Universiteit Leiden

ICT in Business

A framework for a centralized external RPA orchestrator

Name: Diederik Oprel
Student-no: 3124878

Date: 05/01/2025

1st supervisor: Drs. J.B. Kruiswijk
2nd supervisor: Dr. P.W.H. van der Putten

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgements

This thesis has been a slowly progressing project besides my full-time job. Therefore, I would like to thank my primary supervisor, Bas Kruiswijk, for his patience, positivity, use of his connections and continued feedback while guiding me through this thesis. I would also like to thank my second supervisor, Dr. Peter van der Putten, for his insightful comments based on his industry experience, feedback and for connecting me with relevant people in the automation industry.

Additionally, I would like to thank everyone who took the time to complete the survey and participate in the interviews.

Finally, I would like to thank my girlfriend, Amber, and my parents for supporting me and caring for things so I could focus on completing my thesis.

Abstract

Robotic Process Automation, commonly referred to by its abbreviation RPA, is a software tool for creating low-code business process automation. It exists as a non-intrusive form of automation, mimicking the interactions of a standard user with a computer. The developed processes can be scheduled to run at certain intervals per the business's needs. However, insufficient control regarding the execution of processes developed with RPA is reducing the tools' effectiveness. This thesis investigates the presence of problems with Robotic Process Automation orchestration and how these can be addressed by introducing an improved external orchestrator.

The research sets out to address the question, 'How can RPA reliability be improved by a vendor-agnostic improved process orchestrator?'. The sub-questions divide the research into the following topics: the consequences of the current situation, requirements for an improved situation, improvements in that situation, and the structured way to address it via a framework.

A survey was conducted and answered by 33 RPA professionals, and seven semi-structured interviews were held with RPA professionals. Both groups were asked about their struggles with RPA orchestration, what features were missing, and how this would improve the reliability of their deployment. By analysing the suggested features, a framework was developed to catalogue and group the features. Based on the features, a prototype was developed to showcase the feasibility of implementing an external orchestrator alongside an existing RPA tool.

The survey and interview analysis shows a wide variety of RPA tools being used, with a strong recognition of orchestration problems despite the tool in operation. A wide set of suggested features which could be grouped in similar ways as the three-tier architecture, on which the framework is based. The analysis of the features boiled down to customizability as the main requirement, due to the variety of business processes and runtimes. The prototype successfully showed the feasibility of implementation using a subset of the features with limited tools and time, resulting in a valid framework guiding the development of an orchestrator.

Table of Contents

List of figures

List of tables

Definition of terms

Abbreviation	**Term**	**Description**
RPA	Robotic Process Automation	Software method to automate digital processes
CoE	Center of Excellence	The department which focuses on RPA
PPHE	Park Plaza Hotels Europe	International hotel and real estate group where the research is being performed
-	Script	The order of automated actions created by a developer in the RPA tool.
-	Robot / Virtual Agent / Virtual Worker / Bot	The instance of an RPA script performing the automated actions.

1. Introduction

Businesses strive for greater automation and efficiency to reduce costs, minimise human errors, and address workforce shortages. One way of achieving these goals is through Robotic Process Automation, a non-intrusive way of automating software interactions mimicking a user's inputs.

1.1. Personal motivation

While RPA has significant potential, commercial implementations face downsides. One major issue is the limited flexibility of built-in process orchestration, affecting the robustness of RPA operations. Errors or limitations can disrupt process scheduling without properly alerting stakeholders or mitigating impacts, leading to higher maintenance loads and limiting scalability. Multi-platform deployments further complicate management due to fragmented alerts and dashboards.

An open-source, vendor-agnostic orchestration platform could address these issues by centralising RPA monitoring and allowing for more complex and responsive process orchestration. This topic is highly relevant for many RPA professionals and is frequently discussed at industry meetings and events.

The importance of automation in today's world cannot be understated. It significantly boosts company output and, broadly speaking, enhances human productivity. For example, within RPA, the NHS has been a prominent adopter, scaling healthcare services across the UK. The RPA market is estimated to be worth 18 billion dollars in 2024, growing at 17.1% annually. (Fortune Business Insights, 2024)

Despite the rapid growth of RPA, formal theoretical research in this area is limited. This thesis presents novel work in the field, serving as a starting point for further research and providing a theoretical framework for orchestration development.

1.2. Research aims and objectives

The primary research question is as follows:

How can RPA reliability be improved by a vendor-agnostic improved process orchestrator?

