundation and practical feedback is used. A theoretical background is used to design an RPA DLC method. For this, the research performs an extensive multi-vocal literature review (MLR) on existing RPA papers focused on RM, testing practices, and DLC methods. The findings from the literature study are combined with the insights gained from the problem-centred interviews and the multiple-case study to design an RPA testing method. The designed RPA method is validated and evaluated with RPA experts at Ciphix.

This study focuses on a design problem since there is a need for a novel RPA DLC method. Wieringa (2014) described a design problem as a problem to design an artefact with certain requirements that can achieve some stakeholder goals. The stakeholders in this context are the academic field of RPA and RPA practitioners. The following research objective is formulated which describes the artefact to be designed.

Design a method that effectively supports the alignment between requirement management and testing practices in the RPA development life cycle

The following sub-questions have been formulated to reach this objective:

Sub Questions:

1. ***What development life cycle phases are relevant to align requirements with testing practices?***

   The importance of RM and testing practices in RPA cannot be overstated as they are crucial to determining project scope, designing, and developing the project, and ensuring all requirements are met. As both RM and testing practices have an impact on multiple phases of a project, it is essential to research and incorporates them into a novel RPA DLC method that aligns the requirements and testing activities. To achieve this, an MLR is conducted to explore the different phases that benefit from this alignment and to develop a comprehensive RPA DLC method.

2. ***What methods or techniques focused on aligning requirements and testing practices are suitable for RPA?***

   The aim of this sub-question is to gain knowledge of the various methods that exist for aligning RM and testing practices. In order to achieve this, an extensive MLR is conducted. The focus of the study is to identify the differences among the existing methods and to assess their suitability for RPA. Based on the research question, MLR, and problem-centred interviews, requirements for the RPA DLC are generated. These requirements are used to analyse the suitability of the different

[1] https://www.uu.nl/en retrieved on April 26, 2023
[2] https://ciphix.io/ retrieved on April 26, 2023

methods and to determine which method is most appropriate for RPA.

3. ***How should requirements and testing practices be formalized and documented to create alignment between them?***

   To enable effective alignment between requirement management and testing practices in RPA projects, it is crucial to formalise and document these practices appropriately. Proper documentation is essential in any DLC, and it should be given sufficient consideration in this research. This sub-question aims to address how alignment between RM and testing practices can be incorporated into the RPA method through the utilization of literature review and multiple-case study.

4. ***How can we design an effective RPA method that aligns requirement management and testing practices?***

   The answers from the previous sub-questions are combined to design an RPA method that aligns requirement management with testing practices. A quality assessment is performed with RPA experts validating and evaluating the RPA method.

5. ***How does the designed RPA method integrate with the current documentation and tools used within RPA projects?***

   In order to assess the effectiveness and suitability of the RPA testing method designed in the previous sub-question, it is important to evaluate its integration with the project environment, including documentation and tool use. This evaluation provides a more comprehensive understanding of the RPA testing method in practice. To achieve this, expert interviews are conducted to analyze and evaluate the integration of the RPA method with its environment.

2. Research design

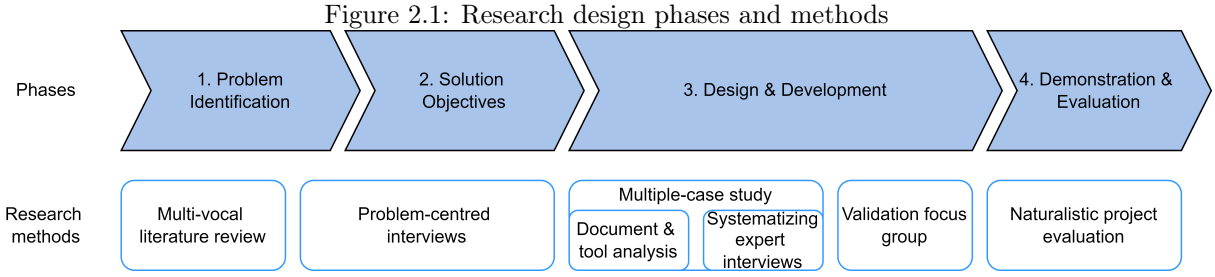
This study is set up as a design science research with an RPA testing method as an artefact. We follow the design science research methodology created by Peffers et al. (2007). It consists of different phases that are focused on creating and evaluating an artefact (i.e., method). This chapter first describes these phases and the different research methods related to each phase. Secondly, the evaluation approach is explained describing the evaluation strategy and quality assessment criteria on which the RPA testing method is evaluated. Thirdly, the validity threats are discussed.

2.1 Research phases

The different phases of the design science research methodology from Peffers et al. (2007) are described as follows:

Problem Identification	Identify and justify the need for an RPA testing method
Solution Objectives	Review literature & conduct expert interviews to asses RPA testing methods
Design & development	Design & validate method that aligns requirements and testing practices
Demonstration & evaluation	Apply and evaluate the method in a naturalistic case study
Communication	Write the final version of the thesis

Figure 2.1: Research design phases and methods



In order to execute the different phases, different research methods are used. This is depicted in Figure 2.1. For the case studies, the different data collection methods used are also specified. The next section describes what happens in these different phases.

Problem Identification

In the problem identification phase, we identify the problem and what value the newly designed artefact can provide (Peffers et al., 2007). A multi-vocal literature review (MLR) is conducted to provide a clear picture of the current research on RPA documentation, requirements, and testing to identify a gap. Furthermore, an extended literature search is done to identify different testing methods that can align requirements and testing practices. Besides the MLR, problem-centred expert interviews are conducted to understand the limitations of RPA requirement management, requirements alignment with testing, and RPA testing practices. The problem-centred interviews and the MLR form the building blocks for the solution objectives.

Solution Objectives

From the problem-centred interviews, we gain some insights into crucial aspects of RPA projects. These are part of the construction of the first iteration of the RPA testing method. Additionally, the MLR identifies crucial aspects of testing methods. Together, the MLR and interview insights identify a suitable RPA testing method that can align requirements and testing practices. This first iteration is used as the initial method for the next phase.

Design & Development

The design & development phase consists of an extensive multiple-case study and design validation interviews. The multiple-case study performs a thorough analysis of different RPA projects at Ciphix by performing document & tool analysis and systematizing interviews. These two methods are described in more detail in Section 2.2.3. The second step is the design validation interviews. Validation is aimed at justifying the contribution of a method to stakeholders before implementation (Wieringa, 2014). The design validation interviews are explained in more depth in Section 2.2.1. The aim of the multiple-case study is to analyse the use of traceability within RPA projects and to understand their requirement analysis, design, development, and testing procedures and standards. The case study should provide insights into the procedure to generate guidelines for the different components of the RPA testing method. The design validation interviews are focused on validating the RPA test method and its assisted guidelines. Furthermore, the experts can generate feedback on the RPA testing method. The feedback can be used to improve the RPA testing method. This phase leads to the redesigned and validated method that is used in the demonstration and evaluation phase.

Demonstration & Evaluation

The created RPA method needs to be demonstrated. This is an essential part of design science research and should confirm that the artefact (i.e., method) can solve the organizational problem (i.e., limited testing standards and a lack of alignment between requirements and tests) (Peffers et al., 2007). For this, the validation and evaluation approach by Wieringa (2014) is used. The goal of the validation is to assess whether the RPA method is deemed appropriate and suitable for practical application (Wieringa, 2014). The objective of the evaluation is to assess the effectiveness of the RPA method after its implementation in a real-world project (Wieringa, 2014).

2.2 Research methods

This section discusses the various research methods used in the study. First, it describes the expert interviews. Second, it explains the MLR. Third, it describes the validation focus group. Finally, the section discusses the case study evaluation.

2.2.1 Expert interviews

There are a few individual types of expert interviews used in the research. The first is the problem-centred interviews in the problem identification phase. The second is the systematizing expert interviews conducted as part of the multiple-case study which is discussed in Section 2.2.2. For analysis purposes, all interviews are recorded and transcribed. The following criteria are used to select participants for the interviews:

- Participant has at least one year of working experience with RPA
- Participant is involved in the development of RPA with requirement management, design, and testing (i.e., domain expert)

Problem-centred interviews

The *problem-centred interviews* use a mix of deductive and inductive reasoning for its interviews (Bogner et al., 2009; Witzel & Reiter, 2012). Deductive means that the interviewer uses predetermined concepts as a framework to guide the interview and understand these concepts to a greater extent (Witzel & Reiter, 2012). On the other hand, inductive reasoning takes a more open approach without predetermined concepts and the questions try to discover novel ideas or issues not discovered before. It requires structured knowledge of the concepts while simultaneously being open and flexible in the face of different concept necessities (Witzel & Reiter, 2012). This approach is chosen since the interviews aim to understand the current state of requirement analysis, translating them into test cases and the test practices in general. This is compared and evaluated against the findings from the literature study. Furthermore, the interviews try to discover current struggles or inefficiencies within RPA projects.

2.2.2 Multi-vocal Literature review

RPA is a relatively new field and academic literature has been lacking (Enríquez et al., 2020; Syed et al., 2020). RPA is application-oriented (Ng et al., 2021) and a growing group of practitioners is contributing to RPA research through grey literature (Chugh et al., 2022). However, few systematic reviews of grey literature for RPA are completed (Chugh et al., 2022). Therefore, with limited academic literature and a group of practitioners contributing to grey literature, a combination of grey and white papers is used for the literature study. This constitutes a multi-vocal literature review (MLR) (Garousi et al., 2019). This combination provides insights into the 'state-of-the art' and 'practice' of RPA and should give a full overview of the current developments within RPA research. The MLR structure provided by Garousi et al. (2019) is used. A full explanation of the structure and how the MLR is executed is explained in Chapter 3.

2.2.3 Multiple-case study design

For the design and development phase, a multiple-case study is used rather than a single case study since it increases the completeness and generalizability of the results (Runeson & Höst, 2009). Additionally, a case study allows studying RPA testing practices in a realistic setting which is an advantage for a design science approach (Wieringa, 2014). For the multiple-case study, triangulation — combining multiple sources of data (Runeson & Höst, 2009) — is used by performing a: document & tool analysis and systematizing interviews. The document & tool analysis shows the current practices of RPA development and the format of requirements and testing scenarios. Furthermore, this document & tool analysis is used to understand the integration and contribution of the RPA testing method with the current use of documentation and tools. The systematizing interviews are conducted with experts involved in the case for a better understanding of the current testing practices and to pinpoint issues. The multiple-case study is retrospective and observational in nature since finished projects are used and no changes are made to these projects. For the case study four projects are selected according to the following criteria:

- Complexity of the case described by the number of process steps and number of applications used

  High-complexity is more than 5 applications, more than 50 steps

  Medium-complexity is between 3-5 applications, between 25-50 steps

  Low-complexity is less than 3 applications, less than 25 steps

- Select 2 high and 2 low complexity cases
- Experts must differ for every case within the same complexity level

In total four cases are chosen with two in the high complexity group and two in the low complexity group. This allows for a thorough analysis. We perform a within-case analysis and an across-case analysis (Ayres et al., 2003). The within-case approach studies an single case in-depth. The across-case approach helps to discover similarities and mismatches between cases which increases generalizability (Ayres et al., 2003).

Document & tool analysis

Document & tool analysis is used to gain understanding and develop knowledge of the current use of documentation and tools within an RPA company. The analysed documents can take various forms but in this research, it is limited to formal requirements, solutions, and testing documents. This excludes slide decks, emails, or other informal documents. The document study is aimed at understanding the current use of traceability within RPA project documentation and generating guidelines for good practices. The tools are analysed to see how they are used in relation to the documentation. An evaluation is done on the limitations and benefits of the documents and tools and how they integrate with the developed RPA testing method. This evaluation is focused on but not limited to understanding the *functionality* and *completeness* criteria specified in Table 2.1.

Systematizing interviews

The systematizing interviews are focused on the systematic retrieval of information and gaining access to exclusive knowledge that the expert has (Bogner et al., 2009). These interviews are recorded and transcribed for analysis purposes. In these interviews, the object of the study is not the expert itself, but the information the expert can provide. A semi-structured interview is executed to first focus on specific topics but also leave space for elaboration so that new concepts, issues, or theories can emerge from the data (Bryman, 2016). The systematizing interviews have multiple purposes. Firstly, guidelines for the RPA testing methods need to be provided. Secondly, the designed RPA-VV method needs to be discussed and verified to provide additional feedback and the integration of the RPA-VV method with the documentation and tools needs to be evaluated.

2.2.4 Validation focus group

The *validation focus group* is used to validate the RPA testing method. Validation is aimed at justifying the contribution of the method to stakeholders before implementation (Wieringa, 2014). The method is validated according to the criteria in Table 2.1. Additionally, the interviewees can still suggest changes to the RPA testing method to improve its completeness and effectiveness. The focus group consists of multiple solution architects (SAs) with some having already participated in the case study and some being new to the designed RPA method. Convenience sampling is used since the total size of the group of SAs is insufficient to exclude any from the focus group.

2.2.5 Naturalistic project evaluation

The naturalistic project evaluation consists of two projects to evaluate the RPA method. Evaluation is the investigation of the method after implementation (Wieringa, 2014). Therefore, the RPA method is applied in these two projects and focus groups with the project team are planned to evaluate the method in this naturalistic setting. A naturalistic evaluation means exploring the performance in a real environment (Venable et al., 2016) and in our case, the method is deployed in a real project at Ciphix. The focus group aids in the evaluation by gathering relevant feedback on the quality of the method based on the quality assessment criteria defined in Table 2.1. For the selection of the projects, a convenience sampling approach is used. This means the case is selected on minimum time and effort to accommodate the research restrictions (Shakir, 2002). This is done in consideration of the restricted time period in which the evaluation needs to happen. Therefore we are limited to cases that start and end within this specific time period.

2.3 Evaluation

We follow the four-step framework provided by Venable et al. (2016) to set up and explain our evaluation approach. Their framework consists of the following four steps:

1. Explicate the goals of the evaluation
2. Choose the evaluation strategy or strategies
3. Determine the properties to evaluate
4. Design the individual evaluation episode(s)

Goals of evaluation

Venable et al. (2016) explain three goals for the evaluation of a method. One of the goals is to establish the *rigour* of the method to determine how the method works in a real situation (Venable et al., 2016). *Reduce risk* is a second important evaluation goal (Venable et al., 2016). This indicates to use of evaluation early in the design process to identify areas of improvement and use the evaluation as support in the development phase to increase the quality (Venable et al., 2016). *efficiency* is the final goal addressed by (Venable et al., 2016). This considers the depth of the analysis while taking time and resources into account. How these three goals are addressed is explained in the next section.

Evaluation strategy

The strategy that is followed in this research is called *Human Risk & Effectiveness* (Venable et al., 2016). This is a suitable strategy if (1) there are resources available to evaluate in a real context and (2) the artefact its utility and benefits over the long run need to be evaluated (Venable et al., 2016). When connecting this to the before mentioned goals of evaluation, the *rigour* is applied by performing the naturalistic evaluation project described in Section 2.2.5. To reduce *Reduce risk*, early problem-centred interviews and a design validation focus group are incorporated into the research method. Considering the *efficiency* and *rigour* together. The decision has been made to have an extensive case study to design the method and a smaller but naturalistic evaluation. This can provide a well-researched and validated method and evaluate how the method works in a real situation. This uses the full potential of efficiently validating and evaluating the RPA testing method.

Properties to evaluate

The quality assessment criteria are based on the evaluation criteria by Prat et al. (2015). The quality assessment criteria are defined in Table 2.1. The evaluation approach is described in depth in Section 2.3.1.

Evaluation episodes

These episodes are already discussed in-depth in Section 2.2, but for consistency are shortly mentioned here. The first episode of is the *problem-centred interviews* to determine the issues within RPA projects their requirements and testing and compare them with literature findings. The second episode is the *validation focus group* to receive feedback on and validate the constructed RPA method. The final evaluation episode is the *naturalistic project evaluation* which consists of applying the method in practice and a*focus group*. This is the final episode that evaluates the RPA testing method in a naturalistic setting.

2.3.1 Evaluation approach

The evaluation approach is based on the method evaluation criteria of Prat et al. (2015). Prat et al. (2015) differentiate different aspects on which a method can be assessed. Three aspects are assessed which are the method *Structure, activity* and *environment*. *Environment* is further categorised in *people* and *technology* (Prat et al., 2015). These three categories assess how the method impacts and integrates with the *Business, people,* and *technology* environment. *Structure* criteria assess the relation, hierarchy, and dependence of different elements of the method (Prat et al., 2015). *Activity* criteria evaluate the dynamics of the method by assessing the input, transformations, and outputs of the method (Prat et al., 2015). All the criteria related to these aspects are described in Table 2.1 and are assessed within an interview, questionnaire, or focus group. This provides a conclusion on the value and quality of the RPA testing method.

Table 2.1: Evaluation criteria by Prat et al. (2015)

Criteria	**Aspect**	**Description** (Prat et al., 2015, pp. 266, 267)
Completeness	Structure & Activity	The degree to which the structure and activities of the artefact contains all necessary elements and relationships between elements
Consistency	Structure & Activity	The degree of uniformity, standardization, and freedom from contradiction among the elements of the structure and activity of the artefact
Functionality	Activity	The capability of the artefact to provide functions which meet stated and implied needs when the artefact is used under specified conditions
Simplicity	Structure & Activity	The degree to which the structure and activities of the artefact contain the minimal number of elements and relationships between elements
Usefulness	People Environment	The degree to which the artefact positively impacts the task performance of individuals
Ease of use	People Environment	The degree to which the use of the artefact by individuals is free of effort

2.4 Threats to validity

The four types of validity threats described by Wohlin et al. (2012) are important to address to ensure the rigour of the research. The four types are *construct, internal, external,* and *reliability.* How they are addressed within the research design is explained next.

2.4.1 Internal validity

According to Wohlin et al. (2012), *internal validity* refers to the causal relationship between the independent and dependent variables and is mainly concerned with the accuracy of the obtained results. In this study, a systematic literature review was conducted to select a suitable method for the RPA DLC, and seven requirements based on the literature study and problem-centred interviews were used for the selection. Additionally, the multiple-case study had different complexity and solution architects (SAs) as selection criteria for the included cases. However, convenience sampling was used for the selection of participants in the validation and evaluation stages due to the relatively small size of the SA group and the need for the evaluation to occur within a specific time frame. Despite this, the evaluation was carried out in a naturalistic setting to determine the effectiveness of the RPA method in a real-life scenario.

2.4.2 External validity

The concern of *external validity* is to generalize the research findings beyond the study sample (Wohlin et al., 2012). In this research, all participants were domain experts to ensure their understanding of the topic under evaluation. However, since the research focused solely on the RPA domain, its direct application to other domains is limited. Moreover, the sample size of the focus group was small, and the experts were from one organization, which limits the generalizability to other populations. Nevertheless, although the research focused on RPA organizations that outsource their services, it can still be applied to organizations that develop their RPA solutions internally. In such cases, only the prerequisites of the method that relate to the client need to be performed by the organization's experts. Furthermore, the research combined a theoretical foundation with practical evaluation, which enhances its suitability for actual RPA projects and provides insight into the effect of the method on naturalistic RPA projects.

2.4.3 Construct validity

According to Wohlin et al. (2012), construct validity is concerned with the relationship between the theory and observation and evaluates whether the measures used in the research actually measure what they are intended to measure. To reduce this threat, the current study analysed multiple methods that can

help align requirements and testing. Seven requirements of the method were constructed based on the strengths and weaknesses of the method found in the literature and through problem-centred interviews with RPA experts. The selected method was further re-designed through a multiple-case study that used data triangulation (Runeson & Höst, 2009). The validation focus group further improved the method and analysed if the method was expected to be suitable for practice. Multiple criteria were selected to analyse its suitability, and a final naturalistic project evaluation was conducted to analyse the effectiveness of the method in practice, which also used criteria as measurements. However, the validation and evaluation were limited within a time frame, and not all intended criteria were analysed, which reduced the construct validity of the method.

2.4.4 Reliability

The *reliability* of research refers to the consistency and reproducibility of research findings (Wohlin et al., 2012). This includes the data collection method, instruments used, and measurements. The research in question has taken care to ensure reliability. A clear protocol or approach has been developed for all research methods used, including an extensive literature study protocol and multiple-case study protocol. Recordings have been made and transcribed for all interviews and focus groups. Data triangulation has been employed in the multiple-case study to enhance reliability. Moreover, several qualitative criteria selected by Prat et al. (2015) are applied to different research methods, and the criteria are qualitatively analyzed with RPA experts. However, the feedback received from experts can be biased or influenced by their own standards within RPA projects. To address this, the interviews are controlled with framed questions for specific criteria, and multiple interviews with different experts are conducted.

Literature study

3. Literature research protocol

We perform a multi-vocalregu literature review (MLR) to investigate the current state-of-the-art for testing and documentation practices for RPA. An MLR consists of a systematic review of grey and white literature. This approach is chosen since RPA has a growing group of practitioners that contribute to RPA research through grey literature (Chugh et al., 2022). The goals of the MLR can be explained by the following points:

- Gain insights into the existing research of RPA requirements and testing by:
  - Researching current testing methods of RPA
  - Researching requirement management for RPA
  - Researching RPA documentation related to requirements management and testing
- Research requirement traceability
- Identifying and evaluating testing methods on their suitability to align requirements and testing.

3.1 Approach

The MLR structure provided by Garousi et al. (2019) is followed. This consists of the following three phases: *Planning MLR, Conducting MLR*, and *reporting the review*. The planning phase consists of explaining the need for and goals of the MLR which is described in the previous section. The second phase *conducting MLR* focuses on the search process through source selection, quality assessment, and data extraction (Garousi et al., 2019). The last phase is focused on reporting the findings which is important to ensure the usefulness to the target audience of what is reported (Garousi et al., 2019).

3.2 Conducting MLR

The goals of the MLR are mentioned above. To fulfil these goals, Garousi et al. (2019) specifies the *conducting MLR* phase in which MLR questions need to be formulated. These form the basis of what we want to discover during the MLR and should all be answered. The questions are as follows:

- What is the current most used development life cycle method for RPA?
- What are important aspects of testing?
- What testing methods are used for RPA?
- What is requirement tractability or alignment?
- How are process requirements defined or formatted in RPA projects?
- What methods can help in aligning requirements and testing practices?
- What documentation is used for the requirement, design, and testing phase in RPA projects?

From these MLR questions, the initial search terms were created. The search engine used is Google Scholar for scientific publications and Google for grey literature. The initial search terms are:

"Software development life cycle" AND "Robotic process automation" OR RPA ; “Software testing techniques” ; "Software testing methods" ; Testing of "robotic process automation" ; "robotic process automation" AND requirements OR process definition ; "robotic process automation" AND Documentation OR Document ; "robotic process automation" OR RPA AND “process definition document” OR “process design document” OR PDD; "robotic process automation" OR RPA AND “solution design document” OR SDD ; "robotic process automation" OR RPA AND “test plan document” OR TPD ; “Effectiveness metrics” for software testing OR Software “testing metrics” ; “Requirements traceability" OR "requirement alignment" AND testing ;

While with these search terms, some papers on testing methods for RPA were found. The pool was still limited. Therefore, the backward snowballing approach (Wohlin, 2014) was used to extend the pool with relevant literature and explore mentioned topics in more depth. Additionally, the literature pool did not include proper testing methods to align requirements and testing practices. Therefore, an extended search was performed that focused on the different development or testing methods that could align requirements and testing better. These extended terms include:

“Requirements-based testing” ; “Model-Based Testing” ; “Requirement traceability testing” ; “Test Driven Development” ; “Waterfall model” AND testing ; "V-model" AND testing ;

Stopping criteria

When to stop searching is an important aspect of an MLR. Garousi et al. (2019) describe this as the exhaustion-stopping criteria. For this MLR, we have chosen to stop at the last page of Google or Google Scholar where any relevant literature was found based on the inclusion and exclusion criteria described in the next section.

Selection criteria

All the papers that were found from the initial and extended search terms formed the initial literature pool. The number of papers per topic for this pool is shown in 3.1. The next step was analysing the relevance of these papers. This is done by reading the title and abstract of these papers and analysing them based on the inclusion and exclusion criteria — and quality assessment criteria for grey literature. These criteria are:

Inclusion criteria:

- Papers discussing testing techniques, types, and levels
- Papers discussing requirement management
- Papers discussing requirements traceability
- Papers discussing methods or approaches to align requirements and testing practices
- Papers discussing RPA documentation, development, requirements, or testing

Exclusion criteria:

- Written in a language other than English
- Papers discussing using RPA technology for testing software
- Papers using RPA in a different domain or field than RPA projects
- Non-organizational reports or slide decks such as news releases or specific product brochures

The quality assessment criteria for the grey literature are based on guidelines provided by Garousi et al. (2019):

Quality criteria:

Methodology:

Does the work cover a specific question?

Objectivity:

Does the work seem to be balanced in its presentation?

Novelty:

Does it enrich or add something unique to the research?

Does it strengthen or refute a current position?

This created a relevant pool of literature shown in Table 3.1. This has been further examined by a full-text analysis and only papers that were helping in answering one of the MLR questions were selected. These papers form the final pool of literature shown in Table 3.1 and are included in the MLR in Chapter 4.

Table 3.1: Literature review pools and result

Literature review step	**Concept**	**# sources**	**# scientific publication**	**# grey literature**
Initial pool	RPA development life cycle	70	40	30
	Testing	134	104	30
	Requirements	30	20	10
	Documentation	110	70	40
	Alignment requirements & testing	30	30	0
Relevant pool	RPA development life cycle	17	9	8
	Testing	55	44	11
	Requirements	9	6	3
	Documentation	49	29	20
	Alignment requirements & testing	13	13	0
Final pool	RPA development life cycle	9	5	4
	Testing	24	22	2
	Requirements	8	6	2
	Documentation	7	2	5
	Alignment requirements & testing	5	5	0

4. Multi-vocal literature review

Based on the research protocol defined in Chapter 3, the MLR was conducted. All the MLR questions formulated in Chapter 3 are answered in the different sections of this chapter. Firstly, what a development life cycle (DLC) is and how RPA uses a DLC is discussed in Section 4.1. Secondly. requirement management, requirement traceability, and how this is incorporated in RPA projects is specified in Section 4.2. Thirdly, RPA documentation is explained in Section 4.3. Fourthly, in Section 4.4 testing is discussed by focusing on general testing aspects, RPA testing, and testing methods that can help in aligning requirements and testing practices. Lastly, in Section 4.5 a discussion about the differences between the testing methods is reported.

4.1 Development life cycle

This section first introduces the development life cycle (DLC) and different DLCs are explained. After, the RPA DLCs described in the literature are discussed in detail. The latter section answers the MLR question: *What is the current most used development life cycle method for RPA?*.

The development life cycle (DLC) or often referred to as the software development life cycle outlines each step to analyse, create and deploy a —software— application (Servicenow, n.d.; Tran & Ho Tran Minh, 2018). Since RPA has some differences from standard software projects, the term DLC is used from here on out. DLC is an important aspect of quality assurance for a project (Cernat et al., 2020). Tran and Ho Tran Minh (2018) describe that the aim of DLC is to increase the quality of a development process. Emorphis (2020) describes it as a holistic view of the various stages to make the implementation of an application simple and systematic. Consulting (2021) adds that it can help in allocated resources and management of the whole development cycle and should aim to provide full tractability of the process from beginning to end.

There are different types of methods that fall under the DLC. These different methods include but are not limited to *Agile, DevOps, waterfall,* and *V-model* (Servicenow, n.d.). *Agile* focuses strongly on user input and experience working with quick cycles and incorporating feedback. *DevOps* is similar to agile by gathering and incorporating feedback, its unique characteristic is that it has a team consisting of developers, testers, and operational personnel that have close communication with each other(Servicenow, n.d.). The *Waterfall* model is a linear development method completing each step consecutively from planning to deployment. The *V-model* is similar to the waterfall model since it executes its phases sequentially. However, it has an emphasis on verification within the planning and design phase and links every phase with a type of testing for validation (Servicenow, n.d.).

4.1.1 Development life cycle for RPA

Since RPA is still a relatively new field its DLC is also still maturing (Cernat et al., 2020). However currently, most of the DLC used for RPA are based on or follow a similar structure to the waterfall model (Cewe et al., 2017; Tran & Ho Tran Minh, 2018). For RPA, the emphasis is of course on process description and process analysis rather than software (Orynbayeva, 2019). Different approaches to the RPA DLC are described in the literature and are shown in Table 4.1.

In Table 4.2 the different RPA DLCs have been grouped according to similarities within their phase description. This shows a total of eight different phases described by all the authors. On average, RPA DLC consists of six phases. Group 1 is the planning or discovery group which is described by Ghouse and Sipos (2022), Emorphis (2020), and Orynbayeva (2019, #7). It focuses on understanding customer demands, creating a project plan, and assigning a team. Group 2 focuses on requirements gathering or analysis of the process. This phase analyses the full process and documents any exceptions, criteria, or other important aspects that need to be understood of the process (Ghouse & Sipos, 2022; Montero et al., 2019; Orynbayeva, 2019; Tran & Ho Tran Minh, 2018). Ghouse and Sipos (2022), Orynbayeva (2019), and Tran and Ho Tran Minh (2018) specify on including a process description document (PDD) here. Group 3 is the design phase that all authors except Reddy (2022) include. This phase describes

Table 4.1: Development life cycle of RPA according to different studies

Source #	Author(s)	Phases	# of phases
1	(Ghouse & Sipos, 2022)	Discovery, Design, Development, User acceptance test, deployment, execution of bots	6
2	(Montero et al., 2019)	Analysis, Design, Development, Deployment, Testing, Monitoring	6
3	(Tran & Ho Tran Minh, 2018)	Requirements gathering and Analysis, Design, Implementation, Integration and testing, Deployment, Maintenance	6
4	(Emorphis, 2020)	Discovery, Solution Design, Development, User acceptance test, Deployment & maintenance, execute bot, support & maintenance	7
5	(Reddy, 2022)	Analysis, Development, Testing, Deployment, Maintenance	5
6	(Orynbayeva, 2019, #6): Blue Prism	Define process, Design, Development, Testing, User acceptance test, Deployment, Maintenance	7
7	(Orynbayeva, 2019, #7): UiPath	Planning, Analysis, Solution Design, Development, Testing, Hyper-care	6

Table 4.2: Groups of RPA DLC phases

Group #	Group	source #	Description
1	Planning, Discovery	1,4,7	Create project plan and assign team
2	Requirements gathering, Analysis, Define process	1-3, 6, 7	Analyse process requirements
3	(Solution) Design	1-5, 7	Create full understanding and documentation of the design
4	Development, Coding, Implementation	1-7	develop all functionality of the bot
5	User acceptance test, testing, integration and testing	1-7	Perform all tests on the bot
6	Deployment, Hyper-care	1-7	Deploy bot in a production environment to validate it can run successfully
7	Execution of bots	1,4	Describes that bot performs the process
8	Operation, Maintenance, Support. Monitoring	2-6	Monitor to bot while running to identify any unforeseen issues. If they arise, provide support to fix these issues

and documents the full functionality in a detailed document. This document is called the solution design document (SDD) (Ghouse & Sipos, 2022). Group 4 is included by all authors which is the development phase. This phase is crucial since without it no bot would be created. Overall all authors agree on this phase. The only exception is Orynbayeva (2019, #6) which describes its development and testing phase in parallel. Orynbayeva (2019, #6) only leaves the user acceptance test (UAT) as a separate testing phase in its DLC. This brings us to the next phase which is the testing phase. All authors agree on the inclusion of this phase in which all functionality is validated through testing. After testing is complete, the sixth phase describes the deployment of the bot in the production environment. Orynbayeva (2019, #7) calls this the hyper-care phase because this phase is still closely monitored to check if the bot runs smoothly and without any issues in the production environment. Group 7 is only included by Emorphis (2020) and Ghouse and Sipos (2022) described that the bot is live and performing its intended process and producing output. The other authors assume that if the deployment phase is successful the bot is ready to execute anyways. Group 8 is the last and is the maintenance and support phase. All unforeseen issues that require maintenance or upgrades for the bot are handled in this phase (Ghouse & Sipos, 2022;

Montero et al., 2019).

4.2 Requirement management

This section answers the MLR questions: *What is requirement traceability or alignment* in Section 4.2.1 and *How are process requirements defined or formatted in RPA projects* in Section 4.2.2. Firstly, an introduction is provided on requirement management usage and benefits. Secondly, requirements traceability is discussed. Lastly, RPA requirements and the use of traceability within RPA projects are discussed.

Requirement management is the transformation of customer needs into the application specifications (Jiao & Chen, 2006). These needs need to be defined in a structured format. Accurate requirements are important since it forms the basis for the further design, development, and testing of the application (Song, 2017). Requirement management is important to understand the scope and target of the project. Extensive studies have shown the relevance of careful assessment of requirements for the success of a project (Jiao & Chen, 2006). Often, a poor understanding of requirements and inaccurate assumptions have a big negative impact on design, code quality, and project time and cost (Jiao & Chen, 2006). Mogyorodi (2001) and Skoković and Rakić-Skoković (2010) describe that requirements are the main cause of defects in projects. In total, 56% is due to requirements issues, 27% is due to issues in design, and only 7% because of issues in the code. Of all requirement issues, about half are from poorly written requirements, and the other half are from incomplete requirement specifications. Faults in requirements lead to more work which in turn causes higher project costs and delays (Skoković & Rakić-Skoković, 2010). Mogyorodi (2001) describes three reasons why projects fail: (1) Requirements are incomplete, (2) scope creep (i.e., requirements changes) happens too often and is not handled correctly, and (3) a lack of customer or user input for the requirements. Fixing defects from requirements is often very costly since it requires changes in the requirements, design, code, and test cases (Mogyorodi, 2001). The later a defect is found the more costly it becomes. Mogyorodi (2001) shows the cost ratio related to the phase in which an issue is found. This is depicted in Table 4.3.

Table 4.3: Cost ratio of defects found in different phases (Mogyorodi, 2001)

Phase in which defect was found	**Cost ratio**
Requirements	1
Design	3-6
Coding	10
Unit/Integration testing	15-40
System/acceptance testing	30-70
Production	40-1000

Difficulties of requirement management

Jiao and Chen (2006) mention a few reasons why requirement gathering can be difficult. Firstly, requirements from customers or often qualitative and imprecise due to their linguistic origin. Secondly, requirements are negotiable which can create conflict with one another. This makes it necessary to make trade-offs. Third and last, differences in semantics or terminology can make it more difficult to convey the information of the requirements effectively. Additionally, this can lead to abstract terms for the requirements which can lead to vague assumptions for the requirements (Jiao & Chen, 2006).

4.2.1 Requirement traceability

Requirement traceability (RT) is focused on being able to trace a requirement from creation in the analysis phase until fulfilment in deployment (Torkar et al., 2012). This can help to link or align requirements to other artefacts of the application such as design documentation, models, source code, test scenarios, and test cases among others (Ziftci & Krüger, 2013). This can be helpful to assist stakeholders to find the origin or rationale behind a requirement (Bouillon et al., 2013) and shows the impact of a requirement on the project (Torkar et al., 2012). With RT, data can be linked between artefacts which can then

be monitored and controlled (Kirova et al., 2008). Barmi et al. (2011) mention that RT can be used in different steps of the V-model that aid in requirements validation and verification.

Benefits and challenges of requirement traceability

There are some benefits and challenges related to RT. One benefit is that RT can help in change management throughout the DLC (Barmi et al., 2011; Kirova et al., 2008). For example, scope creep can have a big impact on a project when it occurs but with RT an accurate response can be set in place to see the impact on the project's progress and apply needed changes. A second benefit is that it can help with tracking requirement coverage and requirement validation through linked test cases (Barmi et al., 2011). When a test fails it can be traced back to a specific functionality or code in the module which can help understand and hopefully solve issues faster. Furthermore, this alignment can be used for functional and non-functional requirements (Barmi et al., 2011). These benefits show how RT helps to verify the consistency and completeness of requirements throughout the whole DLC and with this improves an application's quality (Kirova et al., 2008). Bouillon et al. (2013) conducted a survey with practitioners that used RT in their DLC which showed the following:

- 80% agree that RT has expected development benefits.
- 60% agree that RT is important for the development process.
- 50% agree that RT its benefits out weight the cost.

Especially the last point is interesting since only half think that RT its benefits outweigh the cost. Torkar et al. (2012) mention that one of the tricky parts of RT is to balance the benefits obtained and the effort spent on RT. Finding this balance can be challenging due to a few reasons. Kirova et al. (2008) and Torkar et al. (2012) mention that identifying dependencies or links between artefacts is done manually which is very time-consuming and costly. Additionally, the level of granularity of requirements can be tricky to be able to trace throughout the project (Kirova et al., 2008). There are still a wide variety of implementation and standards for RT (Kirova et al., 2008) which can also be a reason why only half of the practitioners think that the benefits outweigh the costs (Bouillon et al., 2013).

To balance the benefits and costs of RT as a business, it is important to choose a fitting RT strategy. Different factors are of influence finding a fitting RT strategy. These business aspects are best described by Kirova et al. (2008) and include *Business, Industry, Development life cycle, Project*, and *Technology*. The *business* factor entails the business objectives and the commitment to quality standards. Also, the complexity of the business structure and the collaborative nature of teams are part of this factor. The second factor is *industry* and discusses the maturity of the industry or market has business entered. The third factor is *development life cycle* and encompasses the maturity of the DLC in regards to established processes, standards, and frameworks(Kirova et al., 2008). This also includes semantic information available for artefacts that are used within the DLC and the related documentation. The fourth factor is the *project* which comprises the size and complexity of the projects and the team's experience. The fifth and final factor is *technology* which covers tools used by the team or organization (Kirova et al., 2008). Useful tools that are often used include project management tools, documentation tools, testing tools, and integrated development environment (Bouillon et al., 2013).

When a business understands the above-mentioned factors, a suitable RT approach can be chosen. Torkar et al. (2012) describe multiple approaches of RT. These include but are not limited to the *rule-based, scenario-based, process-centred* approaches and *traceability matrices*. Starting with the latter, *traceability matrices* are used as visualization tool (Ziftci & Krüger, 2013) to define relationships between requirements and other artefacts such as design modules code, or tests (Torkar et al., 2012). These links are often created manually. While it is a clear visualization of the alignment between requirements and other artefacts, Torkar et al. (2012) mention that traceability matrices can suffer from scalability and maintenance problems. The *rule-based* RT approach comprises automatically generated traceability links using specified rules (Torkar et al., 2012). A rule can be requirement-to-object-model traceability to link artefacts in different documents. Semantic tags can be used to link requirements to other artefacts. *Scenario-based* RT uses scenarios to model system functionality and generates test cases from these scenarios (Torkar et al., 2012). These generated test scenarios are linked with requirements and other parts of the code.

Another RT approach is the *process-centred* environment in which non-functional and functional requirements can be linked (Torkar et al., 2012). Traceability tasks need to be defined and checked with an architectural assessment method. This can be combined with a method such as *keywords and ontology* which provides the traceability between documents and models for the non-functional requirements. The use of specific keywords throughout the whole DLC is important for this approach.

To gain the most benefit from RT, Uusitalo et al. (2008) describes five practices to align requirements and testing practices. The first is *Early test participation*. This consists of involving testers in the analysis phase of a project. This ensures that test planning is taken into account in the analysis. The second practice is including *testers in requirement reviews*. This is beneficial since testers have different viewpoints than requirement analysts, which can help to surface deficiencies or requirements that are difficult to validate (Uusitalo et al., 2008). A third practice is *trace tests to requirements*. This is critical because sometimes tests were not linked to any requirements or some requirements were not linked to any test. This practice can improve test coverage and change management can be efficiently traced to test cases. The fourth practice is *linking testers with requirement owners*. This is important because documentation alone is not always enough. Person-to-person meetings can improve communication and reduce assumptions made by testers. The fifth and last practice is *achieve superior knowledge of the application* as a tester. This emphasizes the fact that testers need to fully understand the application and its requirements to aid both designing test cases and performing testing (Uusitalo et al., 2008).

4.2.2 RPA requirements

Since RPA focuses on processes their requirements are analysed and formulated differently than other software projects. The first step for RPA projects evaluates different processes on their suitability for automation (Leeuwen, 2022; Tran & Ho Tran Minh, 2018). This is related to the planning phase described in Table 4.2. For this, different criteria are checked. For example, if the process is rule-based and structured, repetitive, currently manual, multiple systems are used (Leeuwen, 2022), and a standardized process (Fernando, 2020). If a process is chosen, the requirements gathering starts. This requirement gathering is often done via a group discussion (Tran & Ho Tran Minh, 2018) involving all stakeholders (Leeuwen, 2022; Tran & Ho Tran Minh, 2018). These stakeholders include business analysts, solution architects, and domain experts (Fernando, 2020). The stakeholders discuss the whole process and its different steps are analysed (Kirchmer & Franz, 2019). Tran and Ho Tran Minh (2018) describe RPA project requirements in two categories. The first is resource requirements and the second is validation requirements. Resource requirements include the software applications used in the process, the RPA platform used for the development and deployment of the bot, and a testing environment. The latter is important to validate that the bot can run and find issues. The testing environment consists of all software, hardware, and network components simulating the production environment (Tran & Ho Tran Minh, 2018). The validation requirements consist of the entire bot and its functionality. This is complete if all happy and alternative scenarios from the process are defined including all known and unknown exceptions (Tran & Ho Tran Minh, 2018). All the gathered requirements are written down in the process definition document (PDD) in the form of the application requirements and the as-is process description (Fernando, 2020). The PDD is discussed in more detail in Section 4.3. It is important to understand the scope of the project correctly. Asking the right questions during the stakeholder group discussion is crucial to understand the full process scenarios and all manual steps (Harita, 2019). Every sub-task should be discussed and connected to other tasks to form all the happy scenarios, alternative scenarios, and other exceptions (Harita, 2019). If the whole process is understood correctly, optimization opportunities can be discussed (Kirchmer & Franz, 2019) to improve how the bot performs the process (Jovanović et al., 2018). This can be described in a detailed to-be process description in the PDD (Jovanović et al., 2018). When all of this requirements gathering is complete, the whole end-to-end process and all exceptions are properly documented so the scope and timeline of the project are clear (Harita, 2019). After this phase, the project analysis is complete and everything is in place to start designing and developing the bot.

Requirement traceability is a technique used within general software development but is not commonly used within RPA. No grey or white papers could be found on the topic. This shows a gap within the field of RPA research. The closest resemblance to some form of traceability could be found in RPA documentation. The different types of RPA documentation and the links between those documents are

described in Section 4.3

4.3 RPA documentation

This section answers the MLR question: *What documentation is used for the requirement, design, and testing phase in RPA projects?* This research only discusses the Process Definition Document (PDD), Solution Design Document (SDD), and Test Plan Document (TPD) since these are focused on the requirement, design, and testing phase of RPA projects. Choudhuri (2021) mentions that RPA documentation is a crucial part of a project. They are essential for most steps in the DLC and ensure that key information and knowledge are available for future developers and users. However, little academic research has been conducted on RPA documentation. There are papers that mention the importance of documenting the project in great detail (Goris, 2019) and that issues with having incomplete documentation occur in RPA projects (Prucha, 2021). Nevertheless, no proper description of the elements included in RPA documentation is written in academia. Therefore, the explanation of RPA documentation is mainly based on grey literature.

Process definition document

The process definition document (PDD) outlines the sequential steps of the process from end-to-end (Goris, 2019). This includes all tasks and deviations from the main process. In the creation of the document, the business analyst or solution architect and subject matter expert (SME) (i.e., domain expert) are involved (Bezemer, 2019). They come together in a group discussion to analyse and document the process correctly. UiPath (2022) describes the following elements for the PDD:

Table 4.4: PDD components (UiPath, 2022)

PDD Element	Description
Purpose of the document	Outlines process to be automated and includes elements of the document
Process key contacts	Describes stakeholders and their contact details
As-Is process description	Shows a model of the process overview and describes detailed process steps with screenshots of UI elements
To-Be process description	Expected design of process after changes or improvements
Application criteria	Comprehensive list of all applications used and the environment in which they run
In scope for project	All activities and tasks included in the project that are automated
Out of scope for project	All activities that are not included in the project
Business exceptions handling	List of process exceptions and how they are handled. This includes the solutions for known and unknown exceptions
Application error and exception handling	list of application or environment exceptions and how they are handled. This includes the solutions for known and unknown exceptions
Document approval	Section that provides the PDD sign-off by stakeholders

Bezemer (2019) also describes some essentials that contribute to a high-quality PDD. Firstly, within the PDD the process description should contain a high-level description and a step-by-step description with screenshots (Bezemer, 2019). Secondly, models or flowcharts of the process should be properly linked with the description (Bezemer, 2019). This is related to RT. Lastly, Bezemer (2019) stresses the importance of properly documenting all the process scenarios and all the different exceptions thoroughly. The PDD is the first documentation that is created for an RPA project and is crucial for the design and the development of the bot(Choudhuri, 2021).

Test plan document

The test plan document (TPD) describes the what, when, how and more of a project's testing (Choudary, 2020). The TPD includes all the process scenarios that can be formulated from the PDD steps. These

scenarios are used for the design in the SDD and tested after development. The TPD serves as a roadmap for the design and the testing process to control the possible risks. Hamilton (2022) describes different components that should be included in the TPD which are shown in Table 4.5

Table 4.5: TPD components (Hamilton, 2022)

TPD component	Description
Scope	Defines the features, FRs and NFRs that are (in-scope) or are not (out of scope) tested
Quality objectives	Description of the objectives that ensure testing quality and define when the testing is done
Roles and responsibilities	Detailed description of the roles and their task responsibilities
Test methodology	Defines the type of tests that need to be performed and when in the DLC. It also describes how bugs need to be tracked and handled
Test deliverables	List of all test artefacts that are delivered (e.g. Test cases, requirements traceability matrix, bug reports, test metrics, etc.)
Resources & tools	Mentions the hardware and software required for testing (testing environment) and tools used (e.g. requirement tracking tool, bug tracking tool, automation tool, etc.)

The TPD plays a crucial role in testing and can make or break a project (Choudary, 2020). Therefore, a clear understanding of the components mentioned in Table 4.5 is important. Proper documentation and understanding of every component for an RPA project can increase and ensure a project's quality (Hamilton, 2022).

Solution design document

The Solution Design Document (SDD) is crucial for every process that is being automated (Vinocha, 2022). The SDD prepares the solution with flows and design diagrams (Kumar, 2022). The SDD contains the architectural design based on the process steps defined in the PDD and the process scenarios defined in the TPD. These are translated to exact actions the bot is going to perform such as screen recordings, clicking on keys, buttons, drop-downs, etc. (Kumar, 2022). The SDD is the blueprint for the developers of the project (Vinocha, 2022) and allows for insights that could be missed without the SDD (Choudhuri, 2021). The SDD shows how the applications are involved and how logical decision exceptions are implemented (Choudhuri, 2021).

4.4 Testing

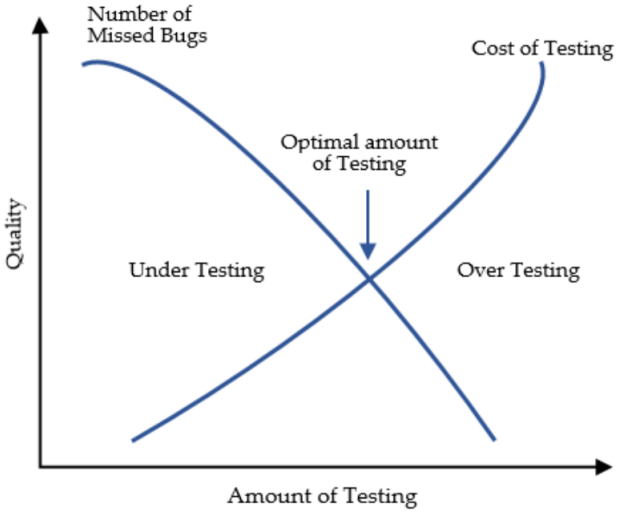
This section answers the MLR questions: *What are important aspects of testing?* in Section 4.4.1, *what testing methods are used for RPA?* in Section 4.4.2, and *w hat methods can help in aligning requirements and testing practices?* in Section 4.4.3. First, a background on testing is provided highlighting different testing activities, types, and levels. Secondly, the testing for RPA is discussed and novel testing methods are analysed. Thirdly, different testing methods that might be suitable for aligning requirements and testing are mentioned. Finally, in Section 4.5, a discussion is provided highlighting the benefits and limitations of the different testing methods in regard to their structure, traceability approach, and testing procedure.

4.4.1 Testing background

Testing is an important part of any DLC. If testing is planned well it can increase the quality of the product hugely but if testing is executed poorly it can harm the product or application (Ratilainen et al., 2019). Testing aims to uncover defects and can show the presence of defects but does not show their absence (Ratilainen et al., 2019). Therefore, it is important to spend enough time and effort on testing to

uncover errors. In Figure 4.1 the quantity of errors is depicted against the amount of testing where with too little testing many bugs remain and with over-testing the cost of testing becomes extremely high. Finding this optimum point of testing is important to ensure a high-quality application without spending too many hours on testing (Jamil et al., 2016). Testing should be integrated within the development process and is important to validate any DLC phase (Mařík et al., 2000).

Figure 4.1: Optimum effort for testing (Jamil et al., 2016)



Testing activities

The testing process is mentioned by Jamil et al. (2016) and Ratilainen et al. (2019). While both authors cover the same topics, some differences in naming are used. For the example of these testing activities, the naming of Ratilainen et al. (2019) is used. The activities consist of the following:

Table 4.6: Testing activities (Ratilainen et al., 2019)

Name	Description
Test planning	Goal is to develop, record, and communicate to stakeholders the scope of testing and the approach in testing towards the testing target.
Test monitoring	Monitors the testing phases and checks if it is going according to the test plan and expected progress.
Test design	Determines how the features need to be tested and designs the test scenarios and cases.
Test execution	Test procedures are performed, recorded and results are determined.
Test analysis	Includes the reporting of issues and getting relevant incident information to stakeholders so that issues can be resolved.

Goals of testing

There are multiple goals for testing. Ratilainen et al. (2019) and Sawant et al. (2012) mention three main ones. These are (1) evaluate and validate requirements, (2) inform the quality of the project, and (3) test to uncover issues. Sawant et al. (2012) add that to ensure quality and retain knowledge a traceable document should be kept and tests should include valid and invalid conditions. Additionally, both Ratilainen et al. (2019) and Sawant et al. (2012) add the importance of starting planning for testing early in the DLC. Dave (2020) describes five pillars that are crucial for RPA testing. These are (1) Requirement understanding, (2) Test data, (3) test scenarios and cases, (4) test execution, and (5) Defect management. (1) requirements management is very important to perform the subsequent test activities. After this step, the whole process and its requirements are clear and well-documented. (2) test data

is important for correct and complete test execution. Incorrect data leads to incorrect test results and reports which can lead to defects being missed. This data needs to be relevant to the current process that is being automated. (3) Test scenarios are created from the requirements while the test cases test these scenarios with the test data. This pillar makes sure that the test can be properly executed and that all requirements, rules, and exceptions are covered by the tests. (4) Execution performs the actual testing and reports the issues found. The last pillar is (5) defect management which makes sure that the defects found are actually fixed before a bot is deployed.

Testing techniques, types, and levels

There are different testing types and techniques to ensure the project's quality and the project's testing objectives. Not all authors agree on the extent and difference between testing types or techniques. First of all, Ratilainen et al. (2019) and Sawant et al. (2012) make a difference between dynamic and static testing techniques. Static testing is described as a manual exploration early in the DLC (Sawant et al., 2012) in which reviews or inspections are done on documents, requirements, code, or test plans (Ratilainen et al., 2019). On the other hand, dynamic testing involves running test scripts with software (Ratilainen et al., 2019) to check for correctness, performance or reliability within the application (Sawant et al., 2012). The second distinction of testing techniques is made by Jamil et al. (2016) and Sawant et al. (2012) who mention *black box* vs *white box* testing. *Black box* testing is focused on the functionality of the application without knowing the internal structure (Jamil et al., 2016; Sawant et al., 2012). These tests need to make sure that the application meets the specified requirements (Jamil et al., 2016) by verifying and validating that the correct input is accepted and generates the correct output (Sawant et al., 2012). *White box* testing is checking the system while being able to view the internal structure and code of the application (Jamil et al., 2016; Sawant et al., 2012). This testing technique is very effective in testing the application's logical decisions, loops, and internal data structure (Jamil et al., 2016; Sawant et al., 2012). These two techniques can be used within different testing types.

Not all authors agree on the extent of testing types. This paragraph describes the different testing types mentioned by Jamil et al. (2016), Kaur and Singh (2014), and Ratilainen et al. (2019). Firstly, Kaur and Singh (2014) and Ratilainen et al. (2019) both mention the *functional* and *regression* testing types. *Functional* testing can be viewed as a *black box* technique and checks if the system operates as predetermined by its requirements (Kaur & Singh, 2014; Ratilainen et al., 2019). It emphasizes the external behaviour of the system and checks if the input provides the correct output (Kaur & Singh, 2014). It can be measured by test coverage of the required functionality of the system (Ratilainen et al., 2019). *Regression* testing (i.e., change-related testing (Ratilainen et al., 2019)) is defined by testing the entire system after an update to fix an error or if new functionality needs to be added. It needs to test to see if the errors are fixed, the new functionality works properly, and all existing functionality still functions as before (Kaur & Singh, 2014; Ratilainen et al., 2019). Ratilainen et al. (2019) also mention the *non-functional* testing type. *Non-functional* tests target non-functional requirements (NFRs) such as stability, usability, performance, security, reliability, robustness, and data integrity among others. Writing test cases for this often requires sufficient knowledge and skill. Kaur and Singh (2014) also mention the testing types *Random, control-flow* and *data-flow* testing. *Random* testing is a black-box testing technique that creates and tests without structure. It is often rapidly done and limits the correlation due to its random nature. By itself, *random* testing is not a good testing type (Kaur & Singh, 2014). *Control-flow* testing is useful for white-box testing and assures that decisions and conditions within the code are executed. To execute all decisions the application's branches must be taken at least once. To execute all conditions, all decisions and their related conditions need to be tried for every value for this condition (Kaur & Singh, 2014). This means testing both the valid and invalid condition values. The last testing type discussed is *data-flow* testing. *Data-flow* testing focuses on the variable usage and what value they have in every position of the application. It can help to check if variables are defined correctly, used in the correct places and if their values correspond to what is expected (Kaur & Singh, 2014). This testing type uses a *white box* testing technique. From this paragraph, it is clear that many different testing types exist and that there is no consensus yet on what their importance is.

When we look at different testing levels there are four different levels described by Ratilainen et al. (2019) and Sawant et al. (2012). These four types are *unit, integration, system,* and *acceptance* testing. *Unit*

Table 4.7: Testing types

Testing type	Description
Functional	Test to check if the application has all previously specified requirements
Regression	Full application test to check if new features and old features still function properly
Non-functional	Test quality properties of application (e.g, stability, robustness, usability, etc.)
Random	Unstructured test case generation for rapid uncorrelated testing
Control-flow	Test all conditions and branches of an application
Data-flow	Test variable values though out the flow of the application

Table 4.8: Testing levels

Testing level	Description
Unit	Test a single module in isolation
Integration	Test integration of modules and their interdependencies
System	Test the whole application for internal and external integrity including functional and non-functional requirements
Acceptance	Test readiness of the application together with the customer or user. showcase all specified requirements effortlessly

testing focuses on a single module. The module is tested in isolation from the rest of the application to check if it has any errors and can perform its intended functionality (Ratilainen et al., 2019; Sawant et al., 2012). The test cases are often written and executed by the developer within the development environment and defects are fixed immediately (Ratilainen et al., 2019). It is usually seen as *white-box* testing since the test execution includes evaluating the implementation of the code (Sawant et al., 2012). *Integration* testing focuses on testing the integration of different modules together and interactions between them (Ratilainen et al., 2019). It can help define and test the program's structure and uncover issues related to the program interfacing (Sawant et al., 2012). *System* testing, also called end-to-end testing, is a test that happens when the whole application is finished. It tests each and every perspective from the application from beginning to end (Jamil et al., 2016). It is the first check to see if it is compliant with its specified requirements and the application is checked for correctness (Sawant et al., 2012). *System* testing includes end-to-end tasks of the system and contains functional and non-functional requirements to test (Ratilainen et al., 2019). With this, it is focused on gathering information on the whole quality of the system. Furthermore, it is important that the test environment is as close as possible to the production environment (Ratilainen et al., 2019). The last test level discussed is *Acceptance* testing or *user acceptance* testing. It is used to determine the readiness of the application to be deployed and used by end-users (Ratilainen et al., 2019). It is often carried out with the user or customer (Sawant et al., 2012). *Acceptance* testing falls under *black box* testing since the user is not interested in its internal structure but rather in it functioning properly. It should therefore be able to showcase all of its specified requirements effortlessly. Finding a considerable number of errors in this phase is a big risk to deployment since it should be the last testing phase to get the go-ahead from the customer to deploy the application (Ratilainen et al., 2019).