This question can be divided into the following sub-questions:

RQ1. What are the consequences of an unreliable RPA environment?
RQ2. What are the requirements for an improved RPA orchestrator?
RQ3. How would these requirements translate into a high-level framework for RPA orchestration?
RQ4. How does this improve the reliability of an RPA deployment?

The following hypotheses have been proposed to cover the primary research question:

H0 - *An improved RPA scheduler does not measurably affect reliability.*
H1 - *An improved RPA scheduler increases reliability due to its extended feature set.*
H2 - *An improved RPA scheduler decreases reliability due to having an extra system component failing or requiring end-user support/explanation.*

1.3. Scope

This research primarily focuses on the design science of creating and validating a framework for RPA orchestrations. Therefore, the primary deliverables will be:

- A high-level framework for RPA orchestration compatible with major RPA tools.
- A prototype of the orchestrator in PPHE's RPA environment implementing a subset of the framework's features.
    a. A limited assessment and reflection on the impact of the prototype on PPHE's RPA operations.

The literature review will solely discuss relevant topics for the rationalization and design of such a framework. It may omit other aspects of discussed topics such as RPA and not provide a full understanding of the underlying technology.

1.4. Structure of the thesis

This thesis will first explore available literature on the topic of robotic process automation and its aspects relating to orchestration. Afterward, based on the literature a survey and interview will be composed and conducted. The surveys and interviews will collect answers from RPA professionals, and the results will be dissected. The answers will be analysed to form major takeaways contributing to answering the research questions. From the answers, a list of features will be created, which forms the basis for the RPA orchestrator framework. These features will then be modelled and developed into a prototype to provide an example and show the feasibility of developing such a framework. Then, the primary takeaways will be reiterated in the discussion. Finally, the conclusion will provide closure regarding the actual results of this thesis.

2. Methodology

The posed research questions can be generalised into three research categories, firstly RQ1 and RQ2 are about obtaining information, secondly, RQ3 is about creating new work based on this information, and finally, RQ4 is about validating the created work.

These categories will be structured into a research approach consisting of 5 stages, primarily rooted in design science:

1. Theoretical framework
2. Survey
3. Framework
4. Interviews
5. Prototype

Design science is the practice of creating new methods, models or systems to help address the identified problems. This approach has been chosen due to the focus of this research on providing a basis for exploring orchestration requirements and starting the development of a custom orchestrator. The framework will be designed and serve as such a jumping-off point. Considering Figure 1 the kind of design science contribution which will be made will likely exist within the improvement domain. Where the problems will be known, and a new solution will be created to address the issue. (Johannesson & Perjons, 2014)

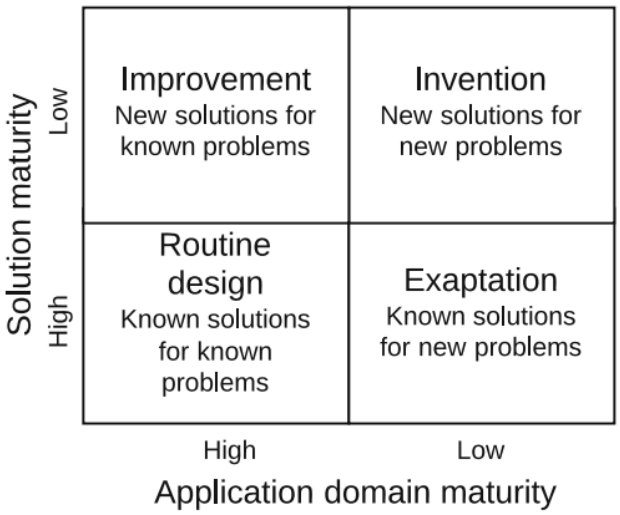

Figure 1 - Kinds of design science contributions (Johannesson & Perjons, 2014)

These five steps have been chosen to provide a robust and representative conclusion. As the framework will be one of the primary outputs of this research, the interviews and prototype allow for validation of the prototype. By consulting external experts and putting the framework to use, a generic reusable framework can be constructed built upon a solid foundation of knowledge.

2.1. Theoretical background

To provide a frame of reference and introduction to the relevant topics, the available literature will be reviewed and condensed into a theoretical background. The literature review will also provide early explanations for research question 1, stating the consequences of an unreliable RPA environment. Moreover, relevant forms and frameworks regarding orchestration will be investigated. Finally, it will provide a foundation for the interviews and survey, helping design questions which can be asked in these stages.