4.4.2 RPA testing and waterfall model

Testing is a crucial phase in RPA development to ensure the quality of a project (Cernat et al., 2020). When looking at what types of defects show up most in RPA projects we can put them into three main

categories: Application issues (42%), Infrastructure issues (28%), and Robot issues (24%) (Weishaar, 2022). Since RPA is a relatively new field, little research has been done on the best testing procedures (Enríquez et al., 2020). Currently, most testing for RPA is still done manually (Cernat et al., 2020) and according to the waterfall model approach (Cewe et al., 2017; Tran & Ho Tran Minh, 2018). The testing phase is important to verify if functionality works as intended and if there are no deviations. According to the waterfall model, testing is one of the last phases and occurs all at once after all the previous phases are finished (Balaji & Murugaiyan, 2012; Ghouse & Sipos, 2022). This means that the full testing of all hardware and software configurations are checked to see if it works as intended is checked after development (Petersen et al., 2009).

When looking at RPA testing and the waterfall model some limitations can be discussed. Firstly, the waterfall model is a step-by-step approach in which every phase is finished before the next starts. This results in the fact that testing is not really integrated into the development process and if scope creep happens during development it is more difficult to deal with (UiPath, n.d.). Secondly, the whole robot is tested all at once at the end of the DLC which is a big effort. Because of this, priority is often given to functional testing rather than non-functional (i.e., quality requirement testing) (Petersen et al., 2009). Additionally, if unexpected issues arise, they are often harder to fix and lead to higher costs or schedule overrun (Petersen et al., 2009). Thirdly, if development is delayed, in an effort to stay on schedule project might compromise its testing efforts (Petersen et al., 2009). Moreover, insufficient testing can lead to higher maintenance time and cost (UiPath, n.d.).

While very little research has been done on RPA testing there are some themes addressed in academic literature. The papers published on RPA testing focus on creating testing environments or on different testing method approaches. Jiménez-Ramírez et al. (2020) and Montero et al. (2019) focus on the fact that a testing environment for RPA projects is not always available. A testing environment is necessary to run fake applications to mimic the environment the robot runs on in production (Montero et al., 2019). If there is a discrepancy between the test and production environment issues can leak through to production (UiPath, n.d.). Additionally, without a test environment available, bots need to be tested in the production environment which poses a high-risk (Jiménez-Ramírez et al., 2020). Jiménez-Ramírez et al. (2020) and Montero et al. (2019) both propose to automatically generate a testing environment based on a User-interface log (UI log). The UI log contains the interactions between humans and the system (Montero et al., 2019). Therefore, this environment would mimic and present images of an application screen based on the UI-log actions. One limitation of this approach is that there needs to be a complete UI log available to be able to create this environment which is not always the case (Jiménez-Ramírez et al., 2020; Montero et al., 2019). Other research on RPA testing focuses on applying a different testing method. A paper by Cewe et al. (2017) focuses on using a test-driven development (TDD) approach. They propose this because they describe a waterfall model as inefficient with much over-engineered development and documentation (Cewe et al., 2017). TDD is an agile development approach in which test cases are written before development. Every test case is written and developed until it is passed. Only then a new test case is selected to be developed. This is where the name test-driven development comes from. Currently, this testing approach is in the concept stage and has not been tested in a case study or in practice (Cewe et al., 2017). Another approach focuses on automation testing with the use of model-based testing (MBT) (Cernat et al., 2020). Currently, testing scenarios and cases are generated and executed manually. Within MBT, a model is built that represents the application under test and this model can automatically generate test cases and data. However, this research also only has a theoretical foundation and has not been tested yet in practice. Cernat et al. (2020) describe a few limitations that limit the research. Firstly, there is a lack of models and creating these models requires a lot of knowledge. Secondly, it is difficult to integrate these models into the current testing tools that are used for RPA. Cernat et al. (2020) are currently trying to develop an MBT tool themselves in a PhD project.

As this section shows, there is very limited research currently conducted on RPA testing and an RPA literature study by Enríquez et al. (2020) showed that only 5% of the studies focused on testing. Therefore, in the search for suitable testing approaches that can align requirements and testing practices, an extended search has been conducted on testing approaches that could achieve this. This is described in detail in the next section

4.4.3 Testing approaches to align requirements and testing

The systematic literature review identified a couple of methods and approaches that can help in the alignment between requirements and RPA testing. Of course, Requirement traceability (RT) is part of this but this has already been discussed in depth in Section 4.2.1. This section goes more in-depth into the already defined method of *TDD*. While *MBT* is a testing method discussed in RPA literature, its focus is mainly on automation testing and not focused on requirements traceability. Furthermore, Cernat et al. (2020) pointed out limitations that are hard to address within the time scope of this research. Therefore, it is not discussed in depth in this section. However, the *Requirement-based testing* (RBT) method and the *V-model* is discussed since these are aimed at aligning requirements with testing.

Test driven development

Test-driven development (TDD) is focused on creating unit test cases prior to coding starts (George & Williams, 2004; Janzen & Saiedian, 2005). This happens in small, rapid iterations in which the minimum to pass a unit test is coded. After this, the next iteration of writing the unit test and coding the solution is performed (Janzen & Saiedian, 2005). For TDD an important rule is that 'If you can't write a test for what you are about to code, then you should not even be thinking about coding' (George & Williams, 2004, p. 337). TDD mainly focused on integrating testing first and then code. There is no clear model or structure besides what is meant before which makes TDD more a testing and development technique than a testing model (Janzen & Saiedian, 2005). TDD promotes a strong integration between testing and development (George & Williams, 2004). This allows for continuous integration testing throughout the project. George and Williams (2004) describes a couple of benefits of TDD. It allows for efficient coding and high testability of the code. Measuring the code quality can be done by monitoring the *number of defects found*, *defects per test case*, and *effort required to fix defects*. An experiment performed comparing TDD with waterfall development showed that TDD had higher code quality and passed 18% more tests but also took 16% longer for development (George & Williams, 2004). Nevertheless, while it is expected that TDD increases test quality, according to Turhan et al. (2010) there is insufficient evidence from industrial use. TDD is described extensively in literature but is used relatively little in practice (Turhan et al., 2010).

George and Williams (2004) also describe some concerns of TDD. Firstly, there is a lack of upfront design with TDD which can lead to a lack of conceptual integrity. Secondly, TDD is high-risk-reward. If it works it can lead to time and cost savings, but if it fails there is no fallback on explicit design and documentation. Third and Last, TDD requires the ability to create test cases by developers beforehand. Furthermore, not all code structures are easy for TDD such as for graphical use interfaces (George & Williams, 2004). This might limit its applicability for RPA.

Requirement-based testing

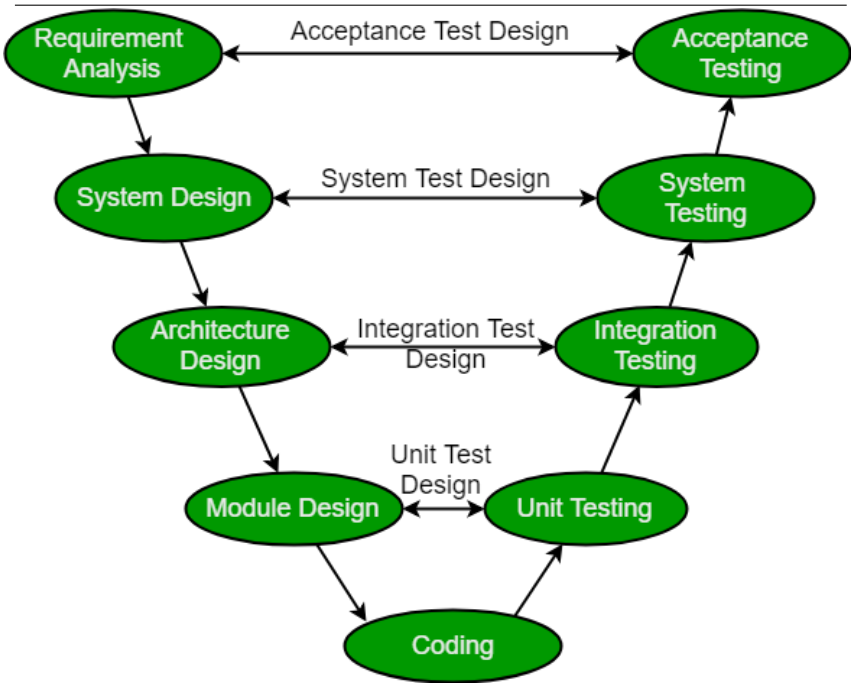
Requirement-based testing (RBT) tries to address the fact that incorrect or incomplete requirements are often the root cause of project failure. So RBT tries to make sure that requirements are correct and linked to testing so that they are properly validated (Mogyorodi, 2001). RBT tries to address two main issues. Firstly, focusing on correct and complete requirements (Mogyorodi, 2001) to ensure a high quality of requirements specification (Skoković & Rakić-Skoković, 2010). Secondly, the design of sufficient test cases to meet the requirements and validate them (Skoković & Rakić-Skoković, 2010). The first addressed issues should lead to early defect detection (Skoković & Rakić-Skoković, 2010). The earlier issues are found the least costly they are according to Mogyorodi (2001) (see Table 4.3). RBT is closely related to requirement traceability since it is focused on generating test scenarios and cases from a linked set of requirements. This traceability is crucial to monitor progress and test coverage as well as managing changes for requirements and related test cases (Skoković & Rakić-Skoković, 2010).

Skoković and Rakić-Skoković (2010) describe the different steps of RBT. These different steps focus on verifying the requirements with stakeholders and generating test cases based on these requirements. Furthermore, RBT focuses on the following testing activities: *Define test completion criteria*, *Design test cases* and *Verify test coverage* (Skoković & Rakić-Skoković, 2010). To do this, requirements need to be clearly written and testable. Mogyorodi (2001) mentions qualities of testable requirements such as

deterministic, traceable, correct, complete, lends itself to change control, feasible, logically consistent, and *non-redundant.*. These qualities should be considered when writing down requirements in the analysis phase. Lastly, Skoković and Rakić-Skoković (2010) mentions a few metrics for RBT such as *Percentage of requirements reviewed, Percentage of requirements with formal representation,* and *Percentage of formal requirements covered by formal test cases.* These metrics can provide insights into the effectiveness of the requirements alignment with testing and the completeness of the testing phase.

Figure 4.2: V-model[1]

[1] Source: https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/



V-model and W-model

The V-Model or *validation & verification* model (Balaji & Murugaiyan, 2012) is a life-cycle project process model (Johansson & Bucanac, 1999). It tries to reach project goals at the project level through the different DLC phases. Every phase of the DLC verifies the previous one and all DLC phases are validated with a specific testing type (Balaji & Murugaiyan, 2012). The V-model is similar to the waterfall model since it relies on completing a step before moving on to the next one (Balaji & Murugaiyan, 2012). Every phase in the V-model is linked with a type of test. For example, the requirements-gathering phase is validated by the acceptance test. These phases and their linked test type are shown in Figure 4.2. Through this structure, the V-model intends to discover issues as early as possible and have a closely integrated testing phase that is aligned with different steps in the DLC (Shuping & Ling, 2008).

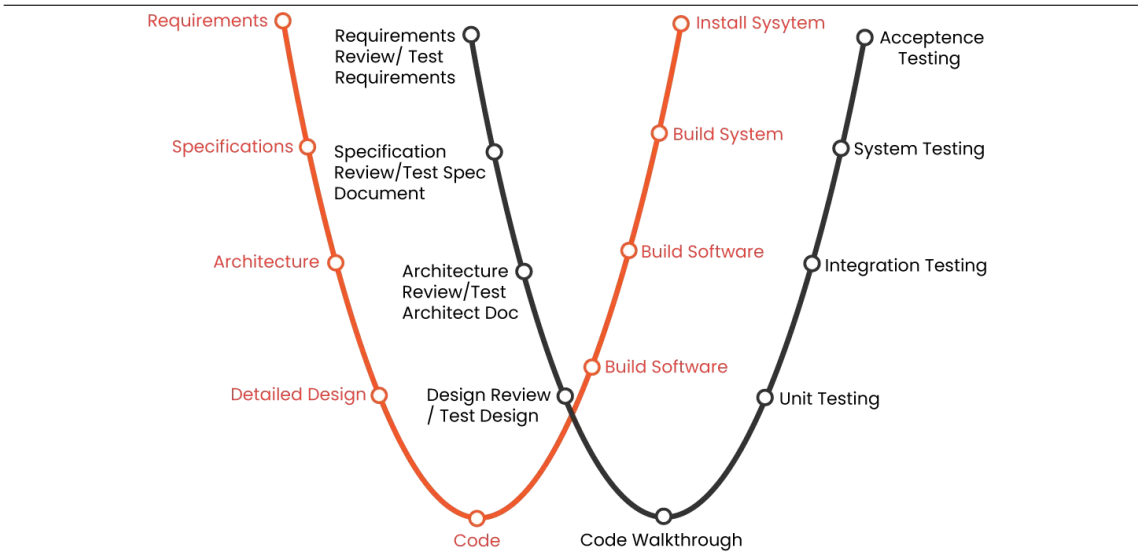
There are a couple of advantages and disadvantages described for the V-model. Balaji and Murugaiyan (2012) mention the similarity to the waterfall model, testing involved in the requirements phase, and the possibility of requirements change in any phase advantages for the V-model. (Regulwar et al., 2010) mentions two advantages. Firstly that the V-model can find defects in the early stages which reduces the costs to fix them. Secondly, since the V-model is project independent, it can be tailored to the needs of specific projects. However, also some disadvantages are described for the V-model. Balaji and Murugaiyan (2012) mention 3 disadvantages. Firstly that the V-model is very rigid and not flexible. Secondly, if changes happen both requirements and test documents need to be updated. Thirdly, short projects benefit less from the V-model since the V-model requires reviews at every stage. Johansson and Bucanac (1999) mention that while the V-model covers DLC activities, they are described in quite an abstract plain with little detail. Regulwar et al. (2010) mention that the V-model depicts a one-to-one

relationship between documents and test activities which is not always the case. because of these disadvantages, many different variations or upgrades of the V-model have been proposed by different authors.

Shuping and Ling (2008) propose an upgrade to the V-model in which test stages are overlapping and stresses the importance of the acceptance test. Regulwar et al. (2010) propose a different variation to the V-model. This is called the W-model. The W-model still presents a standard DLC with every phase linked with a test activity (Regulwar et al., 2010). The W-model is shown in Figure 4.3 and consists of two V's forming the W. The first V (shown in black) describes the phases and their documentation going down and then the development for these phases. The second V (shown in red) starts with the validation of the documentation going down and the verification of development with testing going up. The function of the tests is to determine if the objectives of development are met and the deliverable meets its requirements (Regulwar et al., 2010). Just like the V-model, the W-model is project independent and can be adjusted if a project has different development stages. Regulwar et al. (2010, p. 135) mention that the important thing about the W-model is that it "focuses specifically on the product risks of concern at the point where testing can be most effective".

Figure 4.3: W-model[1]

[1] Source: https://shiftasia.com/column/difference-between-v-model-and-w-model-in-software-testing/



4.5 Discussion on testing methods and traceability

This section compares the different testing methods on their benefits and weaknesses. This forms the foundation for the solution objectives in Section 6.1. All methods or techniques that have been discussed in the MLR are included. This includes *TDD, W-model, RBT, RT, MBT,* and *waterfall model*. The V-model has also been discussed but because of its great overlap with the W-model is not included. Also, the MBT and the waterfall model have already shown limitations to be implemented but for completeness are part of the discussion. The discussion focuses on which *DLC phases each method includes*, their *approach on traceability*, and their *procedure for testing*.

4.5.1 Development life cycle phases

TDD is a 7-step process in which requirements are converted to test cases which are in turn coded and implemented (Turhan et al., 2010). TDD does not have a clear design phase and mainly actually only uses requirement analysis, testing and coding as phases. TDD has clear but not very detailed steps and does not have a clear visual representation of its phases are procedures. Developers need a high understanding of the system they are coding to create test cases and code without a clear design (George & Williams, 2004). TDD is considered an agile development process (Cewe et al., 2017). This is quite a different approach than the waterfall model that is currently used for RPA (Tran & Ho Tran Minh, 2018).

The **W-model** incorporates multiple DLC phases including requirements analysis, design, development, and testing (Regulwar et al., 2010). It has quite visual elements but these elements are not explained in great detail leaving space for interpretation. The W-model tries to discover issues as early as possible in the DLC (Shuping & Ling, 2008) which reduces the cost of fixing these issues significantly (Mogyorodi, 2001). Furthermore, the W-model is focused on linking different design steps with verification through reviews and development with validation through different testing levels (Regulwar et al., 2010). This is also the reason why the W-model is sometimes referred to as the verification and validation model(Balaji & Murugaiyan, 2012). The W-model its phases and elements are visible in Figure 4.3. It is based on the waterfall model since it also follows sequential steps in its phases and tries to complete a phase before starting the next.

The **RBT** method consists of a 12-step process to align requirements with test cases (Skoković & Rakić-Skoković, 2010). It is very detailed in its description of these steps but has no visual representation. Since the RBT is only focused on aligning requirements with test cases, the DLC phases it covers are simply requirement analysis and testing. The RBT method does include a clear explanation of different reviews of the requirements. just like the W-model, RBT is focused on uncovering issues as early as possible in the DLC (Skoković & Rakić-Skoković, 2010) which reduces its fixing cost (Mogyorodi, 2001).

RT is mainly focused on linking requirements with different artefacts in a project including artefacts for design, development, and testing (Ziftci & Krüger, 2013). RT does not have a visual model of some sort, it does focus on linking requirements to the other mentioned phases. Therefore it can be said that RT includes the following DLC phases: requirements analysis, design, development, and testing. However, it does not focus on how to perform these phases, it only focuses on traceability within these phases. For RT, it is finding a balance between effort put into linking all project's artefacts and the benefit gained from this (Torkar et al., 2012)

MBT does not clearly specify different DLC phases. It is mainly focused on creating a model from the application that is developed. This model automatically generates test cases (Cernat et al., 2020). Because of this, it can be said that MBT only focuses on the testing phase. MBT does have a model which can be visualized. However, for the construction of this model are no clear guidelines provided.

The **waterfall model** has been researched and described a lot in literature and because of this different versions of the waterfall model exist. Nevertheless, the model at least includes an analysis, design, development, testing, and maintenance phase. These phases are performed in a sequential order and the next phase can only start if the previous one is finished. This means that the waterfall model cannot handle scope creep well (UiPath, n.d.). The waterfall model provides clear visual elements of these phases

but does not include guidelines as support for these elements.

4.5.2 Traceability approach

Firstly, **TDD** uses little documentation for its requirement analysis, design, or coding (George & Williams, 2004) which is a big part of traceability (Ziftci & Krüger, 2013). However, TDD does use a structure in which tests are created for a specific test case and from this code is built. This shows some link between a requirement, test, and code. Nevertheless, no clear procedure for proper traceability is provided for this. Therefore, no clear standpoint can be appointed for TDD and its use of traceability.

The traceability for the **W-model** is also unclear. The W-model has a clear structure going through analysis and design with proper reviews in place as verification and a clear development structure with link test level as validation (Regulwar et al., 2010). However, it does not specify linking requirements to a spot in design or development. Therefore, the W-model cannot directly be categorized as a model that includes traceability. It is unclear if these links within the model can include traceability aspects.

The **RBT** method is focused on creating links or traces from requirements to testing. It accomplishes this through a clear structure of requirements reviews, test generation based on requirements, and metrics to verify to what extent the requirements are covered by tests (Skoković & Rakić-Skoković, 2010). Therefore, it can be said that RBT uses traceability to align requirements to testing.

RT, just like RBT, has its main core focus on traceability. Compared to RBT, RT has a bigger scope since it includes all artefacts related to design, development, and testing. Furthermore, RT explains different techniques such as *requirement traceability matrix (RTM), rule-based,* and *process-based* approach(Torkar et al., 2012). Simple naming tags can also be used as a traceability approach. Most of these techniques need some form of tool to use them. However, further research should be done to identify the clear advantages and disadvantages of these different techniques.

MBT has no clear relation with traceability. While MBT creates a model based on the developed system, there is no procedure in place that checks if the model includes all the requirements or functionality. This means that MBT does not include a traceability approach.

The **waterfall model** also has no relationship with traceability. The waterfall model only focuses on the sequential phases of a software project but does not specify any relation or link between these phases. Therefore it can be concluded with certainty that the waterfall model does not include requirement traceability.

4.5.3 Testing procedure

TDD promotes a strong integration between testing and development through its unique and iterative process of first creating test cases and then developing the code (George & Williams, 2004). TDD is mainly focused on writing unit tests and does not address any other levels of testing.

The **W-model** includes 4 types of testing that are linked to different phases of development. These types are *unit, integration, system* and *acceptance* testing. Figure 4.3 shows this. The W-model is the only one that indicates these different levels of testing. The W-model also specifies to function of validation of testing and their relation to different design and development steps. These different testing levels are clearly defined in sequential order.

RBT is focused on converting requirements to tests which can be seen as functional and non-functional testing (Ratilainen et al., 2019). RBT does not specify other types of testing nor distinguishes different testing levels. RBT does specify quality attributes of requirements that make them more testable and provides metrics to check if the tests adequately cover the requirements.

RT its main focus is on linking requirements to for example test cases. However, no standard way of testing or any other guidelines on how to perform these test cases is provided. Also, no differentiation

between types or levels of testing is specified.

MBT is mainly focused on automatically generating test cases (Cernat et al., 2020) to test all the functionality. Through this, it has integrated testing within its development process and a clear standardized method for testing. However, Cernat et al. (2020) mentioned that their approach is still theoretical and pointed out a few limitations. Firstly, there is a lack of knowledge of how to create proper models and no tooling for this. Secondly, it is difficult to integrate these models into the current testing tools for RPA (Cernat et al., 2020).

The **waterfall model** includes a testing phase at the end of a project. Before testing starts, development has to be finished completely and all the testing happens in one go. This means that there is no integration between testing and development or another phase that came before the testing phase. This can lead to issues that are harder to fix (Petersen et al., 2009).

Method design

5. Problem-centred interviews

This chapter discusses problem-centred interviews. The problem-centred interviews were conducted with two solution architects (SAs) and one head of support. The interviewees are labelled Interviewee SA1, Interviewee SA2, and Interviewee Support1 respectively. They all signed a consent form which can be found in Appendix A1 in Section 16.1. The interviews were focused on the current requirements, testing practices in RPA, and the translation of these requirements into scenarios. The questions can be found in Appendix A2 in Section 16.2. The results are summarized in Section 5 and emphasize three main categories for which problems were discovered. These categories are *importance of requirement analysis, responsibilities of customer*, and *struggles with testing*. Within these three categories, multiple issues were highlighted, which are discussed in the next sections. The insights from these interviews are addressed in the last section and shown in Table 5.4

5.1 Importance of analysis

5.1.1 Discover all scenarios and exceptions in the analysis phase

Discover all scenarios and exceptions in the analysis phase is important to have a complete picture of what is in or out of scope for the project. Interviewee SA1 mentioned that "it is important to ask difficult questions [about the process] so that step by step all exceptions are explained". "The more detail in the analysis phase about exceptions the easier development" —Interviewee SA1. All exceptions define what scenarios exist within the process. "We with the client need to define a scenario for every flow through the process and every business rule exception is a scenario" —Interviewee SA1. It is important to understand all scenarios that should be developed. "If exceptions are discovered later this means that more scenarios might need to be incorporated. These exceptions then need to be discussed with the stakeholders to see if they are in or out of the scope of the project. If they are considered in scope, scope creep occurs.

5.1.2 Scope creep

Scope creep means that changes are made to the requirements or scenarios that are implemented in the project. "If exceptions are taken into account it is a form of scope creep" —Interviewee SA1. "Scope creep can be an issue" —Interviewee SA2. This can cause delays in development and testing for the project. For example, Interviewee SA2 mentioned that "a lot of changes were integrated that conflicted with the original code".

5.2 Responsibilities of the customer

The customer plays a big role in the success of a project. They are responsible for a clear description of the process, signing-off project documents, and proving correct and complete test data and environment. Issues with their responsibility are discussed.

5.2.1 Issues in the analysis phase

Issues in the analysis phase can be caused because of incomplete descriptions of the process or disagreements between subject matter experts (SMEs). One interviewee gave a clear reason why an incomplete process description was provided. "The process was discussed by a small group of SMEs that did not discuss all scenarios" —Interviewee SA2. This means that exceptions to the process were missed in the analysis phase. Another interviewee pointed out an issue that was caused because of a disagreement between SMEs about the process steps. "Last project there was a lot of discussion about the different process steps and how they are performed. The explanation of the process was vague. [...] If they don't agree with how their own process is performed they need to have a discussion because we cannot automate it then" —Interviewee SA1.

5.2.2 Proper document sign-off with customer

Proper document sign-off with customer was something that the interviewees missed. This was related to customers not checking documentation thoroughly or claiming that documentation was changed after sign-off. The SA translates the requirements into process scenarios. However, the customer does not always take the time or effort to check if this is done correctly. "Most of the time we translate it and the business says it looks good without actually reading or understanding it. Maybe it is a bit complicated or they are not invested enough. Sometimes you see they don't care" —Interviewee SA2. The customers are the experts in the process and if they do not check the documentation it can cause issues later in development. Another issue is that documentation is currently signed off online by the customer and this online document can still change. This causes that "we are sometimes accused that the document changed after the customer signed it off. [...] So basically an official sign with the client is something I miss" —Interviewee SA2.

5.2.3 Issues with providing test infrastructure, test applications, or relevant test data

Issues with providing test infrastructure, test applications, or relevant test data can have a big impact on the project's quality and success. First of all, the customer is responsible to provide relevant test data but this does not always happen. "You always ask for relevant and up-to-date test data but this is not always the case. The test data should be the same as in production" —Interviewee SA1. Another issue can be if the customer provides incorrect test data for a specific scenario. "Last project, the client needed to provide a set of emails that contained specific data that the robot searched on. However, the scenarios defined (happy flow) ended in business rules exceptions because of this incorrect data" —Interviewee SA1. This can cause delays in testing. The last issue discussed is related to the bot infrastructure and applications. The bot runs on system infrastructures and uses applications. Understanding how the applications function is quite important because "we are automating through the user interface of applications" —Interviewee SA1. The applications and their version is discussed at the beginning of the analyses phase. "We need to know what version of the application is used in the process and we need to document on which version we build the bot" —Interviewee SA2. To test if the but functions correctly, a test environment consists of the system infrastructure and applications the bot uses. All interviewees mentioned that there can be issues with the test environment. The interviewees all gave the same reason which was best summarized by Interviewee Support1. The "test environment is not provided by the customer or the test environment is not the same as the production environment" —Interviewee Support1. If the test environment is not the same, this means that the version of applications on which the bot is tested does not match the version on which the bot runs in production. All the above-mentioned issues can cause struggles with testing which is discussed in the next section.

5.3 Struggles with testing

5.3.1 Incorrect test data or, test infrastructure. or test applications

Incorrect test data or, test infrastructure. or test applications can cause issues for testing. If the customer does not provide relevant test data it is possible that issues only show up after deployment. "Exceptions that did not occur in the test data will show up in production" —Interviewee SA1. Considering the test environment, there were two issues that could occur. One was no correlation with the production environment and two was no test environment available at all. For the first, "if the test and production environment do not correlate we can pass a test in the test environment but still fail in the production environment" —Interviewee Support1. This means that the provided test applications are different from the applications on which the bot runs. This can result in issues only being discovered after deployment. If no test environment is available at all, development and testing need to happen in the production environment which is a high risk. "Testing in production feels high risk because you don't know what connections are in place and what the consequences are if you change something" —Interviewee SA2.

5.3.2 Lack of testing knowledge and traceability

Lack of testing knowledge and traceability is another struggle that the interviewees pointed out. "In general we don't have a lot of knowledge about testing" —Interviewee SA1. Interviewee SA2 mentioned that a clear standard for testing is missing. "I want a more standardized way of testing and going through a testing pipeline" —Interviewee SA2. Currently, no specific technique is used to go from requirements to the process scenarios that need to be developed and tested. "No technique is used to set up or define the scenarios. Use common sense" —Interviewee SA2. Furthermore, Interviewee SA2 mentioned that testing could be more integrated into the development. "I would like to see in our development framework a standard form of testing so we integrate testing more into development" —Interviewee SA2. Lastly, Interviewee SA1 mentioned that there are some traceability links in the documentation but these are not complete. "All scenarios are listed in the TPD. However, the scenarios are not linked to specific steps in the PDD. On a higher level though, the scenarios are linked to the design and all scenarios are tested" —Interviewee SA1. Therefore, a better understanding of traceability and its usefulness should be researched.

5.4 Insights from interviews

From the issues described in the previous three sections, a few insights have been gained that help define the solution objectives for the RPA testing method in Chapter 6. The insights are described in Table 5.1.

Table 5.1: Insights from problem-centred interviews

Insight	**Description**
Importance process analysis	A complete and thorough process analysis is important to understand the whole process and to specify all possible scenarios and exceptions of the process (i.e., main scenario and exception flow scenarios). (Section 5.1.1 and 5.2.1)
Importance infrastructure and application specifications	A robot needs to run on hardware infrastructure and software applications. These need to be clearly specified and a test environment needs to be provided for proper testing. (Section 5.2.3 and 5.3.1)
Importance of customer reviews and sign-off	All insights mentioned above need customer input. Therefore, it is important to have a thorough and proper review with the customer and an official documentation sign-off. (Section 5.2.2)
A better understanding of document traceability	Traceability within and between documentation is used but this is not consistent throughout all documents. Its applicability and usefulness for RPA need to be researched in more depth. (Section 5.3.2)
No clear standard for testing	There is a lack of testing knowledge or standards for testing. (Section 5.3.2)

6. Solution objectives

This chapter focuses on creating the first iteration of the RPA testing method. This first iteration is based on the problem-centered interviews from Chapter 5 and the literature study in Chapter 4.

6.1 First version of RPA method

Based on the main research question, the insights from Chapter 5, and the discussion in Section 4.5, the first version of the RPA testing method is formulated. This section explains the reasoning and selection process that led to the first version of the RPA testing method.

6.1.1 RPA method selection process

Requirements of the RPA method

Table 6.1 outlines the seven requirements used to establish the RPA DLC method. These requirements were formulated based on findings from the problem-centred interviews and MLR. The requirements address issues such as specifying the inclusion and exclusion of DLC phases, traceability, multiple requirements focused on testing, visual elements, and clear guidelines. Further details on each of these requirements can be found in Table 6.1.

Table 6.1: Requirements for RPA method

Requirement #	Requirement name	Description
1	RPA DLC phases	The RPA method should include all the DLC phases that are related to requirement management or testing
2	Requirement traceability	The RPA method should be able to trace requirements throughout the included DLC phases and align them with test scenarios
3	Integrated static testing	The RPA method should verify the requirements and design through proper reviews of the requirements and design with the customer
4	Integrated dynamic testing	The RPA method should validate the requirements, design, and code through proper testing
5	Standardized testing	The RPA method should include testing standards
6	Visual elements	The RPA method should have clear visual elements that are easy to read and understand
7	Clear guidelines	The RPA method should have clear guidelines for each element to explain them in detail and describe the activities that need to be performed.

RPA DLC phases in scope for the RPA method

To be able to meet Requirement 1, the different DLC phases that are in and out of scope need to be determined. To determine which phases are in scope we looked at the research question and the description of the different phases. All phases that contributed or benefited from requirement management, requirements traceability, or testing were included. Table 6.2 shows all the eight RPA DLC phases mentioned in Table 4.2 and specifies whether a phase is considered in or out of scope. The Table shows that five phases are considered in scope and three are considered out of scope. The included DLC phases are *Requirement management. Design, Development, Testing* and *Hyper-care*. This means that these five phases form the basis of Requirement 1.

Table 6.2: DLC phases included in the RPA method

Phase	In/Out of scope	Reasoning
Planning	Out of scope	Focused on the project plan and process selection rather than requirements or testing
Requirement analysis	In scope	Requirement analysis is an essential part of the Research question
Design	In scope	Design is constructed from the requirements. Requirement traceability is important to include within the design. This can verify if all the specified requirements or included in the solution design
Development	In scope	Consists of implementing the specified requirements. Requirement traceability can also be applied in the code. Testing its main purpose is to validate that the developed process works and has the intended functional requirements
Testing	In scope	Testing is an essential part of the RQ
Hyper-care	In scope	Essential part of analysing if requirements and exceptions are incorporated and work correctly in the production environment
Execution of bots	Out of scope	Only describes the bot going live and is not part of the research question
Support & maintenance	Out of scope	Support is not related to requirement analyses or management and their job description falls out of the scope of this research.

RPA method selection

To decide which method or methods are selected to base the RPA testing method on, a comparison is made between all of them. This comparison is shown in Table 6.3. The method that addresses the most requirements from Table 6.1 is selected. All methods discussed in this research are included in the comparison. The requirements are answered using Chapter 4 and specifically the discussion in Section 4.5.

Table 6.3: Comparison of methods

Table 6.1 requirements	**TDD**	**W-model**	**RBT**	**RT**	**MBT**	**Waterfall**	Legend:
1	?	+	-	+	-	+	+ = Requirement addressed
2	?	?	+	+	-	-	- = Requirement not addressed
3	-	+	+	-	-	-	? = Requirement insufficiently addressed
4	+	+	+	-	+	?	
5	+	+	-	-	+	+	
6	-	+	-	-	+	+	
7	-	-	+	+	?	-	

Overall, the W-model scores best across all metrics. It only scored a '-' on Requirement 7 and a '?' on Requirement 2 (i.e., traceability). Since no other method or technique scored positive on as many requirements, the W-model is the main component of the solution objective for the RPA method. However, to fulfil all requirements, the W-model needs to be extended to adhere to all requirements. This means that the role of documentation within the RPA DLC is an essential part of the RPA method to analyse how traceability can be included. Furthermore, the multiple-case study should result in guidelines that can support the RPA method.

6.1.2 RPA Verification and Validation method

The W-method is also called the *verification and validation model* because of its approach to verifying the requirements and design with reviews and validating the development with testing. These are also two requirements of the RPA method (Requirements 3 and 4) shown in Table 6.1. Since the focus of the RPA testing method lies on this verification and validation, the developed RPA method is called *RPA Verification and Validation method* (RPA VV-method). Based on the elements of the W-model, the first iteration of the RPA VV-method is created. The first version of the RPA VV-method is shown in Figure 6.1. The RPA VV-method shows a few modifications on the W-model. The W-model and its elements have been converted to RPA elements. These elements and their purpose can be seen in Table 6.4. These elements are based on the problem-centred interview insights shown in Table 5.4 and the MLR in Chapter 4. The RPA VV-method focuses strongly on verification and validation. This is visualized in Figure 6.1 by the arrows between for example requirement analysis and review requirements as verification, and validate bot and UAT as validation. Furthermore, the sequential steps going toward coding and from coding the different development steps are visualized by the bigger arrows. This is the RPA-VV method that is used for the interviews during the multiple-case study in Chapter 2.2.3.

Figure 6.1: First version of the RPA verification and validation method

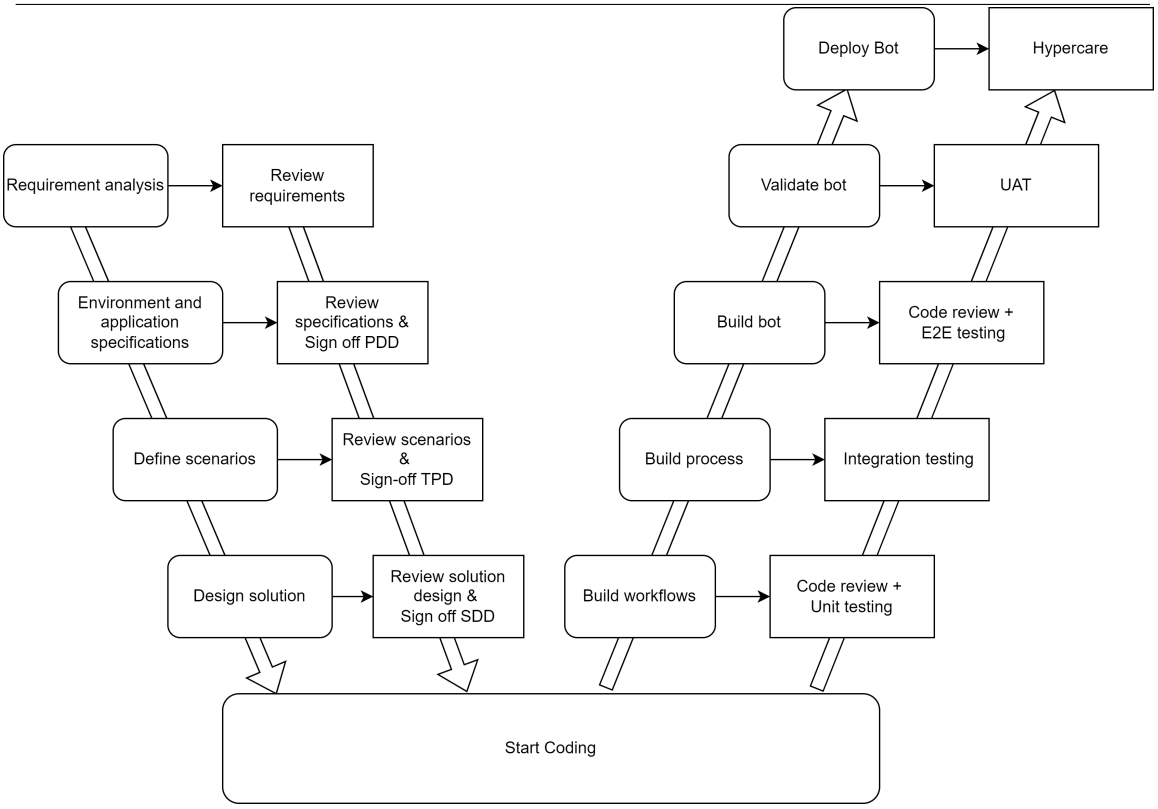


Table 6.4: RPA VV-method components

RPA VV-method component	**Description**
Requirement analysis	First analysis of process
Infrastructure and application specifications	Clarify and document infrastructure and applications specifications such as their version and test environment availability
Define scenarios	Define process scenarios of the process based on requirements
Design solution	Design solution based on scenarios and system specifications
Review requirements	Verify process requirements together with customer
Review system specifications	Verify process specification with customer and sign-off PDD
Review scenarios	Verify scenarios with the customer on correctness and completeness and sign-off TPD
Review solution design	Verify solution with the customer and sign-off SDD
Coding	Start coding the solution
Build workflows	Develop a single task in a workflow
Build process scenarios	Integrate all workflows of a scenario
Build bot	Integrate all scenarios of the process
Validate bot	Validate that the bot has all specified functionality
Deploy bot	Deploy bot in production environment
Unit testing	Test every workflow individually
Integration testing	Test that the integration of different workflows of a scenario work correctly
End-to-End testing	Test the entire process with all its different scenarios applications, and infrastructure
UAT	Test all the process scenarios together with the customer
Hyper-care	Monitor that the bot works correctly in the production environment and can complete all the scenarios successfully

7. Multiple-case study approach

The aim of the multiple-case study is to obtain technical and processual knowledge about the RPA development life cycle (DLC). Additionally, we intend to fully understand the use of documentation and tools within the RPA DLC. This chapter explains the approach that is taken for the multiple-case study. Below, an overview is provided of the objectives, the different analysis methods, and the criteria that are used for this analysis. A more extensive case study protocol is provided in Appendix B2 (17.2). The case study protocol explains how the cases are selected for the multiple-case study and which documentation and tools are included.

7.1 Overview

The multiple-case study consists of four projects that are described in Table 7.2. These four projects are analyzed through three different methods. The first is a thorough documentation and tool analysis, which is retrospective. The second method is a survey with the solution architects, and the third method is multiple systematic expert interviews. The interviews are aimed at gaining a better understanding of the RPA DLC and a deeper understanding of the use of documentation and tools. The result of the multiple-case study is best described by the following two objectives shown in table 7.1.

Table 7.1: Objectives description

Objective	Sources	Result Chapter
Improve and modify the activities of the RPA-VV method.	Systematic expert interviews	Chapter 9
Generate guidelines for the RPA-VV method and RPA project documentation.	All analyses methods	Chapter 10

Table 7.2: Case description

Case #	Case name	Department	Complexity level	Duration
1	Car rental reservation	Customer service	Low	18-04-2022 until 13-05-2022
2	XDM Import	Healthcare	High	19-07-2021 until 28-08-2021
3	Downloading From Sap	Supply chain	Low	04-07-2022 until 29-07-2022
4	RVO requests	Recruitment	High	02-05-2022 until 01-07-2022

7.1.1 Criteria

The documentation and tools are analysed based on the criteria described in Table 7.3. These criteria are incorporated within the survey and form the basis of the results section in Chapter 8.

Table 7.3: Documentation & tool analysis criteria

Criteria	Document analysis	Tool analysis
Completeness	Included	NOT included
Consistency	Included	NOT included
Functionality	NOT included	Included
Usefulness	Included	Included
Ease of use	Included	Included

7.1.2 Different analysis methods

This section describes the approach for the three different methods that are used within the multiple-case study in more depth.

Document and tool analysis

The document and tool analysis focused on the different tools and documents included in every project. The goal of this analysis was to fully understand the purpose of all documents and tools used within the RPA DLC and analyze their use based on the criteria described in Table 7.3. Only completeness and consistency could be analyzed in this method since ease of use and usefulness can only be judged by the practitioners themselves. Thus, the other two analysis methods were required for this. Which tools and documents are included is shown in the case study protocol in Appendix B2 in Section 17.2. Since the projects are already completed, this analysis is retrospective in nature. The insights from the document and tool analysis are written down in Chapter 8 in a within- and across-case section. Furthermore, the results are used as inspiration for the systematic expert interviews and form the foundation for the guidelines in Chapter 10.

Survey

The aim of the survey was to obtain expert opinions on the use of documentation and tools in RPA projects. To assess the documentation and tools, a set of criteria outlined in Table 7.3 were used, and sixty-two questions were asked. The survey paid particular attention to the usefulness and ease of use of the documentation and tools, as these aspects could not be evaluated by analyzing the documents and tools alone. The survey questions can be found in Appendix B3, Section 17.3. A total of five RPA experts were interviewed, and the survey responses are presented in Appendix B4 (17.5). Some of the insights of the survey are discussed in the across-case analysis in Section 8.2. Additionally, the survey results served as a reference for the systematic expert interviews and laid the groundwork for the guidelines presented in Chapter 10.

Systematizing expert interviews

Several expert interviews were conducted as part of the multiple-case study. The interviews consisted of two parts with the solution architects of the different projects within the multiple-case study as participants. The interview database is available in Appendix B1, Section 17.1.1. The first part of the interviews focused on comparing the RPA development life cycle (DLC) with the RPA-VV method proposed in Section 6.1, using the initial iteration of the RPA-VV method visible in Figure 6.1. The interviews were conducted with four participants as shown in Table 17.1. All participants filled in the consent form shown in Appendix B5 in Section 17.5. The interview lasted for an hour and all the different phases were discussed in a structured manner. The interview questions can be found in Appendix B6 in Section 17.6. The second part of the interview involved a thorough investigation of the use of documentation and tools within the RPA DLC. The interview lasted for thirty minutes, and the questions were modified according to the results of the document analysis and survey responses. Therefore, the interview questions were adjusted for all five Interviewees as shown in Table 17.1. Their interview questions are available in Appendix B7 in Section 17.7. Some of the insights from these interviews are described in Chapter 8. The interviews were transcribed and coded using Nvivo[1]. The coding was based on the following themes: *prerequisites DLC phase, essential activities, roles involved*, and *definition of completeness for a DLC phase.* These themes have led to improvements in the RPA-VV method described in Chapter 9. Additionally, these interviews, together with the results of the document and tool analysis and the survey, provide support for the guidelines in Chapter 10.

[1] https://lumivero.com/products/nvivo/ retrieved on April 26, 2023

8. Multiple-case study analysis

This chapter describes the different analyses from the multiple-case study. Both the within-case analysis and across-case analysis in Sections 8.1 and 8.2 are discussed respectively. Both the document and tool analysis and the survey have contributed to these results. The within-case analysis closely examines three different pieces of documentation found in one project, namely PDD, SDD, and TPD. The across-case analysis, on the other hand, compares identical documents from different cases and covers the usage of tools in projects. The tool analysis includes UiPath Studio, Azure DevOps, and Monday.com. Although UiPath Studio is used for testing and development, it could not be incorporated because access could not be given. Hence, the across-case analysis only provides a small introduction to the tool itself. Furthermore, access could not be granted to the Azure DevOps environment of Cases 3 and 4 since it was hosted on a private server of the customer. Thus, only Cases 1 and 2 are included in the across-case analysis for Azure DevOps. Lastly, the limitations of the study and potential future work are discussed.

8.1 Within-case analysis

Below is a discussion of the four cases and their within-case analysis. The within-case analysis dives into a single case to understand the use of the documentation within that case (Ayres et al., 2003). The PDD, SDD, and TPD are analysed for every case on their completeness and consistency. The ease of use and usefulness are also part of the document analysis but could not be judged solely on the documentation. Therefore, only the completeness and consistency are discussed in this section. The different documents that are analysed are given a database number. This is added to Appendix B1 (17.2). A Summary of the completeness and consistency of the within-case analysis is shown in Table 8.1. They are categorized in *high, medium*, and *low*. They were categorized as high if no more than one discrepancy was found. Medium if between two and five discrepancies were found. If more than five discrepancies were found they were categorized as low. A more extensive explanation of the four cases and their results are described in the following sections.

Table 8.1: Insights document and tool analysis

Case	Document	Completeness	Consistency
Case 1	PDD	High	Medium
	SDD	Low	Medium
	TPD	Medium	Medium
Case 2	PDD	High	Medium
	SDD	High	Medium
	TPD	Absent	N.A
Case 3	PDD	High	Low
	SDD	High	Low
	TPD	Absent	N.A
Case 4	PDD	High	Medium
	SDD	High	Low
	TPD	High	Medium

Case 1: Car rental reservation

Case introduction

Case 1 is about automating the car rental process of a customer service centre. Data fields had to be copied back and forth between the two applications that were used in the process. The bot was attended, which means that the user can interact with the bot and activate it as needed. Due to the fact that there were just two applications and 10 steps in the process, it was classified as low complexity. Five business-rule exceptions (BE) were displayed in the PDD, while seven test cases were in the TPD. Three documents, the PDD (D1), SDD (D2), and TPD (D3), are relevant to this case.

Results

The PDD (D1) is entirely filled out in regard to completeness. The SDD (D2) and TPD (D3), however, lacked certain components. In the TPD, the test data and test results were absent, whereas for the SDD (D2), only the solution design was present and all other components were absent. This demonstrates that not all of the documentation for this case was complete.

Regarding consistency, a standardized format was used throughout all of the documents. However, several discrepancies or missing linkages between the PDD (D1), SDD (D2), and TPD (D3) were found. The various scenarios and exceptions that are part of a project are listed in the PDD (D1). All of these should have been tested, but the TPD lacked several scenarios (D3). This demonstrates that the information is not always consistent across documents. Also, the solution diagram in the SDD (D2) included the exceptions that were specified in the PDD (D1). Yet, their names were different in the documents, which was another discrepancy discovered.

Case 2: XDM Import

Case introduction

Case 2 involved an automated process for importing XDM photos into the HIX application. For this, it had to discover the relevant patient in Hix, create a file and upload the images into it. The process was regarded to be complicated. Although there are only nineteen process stages and one application, there are numerous decision points that can result in different process flows. There was a two-week delay in the project. The PDD (D6), SDD (D7), and TPD (D8) are the related case documents.

Results

We discovered that the PDD (D6) and SDD (D7) for Case 2 are both nearly complete when evaluating how detailed the documentation is. Compared to the template, just one PDD element and two SDD elements were missing. The SDD also has a substantial appendix with a list of the solution diagram components and information on their variables (D7). For Case 2, the TPD (D8) was completely blank. This indicates that it is incomplete and that nothing regarding consistency can be said.

Several problems could be found when the consistency between the PDD and SDD is taken into account. The PDD appeared to describe different process steps than those depicted in the SDD. As a result, the consistency is considered to be limited (D6, D7).

Case 3: Downloading from SAP

Case introduction

Case 3 involves a process that downloads three different files of transactions from SAP. It was classified as a low-complexity project since it only includes eight steps, uses one application and has no decision points indicating different flows. The documentation of this case is as follows: PDD (D11), SDD (D12), and TPD (D13).

Results

The PDD (D11) and SDD (D12) are both rather well filled up when completeness is taken into account. The PDD lacked the prerequisite component but this could be because there were no prerequisites for this project (D11). The SDD only lacked the debugging tips components, which might have been because there weren't any for this project as well. As a result, it can be argued that the PDD (D11) and SDD (D12) are very full. The TPD (D13) is entirely blank and thus incomplete.

Upon examining the consistency of the documentation, we were unable to locate any discrepancies within either the PDD (D11) or SDD (D12). However, there were multiple contradictions between them. First off, the PDD and SDD used different naming conventions for the same exceptions, making it challenging

to identify their connection. This was the case for four of the exceptions. Moreover, the PDD and SDD both specified the robot's start time, although they did so at 6 AM and 7 AM, respectively (D11, D12).

Case 4: RVO requests

Case introduction

RVO queries are automated by Case 4. Employees currently spend too much time on this repetitious work, which is why an RPA project was chosen for it. Case 4 is categorized as a high-complexity process since it makes use of three applications, has 31 phases in the process, and has five exceptions that must be taken into account. The PDD (D15), SDD (D16), and TPD (D17) are the documents relevant to this case.

Results

The PDD (D15) is pretty elaborately filled in when completeness is taken into account. Each component is filled in with thorough documentation of all the process phases, exceptions, and input data. The SDD (D16) also had a comprehensive solution diagram and almost all components were filled in. The environmental details and debugging tips were the only components that were empty. Moreover, the TPD (D17) was largely filled in. A comprehensive list of all the exceptions that were encountered during development was documented, and sixteen test scenarios were included and tested. Also, a concise description of the various input data was given. The UAT and hypercare data were the only things that were missing.

No inconsistencies were observed in the PDD (D15) or TPD (D17), but some inconsistencies were discovered in the SDD (D16). Similar exceptions were categorized as system exceptions and business rule exceptions (BRE). Moreover, although there were two alternative catches for the BRE described, only one was shown in the solution diagram. After comparing the three papers for consistency, we found that the PDD (D15) and SDD (D16) have different naming practices. Also, we were unable to discover the BREs in the SDD that were indicated in the PDD. Moreover, differing naming standards between the identical cases in the PDD (D15) and TPD (D17) were found.

8.2 Across-case analysis

This section discusses the across-case analysis. It is divided into a documentation and tool part. The documentation part focuses on the difference between the documents and includes all previously mentioned documentation (D1-D3, D6-D8, D11-D13, D15-D17). The tool analysis includes all the tools for all cases which are Monday.com (D4, D9, D14, D18) and Azure DevOps (D5, D10). For the tool analysis, a short introduction of the tools is provided and their functionality for the RPA DLC is explained. Furthermore, the results of the survey are shortly addressed.

8.2.1 Documentation

We discovered a significant disparity in the degree of completeness when we take all the cases into account. The only document that was persistently complete was the PDD which is shown in Table 8.1. This can be related to its usefulness which score a 4.2 in the survey which is high. These results are shown in Table 8.2. While all cases use the same standardized format, this was not used to the same extent. Cases 1 and 4 were overall filled in the best with the other two projects missing the entire TPD. The TPD and SDD were filled in the worst while this did differ widely per case. The SDD completeness ranged from low to high and the TPD form absent, to medium, to high. Especially the fact that the TPD was absent in two cases is apparent since its usefulness in the survey scored a 4.2 which is high. This is shown in Table 8.2. For the SDD, the only elements that were always filled in were the solution diagram and environmental details. When considering completeness, this big difference between cases is quite apparent.

All cases showed a form of inconsistency within their documentation. This was mainly clear in the different scenarios or exceptions mentioned for a robot in the PDD, SDD and TPD that used different words

to describe them. This made it difficult or impossible to link these scenarios or exceptions together between the documents. Other reasons for this inconsistency were missing or additional scenarios. Another inconsistency found was how the process steps in the PDD were documented. The main differences were the details mentioned for every process step and the format of how alternative scenarios were written down.

When considering the usefulness and ease of use in Table 7.3 we can see that the usefulness of all documents is relatively high. Only the SDD scored a 3.3 with a high standard deviation suggesting that the participants did not fully agree on the usefulness of the SDD. The ease of use had a medium score between 2.9 for the SDD and 3.8 for the PDD. However, all had a high standard deviation above 1 indicating that also for the ease of use, the participants did not fully agree with each other.

Table 8.2: Documentation survey statistic

Criteria		**PDD**	**SDD**	**TPD**
Usefulness	Mean	4.2	3.3	4.2
	Standard deviation	0.8	1.2	0.7
Ease of use	Mean	3.8	2.9	3.7
	Standard deviation	1.2	1.3	1.1

8.2.2 Tools

Monday.com[1] is a tool for project planning and team communication and offers a range of features, including the ability to create tasks, assign owners, and change task statuses. For each of the several stages of the development life cycle, a common template is utilized inside the cases, and the essential tasks are listed here. This template is used for all projects.

We discovered that not every case made full use of the tool. Case 1 made the most of the tool possible across all projects. The majority of the tasks had their status changed to "done" and sub-tasks had been added for the development sprint with all the different workflows of the process. Code reviews were additionally incorporated for all workflows. Cases 2 and 3 made the least use of the tool and had a lot of unassigned, open, or incomplete tasks on the project planning template. Moreover, no tasks were established for the various workflows, and no code review was found. This suggests that either they are not carried out or Monday.com is not used for its stated project planning purpose. The project planning template was used in Case 4 and the majority of the tasks at the start of the development life cycle were completed. Several jobs were missing in the later phases, and no more code reviews were performed. Generally, we can claim that the completeness is limited and that the amount of use of Monday.com varies greatly from case to case. This is confirmed with the survey results shown in Table 8.3 that showed a relatively high score for Monday.com but also a high standard deviation.

Table 8.3: Tool survey statistic

		UiPath studio	**Azure DevOps**	**Monday.com**
Functionality	Mean	4	4.6	4
	Standard deviation	0.9	0.6	1.3
Usefulness	Mean	4.4	4.1	4.1
	Standard deviation	0.8	0.9	1.2
Ease of use	Mean	3.9	4.4	4
	Standard deviation	0.9	0.5	1

Using Azure DevOps[2], the various project team members can work closely together on the development of the project. Because it is a cloud service, anyone with access to the drive can access the data from any location. Among other things, it provides pipelines, test plan functionality, and code management.

[1] https://monday.com/ retrieved on April 26, 2023
[2] https://azure.microsoft.com/en-us/products/devops/ retrieved on April 26, 2023

It is only utilized in cases for its code management capabilities to save and manage the robot. To do this, several repositories for the project its data, frameworks, and different applications are used (D5, D10). Case 1 (D5) appeared to be better organized than Case 2 (D10) since Case 2 contained numerous code fragments and test cases that were commented out. This is regarded to be bad practice (Pham & Yang, 2020). This reveals some inconsistency between the two cases. Considering the completeness, the tool its code management functionality is adequately utilized. But, in general, projects do not use the full capabilities of the feature within the tool. This is supported by the survey results of survey question fifty-two shown in Appendix B5 in Table 17.5 that asked if the tool should be used for more functionality. Overall, Azure DevOps scored high on their functionality, usefulness, and ease of use which is shown in Table 8.3.