2.2. Survey

The survey will be the primary source of information for answering research questions 1, 2 and 3, the consequences of an unreliable environment, the requirements for an improved orchestrator, and how these requirements would translate into a high-level framework. The survey will be shaped based on the literature review. Additionally, the survey will serve as a primary way to source interview candidates.

2.3. Framework

The framework will be created by analysing the survey responses and extracting common themes between features. This will form a framework where the features can be plotted and categorised. These categories will encompass all or the majority of features which can reasonably be contained within an orchestrator.

The features which are to be plotted inside this framework will be selected when they are commonly mentioned or deemed highly impactful for an RPA environment. Not all possible or already present features will be listed in the framework, only the ones mentioned in this research. However, it should allow space for expansion and additions based on existing or future developments.

The features will individually be described in a list, detailing their purpose and use within an RPA environment.

This process will provide an answer to research question 3, asking how requirements would translate into a high-level RPA orchestration framework.

2.4. Interviews

The interviews will be designed to gain more insight into the relationship between orchestration and RPA. It will be a semi-structured interview, containing a list of questions that need to be answered but have the ability to go off-script into related topics.

For the interviews RPA professionals will be chosen who have experience with multiple RPA tools to provide a wide field of experience.

During the interview, the aforementioned framework will be subjected to review by these RPA experts. They will be given the opportunity to review and comment on the framework, testing whether it can encompass all orchestration features.

2.5. Prototype

The prototype will seek to prove the validity of the framework by implementing a subset of the framework in the RPA department of PPHE. Being our available testing platform, PPHE's environment will be used to showcase the feasibility of implementation. For this reason, the feature selection will be founded on the needs of PPHE, as every company and tool might have different requirements. However, a broad spread of features will be targeted to provide insight into each section of the framework. To provide this broad spread an additional number of features will solely be modelled and not technically implemented. These models will show how the feature should integrate and work from a theoretical point of view.

This implementation will be limited but prove the feasibility and based on the experiences within the company provide an answer to research question 4 and the main research question, asking how an improved RPA orchestrator improves RPA deployment reliability. This answer will be in line with one of the hypotheses posed in the 'research aims and objectives' section.

3. Theoretical background

The following chapter will provide a theoretical foundation for the research and provide an overview of related work.

A study in 2019 established that a significant gap exists in the amount of literature regarding RPA. The research which has been performed primarily focuses on case studies regarding RPA implementations. At the time little theoretical research had been done and no theoretical frameworks were formed. (Ivančić, Suša Vugec, & Bosilj Vukšić, 2019) While more contributions have been made since this time, the field remains a niche technology.

3.1. Robotic Process Automation

The term Robotic Process Automation (RPA) refers to the mimicking of human interactions with a computer by software. RPA scripts complete processes in the same way a human would, by navigating through the user interface and performing the necessary interactions in a non-invasive manner. The software instances performing these processes are often referred to as robots, bots, or digital workers. These terms just reference the single license of the RPA software residing and operating on a PC or server.

These scripts are designed by professional or citizen developers in an often low-code environment. Thus, making the barrier to entry low and the processes able to be automated by the humans who originally performed the process, so-called 'citizen developers'.

Certain processes will never be ready for automation, RPA is designed to run unattended and be simple in terms of development. RPA can therefore not make complex decisions or deal with exceptional cases. Table 1 shows the commonly required or beneficial characteristics of processes suitable for RPA. The processes for which RPA is commonly suited can also be referred to as 'swivel chair processes', where the processes concern the data retrieval, manipulation, and uploading between an array of different systems. (Willcocks, Lacity, & Craig, 2015)

Criteria	**Details**
Create business value	A process should provide sufficient business value to be worth the time to automate and maintain.
High volume / Transactional	RPA is well suited for dealing with processes which need to perform many smaller identical tasks. This can even be parallelized across multiple bots.
Prone to error	Processes which are prone to human error, such as data entry/migration, are ideal for RPA, as the software will not make such errors.
Repetitive	Repetitiveness makes scripting easier, as it can loop through the same actions.
Rule-based	The process should be able to follow clearly defined rules as RPA by itself is not able to make complex decisions.
Predictable / Stable	The applications and procedures should be stable to keep maintenance and process alterations to a minimum.
System diverse	RPA can easily interact with multiple systems consecutively and share data between them.
Structured data	The data source for the process should be well-structured and easily interpretable by a machine.