UiPath studio[3] offers advanced automation software to build robots. Additionally, it offers a wide range of tools to manage and test them. It offers testing functionality in the form of generating test cases for workflows and a test management tool that offers more testing functionality. Nevertheless, from the interviews, it was apparent that this test management tool is not used —Interviewee 2B. Within projects only the development is used and the workflows are run as a form of testing. This limited the insights into testing since no record in the form of a test case is generated. Nevertheless, UiPath studio scored high in functionality, usefulness and ease of use with standard deviations under the 1 shown in Table 8.3.

[3] https://www.uipath.com/product/studio retrieved on April 26, 2023

9. Improved RPA-VV method

Through the systematizing expert interviews, it was determined that the first iteration of the RPA-VV method in Figure 6.1 needed a few changes. There were two main issues that needed attention. The first was that the W-model structure shown in Figure 6.1 is not suitable for the RPA development life cycle. The second issue was related to the names of the activities within the first iteration RPA-VV method. The following section further explains the two issues and present their solution.

W-model structure not suitable for the RPA development life cycle

The original W-model itself is structured in the form of a 'W' as can be seen in Figure 4.3 since it links the first phase of requirements to the last phase of acceptance testing. However, during the interviews, it was apparent that this link does not exist as clearly for RPA. A second reason why the W-model structure has limited benefits for RPA is that within the RPA development life cycle (DLC) there are essential deliverables that are crucial for different phases of the project. "There are prerequisites or deliverables from previous activities that should be finished before we continue with the project" — Interviewee 1A. Some were already incorporated in the first iteration of the RPA-VV method in Figure 6.1, such as 'Sign-off PDD'. Nevertheless, during the interviews, it became clear that there were more essential deliverables that should be included. The structure of the W-model is not suitable for including a full list of these deliverables. Lastly, the W-model has no formatting rules specified which limits adding additional activities or deliverables to the model in a well-ordered manner. Therefore, with limited reason to use the W-model structure and with the necessity to include deliverables a different modelling format is needed.

To incorporate the deliverables of the RPA development life cycle within the RPA-VV method, we have opted to use the process-deliverable diagram by van de Weerd and Brinkkemper (2009). The original process-deliverable diagram shows the development life cycle on the left with its phases and activities. The related deliverables are visible on the right. This structure can be seen in Appendix C1 (18.1). Nevertheless, while the deliverables are essential for the RPA DLC, the relation between an activity and deliverables is difficult to see. Since the RPA-VV method should be used by RPA practitioners we have decided to simplify the structure of the process-deliverable diagram. The phases and activities are still the same but the deliverables have been replaced with a *checklist*. These checklist items resemble the deliverables and need to be checked off to finish an activity. This structure is used for the RPA-VV method that is used in the validation and evaluation and can be viewed in Figure 9.

Ambiguous names for the activities in the RPA-VV method

The second issue was that the names of the different activities were too ambiguous to derive their intended functionality. The names of components for the first iteration of the RPA-VV method shown in Figure 6.1 were based on the W-model itself and literature found on the RPA DLC. However, the Interviewees had different interpretations of what activities were part of which component. "Build a bot and validate bot are not very clear names for these phases of the project." — Interviewee 2A. To reduce the different interpretations of the phases and activities, names are adjusted to suggestions made my the experts during the interviews. Additionally, descriptions are beneficial to learn the details of an item (Williams & Lombrozo, 2010). Therefore, Table 9.1 and 9.2 are added to explain all the details of the different activities of the RPA-VV method.

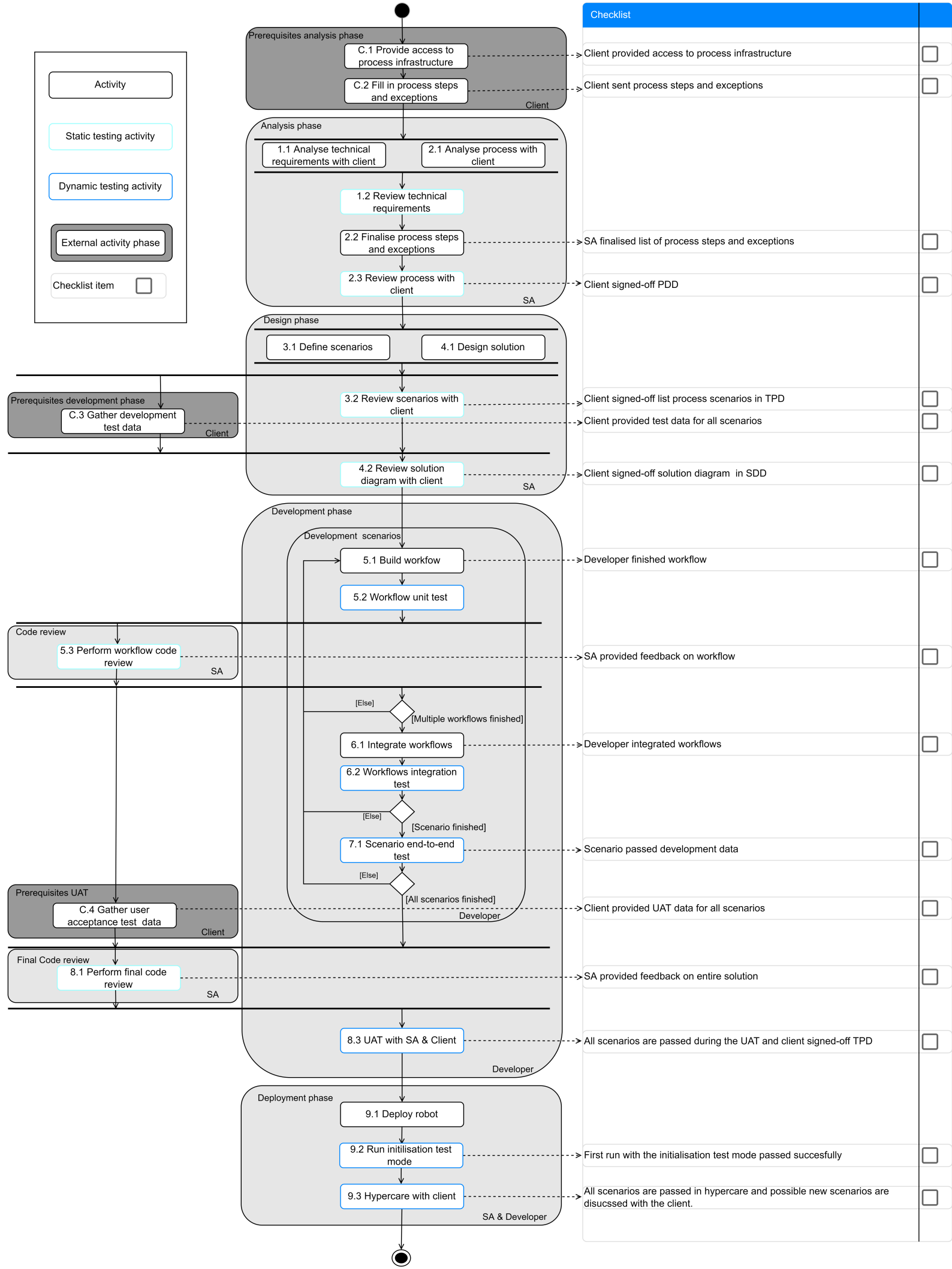

Activity
Static testing activity
Dynamic testing activity
External activity phase
Checklist item
Prerequisites analysis phase
C.1 Provide access to process infrastructure
C.2 Fill in process steps and exceptions
Client
Analysis phase
1.1 Analyse technical requirements with client
2.1 Analyse process with client
1.2 Review technical requirements
2.2 Finalise process steps and exceptions
2.3 Review process with client
SA
Design phase
3.1 Define scenarios
4.1 Design solution
Prerequisites development phase
C.3 Gather development test data
Client
3.2 Review scenarios with client
4.2 Review solution diagram with client
SA
Development phase
Development scenarios
5.1 Build workfow
5.2 Workflow unit test
Code review
5.3 Perform workflow code review
SA
[Else]
[Multiple workflows finished]
6.1 Integrate workflows
6.2 Workflows integration test
[Else]
[Scenario finished]
7.1 Scenario end-to-end test
[Else]
[All scenarios finished]
Developer
Prerequisites UAT
C.4 Gather user acceptance test data
Client
Final Code review
8.1 Perform final code review
SA
8.3 UAT with SA & Client
Developer
Deployment phase
9.1 Deploy robot
9.2 Run initilisation test mode
9.3 Hypercare with client
SA & Developer
Checklist
Client provided access to process infrastructure
Client sent process steps and exceptions
SA finalised list of process steps and exceptions
Client signed-off PDD
Client signed-off list process scenarios in TPD
Client provided test data for all scenarios
Client signed-off solution diagram in SDD
Developer finished workflow
SA provided feedback on workflow
Developer integrated workflows
Scenario passed development data
Client provided UAT data for all scenarios
SA provided feedback on entire solution
All scenarios are passed during the UAT and client signed-off TPD
First run with the initialisation test mode passed succesfully
All scenarios are passed in hypercare and possible new scenarios are disucssed with the client.

Table 9.1:

Activity nr.	Deliverable	Component name	Description	Interviewee
C.1	Client provided access to process infrastructure	Provide access to process infrastructure	Provide accounts with the correct rights to access applications and environments such as databases	[1A, 3A, 4A]
C.2	Client sent process steps and exceptions	Fill in process steps and exceptions	Client aligns itself on the process steps and documents the process steps and exceptions.	[1A, 2A, 4A]
1.1		Analyse technical requirements with client	Client provides insights into all technical requirements such as applications and environments	[1A, 3A, 4A]
1.2		Review technical requirements	Verify that access is granted to applications and understand how process steps are performed in the application	[2A - 4A]
2.1		Analyse process with client	Client provides insights into all the steps and process exceptions that need to be automated.	[1A - 4A]
2.2	SA finalised list of process steps and exceptions	Finalise process steps and exceptions	Fill in PDD with all previously gathered information about the process	[1A, 2A, 4A]
2.3	Client signed-off PDD	Review process with client	Verify with client that process steps and exceptions are understood and documented correctly and sign-off PDD	[1A, 3A, 4A]
3.1		Define scenarios	Define a list of scenarios which includes all happy flows, alternative flows, and exceptions.	[1A, 2A, 3A, 4A]
3.2	Client signed-off list process scenarios	Review scenarios with client	Client reviews and checks if the list of scenarios is complete and provides feedback. The client signs-off the list	[1A, 2A, 3A]
4.1		Design solution	Create the technical blueprint of how the robot will automate the process	[1A, 2A, 3A, 4A]
4.2	Client signed-off solution design	Review solution design with client	Verify the solution diagram with the client to check if the input and output is correct and steps in between are correct.	[1A - 4A]
C.3	Client provided test data for all scenarios	Gather test data	Provide test data for all scenarios	[1A - 4A]
5.1	Developer Finished Workflow	Build workflow	Develop the different workflows from the design one by one	[1A, 2A]
5.2		Workflow unit test	Debug the workflow and make sure it can perform all its required actions	[1A, 2A]
5.3	SA provided feedback on Workflow	Perform workflow code review	Review the workflow and verify it adheres to the engineering principles	[2A, 3A]

Table 9.2:

Activity nr.	Deliverable	Component name	Description	Interviewee
6.1	Developer Integrated workflows	Integrate workflows	Combine a set of workflows together	[1A - 4A]
6.2		Workflows integration test	Test if workflows are integrated correctly and debug if necessary	[1A, 2A]
7.1	Process scenario passed development data	Scenario end-to-end test	Test if a process scenario can run and produce the expected outcome	[1A - 4A]
8.1	SA provided feedback on entire solution	Perform final code review	Review the entire solution and give feedback if necessary	[1A - 4A]
C.4	Client provided UAT data for all scenarios	Gather UAT data	Client gathers data for all the test scenarios that are performed during the UAT	[1A - 4A]
8.2	All scenarios are passed during the UAT was successful and client signed-off TPD	UAT with SA & Client	Showcase to the client that the robot can perform all scenarios	[1A - 4A]
9.1		Deploy bot	Move robot to the production environment and check if the robot can can function correctly in debug mode	[1A - 4A]
9.2	First run with the initialisation test mode passed successfully			
9.3	All scenarios are passed in hypercare and possible new scenarios are discussed with the client.	Hypercare with client	Let the robot run on production and validate it can perform all scenarios correctly. Debug if necessary	[1A - 4A]

10. Guidelines

This chapter focuses on the insights gained from the different parts of this research including the problem-centred interviews and multiple-case study. The multiple case study consists the documentation and tool analysis, survey, and systematic interviews. This chapter includes nine issues that were discovered. These issues are described and supported by at least one guideline. A guideline is a specific statement of instructions and can be understood as a statement to offer advice or to plan a course of action about how to reduce the negative impact of the issue (Vanclay, 2003).

The following sections include a description of issues and related guidelines. The sections are aimed at addressing two distinct aspects of the research. Specifically, Sections 10.1, 10.2, and 10.3 focus on issues discovered during various activities of the RPA development life cycle. The guidelines provided in these sections offer courses of action or advice to minimize the negative effects of these issues. Notably, these guidelines are also integrated into the RPA-VV method. On the other hand, Sections 10.4, 10.5, and 10.6 describe issues related to the documentation and tooling employed in the RPA development life cycle. The related guidelines serve as advice and are not integrated into the RPA-VV method. Each section is structured such that the underlying reasoning behind the issues is first outlined, followed by the formulation of different guidelines with related sources and an illustrative quote. In cases where further explanations or instructions are necessary, they are also provided.

10.1 Client input for the project

Issue I: The quality and success of an RPA project are limited without the client's input in the form of multiple contact points and deliverables.

In the RPA development process, the client plays a crucial role in ensuring project success due to their extensive process knowledge. Multiple points of contact with the client occur throughout the project, beginning with the analysis phase where the client acts as the process expert. As stated by Interviewee 4A, "The client showcases the process of what we are going to automate. [...] We try to discover everything we do not know and they do know." This reinforces the significance of a comprehensive analysis phase, which was also emphasized in problem-centred interviews. Additionally, the client is responsible for providing crucial deliverables, such as test data, which are necessary for project progress. According to Interviewee 3B, incomplete or missing deliverables can impede project progress, "We need enough test data to test everything. Sometimes the customer does not have data prepared and this can be an issue for testing." Discussions, reviews, and deliverables are among the various contact points with the client and are essential for project quality and success. Guidelines 3, 4, and 5 outline the necessary steps to establish effective communication and collaboration with the client.

Guideline 1: Clear communication with the client is important since their subject matter experts and process owner are closely involved in all project phases.

Expert interviews	Interviewees 1A-4A
Illustrative quote	"There needs to be a minimum of one subject matter expert (SME) and one project owner present. The client needs to provide an SME that knows the process well and is aligned with the other SME's [...]. "They are involved in the analyses phase [...], reviewing process scenarios [...], and important for the UAT" — Interviewee 1A.

The subject matter expert (SME) and product owner are closely involved throughout the DLC. Their role is crucial within the analyse phase which is shown in the RPA-VV method in Figure 9 by their involvement in Activities 1.1 and 2.1. Additionally, when issues arise during the project they should serve as the contact point for discussion. From here on out they are called 'client'. Guidelines 4 and 5 explain multiple important tasks within the RPA DLC that the client has.

Guideline 2: The client needs to verify and validate deliverables through reviews and official sign-offs.

Expert interviews	Interviewees 1A-4A, 1B, SA1, SA2
Illustrative quote	"We challenge the client to describe all scenarios and verify all scenarios are complete. Their input can be beneficial, If it is clear for them it can improve the quality" — Interviewee 1B.

The RPA-VV method utilizes proper static and dynamic testing to achieve verification and validation. The involvement of the client, as the process expert, is crucial to effectively verify and validate the project deliverables. Verification ensures the accuracy of the design while validation assesses whether the delivered product meets the actual needs and expectations. The verification is carried out through reviews, in which the client actively participates. The RPA-VV method features three reviews that are conducted with the client to verify the process steps (Activity 2.3), process scenarios (Activity 3.2), and solution design (Activity 4.2), as presented in Figure 9. In addition, the client is involved in validating the robot's performance during User Acceptance Testing (Activity 8.3) and Hypercare (Activity 9.3). Formal sign-off by the client is required during these reviews to ensure the successful completion of these activities, as indicated in the RPA-VV method's checklist items. Timely planning of these client contact points is essential to avoid any potential oversight and to enable the incorporation of feedback into the project.

Guideline 3: The client is responsible to deliver one test data set for development and one data set for the user acceptance test. The data set should contain the data for all process scenarios.

Expert interviews	Interviewees 1A-4A
Illustrative quote	"We need to make sure we have a list of all the test scenarios present because we need test data for every one of those scenarios" — Interviewee 1A.

Test data plays a crucial role in dynamic testing activities during development. Given that the client is the owner and expert of the process, they are responsible for providing the necessary data related to the process. Specifically, the client should gather a data set that covers all the scenarios agreed upon during the review of the process scenarios (Activity 3.2). It is imperative to obtain data for all scenarios to ensure that they are covered by test cases and thoroughly tested. Early in the development life cycle, the client should be informed about their role in providing test data, allowing them enough time to gather all the required data. The client needs to gather two sets of data, one for development purposes which forms the basis for unit, integration, and end-to-end tests. Since the robot is developed using the first data set, a second data set is necessary for user acceptance testing to verify that the robot can also function correctly with different data.

10.2 Testing standardisation

Issue II: There is a limited overview of what is tested and no clear standardisation for testing practices during development

Dynamic testing is performed mainly by developers during projects and the Interviewees had little understanding of the extent of testing performed by the developers. "We should be doing testing such as integration testing but this is often in the hands of the developer. It is difficult to check if this happens" — Interviewee 4A. This also shows that integration testing as one of the four different testing levels is not always performed. Furthermore, not all Interviewees understood the correct meaning of the four different levels of testing. "We should unit test all scenarios" — Interviewee 3A. A unit is the smallest part of the code while scenarios are often way bigger. This shows a lack of understanding and standardisation for dynamic testing. Additionally, a lack of standardisation was also visible in the way code reviews were performed. No Interviewee had the same interpretation of how big the size of the code should be under review. Interviewee 2A mentioned "review the workflow to provide feedback" while Interviewee 4A stated, "code review is mainly done once during a sprint". Therefore, the following guidelines are formulated to create a more standardised format for code reviews and dynamic testing.

Guideline 4: The different granularity levels within RPA development need to be aligned with their own level of testing.

Literature	(Ratilainen et al., 2019; Sawant et al., 2012; Shuping & Ling, 2008)
Illustrative quote	"Test levels are instances of a test process performed to relation of development level. [...] Every test level is suitable for different stages of development from individual components to a system ready for production (Ratilainen et al., 2019, p. 22).

In the RPA-VV method shown in Figure 9, dynamic testing is a central activity, with the levels of testing being fundamental. These levels include unit testing (Activity 5.2), integration testing (Activity 6.2), end-to-end testing (Activity 7.1), and user acceptance testing (Activity 8.3). In the case of RPA, there is an additional critical level of testing called hypercare (Activity 9.3) that takes place during the deployment phase. As noted in a previous study (Ratilainen et al., 2019), each level of testing is related to a specific development state.

For RPA, the first level of testing, *Unit testing*, is associated with a single workflow of the robot, such as opening applications, converting an IBAN, or creating an invoice. The next level, *Integration testing*, validates that these sets of workflows can function together and that different variables can be passed between workflows. The third level, *End-to-end testing*, checks if the robot can use the input data and correctly convert it to the expected output when an entire scenario of the process is finished. The fourth level, *User acceptance testing* (UAT), involves testing the robot with a new data set in the presence of the client to showcase its ability to perform all process scenarios. Upon successful completion of the UAT, the robot is ready for deployment on the production environment, and the final testing level, *Hypercare*, requires the robot to perform all process scenarios with real-time production data. All five levels of dynamic testing are critical for ensuring the proper and high-quality completion of the RPA project. To ensure that all levels of testing are performed correctly and to establish a standardised way of working, clear communication between the solution architect and developer is crucial. They should discuss how the testing will be performed and what is expected from the test cases.

Guideline 5: The primary code review should be performed on the granularity level of workflows.

Literature	(Bosu et al., 2015)
Illustrative quote	"Review effectiveness decreases with the number of files in the change set. [...] We recommend that developers submit smaller and incremental changes whenever possible, in contrast to waiting for a large feature to be completed." (Bosu et al., 2015, p. 10)

During the development process, static testing plays a crucial role in ensuring the implementation of the best possible solution and adherence to engineering principles established by the organization. This is done through code reviews. As highlighted by Bosu et al. (2015), smaller code chunks submitted for review lead to more effective and useful code reviews. Hence, the RPA-VV method includes two code reviews, as depicted in Figure 9, to enhance the quality of the development process. The first code review, conducted at the workflow level, is depicted in Activity 5.3, while the second and final review is conducted in Activity 8.1 to ensure the final product is thoroughly inspected and prepared for UAT.

Guideline 6: The handling of application exceptions and the reporting of the robot are their own test scenario and should be included in the end-to-end, user acceptance test, and hypercare.

Expert interviews	Interviewee 1A-4A
Illustrative quote	"We need to make sure that all scenarios are included in the robot with correct input, output, and that all reporting is included — Interviewee 1A.

In addition to the typical process scenarios and business exceptions that a robot executes, there are two other important components of the robot that require validation through testing. The first component concerns the applications utilized by the robot, as unexpected errors may arise, such as a frozen page, which must be handled appropriately by the robot. Hence, these applications should have their own

test scenarios. Furthermore, the robot generates reports for the client when issues occur or to provide information about all cases handled during a specific time period. Such reporting constitutes an essential part of the robot, necessitating its own test scenario. Thus, these components are tested and incorporated into various testing levels, which are further elaborated in Guideline 9.

Guideline 7: All the scenarios discussed in the design phase and encountered during development should be tested during end-to-end, user acceptance testing (UAT), and hypercare. The scenarios need to be officially signed off in the Test plan document during all three testing activities.

Expert interviews	Interviewee 1A-4A
Illustrative quote	"Developer is performing an end-to-end test of the entire solution. This involves all scenarios. [...] We let the robot perform all the scenarios live in an acceptance environment [during the UAT]. Then the client can provide feedback on how the robot performs every scenario. Different parts of the TPD are marked that all scenarios are tested." — Interviewee 1A.

As previously discussed, the scenarios of the project play a central role in what is tested and validated. Three moments arise in which all the process scenarios are tested. These include the end-to-end test (Activity 7.1), the user acceptance test (UAT) (Activity 8.3), and the hypercare (Activity 9.3). It is crucial that all scenarios discussed, as well as any possible additional scenarios encountered during development, are included in these tests. Throughout all three stages, the tests should be meticulously documented in the test plan document (TPD) and signed off. The UAT, being the final test before deployment, occurs together with the client. Here, the client validates that the robot has the correct functionality and signs off the TPD. Subsequently, the hypercare is performed, and again, all scenarios are documented in the TPD and signed off by the client.

10.3 Discrepancies within production environment

Issue III: Discrepancies between test and production environment are common during deployment

During deployment, the robot interacts with the production environment and utilizes real-time data for the first time. However, there may be discrepancies between the acceptance and production environment, and the data may include scenarios not previously encountered during testing. As Interviewee 3B stated, "It is always possible that there are some discrepancies [between the acceptance and production environment.] That is why we have the hypercare period". In addition, Interviewee 3B emphasized the importance of hypercare, stating that "Often the data is the main cause why a robot fails." Guidelines 10 and 11 are developed to address these issues and provide support during the hypercare period.

Guideline 8: The first run on the production environment needs to happen with the initialisation test mode.

Expert interviews	Interviewee 1A,1B,3A,4A
Illustrative quote	"We want to start the first run in the form of test mode in which we don't make any alterations. Also here we start with only opening and closing the applications in production." — Interviewee 1B.

During the first run of the robot in the production environment, it is crucial to use the initialization test mode. This mode helps to identify any discrepancies or issues that may occur unexpectedly and enables the prompt resolution of these issues. The initialization test mode involves a basic test that checks whether the robot can correctly open and close applications in the production environment. If the developer and solution architect are confident in the robot's performance, the hypercare period begins to ensure that the robot can handle all data and scenarios in the production environment. Guideline 11 provides further details on this aspect.

Guideline 9: All the scenarios not encountered and possible new scenarios that are encountered during the hypercare need to be discussed with the client and clear instructions on how they are handled should be agreed upon.

Expert interviews	Interviewee 1A-4A, 1B, 3B, 4B
Illustrative quote	"We need to pass all scenarios. If not all scenarios were touched, we ask if we can run them with fake data. We need to specify the action we will take if these scenarios show up. Otherwise, those scenarios will not be taken into support for example." — Interviewee 4B.
Illustrative quote	"If exceptions still occur in this phase, we discuss with the client what to do in those scenarios. If they are minor we can perform them. If an entirely different scenario pops up, we might need to change the scope of the project and extend it." — Interviewee 1A.

During hypercare, the robot interacts with real-time data for the first time, making it the final validation stage for ensuring that input data is accurately transformed into the required output. Thus, it is imperative to verify that all scenarios are covered and that the robot can handle all the required scenarios. In the event that some scenarios are not tested, it is necessary to consult with the client and determine whether the hypercare period should be extended or if these scenarios should be addressed in a support role. Furthermore, new, unforeseen scenarios may arise during hypercare, necessitating communication with the client to determine whether to include these scenarios in the project's scope or whether the robot's inability to handle these scenarios will be accepted. It is also essential to collaborate with the client to establish a protocol for addressing any unforeseen issues that may arise.

10.4 Documentation traceability

Issue IV: There is no clear traceability between different deliverables of the development life cycle phases

Within RPA projects there is a clear flow from the process steps and exceptions and retrieving the process scenarios from them. This in turn needs to be translated into a design that can be automated and tested. Interviewees confirmed the link between elements of different documentation but specified that this was not always consistently documented. "Inconsistency is mostly based on the way certain scenarios are formulated. Sometimes we use different wording, or there is a discrepancy between alternative flow, happy flow, business or system exceptions. [...] So specifically the wording between the PDD and TPD as in how the scenarios are described." — Interviewee 1B. Additionally, this link could also not be found in the document analysis during the multiple-case study. The documentation was either incomplete or had inconsistencies between the PDD, SDD, and TPD. This is shown in Chapter 8. During the validation, it was mentioned that the code of a project is also not aligned with the scenarios which could be beneficial. Guidelines 12, 13, and 14 are formulated to help with this issue. They are suggestions on how to improve RPA documentation and are not incorporated within the RPA-VV method.

Guideline 10: There needs to be a standard format how to document happy flows, happy flow alternatives, and exceptions.

Case studies	Case 1-4
Expert interviews	Interviewee 1B-5B
Illustrative quote	"Exceptions are documented in their own column but what is often missing is the action that should be taken if it occurs. [...] Currently, there is no clear standard for how we document happy flow alternatives. The solution architect can use their own implementation. A possible solution is incorporating some visualisation of the different alternative flows" — Interviewee 5B.

The process of RPA development involves a main flow from the beginning to the end, which is commonly known as the happy flow. However, when decision points lead to different outcomes or require different steps, it is called a happy flow alternative. Additionally, exceptions can arise unexpectedly or due to process rules, which can cause the process to terminate before reaching the end. It is important to

document all these happy flows, happy flow alternatives, and exceptions in a standardized format within the PDD. Currently, there are some missing elements in the standardization, as the illustrative quote shows. Developing a standardized format for documenting these components is crucial not only for ensuring consistency but also for linking them to test cases, as elaborated in Guideline 11.

Guideline 11: All the happy flows, happy flow alternatives, and exceptions are linked to one test scenario and should be documented correctly and logged in the project code.

Literature	(Torkar et al., 2012)
Case studies	Case 1-4
Expert interviews	Interviewee 1B-5B
Illustrative quote	"There is a one-to-one relation between a happy flow and test scenario, happy flow alternative and a test scenario, and a business rule exception and a test scenario" — Interviewee 1B.

Within the case studies it was clear that there was an inconsistency between different project documentation. For example, the wording of an exception in the PDD was different than the test scenario in the TPD. As the illustrative quote mentions there is a clear link between the different scenarios from the design phase and the test scenarios that form the validation. To accomplish this traceability, the different process steps and exceptions in the PDD should be aligned with the test scenarios mentioned in the TPD. To achieve this alignment different traceability techniques mentioned by Torkar et al. (2012) could be used. Examples are using the same description for a scenario, using ID tagging for every scenario, or using different traceability tools. Another form of traceability can be between the scenarios in the documentation and the code workflows themselves. If scenarios are mentioned in the logging of the robot the code is aligned with the documentation. By implementing traceability within the documentation and code, all phases of the project are aligned and the process scenarios are linked and traceable during design, development, and testing. This allows for an easy overview of what is going right and what is going wrong in different phases of the project. Additionally, if this traceability is correctly implemented it should provide the possibility to use metrics such as test coverage. This measures how many flows or exceptions are tested by a test case and can provide a quick overview of what parts of the design still need to be tested.

Guideline 12: There needs to be a form of traceability implemented between the process steps in the PDD and the solution design in the SDD.

Case studies	Case 1-4
Expert interviews	Interviewee 1B-5B
Illustrative quote	"We write down which steps from the PDD are part of a component (workflow) within the SDD solution diagram. This is a new way of working which is why it was not present in the cases you analysed. Numbering is important because multiple steps in the PDD can become one workflow in the SDD, but sometimes multiple workflows are needed to describe one step in the PDD" — Interviewee 1B.

During the document analysis of the multiple-case study, we did not encounter any clear link between the PDD process steps and the solution design components in the SDD. Nevertheless, it was mentioned by multiple interviewees that linking them was a preferred way of working. While there is a difference in granularity between the process steps and design components, they are describing the same process and can be linked as such. As the illustrative quote shows, multiple steps can be one component or vice versa. The Interviewees further mentioned that adding this traceability could also benefit support understanding the documentation better."We hear from support that they don't understand the process or it is not consistent. This is very inconvenient for support" — Interviewee 4B.

10.5 Identifying documentation purpose

Issue V: The project documentation is important for multiple stakeholders but is not always completely filled in or is inconsistent.

High-quality documentation plays a crucial role for different stakeholders such as the project team, the client, and the support team. However, an analysis of the project documentation and the survey revealed that the documentation was often incomplete and inconsistent. During the interviews when asked why the documentation was not always complete, it was mentioned that it was mainly due to the behaviour of the project team. "It is a human task to fill them in. [...] Currently, it is mainly a behavioural reason why it is not filled in rather than that it is not essential." — Interviewee 1B. This shows that while documentation is essential for different stakeholders, this is often not filled in to assist the needs of all stakeholders. This was mentioned during the interviews for support since they are not part of the project and the documentation is crucial for them to understand how the robot works. "Often we hear from support that they do not understand the process or it is not consistent. This is very inconvenient for support." — Interviewee 4B. Another issue mentioned was scope creep which was already mentioned during the problem-centred interviews. Scope creep can require new scenarios to be added to the robot which should also be included in the documentation. Scope creep happens regularly but no clear standardisation is in place on how to deal with scope creep or how to update documentation. Guidelines 15 and 16 provide help to deal with the above-mentioned issues. They are not incorporated in the RPA-VV method.

Guideline 13: The elements of documentation that are essential for the project team, client, and later for support need to be identified and updated accordingly.

Case study	Case 1-4 and survey
Expert interviews	Interviewee 1B-5B
Illustrative quote	"The priority is always a robot with good proper documentation. This provides proper documentation that verifies the bot works and helps support after the project is handed over" — Interviewee 4B.
Illustrative quote	"The PDD is important for the project team and the client. It is important to have a high-quality project. [...] The SDD is mainly for the project team and used for development" — Interviewee 3B.

As evidenced by the illustrative quote, high-quality documentation is crucial for ensuring a highly qualitative robot that is verifiable through its documentation. However, the effort required to fill in documentation can be significant, so it is vital to identify which parts of the documentation are essential for different stakeholders and where they should be recorded. The PDD is crucial for understanding the process and verifying whether it is correctly interpreted by the client. Different sections of the PDD are crucial for different groups, as noted by Interviewee 4B. The survey results in Appendix B3 in Section 17.5 showed that all interviewees found the PDD important for the client and for the project team itself. The SDD is mainly internal to create the solution and understand what needs to be developed. The survey shows that the SDD is difficult to understand for the client. The use of the SDD was mainly for the project team. "The SDD is mainly used internally. It is important for development" — Interviewee 3B. The TPD documents all testing and can later be used to verify the robot has been tested properly and functions as intended. "TPD is the only proof the robot is working. [...] The TPD provides proper documentation that verifies the bot works." — Interviewee 4B. The survey also showed that the TPD is important for the client and the project team itself. During the interviews, it was clear that documentation is important for different stakeholders and that a standardisation of how documentation is filled in needs to be created. Clear communication between all stakeholders is important so it is understood which parts of the documentation are crucial for whom and when these should be documented.

Guideline 14: The client needs to be informed when scope creep occurs and clear communication is required on how to proceed. If the scope creep is accepted the documentation needs to be updated accordingly.

Expert interviews	Interviewee 1B-5B
Illustrative quote	"With scope creep, my preference is to adjust all documents. Also because if the robot does not work the PDD is used to see how the process is performed manually" — Interviewee 2B.

In order to address the issues related to scope creep, it is crucial to establish standard procedures for dealing with it. Interviewees identified two important aspects in this regard. First, communication with the client is crucial to determine whether the requested changes can be easily incorporated or whether additional development time will be required. As Interviewee 3B noted, "There should be some discussion with the client to see if this is easy [to implement] or more development time is needed." Second, if changes are made, updating the documentation should be standard practice. As Interviewee 1B explained, "The ideal situation is that if a scope creep happens, all documents are updated accordingly." This highlights the importance of close communication with the client and keeping documentation up to date. By establishing clear procedures and communication channels for dealing with scope creep, the potential issues related to it can be minimized.

10.6 Tooling potential for testing

Issue VI: There is limited use of tooling for testing.

From the literature, we found that testing for RPA is still a manual and laborious task (Cernat et al., 2020). This was confirmed again by the interviews conducted. "All scenarios possible are defined and tested manually. [...] We don't use advanced tooling to check or review this." — Interviewee 2B. Multiple Interviewees mentioned that the use of tooling for testing is still limited for RPA. Manual testing increases the time and effort to create and record the results of test cases (Cernat et al., 2020). This shows that there is potential to increase the quality of the testing for RPA. Guidelines 17 and 18 specify how this potential could be reached. These guidelines are not incorporated in the RPA-VV method.

Guideline 15: The use of tools for testing ensures more rigorous testing and allows to incorporation of testing metrics such as test coverage of the robot.

Case study	Survey
Expert interviews	Interviewee 1B, 2B, 4B, SA2
Illustrative quote	"UiPath has more tools such as a test suite, apps, actions centre, and automation hub. So many options are still available within UiPath" — Interviewee 4B.

As the illustrative quote shows, there are still multiple tools possible to increase to verification and validation of the robot through testing. Multiple Interviewees would increase the use of tools to support testing. Already during the problem-centred Interviewees the need for this was mentioned. "I would like to see in our development framework a standard form of testing so we integrate testing more into development" — Interviewee SA2. Guideline 13 already mentioned the need for traceability between process flows and test cases. Guideline 17 adds to this by using tools that can also show how much of the code is covered by test cases. This makes testing more rigorous and increases the validation of the robot. If not all code is covered, new test cases can be written to verify all code has been passed successfully.

Guideline 16: Generic test cases such as application or system exception handling that have a high potential for automation should be identified.

Literature	(Cernat et al., 2020; Montero et al., 2019)
Expert interviews	Interviewee 1B, 2B, 4B
Illustrative quote	"We perform a lot of manual testing for the robots in Uipath studio. But there might be some more generic testing which we can do within a pipeline making sure all required elements are there. Additionally, actions you need to take before you go to production can be more automated with a pipeline" — Interviewee 1B.

Two articles on RPA already focused on automation testing. Nevertheless, these techniques focused on full automation through difficult techniques such as creating a model under test (Cernat et al., 2020) or developing a test environment (Montero et al., 2019). The former has a very high learning curve and requires experienced programmers. There are also still difficulties mentioned in the paper to transform the model towards a testing tool. The latter RPA testing paper focuses on generating a test environment for all the applications that the robot runs on. However, with the number of different applications used for every new robot that is built, this would add a big load on the development team for every project. Additionally, the solution by Montero et al. (2019) requires a log of the steps of the process which is often not available.

Nevertheless, the Interviewees mentioned possible generic tests that are useful for any robot that is being developed. Examples are opening and closing applications or handling unexpected issues with these applications. Generic test cases could be written and added to a pipeline to automatically test every robot under development. This would automate a part of the testing of the robot. Further possible generic tests should be identified to increase the benefit those testing pipelines could provide.

Method validation & evaluation

11. Validation

The first part of the assessment is the validation of the RPA-VV method and guidelines. Firstly, the approach to the validation focus group is explained in Section 11.1. Secondly, the results are described in Section 11.2 and an overview of the results is provided in Table 11.2. A list of the descriptions of the guidelines can be found in Table 11.1

11.1 Approach

The validation is meant to validate whether the RPA-VV method and guidelines are deemed to be suitable for the RPA DLC and applicable in practice (Wieringa, 2014). The focus group consisted of four RPA solution architects. Half of them were already interviewed during the multiple-case study, and the other half were new and had never seen the RPA-VV method before. During the focus group, every phase and its related guidelines were analyzed one by one. A discussion between the participants was started by asking questions related to one of the criteria. The criteria included in this validation were *simplicity, completeness, usefulness*, and *ease of use*. Simplicity was validated by asking if the phase or guideline was clearly written down and if changes to the description could be made. The other criteria were also tested by asking questions relating to the criteria. These questions can be found in Appendix D2 in Section 19.2.

11.2 Results

In general, the RPA-VV method and guidelines were received well. Especially, the visual representation of the RPA DLC was deemed useful. Participants mentioned that it provides a quick overview of the different activities in each phase, deliverables, and testing activities. Table 11.2 shows the results of the validation focus group for each phase and guideline. It specifies whether the group deemed the phase or guideline to be complete, how easy they expected it to perform, whether the group thought the guideline or phase was useful to perform during the RPA DLC, and if there were any important comments given about the phase or guideline.

Foremost, the usefulness of all phases and guidelines was confirmed by the group. Nevertheless, the dominant concern was the ease of use in applying the method and guidelines in practice. They either mentioned some important notion that is necessary to perform the guideline or method correctly or some difficulties in performing them. These do not disqualify the guidelines or phases but show that in practice, it might be difficult to perform. Therefore, the evaluation is important to again check if this ease of use is indeed a limitation of the RPA-VV method and the guidelines

Table 11.1: Description of all guidelines

Guideline	Description
Guideline 1	Clear communication with the client is important since their subject matter experts and process owner are closely involved in all project phases.
Guideline 2	The client needs to verify and validate deliverables through reviews and official sign-offs.
Guideline 3	The client is responsible to deliver one test data set for development and one data set for the user acceptance test. The data set should contain the data for all process scenarios.
Guideline 4	The different granularity levels within RPA development need to be aligned with their own level of testing.
Guideline 5	The primary code review should be performed on the granularity level of workflows.
Guideline 6	The handling of application exceptions and the reporting of the robot are their own test scenario and should be included in the end-to-end, user acceptance test, and hypercare.
Guideline 7	All the scenarios discussed in the design phase and encountered during development should be tested during end-to-end, user acceptance testing (UAT), and hypercare. The scenarios need to be officially signed off in the Test plan document during all three testing activities.
Guideline 8	The first run on the production environment needs to happen with the initialisation test mode.
Guideline 9	All the scenarios not encountered and possible new scenarios that are encountered during the hypercare need to be discussed with the client and clear instructions on how they are handled should be agreed upon.
Guideline 10	There needs to be a standard format how to document happy flows, happy flow alternatives, and exceptions.
Guideline 11	All the happy flows, happy flow alternatives, and exceptions are linked to one test scenario and should be documented correctly and logged in the project code.
Guideline 12	There needs to be a form of traceability implemented between the process steps in the PDD and the solution design in the SDD.
Guideline 13	The elements of documentation that are essential for the project team, client, and later for support need to be identified and updated accordingly.
Guideline 14	The client needs to be informed when scope creep occurs and clear communication is required on how to proceed. If the scope creep is accepted the documentation needs to be updated accordingly.
Guideline 15	The use of tools for testing ensures more rigorous testing and allows for the incorporation of testing metrics such as test coverage of the robot.
Guideline 16	Generic test cases such as application or system exception handling that have a high potential for automation should be identified.

Table 11.2: Results of validation focus group

Phase/ Guideline	Complete	Ease of use	Useful	Comment from focus group participants
Guideline 1	Yes	Medium	Yes	
Prerequisite analysis	Yes	Medium	Yes	Applications or process steps are not always made available before analysis. While preferred, the analysis phase can still proceed.
Analysis phase	Yes	High	Yes	
Design phase	Yes	High	Yes	
Guideline 2	Yes	Medium	Yes	Important to organise and take the time for these reviews with the client.
Prerequisite development	Yes	Low	Yes	Difficult for client to gather all data points
Guideline 3	Yes	Low	Yes	Important to emphasize early in the DLC that test data is crucial and the client needs to gather this.
Development phase	Yes	Medium	Yes	
Guideline 4	Yes	Medium	Yes	Clear communication between the Solution architect and developer about expectations and approach of testing is crucial.
Guideline 5	Yes	High	Yes	Workflow level is useful but also the final code review is important to check the stability and robustness of the entire robot.
Guideline 6	Yes	Medium	Yes	The reporting of the robot is a critical element, but it is currently receiving insufficient attention.
Guideline 7	No	Medium	Yes	Include hypercare in the Guideline. Additionally, documentation is often missing or not documented for all test activities. Also, some scenarios can only be tested on production.
Prerequisite UAT	Yes	Medium	Yes	
Deployment phase	No	Medium	Yes	Deploying is separate from actually testing and running the robot. Therefore, deploying and the initialisation test (activity 9.2) needs to be its own activity.
Guideline 8	Yes	High	Yes	Formulation of the guideline is vague and has been improved to initialisation test.
Guideline 9	Yes	Medium	Yes	Need to specify the differences between predefined scenarios that are not encountered and new scenarios. Both should be communicated with the client.
Guideline 10	Yes	Medium	Yes	
Guideline 11	No	Medium	Yes	Guideline needs to include process scenarios in the logging of the project code.
Guideline 12	Yes	Low	Yes	PDD shows the as-is by humans while SDD shows the to-be of the robot. This can differ, making it difficult to link steps or components together.
Guideline 13	Yes	Medium	Yes	Important to agree on when parts of the document need to be updated and for whom.
Guideline 14	Yes	Medium	Yes	There should be clear communication between the team and the client about how scope creep impacts the project and thus the documentation.
Guideline 15	Yes	Medium	Yes	
Guideline 16	Yes	Low	Yes	System exceptions are often difficult to test since they are unknown exceptions.

12. Evaluation

It is important to note that the evaluation included fewer guidelines than the validation. Guidelines 10 to 16, which focus on the use of documentation and tools and offer advice on changing or enriching the standardized format used in projects, were excluded from the evaluation. These guidelines were excluded as it was out of the scope of the project to create a standardized format for projects.

This chapter elaborates on the second part of the assessment, which is the evaluation of the RPA-VV method. Firstly, the approach taken in the evaluation projects is explained in Section 12.1. Secondly, the evaluation of both projects is described in Sections 12.2 and 12.3 respectively. Lastly, the findings and differences between the evaluation of the projects are discussed briefly in Section 12.4.

12.1 Approach

The evaluation encompassed the practical evaluation of the RPA-VV method and guidelines through two projects. Due to the specific time constraints, the projects were selected through convenience sampling. Focus group sessions were held weekly with all team members to provide explanations of the method and related guidelines, and evaluate the previous phase. The criteria included in the evaluation were *completeness* and *ease of use*. Completeness is now measuring to which extent the phase or guidelines was completed rather than if the phase or guideline was missing any activities. Both projects had their first focus group session during the analysis phase, which meant that no support or advice could be offered by the method for the prerequisite phase. Furthermore, both projects had scheduled the review of the analysis and design phase together during the prerequisite phase, which could not be adjusted due to the project timeline. The second project faced delays due to the absence of test data and the unavailability of key stakeholders during a holiday period, which resulted in the delay in performing everything from UAT onward during the evaluation. These limitations impose restrictions on the complete evaluation of the RPA-VV method in practice. Nevertheless, the evaluation uncovered noteworthy results pertaining to the issues confronted by the projects during the DLC.

12.2 Project 1 results

This section shows the results of the evaluation of the first project. A description of the guidelines can be found in Table 11.1. Table 12.1 provides an overview of the results of the project. The different phases and guidelines are described in order of when they occur in the DLC. The Table shows to what extent the project team performed each phase or guideline and if they found it easy to perform. This is shown in two columns which are 'completed' and 'ease of use' respectively.

The following paragraphs present a detailed account of the results of the focus groups which are the foundation of the obtained values presented in Table 12.1. The various phases and guidelines are explicated, highlighting how the team executed them and any potential challenges encountered. The following paragraphs correspond to the respective phases and guidelines displayed in the table. Additionally, where feasible, the rationale or noteworthy observations related to the implementation of a particular guideline or RPA-VV phase are provided.

Guideline 1 emphasizes the importance of involving all stakeholders throughout the various project phases. However, the project team encountered issues due to unclear responsibility for deliverables, resulting in delays. To improve performance, clear communication with stakeholders is necessary to emphasize the importance of deliverables and who is responsible for them. Unfortunately, the project team found it difficult to ensure the timely delivery of deliverables, as evidenced by the poor performance of the prerequisite phase. Specifically, the team had not received a process overview or any applications, which are prerequisites for this phase. Consequently, the team was ill-prepared for what to expect in the subsequent stages of the project. The team stated that while it is preferred to have both deliverables before the analysis phase, the analysis phase can still proceed without them, albeit requiring a different approach. In such instances, the missing deliverables must be tackled during the analysis phase. Regrettably, inadequate

Table 12.1: Project 1 result overview

Phase or guideline	Completed	Ease of use
Guideline 1	Partly	Low
Prerequisite analysis	No	Low
Analysis phase	Partly	Medium
Design phase	Yes	Low
Guideline 2	Yes	Low
Prerequisite development	No	Low
Guideline 3	Partly	Low
Development phase	Partly	Medium
Guideline 4	Partly	Medium
Guideline 5	Yes	High
Guideline 6	Yes	Medium
Guideline 7	Partly	Medium
Prerequisite UAT	Yes	Medium
Deployment phase	Yes	Medium
Guideline 8	yes	High
Guideline 9	Partly	Low

communication and unclear deadlines resulted in restricted access to the infrastructure and test data, which was only granted during the development phase.

The completion of the analysis phase was only partial, as the review of process steps had been postponed until the design phase. Although the design phase had been entirely executed, it presented some challenges for the project due to the combination of the analysis and design phase reviews into a single session at the end of the design phase. During this meeting, it was discovered that certain steps and scenarios were missing, necessitating changes to the documents even though the design phase had already concluded. Guideline 2 highlights the importance of organising meetings to validate different deliverables of the analysis and design phase, which had been performed. However, the method used resulted in issues, as previously described by the team. Ideally, as mentioned by the team, it would have been preferable to validate the process steps and scenarios before finalising the design. This gives the project team to make any necessary changes and the client to gather the test data required at the beginning of development.

Guideline 3 emphasizes the importance of test data in the development phase, which is also reflected in the RPA-VV method in the prerequisite development phase. However, the project team encountered a delay in receiving the test data and access to the required applications, further underscoring the communication issues with the client and their deliverables. In response, the team suggested starting the process early and emphasizing the necessity of test data before commencing the development phase. Additionally, if the process steps and scenarios are reviewed during the analysis phase, the same meeting can be utilized to finalize the data set, streamlining the development process.

Due to the strict project plan, the development phase commenced before the delivery of the test data, leading to a challenging and brief development phase. Guideline 4 emphasizes the importance of testing code on various levels, but only unit testing was conducted, with integration testing delayed until the end of the development phase. This resulted in integration and end-to-end testing being combined, leading to multiple bugs requiring significant time spent on debugging. This highlights the importance of performing integration testing for every workflow added to the robot to avoid delays. Despite the delay in user acceptance testing (UAT) due to the extra time required for testing, all scenarios were tested successfully, complying with Guideline 6, which emphasizes testing all scenarios. Guideline 5 stresses the need to conduct code reviews on individual workflow levels, and this was performed, providing developers with useful insights. However, Guideline 7 was partially implemented. While the end-to-end and UAT were successful, the hypercare phase was challenging. The deployment phase is explained in the next paragraph to address this issue.

The deployment phase is the final phase. The current project revealed a number of discrepancies between the test and production environments, thereby highlighting the significance of the hypercare phase. Guideline 8, which necessitates verification of the robot's functionality during its first run in the production environment, assumes utmost importance in this regard. Moreover, any discrepancies that persist within the process scenarios should also be duly validated. Both Guidelines 7 and 9 recommend the performance of all scenarios during the hypercare phase. During the evaluation phase of the project, it was noted that the project team faced challenges in verifying the successful completion of all scenarios. The lack of a clear verification method hindered their ability to confirm the passing of all scenarios. Although no issues were detected during the hypercare phase, it is unclear whether the untested scenarios could have revealed potential issues.

12.3 Project 2 results

This section shows the results of the evaluation of the second project. A description of the guidelines can be found in Table 11.1. Table 12.2 provides an overview of the results of the project. The different phases and guidelines are described in order of when they occur in the DLC. The table shows to what extent the project team performed each phase or guideline and if they found it easy to perform. This is shown in two columns which are 'completed' and 'ease of use' respectively.

Table 12.2: Project 2 result overview

Phase or guideline	Completed	Ease of use
Guideline 1	Yes	Medium
Prerequisite analysis	No	Low
Analysis phase	Partly	Medium
Design phase	Yes	Low
Guideline 2	Yes	Medium
prerequisite development	No	Low
Guideline 3	Partly	Low
Guideline 4	Partly	Low
Development phase	Partly	Medium
Guideline 5	Yes	High
Guideline 6	No	N.A
Guideline 7	No	N.A
Prerequisite UAT	No	N.A
Guideline 8	No	N.A
Guideline 9	No	N.A

The following paragraphs provide a comprehensive overview of the results obtained from the focus groups, which served as the basis for the values presented in Table 12.2. The discussion covers the different phases and guidelines, with an emphasis on the team's execution strategy and any challenges that emerged. Each paragraph corresponds to the specific phase or guideline listed in the table. Furthermore, whenever possible, the reasoning or noteworthy observations related to the implementation of a particular guideline or RPA-VV phase are provided.

Guideline 1 underscores the significance of involving all stakeholders throughout the various project phases. In the context of Project 2, stakeholder involvement was carried out successfully. However, there were some minor difficulties encountered, particularly with regard to test data, which is discussed in further detail later. Moreover, there were a few deviations from the usual project execution process. Notably, during the prerequisite phase, it was discovered that development and testing had to be conducted in the production environment, an arrangement which is not ideal but was deemed acceptable for the project. Additionally, the step-by-step process overview, which is a key deliverable of this phase, was not initially provided. Nonetheless, the project team reported that it is acceptable as long as the process overview is created during the analysis phase. Fortunately, this was the case for Project 2.

The analysis phase proceeded relatively smoothly, although the reviews for both the analysis and design phases were combined, resulting in a review of the process steps, process scenarios, and solution design simultaneously during the design phase. This review revealed several issues, which necessitated changes to all of the documentation, leading to a delay in the project timeline. Guideline 2 emphasizes the importance of conducting separate reviews for each of these deliverables, but the project team expressed a preference for combining the reviews for the process steps (PDD) and process scenarios (TPD) since they are closely linked. The solution design, which represents the technical blueprint for the automated process, is less relevant for the client and should be reviewed after the PDD and TPD have been validated. Unfortunately, due to the delays caused by the changes required in the design phase, the development phase had to commence while the design phase was still ongoing, resulting in a significant overlap between these two critical phases of the project.

In accordance with Guideline 3, it is crucial to provide a comprehensive dataset for the development phase that includes data for all possible scenarios. The prerequisite development phase also aims to provide test data for the development phase, as emphasized in Guideline 3. However, this aspect did not proceed smoothly in Project 2. Development had already begun prior to the delivery of the dataset, and when it was eventually received, it contained incorrect formats, resulting in additional complications. As a result, the development phase was prolonged, and progress was hindered. Guideline 4 stresses the need to conduct testing at different testing levels. While the developer attempted to adhere to this guideline, the unavailability of testing data made it extremely challenging to test a workflow. Despite successful unit testing and integration testing for each added workflow to the process flow, some integration tests could only be performed once due to the absence of test data. On another note, Guideline 5 focuses on the review of individual workflows. Both members of the project team found it useful to respectively receive and provide feedback on every workflow. Nevertheless, the delay in the development phase in terms of test data meant that no end-to-end testing was possible, and the UAT had to be postponed. Consequently, the evaluation did not include the end-to-end testing and UAT, and Guidelines 6 to 9, which focus on the delayed phases, could not be evaluated either.

12.4 Discussion evaluation

Although the visual representation of the RPA-VV method was positively received in the evaluation, as it enabled the easy identification of critical deliverables or testing activities within the RPA DLC, the overall assessment was challenging. Despite both projects being classified as simple processes, the project teams found them to be the most chaotic during their time as RPA experts. Nevertheless, the chaotic environment provided an interesting setting to evaluate the RPA-VV method. However, the validation identified the primary concern regarding the ease of use of the method, which was confirmed in the evaluation. In both projects, most phases and guidelines were only partially or not completed, with only a few phases being fully completed. Even for the fully completed phases, their ease of use was often low, resulting in extended or overlapping phases. In the next section, we discuss the progression of the various testing activities in the projects, followed by a discussion of two factors that contributed to deviations from the RPA-VV method and guidelines.

Impact on testing activities

The RPA-VV method consists of a few static and dynamic testing activities. Static testing is performed through client reviews during the analysis and design phases and can lead to issues if all reviews are conducted at the end of the design phase. Both projects experienced challenges in this regard and recommended improvements. These are discussed in the next paragraph. Code review, another static testing activity, was conducted in accordance with the RPA-VV method and proved successful in both projects. Dynamic testing activities take place during development and deployment but were impeded by a lack of testing data during the development phase. Project 1 omitted the integration test, which led to increased debugging at the end of the project. However, the other dynamic testing activities were executed correctly and were successful. Notably, Project 1 could not confirm that all scenarios were tested during hypercare, highlighting the importance of maintaining traceability between scenario documentation and code, as per Guideline 11 of the RPA-VV method. Although not evaluated, it is worth noting that Guideline 11 addresses an issue experienced during this phase. Project 2 also experienced

a delay due to missing test data, but the unit and integration tests were conducted satisfactorily. As a result, the other dynamic testing activities were not performed within the evaluation time frame.

Factors contributing to deviations from the RPA-VV method and Guidelines

In both projects, deviations from the RPA-VV method and guidelines were observed, primarily due to two factors: project planning and client contact points and deliverables. The project phases were planned with stringent timelines assigned to each phase before the project's initiation. Thus, if the deliverables were not met within the given timeline, the subsequent phase would still begin as scheduled. This often resulted in deviations from the RPA-VV method. For instance, both project teams combined all the reviews during the more nuanced analysis and design phase. Although the RPA-VV method separates the reviews, the project teams preferred to integrate process steps and scenarios in one meeting, considering their closely related nature. Together, these steps form the technical blueprint of automation and require review at a later stage, which is the final review in the RPA-VV method. This highlights the more progressive but intertwined nature of these phases, indicating an area for improvement in the RPA-VV method. Another instance where strict planning resulted in deviations was the lack of testing data delivered while the development phase was already underway. The impact of this deviation has been elucidated in the previous paragraph.

The intertwined reviews and the missing testing data are interrelated with the second factor, which concerns the contact points and deliverables of the client during the project. The three reviews are contact points with the client and should be organised with their importance in mind. Furthermore, the lack of testing data provided by the client during the development phase had a significant impact on the project's progress, as previously discussed. Therefore, it is recommended that the importance of test data be emphasized to the client in the RPA-VV method, as it has a substantial influence on the development phase. It is worth noting that the RPA-VV method already incorporates the involvement of the client in the various phases of the project, as specified in Guideline 1. Moreover, the method places a significant emphasis on test data with two separate prerequisite activities, as well as Guideline 3, which further underscores its significance. However, there appears to be no explicit connection between missing activities or guidelines and the deviations observed in the projects. Thus, it is suggested that the deviations are linked to the project outcomes themselves.