Table 1 - Process suitability criteria for RPA (Fung, 2014) (Patri, 2020) (Schuler & Gehring, 2018)

However, not all suitable processes are directly ready for automation, it first requires a critical look and potential business process re-engineering. Automating an inherently flawed process will provide an automated process which is unstable or slow. Many of these processes can be better tailored towards the capabilities of a robot and provide the opportunity to process data in bulk with clearly defined business rules. (Davenport & Brain, 2018)

Due to the nature of RPA primarily interacting with the presentation layer, it can also be inferior to better integrated methods of automation, such as an API or custom automation initiatives. RPA is well suited when an API is not available or too limited, a custom initiative is too expensive and time-intensive, or when a temporary measure should be taken such as data migration between systems.

3.1.1. Reliability and maintenance challenges

Aside from the many upsides RPA can present, it also presents issues regarding reliability.

Since RPA is frequently owned by business users who automate their processes or a centre of excellence, it can be outside of the visibility of IT. This can lead to an unsustainable form of shadow IT, which makes managing malfunctions and security tougher. (Schuler & Gehring, 2018) In a Deloitte case study, IT was shown to be the department least supportive of the implementation of RPA, with only 31% showing support for RPA. (Deloitte, 2017)

When these automations are developed by developers without a formal computing or IT background, certain best practices will be unfamiliar to them and therefore not implemented. These best practices from traditional software development can include but are not limited to proper documentation, error handling, formal testing procedures, and keeping process logs. (Drost, et al., 2020) (Noppen, Beerepoot, Weerd, Jonker, & Reijers, 2020) Limited documentation can also lead to process knowledge being lost after automation as the process is out of sight for developers and business users. This can increase required maintenance and reduce the reliability and robustness of a process. (Schuler & Gehring, 2018)

Apart from the less experienced developers, RPA also remains a relatively immature technology, only starting to gain public interest after 2015. (Kregel, Koch, & Plattfaut, 2021) Due to this immaturity, the knowledge surrounding the maintenance of RPA processes lacks formalisation within organisations. (Noppen, Beerepoot, Weerd, Jonker, & Reijers, 2020)

As previously shown, RPA is often deployed as a stop-gap measure when lacking formal automation infrastructure. Combined with the low-code nature of many RPA platforms, this results in RPA developments being deployed quickly. Due to the quick development cycle, there is limited time for extensive stability testing, with the suggested time of development for a process being 2 weeks, according to consultants at KPMG advising PPHE, the host company for this research.

As mentioned above, RPA primarily interacts with the presentation layer of applications, which is much more prone to changes than an underlying database layer with which formal tools integrate. Therefore, any changes made to the presentation layer can render the scripted automation inoperable, without alerting the development team. This is especially problematic in applications where the development team has no governance regarding updates, such as web-based platforms. Therefore, growth in the number of automations performed also leads to exponential growth in the amount of maintenance required.

3.1.2. Software landscape

The RPA vendor landscape includes just over a dozen notable players recognised by Gartner, a market research firm publishing yearly reports on RPA. Figure 2 shows the prominent vendors and their position in the market, according to Gartner. (Gartner, 2024) Apart from this, there are more tools that are not being recognised, especially open-source tools such as Robocorp, OpenRPA, and TagUI, which have garnered an enthusiastic following. This following can be seen in communities such as ‘I Love Automation’ present on LinkedIn and Discord. (Jensen, n.d.)