In conclusion, the usage of the RPA-VV method becomes challenging due to the strict planning and tight timelines. The projects highlighted the importance of client contact points and deliverables, as well as the intertwined nature of the analysis and design phases which can be improved in the RPA-VV method. Furthermore, it is recommended that client reviews be planned and communicated clearly and that the significance of test data be emphasized to the client. The RPA-VV method already emphasizes client involvement and test data, but deviations may still occur due to project outcomes. Therefore, clear communication with the client throughout the project is crucial.

Discussion & Conclusion

13. Discussion

The RPA-VV method is a structured approach to verify and validate the different phases and deliverables of the development life cycle (DLC) of RPA projects. The focus of this method is on linking the different activities of an RPA DLC phase to different testing activities. This approach ensures that the RPA process analysis and design undergo verification, while the development and deployment are validated, to ensure adherence to expectations. The RPA-VV method provides guidelines to support the different phases, activities, and deliverables.

The discussion chapter of this research project offers a thorough analysis and interpretation of the findings. It begins by emphasizing the theoretical contributions of the study, followed by an exploration of the practical implications. After that, the limitations of the study are discussed before moving on to a final section that outlines potential avenues for future research.

13.1 Theoretical contributions

The significance of RPA in practice is rapidly increasing, yet the academic research in this area is still in its early stages, with only a few years of notable contributions (Syed et al., 2020). A gap in the academic research for RPA was identified by highlighting two critical parts of software development, namely requirement management (RM) and testing activities, and relating them to RPA implementation. These topics are barely covered in the field of academic research on RPA (Enríquez et al., 2020). Although research has highlighted the significance of alignment between RM and testing in standard software development life cycles (Skoković & Rakić-Skoković, 2010), this alignment is often missing. This is also the case for RPA, where the main DLC described is the waterfall method (Cewe et al., 2017). The waterfall method only addresses the analysis and testing phases without addressing their relationship or how RM influences the different phases of the DLC. Therefore, this research aimed to fill this gap by researching novel DLC methods that address the alignment between RM and testing and applying these methods to the RPA DLC. According to Syed et al. (2020), there is a lack of theoretical foundations for objective reasoning in the domain of RPA. Additionally, problem-centred interviews conducted with RPA practitioners indicated the significance of the analysis phase and the lack of a standardized testing approach. Therefore, the research methodology was structured to effectively combine theory and practice. The theoretical foundation was established through a comparative analysis of different methods and techniques that focused on both requirements management (RM) and testing. The W-model (Regulwar et al., 2010), which is widely supported and extensively described in the literature, was selected and is applied to the RPA DLC. It was combined with practical feedback to further develop and evaluate the suitability and effectiveness of the RPA-VV method in practice. Through this approach, the methodology contributes to both the academic and practical aspects of RPA research.

13.2 Practical implications

The RPA-VV method and its guidelines provide a clear visual overview of the RPA DLC to help RPA teams create structure within their projects. It clearly defines the different deliverables of a phase and links different testing activities to every phase to properly verify the analysis of a process and validate its implementation. Firstly, this standardisation can improve the quality of RPA projects and ensures that deliverables are properly verified and validated. Additionally, it can be used to inform both RPA solution architects and developers on how the RPA DLC should look. This can help to monitor adherence to the standardised format during the project. Secondly, the RPA-VV method can be used to inform clients of their responsibilities within the project such as verification reviews and how the project will use their crucial deliverables such as test data. This allows the client to have a better understanding of their role in the project, resulting in a smoother collaboration between the client and the RPA organization.

While the RPA-VV method is developed for RPA organizations that outsource their services, organizations that build their RPA in-house can also use the RPA-VV method. However, the guidelines and phases that are focused on the client should be adjusted to how this works within their organization. It

should be noted that the practical implications of the RPA-VV method are currently still limited due to the strict planning of projects. This will be further discussed in the limitations and future work in Section 13.3.

13.3 Limitations

There are several limitations to this research project that should be considered when interpreting the results. Firstly, the small sample size of the case study, which only included four projects, may reduce the generalizability of the results. Additionally, all participants in the study were experts from a single RPA organization, which could introduce some bias and conflict of interest. Moreover, the developers were not included in the case study and validation, and their input could have been beneficial in the earlier stages of the process. Secondly, time constraints were a significant limitation of the research project. Although sufficient time was allotted for the academic research, the scope of the study was still limited, and the evaluation had to be cut short. As a result, the projects could not be fully evaluated, making it difficult to determine the applicability of the RPA-VV method for RPA projects. Moreover, not all assessments were done using the same criteria, which may have affected the accuracy of the results. The limited time for the interviews and focus groups forced us to only analyze the most important criteria, which limits the construct validity of the research. Finally, the guidelines regarding the use of documentation and tools were not included in the evaluation due to the need to change the format and use of standardized documents. This restriction creates a limitation since creating a new standardization is a research project on its own. Therefore, the applicability of these guidelines in practice is not evaluated.

Several limitations of the RPA-VV method were revealed in the evaluation phase which must be considered. The first identified limitation relates to the ease of use of the RPA-VV method, as project teams encountered difficulties in utilizing it during the evaluation phase. This highlights the need for further refinement to improve the method's usability. Additionally, while the RPA-VV method aims to address different deliverables and includes contact points with the client, it may conflict with the strict planning of RPA projects. Unforeseen issues or changes that arise during project execution can result in deviations from the method its guidelines and phases, which may compromise project quality. The second limitation identified relates to the role of the client in executing the RPA-VV method. The guidelines of the method do not explicitly state the role of the client, and they were not included in the evaluation of the RPA-VV method. This may have contributed to the challenges encountered during the evaluation. Finally, although the RPA-VV method was explained and evaluated, no clear training or further instructions were provided for the implementation of its phases or guidelines. This suggests that additional support may be necessary for project teams to implement the method correctly within their projects.

13.4 Future work

Considering the limitations of the research, there are a few areas future work could focus on. These are separated into improvements for the research projects and possible improvements for the RPA-VV method. Improvements for the research project include a larger and more diverse sample size to increase the generalizability and reduce possible biases in the results. This could involve collaborating with multiple RPA organizations to obtain a more comprehensive understanding of the effectiveness of the RPA-VV method. In addition, involving developers in the case study and validation process would enhance the contribution of various stakeholders to the RPA-VV method, potentially improving its practical applicability and identifying areas for further improvement. Furthermore, involving clients in the execution of the RPA-VV method and evaluation can improve the quality of the project. These areas of future work can be combined with a more extensive evaluation of the RPA-VV method to fully assess the applicability and effectiveness of the RPA-VV method for RPA projects.

Areas for future work on the RPA-VV method may include exploring how to balance the structured approach of the RPA-VV method with the need for project planning. Further research is needed to better understand how practitioners can effectively use the RPA-VV method to make it more suitable for practical implementation. This research could involve identifying potential solutions for unexpected issues and changes that may arise during project execution while maintaining the quality of the final

product. Additionally, future research includes providing clearer training and support for project teams and clients that could help address issues with the implementation of the RPA-VV method. Especially focusing on the guidelines and how they support the implementation of the RPA-VV method. This could result in the development of additional guidelines or clear instructions on how to use the RPA-VV method. Other potential areas of research could evaluate the applicability of the guidelines focused on the documentation and tools used within the RPA DLC in practice.

14. Conclusion

Robotic process automation (RPA) is a relatively new field within process management (Hofmann et al., 2020) with limited academic research on requirement management (RM) and testing within the RPA projects and no research focused on integrating different testing activities within the RPA development life cycle (DLC). Therefore, a design science research project has been conducted to develop a method that can align process requirements with testing activities throughout the RPA DLC. The research resulted in the RPA-VV method which fills the research gap. The method emphasizes the importance of testing throughout the entire process, ensuring that the final RPA solution meets the business requirements and functions correctly. It applied static and dynamic testing activities in a structured manner through the different RPA DLC phases to ensure proper verification and validation of different DLC phases and deliverables. The different sub-questions are answered one by one in the next paragraphs.

In order to address sub-question 1, it was necessary to identify the relevant phases of the development life cycle (DLC) that align requirements with testing practices. This was achieved through a combination of a multi-vocal literature review (MLR) and problem-centred interviews (PCIs). The literature review identified eight RPA DLC phases, out of which five were deemed essential for an effective RPA method that aligns requirements with testing practices. These phases include analysis, design, development, testing, and hypercare.

Sub-question 2 of the study aimed to identify suitable methods or techniques to align requirements and testing practices. To address this, the study compared six different methods or techniques found in the literature and evaluated their suitability. Among these methods, the W-model was found to be the most suitable. This led to the process of transforming the W-model suitable for the RPA DLC, which resulted in the first iteration of the RPA-VV method depicted in Figure 6.1.

Sub-questions 3 of the research aimed to investigate how to formalize and document requirements and testing practices to create alignment between them. To achieve this, the study developed three guidelines that focused on traceability within documentation and other DLC artefacts. The validation focus group confirmed the usefulness of the guidelines, but their ease of use scored low to medium. This result indicates that the guidelines were relatively difficult to perform, mainly because a standard format of the documentation needs to be established. Therefore, the focus group expected a challenging implementation without the standardized format. Future research could focus on developing tools or templates that support traceability within the documentation process and ensure adherence to the standardized format.

Sub-question 4 focused on creating, validating, and evaluating the effectiveness of the RPA-VV method. This study involved a synthesis of theoretical and practical perspectives through an MLR and a multiple-case study. The RPA-VV method was developed with the goal of fulfilling seven requirements, which are outlined in Chapter 6. While the W-model covered most of these requirements, there were two important ones that were not addressed — traceability and clear guidelines. To still meet these requirements, the RPA-VV method was designed to include various documentation deliverables that enable traceability and guidelines that support its use. After the multiple case study, two issues were identified with the RPA-VV method: its format and ambiguous activity names. The activity names were revised, and the format was changed to the process-deliverables format proposed by van de Weerd and Brinkkemper (2009). This structured format enabled the linking of activities to specific deliverables, which is essential for the RPA DLC. The enhanced version of the RPA-VV method is presented in Figure 9, and the associated guidelines are outlined in Chapter 10. The effectiveness of the RPA-VV method was analyzed through validation and evaluation. The validation indicated that all phases and guidelines were deemed useful for the RPA DLC but the execution was expected to be difficult. The evaluation revealed two main issues: strict project planning causing deviations and difficulties with client involvement and deliverables. The evaluation was limited due to deviations from the RPA-VV method and an incomplete assessment. Future research could focus on how to effectively execute the structured approach of the RPA-VV method in project planning and handling issues with deliverables.

The fifth and final sub-question examines the integration of the RPA-VV method with current documen-

tation and tools used in the RPA DLC. The RPA-VV method outlines the delivery of documentation in each phase, and guidelines in Chapter 10 focus on the use of documentation and tools. While tools are not directly addressed, recommendations are provided in the guidelines. Due to the variability of tools used in RPA projects, broad guidelines concentrate on their potential rather than integrating a specific tool. The usefulness of these guidelines was validated but not evaluated, offering opportunities for future research to assess their effectiveness in practice.

In conclusion, the RPA-VV method provides a structured and clear approach to testing and verifying and validating the DLC phases and deliverables within RPA projects. It offers a structured method that identifies the different testing activities, including static and dynamic testing, to be conducted in each phase of the RPA project. Moreover, the method includes guidelines to support the different phases. While the method has shown promising results in terms of its potential for RPA projects, further research is required to examine its practical applications. Future work should focus on how to execute the structured approach of the RPA-VV method in relation to project planning, how to deal with issues if things are not delivered, and how this relates to planning. Additionally, research should explore the integration between the RPA-VV method and current documentation and tools used within the RPA DLC. Ultimately, such research efforts will enable practitioners to gain a better understanding of how the RPA-VV method can be applied in practice, ensuring that RPA project deliverables are properly verified and validated.

15. References

Ayres, L., Kavanaugh, K., & Knafl, K. A. (2003). Within-case and across-case approaches to qualitative data analysis. *Qualitative health research*, *13*(6), 871–883.

Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. v-model vs. agile: A comparative study on sdlc. *International Journal of Information Technology and Business Management*, *2*(1), 26–30.

Barmi, Z. A., Ebrahimi, A. H., & Feldt, R. (2011). Alignment of requirements specification and testing: A systematic mapping study. *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, 476–485.

Bezemer, B. (2019). What makes good process documentation for rpa? [Accessed: 7-11-2022]. https://www.linkedin.com/pulse/what-makes-good-process-documentation-rpa-bastiaan-bezemer

Bogner, A., Littig, B., & Menz, W. (2009). *Interviewing experts*. Springer.

Bosu, A., Greiler, M., & Bird, C. (2015). Characteristics of useful code reviews: An empirical study at microsoft. *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 146–156.

Bouillon, E., Mäder, P., & Philippow, I. (2013). A survey on usage scenarios for requirements traceability in practice. *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 158–173.

Bryman, A. (2016). *Social research methods*. Oxford university press.

Cernat, M., Staicu, A. N., & Stefanescu, A. (2020). Towards automated testing of rpa implementations. *Proceedings of the 11th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*, 21–24.

Cewe, C., Koch, D., & Mertens, R. (2017). Minimal effort requirements engineering for robotic process automation with test driven development and screen recording. *International Conference on Business Process Management*, 642–648.

Choudary, A. (2020). How to write a good test plan in software testing? [Accessed: 7-11-2022]. https://www.edureka.co/blog/test-plan-in-software-testing/

Choudhuri, A. (2021). What are pdd and sdd in rpa? [Accessed: 7-11-2022]. https://www.probegroup.com.au/blog/what-are-pdd-and-sdd-in-rpa#:~:text=An%5C%20acronym%5C%20for%5C%20Solution%5C%20Design,%5C%27to%5C%2Dbe%5C%27%5C%20process

Chugh, R., Macht, S., & Hossain, R. (2022). Robotic process automation: A review of organizational grey literature. *International Journal of Information Systems and Project Management*, *10*(1), 5–26.

Consulting, K. (2021). Business case: Automation of software development life cycle. https://kvalito.ch/project/business-case-automation-of-software-development-life-cycle/

Dave, K. (2020). The five pillars of a successful rpa test plan. https://www.mindfieldsglobal.com/blog/rpa-testing

Döringer, S. (2021). 'the problem-centred expert interview'. combining qualitative interviewing approaches for investigating implicit expert knowledge. *International Journal of Social Research Methodology*, *24*(3), 265–278.

Emorphis, T. (2020). Robotic process automation: Stages of rpa life-cycle. https://medium.com/emorphis-technologies/robotic-process-automation-stages-of-rpa-life-cycle-e0f8180a11d6

Enríquez, J. G., Jiménez-Ramírez, A., Domínguez-Mayo, F. J., & Garcia-Garcia, J. (2020). Robotic process automation: A scientific and industrial systematic mapping study. *IEEE Access*, *8*, 39113–39129.

Fernando, L. (2020). Approach for effective process discovery on rpa projects [Accessed: 7-11-2022]. https://lahirufernando90.medium.com/approach-for-effective-process-discovery-on-rpa-projects-9306dad35914

Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, *106*, 101–121.

George, B., & Williams, L. (2004). A structured experiment of test-driven development. *Information and software Technology*, *46*(5), 337–342.

Ghouse, A., & Sipos, C. (2022). Rpa progression throughout years and futuristic aspects of rpa. *Pollack Periodica*, *17*(1), 30–35.

Goris, V. (2019). *Robotic process automation an assesment of process discovery techniques with the purpose of finding rpa eligible processes.*

Hamilton, T. (2022). Test plan template: Sample document with web application example [Accessed: 7-11-2022]. https://www.guru99.com/test-plan-for-project.html
Harita, P. (2019). Quick guide on requirement gathering phase in an rpa project [Accessed: 7-11-2022]. https://skcript.com/svr/quick-guide-on-requirement-gathering-phase-in-an-rpa-project/
Hofmann, P., Samp, C., & Urbach, N. (2020). Robotic process automation. *Electronic Markets*, *30*(1), 99–106.
Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. (2016). Software testing techniques: A literature review. *2016 6th international conference on information and communication technology for the Muslim world (ICT4M)*, 177–182.
Janzen, D., & Saiedian, H. (2005). Test-driven development concepts, taxonomy, and future direction. *Computer*, *38*(9), 43–50.
Jiao, J., & Chen, C.-H. (2006). Customer requirement management in product development: A review of research issues. *Concurrent Engineering*, *14*(3), 173–185.
Jiménez-Ramírez, A., Chacón-Montero, J., Wojdynsky, T., & Gonzalez Enriquez, J. (2020). Automated testing in robotic process automation projects. *Journal of Software: Evolution and Process*, e2259.
Johansson, C., & Bucanac, C. (1999). The v-model. *IDE, University Of Karlskrona, Ronneby*.
Jovanović, S. Z., Đurić, J. S., & Šibalija, T. V. (2018). Robotic process automation: Overview and opportunities. *International Journal Advanced Quality*, *46*(3-4), 34–39.
Kaur, M., & Singh, R. (2014). A review of software testing techniques. *International Journal of Electronic and Electrical Engineering*, *7*(5), 463–474.
Kirchmer, M., & Franz, P. (2019). Value-driven robotic process automation (rpa). *International Symposium on Business Modeling and Software Design*, 31–46.
Kirova, V., Kirby, N., Kothari, D., & Childress, G. (2008). Effective requirements traceability: Models, tools, and practices. *Bell Labs technical journal*, *12*(4), 143–157.
Kumar, R. (2022). Rpa process identification (part4)-solution design document (sdd) [Accessed: 7-11-2022]. https://www.linkedin.com/pulse/rpa-process-identification-part4-solution-design-document-rahul-kumar
Leeuwen, T. (2022). *Project management methodology for robotic process automation implementation* (B.S. thesis). University of Twente.
Mařík, V., Král, L., & Mařík, R. (2000). Software testing & diagnostics: Theory & practice. *International Conference on Current Trends in Theory and Practice of Computer Science*, 88–114.
Mogyorodi, G. (2001). Requirements-based testing: An overview. *Proceedings 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems. TOOLS 39*, 286–295.
Montero, J. C., Ramirez, A. J., & Enríquez, J. G. (2019). Towards a method for automated testing in robotic process automation projects. *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*, 42–47.
Ng, K. K., Chen, C.-H., Lee, C. K., Jiao, J. R., & Yang, Z.-X. (2021). A systematic literature review on intelligent automation: Aligning concepts from theory, practice, and future perspectives. *Advanced Engineering Informatics*, *47*, 101246.
Orynbayeva, A. (2019). *A governance model for managing robotics process automation (rpa)*.
Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, *24*(3), 45–77.
Petersen, K., Wohlin, C., & Baca, D. (2009). The waterfall model in large-scale development. *International Conference on Product-Focused Software Process Improvement*, 386–400.
Pham, T. M. T., & Yang, J. (2020). The secret life of commented-out source code. *Proceedings of the 28th International Conference on Program Comprehension*, 308–318.
Prat, N., Comyn-Wattiau, I., & Akoka, J. (2015). A taxonomy of evaluation methods for information systems artifacts. *Journal of Management Information Systems*, *32*(3), 229–267.
Prucha, P. (2021). Aspect optimalization of robotic process automation. *CEUR Workshop Proceedings*.
Ratilainen, T., et al. (2019). Guidelines for creating a testing process for a software-case study of comparing testing of two different size of slot game projects.
Reddy, G. (2022). Introduction to robotic process automation. https://www.gcreddy.com/2022/01/introduction-to-robotic-process.html

Regulwar, G. B., Deshmukh, P., Tugnayat, R., Jawandhiya, P., & Gulhane, V. (2010). Variations in v model for software development. *International Journal of Advanced Research in Computer Science*, *1*(2), 134–135.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, *14*(2), 131–164.

Sawant, A. A., Bari, P. H., & Chawan, P. (2012). Software testing techniques and strategies. *International Journal of Engineering Research and Applications (IJERA)*, *2*(3), 980–986.

Servicenow. (n.d.). What is the software development life cycle. https://www.servicenow.com/products/devops/what-is-sdlc.html

Shakir, M. (2002). The selection of case studies: Strategies and their applications to is implementation case studies.

Shuping, L., & Ling, P. (2008). The research of v model in testing embedded software. *2008 International Conference on Computer Science and Information Technology*, 463–466.

Skoković, P., & Rakić-Skoković, M. (2010). Requirements-based testing process in practice. *International Journal of Industrial Engineering and Management (IJIEM)*, *1*(4), 155–161.

Song, W. (2017). Requirement management for product-service systems: Status review and future trends. *Computers in Industry*, *85*, 11–22.

Soybir, S., & Schmidt, C. (2021). Project management and rpa. *The Digital Journey of Banking and Insurance, Volume I: Disruption and DNA*, 289–305.

Syed, R., Suriadi, S., Adams, M., Bandara, W., Leemans, S. J., Ouyang, C., ter Hofstede, A. H., van de Weerd, I., Wynn, M. T., & Reijers, H. A. (2020). Robotic process automation: Contemporary themes and challenges. *Computers in Industry*, *115*, 103162.

Torkar, R., Gorschek, T., Feldt, R., Svahnberg, M., Raja, U. A., & Kamran, K. (2012). Requirements traceability: A systematic review and industry case study. *International Journal of Software Engineering and Knowledge Engineering*, *22*(03), 385–433.

Tran, D., & Ho Tran Minh, T. (2018). Workflow methodology development of rpa solution for a vietnamese bank: A case study of korkia oy.

Turhan, B., Layman, L., Diep, M., Erdogmus, H., & Shull, F. (2010). How effective is test-driven development. *Making Software: What Really Works, and Why We Believe It*, 207–217.

UiPath. (n.d.). Continuous automation, continuous testing. https://start.uipath.com/rs/995-XLT-886/images/Ui_200904_ScalingAutomation-Ebook-v2.pdf

UiPath. (2022). Process definition document (pdd) [Accessed: 7-11-2022]. https://rpalearners.com/wp-content/uploads/2022/03/Process-Definition-Document-PDD.docx

Uusitalo, E. J., Komssi, M., Kauppinen, M., & Davis, A. M. (2008). Linking requirements and testing in practice. *2008 16th IEEE International Requirements Engineering Conference*, 265–270.

van de Weerd, I., & Brinkkemper, S. (2009). Meta-modeling for situational analysis and design methods. In *Handbook of research on modern systems analysis and design technologies and applications* (pp. 35–54). IGI Global.

Vanclay, F. (2003). International principles for social impact assessment. *Impact assessment and project appraisal*, *21*(1), 5–12.

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). Feds: A framework for evaluation in design science research. *European journal of information systems*, *25*(1), 77–89.

Vinocha, T. (2022). What is pdd & sdd in rpa? the two most crucial blueprints in robotics process automation [Accessed: 7-11-2022]. https://www.guvi.in/blog/what-is-pdd-sdd-in-rpa/

Weishaar, G. (2022). Uipath testing summit. https://www.uipath.com/events/testing-automation-summit

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer.

Willcocks, L. P., Lacity, M., & Craig, A. (2015). *The it function and robotic process automation*. The London School of Economics; Political Science.

Williams, J. J., & Lombrozo, T. (2010). The role of explanation in discovery and generalization: Evidence from category learning. *Cognitive science*, *34*(5), 776–806.

Witzel, A., & Reiter, H. (2012). *The problem-centred interview*. Sage.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 1–10.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media. https://doi.org/https://doi.org/10.1007/978-3-642-29044-2

Ziftci, C., & Krüger, I. (2013). Test intents: Enhancing the semantics of requirements traceability links in test cases. *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 1272–1277.

Appendices

16. Appendix A

16.1 A1: Consent form problem-centered interviews

Consent Form Problem-centered interviews

October 2022

By completing this form, you give us permission to use this interview for academic purposes and potential publications. Thank you for helping.

Research Title:	Aligning requirements and test scenarios in an RPA testing method
Researcher:	Mees Mouwen
Objective:	This interview is meant to understand the requirement analyses, translate requirements to testing, and the testing phase itself. Further questions about the documentation of these practices are done. The questions are aimed at uncovering issues that are occurring in these practices
Participation period	60 minutes
Risk of participation	We foresee no risk of participation. The participant is free to withdraw at any moment
Tools used	The interview will be audio recorded.
Data protection:	The data will only be used within the context of this research and shall not be shared with any other parties that are not already involved (Ciphix & University Utrecht). All personal data will be anonymized in (potential) publication

---

Personal Information

Full name	
Team	
Function	
Email	

Agreement

I have read and understood the conditions for participating in this research. I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason. I declare the above to be true and agree to participate by signing this consent form.

Signature Date:

16.2 A2: Problem-centered interview questions

Support interview questions:

General support issues:

1. Are there levels of severity of the issue?
2. What type of issues arises most?
3. Who addresses support issues first? You or the customer?

Types of issues that cause maintenance:

4. How often are exceptions the cause of issues?
5. How often are infrastructure or applications the cause of issues
6. How often are Issues in the code or data the cause for maintenance?

.

Testing practices after maintenance:

7. What types of tests are performed after maintenance?
    a. Are these tests already available or made by you?
    b. Do you ever need to create new test scenarios?
8. Is a testing environment always available?
9. Is a testing verison of the bot available?
10. How do you experience the testing process as support?
11. How would your ideal testing situation for support look like?

Solution architect interview questions:

1. How much experience do you have with RPA development or architecture in years?
2. How complex would je describe your last project in terms of size and difficulty (low, meduim, high)?

Requirements or process analysis

3. What steps are taken to gather a process requirements?
4. How much time was taken to gather the process steps or analyse the process?
5. Is testability already taken into consideration when writing down the requirements?
6. How are stakeholders incorporated in the process selection and analysis of the process?
7. What are issues do you (sometimes) encounter in this requirements analysis What types of requirements are gathered?
8. What is the difference between het as-is process en to-be process?
9. How important are the completion criteria?
10. How important are the application criteria?
    a. Can the application be a reason to decide a process is not suitable?
    b. How do the applications (used in the process) play a role in the testing phase?

Exceptions in process

11. How do you decide what the exceptions are of the process and which to include in development?
12. What is done if (difficult/many) exceptions show up in process analysis phase?
13. What if exceptions show up in the development phase?
14. How are exception documented?

Abstraction level of process steps:

15. To what extent is the abstraction level of requirements taken into account?

Documentation for requirements

16. How are requirements documented?
17. How do you feel about the structure of the PDD?
    a. Are there aspects that work very well?
    b. Which aspects do you sometimes struggle with?
    c. Are there aspects you think are missing from the PDD?

Requirements to test scenarios:

18. How are requirements translated to test scenarios?
    a. When in the project cycle is this done?
    b. What technique or method is used to do this?
    c. What is the role of the PDD in this translation?
19. Are you experiencing any difficulties in this translation?

Testing procedure:

20. What types of tests are performed before the bot goes live?
    a. When in development process are these tests performed?
21. How much time did you spend on testing in your last project not taking unit testing into account?
    a. Are you happy with how the current testing phase looks? Is it efficient?
    b. Was this standard compared to other projects
    c. How would your ideal testing phase look like?
    d. Did you experience any difficulties in your last testing phase?
22. Are there standard test scenarios or test cases for re-used components?

Test generation and documentation

23. How are test cases created?
24. Is testing data always provided or sometimes made by hand?
25. How do you go from test data to a test case?
26. How are test scenarios and cases documented?
27. How are issue logs documented and saved?
    a. Is testing stopped if an issue arises?
28. Are there often still issues uncovered in the UAT?
    a. How do you think you can minimize issues here?
29. Hyper-care phase are still issues uncovered?
    a. How do you think you can minimize issues here?

Environment:

30. Is a bot version created where the code is organised for testing?
    a. Are logging modules added to the bot for testing?
    b. Are (testing) bots with logging modules kept and saved after testing?
31. Is a testing environment always provided?

Effectiveness of testing:

32. Is checked/documented how much of the bot's functionality is covered by a test case?
33. How would you determine if a testing phase is effective?

System / application exceptions:

34. How are issues handled that are related to incorrect data, loss of connection, or page not loading or working?
    a. Are there standard test cases for this?
35. How is tested for these exceptions?

Tools:

36. Are specific tools used for testing?
37. Have you used Uipath test suite before?

Other

38. Are there any other this things that are missing or not going well currently in the requirement analysis phase, testing phase, or documentation?