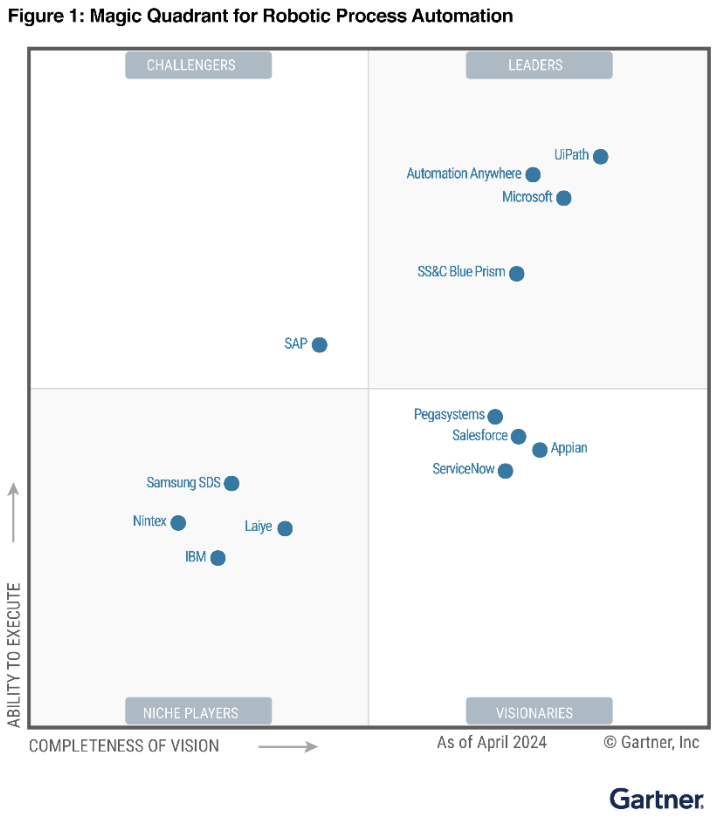

Figure 2 - Gartner Magic Quadrant for RPA vendors (Gartner, 2024)

Due to the differences between tools, some are better in certain areas than others. Therefore, you can also operate multiple different tools at the same time, a multi-platform approach.

An example for choosing multiple tools can be a difference in feature set, such as better intelligent automation or artificial intelligence capabilities. (Kim, 2023) (Keymark, 2023) Different tools also provide different integrations; as PowerAutomate is a Microsoft product, it integrates seamlessly with their Office suite and ChatGPT/CoPilot. (Microsoft, 2023) Another reason could be compatibility with your existing hardware and software stack; certain tools may be better suited for legacy or alternative operating systems, such as an old Windows Server instance or a distribution of Linux. Separate certification programmes and developer networks per tool can affect the availability of certified developers per RPA tool. Finally, the licensing costs also drastically differ per tool; a more expensive tool such as Blue Prism can be deployed for the extensive audit functionality it offers while using a cheaper tool for processes where auditability is of lower concern.

Feature	Blue Prism	UiPath	Automation Anywhere	Microsoft Power Automate
Running processes				
Fixed time process scheduling	X	X	X	X
Queueing scheduled processes		X		
Call process from web/API	X	X	X	X
Trigger process by activity			X	X
Reporting				
Maintain processing logs	X	X	X	X
Show bot utilisation	X	X	X	X
Process status desktop notifications	X	X		

Table 2 - Overview of notable orchestration and scheduling features from top 4 RPA providers (Blue Prism, 2024) (UiPath, 2024) (Microsoft, 2024) (Automation Anywhere, 2021)

3.1.3. Environment

RPA is commonly deployed on-premises or in a hosted cloud environment, but they will both follow similar principles in terms of infrastructure layout. Figure 3 and Figure 4 indicate the characteristics, layout, and responsibilities of an RPA deployment. The primary components for its operation consist of an interactive client, runtime resources, an application server and an SQL database. The developer uses the interactive client to interact with Blue Prism and create automated processes. The runtime resources run the automated processes in an unattended manner, where it is scheduled to log in to the server, perform the necessary actions and log out. The database server stores all information regarding Blue Prism; none of this is stored on the local machines. The application server functions as the connecting piece between these servers, facilitating the communication between the database and all other servers.

Therefore, orchestration is performed from the application server, triggering process information to be sent to the runtime resources based on schedules or commands sent from the interactive client.