17. Appendix B

17.1 B1: Multiple-case study database

17.1.1 Interview database

Table 17.1: Systematic interviews

Database nr.	Purpose	Role interviewee
1A	Method	RPA teamlead
2A	Method	Solution architect
3A	Method	Solution architect
4A	Method	Solution architect
1B	Documentation	RPA teamlead
2B	Documentation	Solution architect
3B	Documentation	Solution architect
4B	Documentation	Solution architect
5B	Documentation	Solution architect

17.1.2 Document database

Table 17.2: Documents and tools used in the multiple-case study

Database nr.	Case	Document or tool	Type	Function
D1	1	Document	PDD	Process description
D2	1	Document	SDD	Solution design
D3	1	Document	TPD	Test plan
D4	1	Tool	Monday.com	Project planning
D5	1	Tool	Azure DevOps	Code management
D6	2	Document	PDD	Process description
D7	2	Document	SDD	Solution design
D8	2	Document	TPD	Test plan
D9	2	Tool	Monday.com	Project planning
D10	2	Tool	Azure DevOps	Code management
D11	3	Document	PDD	Process description
D12	3	Document	SDD	Solution design
D13	3	Document	TPD	Test plan
D14	3	Tool	Monday.com	Project planning
D15	4	Document	PDD	Process description
D16	4	Document	SDD	Solution design
D17	4	Document	TPD	Test plan
D18	4	Tool	Monday.com	Project planning

17.2 B2: Case study protocol

Field procedures

Project selection procedure

The first criterion for the selection of the projects is that the solution architects for all four cases must differ. This is to have a more general view of the working method of different solution architects. The second and last criterion is that the two low-complexity and two high-complexity cases are selected.

This selection allows for an analysis of two similar complexity projects and low plus one high complexity projects. This creates a more robust analysis of the projects. The different levels of complexity are shown below.

High-complexity is more than 5 applications, more than 50 steps

Medium-complexity is between 3-5 applications, between 25-50 steps

Low-complexity is less than 3 applications, less than 25 steps

Case study procedure

The multiple-case study consists of four cases in total. For all cases, a document and tool analysis, survey, and systematic expert interviews are performed. The four cases are analysed in pairs of two iterations as shown below. The reason for performing the case study two iteration is because this allows the first iteration to provide feedback on the survey and interviews which can then be improved if necessary.

1. Document and tool analysis case 1 + 2
2. Survey and systematic expert interviews with experts from 1 + 2
3. Document and tool analysis case 3 + 4
4. Survey and systematic expert interviews with experts from case 3 + 4

Documentation and tool sources

To analyse the different RPA DLC phase and answers the case study objectives, different project documentation and tools are included in the multiple-case study. The included documentation is described in table 17.3 and the included tools are described in table 17.4.

Table 17.3: Documentation

Document	Description
PDD	Describes process and infrastructure requirements, and other conditions necessary for the robot
SDD	Describes the design of the robot and other solutions components
TPD	Describes the test plan, test scenarios, test data, and test results

Table 17.4: Tools

Tool	Function
Monday.com	Project management
Uipath studio	Development & Testing
Azure DevOps	Code management

Process

Data collection methods

To have a solid foundation of data and a logical path to our conclusion, there are a few things that we implement. Firstly, we use multiple sources of evidence to allow for data triangulation (Runeson & Höst, 2009). These sources of evidence include documentation and tool analysis, surveys, and systematizing expert interviews. Secondly, we create a case study database consisting of notes about project documentation and tools, case study documents such as survey responses and interview questions, and transcripts from the interviews. With this, we create a chain of evidence that allows us to trace our findings to our case study.

Analysing evidence

The case study exists of three methods of data collection as mentioned above. These are documentation and tool analysis, surveys, and systematizing expert interviews. All data from these methods is gathered and added to the case study database. The cases are evaluated based on a *within-case analysis* and an *across-case analysis*. The documentation and tool analysis and the survey are deductive in nature since they are analysed based on specific criteria mentioned in table 7.3. The systematizing expert interviews are structured to allow for data comparability (Döringer, 2021). The aim of these interviews are to gain insight in the processual structure of the RPA DLC in practice.

Reporting

The results of the multiple case study are written in Chapter 8 and consists of:

- One report discussing the insights gained from the systematizing interviews
- Four single-case reports discussing the within case-analysis
- One multiple-case report discussing the across-case analysis including the survey

17.3 B3: Survey questions

Survey questions documentation

PDD:

Completeness

1. PDD contains all necessary elements in regard to process requirements
2. Not all elements of the PDD are essential or necessary to be documented
3. PDD is always completely filled out

Consistency:

4. We use a standardized format to fill out the PDD
5. I have experienced inconsistencies between different PDDs
6. PDD information is consistent with the SDD
7. PDD information is consistent with the TPD
8. I have experienced difficulties in a project due to inconsistencies in the PDD

Usefulness:

9. PDD has a positive impact on the tasks I perform during the project
10. The PDD has a positive impact on the client's understanding of the process requirements
11. The PDD has a positive impact on the client's input for the project
12. The usefulness of the PDD is the same for every project

Ease of use:

13. I believe that the PDD is easy to understand for developers
14. I believe that the PDD is easy to understand for clients
15. It does not take a lot of effort to fill out the PDD

SDD

Completeness

16. SDD contains all necessary elements in regard to the solution design
17. Not all elements of the SDD are essential or necessary to be documented
18. SDD is always completely filled out

Consistency:

19. We use a standardized format to fill out the SDD
20. I have experienced inconsistencies between different SDDs
21. SDD information is consistent with the PDD

22. SDD information is consistent with the TPD
23. I have experienced difficulties in a project due to inconsistencies in the SDD

Usefulness:

24. SDD has a positive impact on the tasks I perform during the project
25. The SDD has a positive impact on the client's understanding of the design
26. The SDD has a positive impact on the client's input for the project
27. The usefulness of the SDD is the same for every project

Ease of use:

28. I believe that the SDD is easy to understand for developers
29. I believe that the SDD is easy to understand for clients
30. It does not take a lot of effort to fill out the SDD

TPD

Completeness

31. TPD contains all necessary elements in regard to testing
32. Not all elements of the TPD are essential or necessary to be documented
33. TPD is always completely filled out

Consistency:

34. We use a standardized format to fill out TPD
35. I have experienced inconsistencies between different TPDs
36. TPD information is consistent with the PDD
37. TPD information is consistent with the SDD
38. I have experienced difficulties in a project due to inconsistencies in the TPD

Usefulness:

39. The TPD has a positive impact on the tasks I perform during the project
40. The TPD has a positive impact on the client's understanding of the testing phase
41. The TPD has a positive impact on the client's input for the project
42. The usefulness of the TPD is the same for every project

Ease of use:

43. I believe that the TPD is easy to understand for developers
44. I believe that the TPD is easy to understand for clients
45. It does not take a lot of effort to fill out the TPD

Survey questions tools

Monday.com

Functionality:

46. Monday.com has all the functionality I need for project planning

Usefulness:

47. Monday.com has a positive impact on my ability to perform my project responsibilities
48. The usefulness of Monday.com is the same for every project

Ease of use:

49. Monday.com is easy to use for project planning
50. Writing down tasks on Monday.com does not require a lot of effort

AzureDevops:

Functionality:

51. AzureDevOps has all the functionality I need for code management
52. I would use the functionality of AzureDevOps for more than code management

Usefulness:

53. Azure DevOps has a positive impact on my ability to perform my project responsibilities
54. The usefulness of Azure DevOps is the same for every project

Ease of use:

55. Azure DevOps is easy to use for code management
56. Saving and maintaining in code Azure DevOps does not require a lot of effort

UiPath Studio:

Functionality:

57. UiPath Studio has all the functionality I need for testing
58. I would use the functionality of UiPath studio for more than development and testing

Usefulness:

59. UiPath studio has a positive impact on my ability to perform my project responsibilities
60. The usefulness of UiPath studio is the same for every project

Ease of use:

61. UiPath studio is easy to use for testing
62. Creating and running test cases in UiPath studio does not require a lot of effort

17.4 B4: Survey questions statistics

Table 17.5: Statistics all survey questions

5	Strongly agree
4	Agree
3	Neither agree nor disagree
2	Disagree
1	Strongly disagree

Question	Mean	Standard deviation	Question	Mean	Standard deviation
1	4.6	0.5	32	2.6	1.3
2	3.6	1.5	33	2.6	0.9
3	2.6	0.9	34	3.8	0.8
4	4.4	0.5	35	4	1.2
5	4.2	0.44	36	3.4	0.5
6	3	1	37	3.8	0.4
7	3.4	0.9	38	3	1
8	3.2	0.8	39	4.4	0.5
9	4.2	0.8	40	4.6	0.5
10	4.8	0.4	41	4.2	0.8
11	4.2	0.8	42	3.6	0.9
12	3.6	1.1	43	4.4	0.9
13	4.8	0.4	44	4.2	0.8
14	4.2	0.4	45	2.6	0.9
15	2.4	1.1	46	4	1.4
16	3.8	0.4	47	4.4	0.9
17	4	0.7	48	3.8	1.6
18	1.8	0.8	49	4	1.4
19	4.4	0.5	50	4	1
20	3.6	0.9	51	4.8	0.4
21	3	1	52	4.4	0.9
22	3.4	0.9	53	4.2	0.8
23	3	1	54	4	1.2
24	4.4	0.5	55	4.4	0.5
25	2.8	1.1	56	4.4	0.5
26	3	1.6	57	4.4	0.5
27	3	1.4	58	3.6	1.1
28	4.6	0.5	59	4.4	0.9
29	2	0	60	4.4	0.9
30	2	0.7	61	4.4	0.9
31	3.8	0.4	62	3.4	0.5

17.5 B5: Consent form expert interviews

Consent Form Survey and systematizing interview

January 2022

By completing this form, you give us permission to use the interview and response to the survey for academic purposes and potential publications. Thank you for helping.

Research Title:	Aligning requirements and test scenarios in an RPA testing method.
Researcher:	Mees Mouwen.
Objective:	The survey is aimed at deductively analyzing the use of documentation and tools within Ciphix through different criteria. This is done by providing different statements which can be answered based on a Likert scale. The interview is focused on improving the RPA VV-method by understanding the current working process within Ciphix in regard to the RPA development life cycle (DLC). The interview will use the RPA VV-method to discuss if certain tasks or activities are included in the DLC and how important they are according to the experts.
Participation period	30 minutes (survey) & 60 minutes (interview).
Risk of participation	We foresee no risk of participation. The participant is free to withdraw at any moment.
Tools used	Google Forms is used the create and conduct the survey. The interview will be audio recorded.
Data protection:	The data will only be used within the context of this research and shall not be shared with any other parties that are not already involved (Ciphix & University Utrecht). All personal data will be anonymized in the thesis report and (potential) publication.

Personal Information

Full name	
Team	
Function	
Email	

Agreement

I have read and understood the conditions for participating in this research. I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason. I declare the above to be true and agree to participate by signing this consent form.

Signature Date:

17.6 B6: Expert interviews questions RPA-VV method

RPA VV-method interview: (+-60 min)

Component per component:

1. **What do you think this component entails, please describe in aprox. 10 words?**
    a. **Is this already performed in RPA projects?**

Current projects / how would it be incorporated?

2. **In your current projects, what are essential tasks or practices that should be performed relating to this phase?**
3. **What should you keep in mind or not forget when performing these tasks?**
    a. *Is there a specific order in the tasks?*
4. **Are there prerequisites before you can perform these tasks?**
5. **What roles should are involved now in these tasks?**
6. **When would you consider this phase to be complete?**
7. **How is this phase related to documentation and tools used in RPA projects?**

Component

8. **Is this component in the correct place in the method?**
9. **How would you improve the name of the component to better encompass the description of the phases tasks better?**

10. **Anything missing from the Method that you believe is essential for the development of the bot?**

17.7 B7: Expert interviews questions documentation and tools

Systematizing expert interviews: documentation

1. Interviewee 1B:

1. You say that there is consistency between *PDD and SDD*; *SDD and TPD* but you say neither agree or disagree with consistency PDD and TPD? HOW COME THEY ARE NOT CONISTENT.

    a. Differences between TPD and PDD? →What would be the reason you think? → why easier to be consistent between SDD and other documents?
    b. Should there be consistency between these documents?
        i. What elements are most important to be consistent with each other?
    c. Record changes or inconsistencies of the documentation when it is already signed off? → *Write down notes in SDD if changes need to happen after signing off PDD? → if this results in new scenario also adjust TPD*
        i. Flow from PDD to draft SDD TPD scenarios → finish SDD, record differences with PDD, and adjust TPD scenarios?

2. Scope creep / scope change? → how to deal with it? Where is it documented? Adjust test scenarios?

3. Is there a standard format for how to document alternative happy flows in PDD?, AE, and BE in the PDD?

From document analyses: Related to traceability:

4. Are BE to test scenario 1:1 relation?
    a. Essential to have?
5. Are AE to test scenario 1:1 relation?
    a. Where are AE documented?
6. Are alternative flows to test scenarios 1:1 relation?
    a. Essential?
    b. Where documented?
7. Is this also always the case?
    a. Exceptions when they are not a test scenario or not a BE/AE/ APF?

8. Labeling BE/AE, A happy-flow in PDD, beneficial to create test scenarios in TPD
        i. BE → A happy flow
        ii. Do you ever want to test AE?

9. TPD → All elements are essential → but not always filled out and it takes a lot of effort.
    a. Effort the reason or other reasons it is not filled out?
    b. What can reduce the effort?
    c. Do you think this effort is required to have a high quality robot all the time?
    d. What can help to always have all essential information ready for the TPD?

10. SDD → Strongly agree to: not all SDD elements are essential →Internally or externally?... also not completely filled out → related to project complexity?
    a. Should SDD be simplified or should the focus lie somewhere else?
11. PDD → not all elements are essential?
    a. How come?

12. All documents take a lot of effort to fill out → this acceptable / essential?
    a. What is the reason why things are skipped?
        i. less important or to much effort?

13. Q52: What functionality of Azure do you think can be beneficial for RPA projects?

14. Deal with Discrepancies with an acceptance environment?

---

2. Interviewee 2B

PDD: → All elements are essential (strongly agree), But not always completely filled out,
    a. Why is it not always filled out?
    b. PDD Takes a lot of effort to fill out (strongly agree)
        i. This reason why they are not filled out?
        ii. How can this be solved?
    c. Issues due to inconsistencies in the PDD
        i. → elaborate?
        ii. → how can this be solved?

SDD: Not all elements are essential (agree), not completely filled out (agree) , and takes a lot of effort (Agree)!
        iii. Not being essential the reason why not filled out?
            1. Other reasons?
        iv. Anything changed about the effort? Can things be left out?

TPD: All elements are essential (strongly agree), TPD is always completely filled out (Agree), take a lot of effort (agree)
        v. SA's and developer should put effort in to completely fill it out?
        vi. Not all projects (XDM import), exceptions?

1. How did this happen

vii. **Q38:** Issues due to inconsistencies in the TPD?
1. Elaborate?

Consistencies:

1. PDD & SDD (neither agree nor disagree) ⇒ SDD & PDD are consistent (Agree)
2. PDD & TPD (Agree)
    a. Always?
3. SDD & TPD (Agree)
4. Are they indeed always consistent? Or should they be consistent?
5. What is the reasons when they are not?

d. Should differences or inconsistencies between the PDD and SDD be recorded?
   i. Labeling between documents to keep them consistent?
      1. Already done?
   ii. PDD → SDD is done? → numbering steps of PDD?
   iii. Consistency between PDD and TPD?

e. Is there a standard format for how to document alternative happy flows in PDD?, AE, and BE in the PDD?

From document analyses: Related to traceability:

2. Are BE to test scenario 1:1 relation?
    a. Essential to have?
3. Are AE to test scenario 1:1 relation?
    a. Where are AE documented?
4. Are alternative flows to test scenarios 1:1 relation?

5. Is this also always the case?
    a. Exceptions when they are not a test scenario or not a BE/AE/ AHF?

6. Scope creep / scope change? → how to deal with it? Where is it documented? Adjust test scenarios?

7. Deal with Discrepancies with an acceptance environment?

8. **Q27:** Why do you think the usefulness of the SDD is not the same for every project?

9. **Q52**: What functionality of Azure do you think can be beneficial for RPA projects?

10. **Q58:** What would you use UIPath studio for besides coding and testing?

---

3. Interviewee 3B

1. **PDD**
    a. <u>Experienced issues in the PDD due to Inconsistencies</u>
        i. Type of inconsistencies?
        ii. Issues?

    b. Effort worth it fill out PDD?
        i. Increases quality? Of the project?
        ii. NOT all elements are essential?
        iii. Internal or external?

2. **SDD**
    a. <u>Experienced issues in the SDD due to Inconsistencies</u>
        i. Type of inconsistencies?
        ii. Issues?

    b. Effort worth it fill out SDD
        i. Increases quality? Of the project?
        ii. NOT all elements are essential?
        iii. Internal or external?

3. **TPD**
    a. How important is the TPD? → Do agree it has a positive impact → skipped in case 2.
        i. Was this an exception?
            1. Why?
            2. Where did you keep track of the testing phase and test scenarios?

    b. Effort worth it fill out TPD
        i. Increases quality? Of the project?
        ii. NOT all elements are essential?
        iii. Internal or external?

4. **All docs:**
    a. How important is consistency between documents?

i. PDD is neither consistent nor inconsistent with SDD and TPD → why do you think that?
    1. Should they be?
ii. What happens when a new scenario is found in development? (scope creep)

b. Which documentation is more internal/ which is important for the client?
    i. Which aspects of the documents

5. Other:

a. Standard format how you Document HF **HFA,** and BRE in de PDD?
    1. Certain wording used?
    ii. Standard format for those in the SDD?
    iii. HF 1:1 test scenario
    iv. BE 1:1 test scenario
    v. HFA 1:1 test scenario
    vi. AE / SE (system exception) 1:1 test scenario?

b. Is there currently a method used to translate these to test scenarios?
    i. Can they be easily referred back to?
        1. Link between Test scenario to a BRE or HFA?

c. BRE in SDD & BE in PDD → reason for this difference?

d. How do you currently deal with discrepancies between acceptance and production environment?

---

4. Interviewee 4B:

1. PDD

a. <u>Experienced inconsistency with PDD</u>
    i. What type of inconsistencies?
    ii. What issues?
b. <u>PDD takes a lot of effort</u>
    i. Necessary to high quality project?
    ii. Importance project team, client, support?
c. <u>Not all elements necessary to be documented in PDD</u>
    i. How is this related?

2. SDD

a. <u>Experienced inconsistency with SDD</u>

i. What type of inconsistencies?
ii. What issues?

b. SDD takes a lot of effort
   i. Necessary to high quality project?
   ii. Importance project team, client, support?

c. Not all elements necessary to be documented in SDD
   i. How is this related?

3. TPD

a. All elements are important for TPD → tpd was filled in the worst overal
   i. Why TPD is important, and PDD and SDD unessential elements?
   ii. Importance project team, client, support?
   iii. WHy is this filled in the worst?

b. Effort worth it for the TPD for project quality?
   i. Important for project team, client, support?

4. Doc interactions

a. PDD is not consistent with the SDD
   i. What makes them inconsistent with each other?
   ii. Is there some form of labeling for traceability between PDD and SDD?

b. PDD consistent with TPD
   i. How come?
   ii. Is there some labeling for traceability between PDD and TPD?
   iii. Is there a clear link currently between this flows or exceptions and test scenarios?
   iv. How can you make this more clear?

5. Would labeling flows be beneficial?
   a. HF 1:1 test scenario
   b. BE 1:1 test scenario
   c. HFA 1:1 test scenario
   d. How deal with: AE / SE (system exception) 1:1 test scenario?

6. Standard format how HF, HFA, BE are documented?

7. How important is consistency between documents?
   a. How important is alignment/ traceability between documents?

8. Test data
   a. When is test data currently delivered in projects?
   b. What are the biggest issues you encounter with test data?

9. Usefulness is not the same for every project of the SDD
    a. Why only SDD not the same for every project compared to PDD and TPD?
    b. When are documents more / less useful in your opinion?

10. Monday not enough for project planning
    a. What functionality are you missing for project planning?
    b. What would you use Azure DevOps for?
    c. What use UiPath studio for?

11. BRE in SDD and BE in PDD?

---

5. Interviewee 5B

PDD

1. Hoe belangrijk is het invullen van PDD
2. Kost dit veel tijd en moeite?
3. Wordt het dus ook altijd goed ingevuld?

4. Very extensive → multiple decisions (HFA) & BE → multiple applications
    a. Standard format for HFA? For BE

5. Should HFA (happy flow alternatives) be mentioned within the PDD?
    a. Are they?
    b. What format would you prefer?
    c. Zijn ze labeled?

6. Betere link tussen:
    a. BE 1:1 test scenario
    b. HFA 1:1 test scenario
    c. AE / SE (system exception) 1:1 test scenario?
        i. Verschil AE en SE?
            a.
7. Is there a difference between BE (PDD) and BRE(SDD)
        i. Reason why this is the case?

SDD:

8. Hoe belangrijk is het invullen van PDD
9. Kost dit veel tijd en moeite?
10. Wordt het dus ook altijd goed ingevuld?

11. Zijn de BRE in SDD pas ontdekt door development en in de test data? → cannot be found in PDD
    a. Waar zijn de BE van de PDD gebleven?
12. 2 types of BRE mentioned in SDD? Why?
13. Use a AE, which seems similar to the BRE → why is it AE, and not BRE?

TPD:

14. Hoe belangrijk is het invullen van PDD
15. Kost dit veel tijd en moeite?
16. Wordt het dus ook altijd goed ingevuld?

17. TC12). Images in PDF → can I find this in SDD or PDD?
    a. When was it added?
18. TC3). Why is Myler here its own test scenario while in PDD its mentioned as the same email and I believe scenario?
    a. Data also missing for myler?
19. The exceptions table in TPD → What exactly is this?
    a. Is it found during development or UAT?
    b. Wordt dit altijd bijgehouden?
20. Hoe zij je HFA noemen die alleen ontstaan als andere data er is maar niet een andere scenario worden?
    a. Waar hoort dit nog meer gedocumenteerd te worden?

Consistency:

21. Denk je dat consistency tussen PDD en SDD belangrijk is?
22. SDD en TPD?
23. PDD en TPD?

Test data:

24. Process data op dit moment goed gedocumenteerd?
    a. Mis je hier iets?

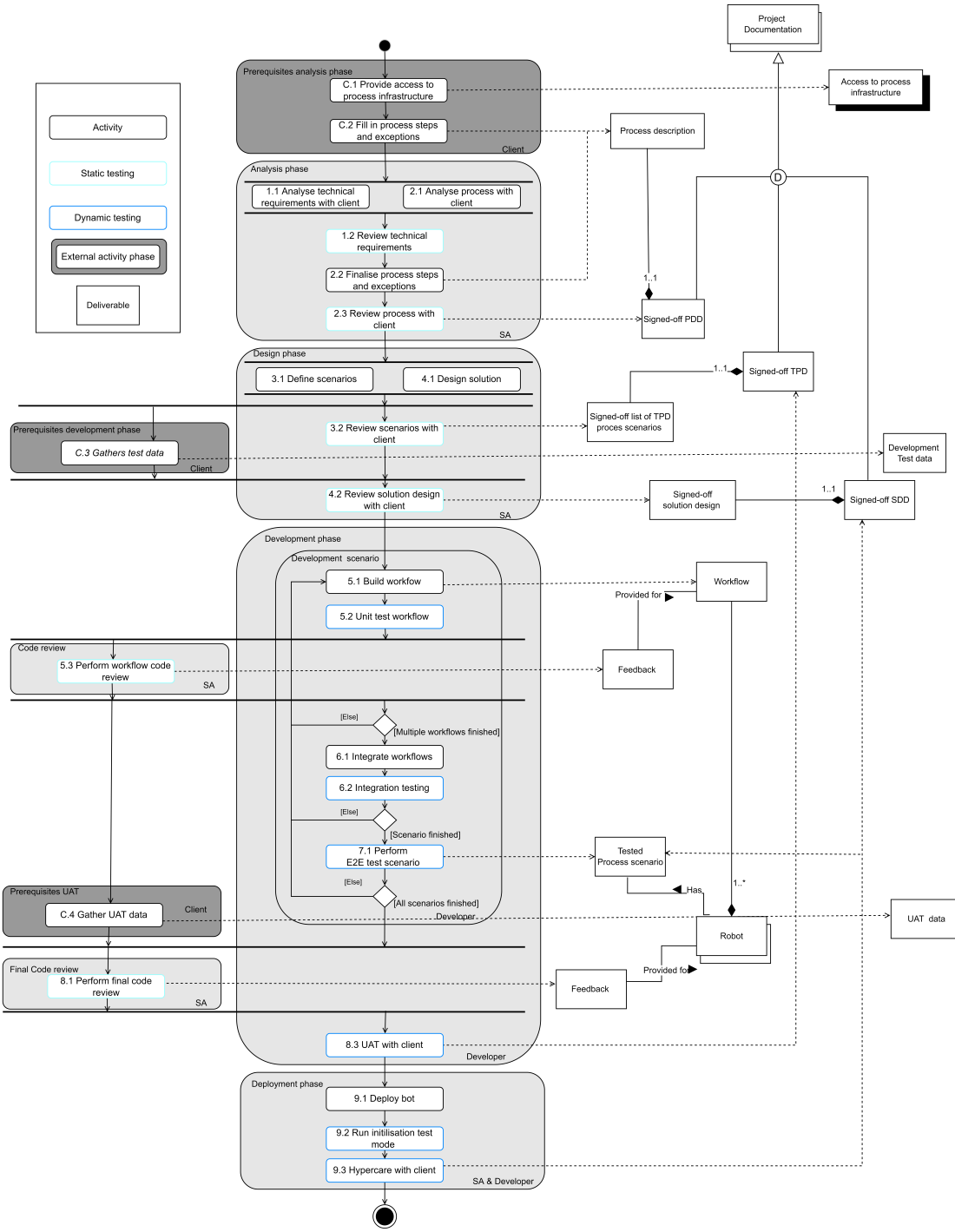
Acceptance environment:

25. In dit project nog problemen hervonden door verschil in acceptatie en productie environment?

---

18. Appendix C

18.1 C1: RPA VV-method PPD format



19. Appendix D

19.1 D1: Consent form focus group

Consent Form focus group

March/April 2023

By completing this form, you give us permission to use focus group for academic purposes. Thank you for participating!

Research Title:	Aligning requirements and testing in RPA projects. Designing an RPA verification and validation method.
Researcher:	Mees Mouwen.
Objective:	The focus group is aimed at validating the RPA-VV method and guidelines through different criteria. All guidelines will be presented and questions are provided to start a discussion between the participants.
Participation period	60 minutes
Risk of participation	We foresee no risk of participation. The participant is free to withdraw at any moment.
Tools used	The focus group will be audio recorded.
Data protection:	The data will only be used within the context of this research and shall not be shared with any other parties that are not already involved (Ciphix & University Utrecht). All personal data will be anonymized in the thesis report and (potential) publication.

Personal Information

Full name	
Team	
Function	

Agreement

I have read and understood the conditions for participating in this research. I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason. I declare the above to be true and agree to participate by signing this consent form.

Signature Date:

19.2 D2: Focus group questions

A. Vragen phases RPA methode:

Simplicity

1. Is de fase duidelijk te begrijpen?

Usefulness

1. Zijn deze activiteiten en deliverables belangrijk om uit te voeren tijdens de RPA development life cycle?
    a. Zo niet, waarom zou je het niet uitvoeren?

Completeness

2. Zijn er essentiële activiteiten of deliverables die ontbreken?
    a. Zo ja, wat mist er?

B. Vragen voor alle Guidelines:

Simplicity

1. Is de fase duidelijk te begrijpen?

Usefulness

3. Zal het toepassen van de guideline de qualiteit van RPA development life cycle verbeteren?
    a. Zo niet, waarom is hij niet nuttig?

Ease of use

4. Is de guideline makkelijk om uit te voeren? Of zie je obstakels tijdens het uitvoeren hiervan?

C. Vragen voor alle issues:

Completeness

5. Mist er een guideline die kan helpen met de issue?
    a. Zo ja, hoe zou je dit omschrijven?

D. Afsluitende vraag:

Functionality

6. Geeft de methode steun om door de RPA development life cycle te gaan? Zou je the methode in de RPA implementatie standaarden verwerken?