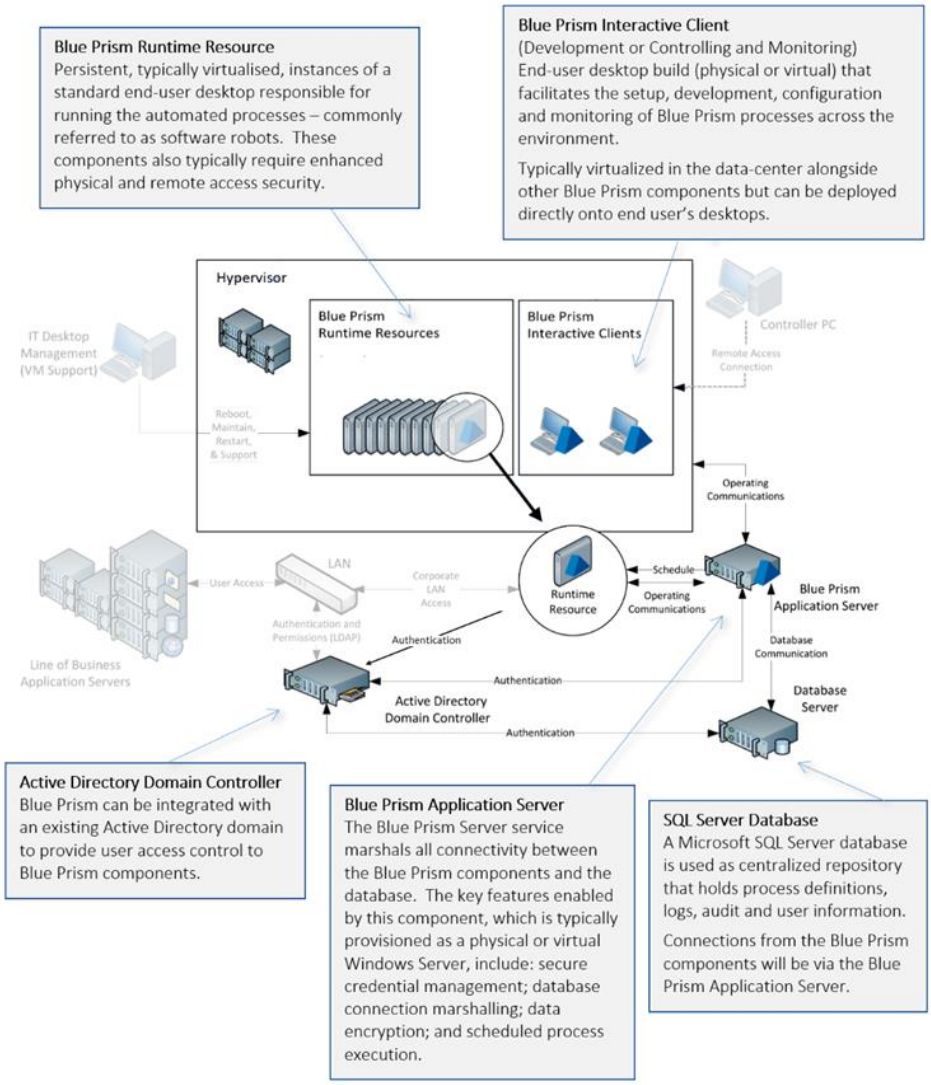

Figure 3 - Blue Prism server infrastructure characteristics (Blue Prism, 2024)

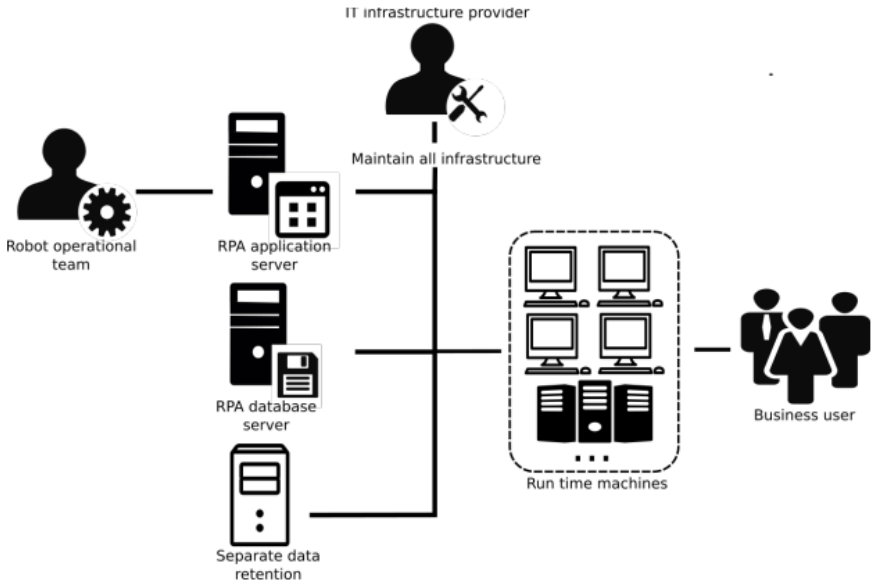

Figure 4 - Typical RPA infrastructure layout (Schuler & Gehring, 2018)

3.2. Orchestration

To better understand the concept of orchestration and how it can be implemented in other systems, we'll consider several frameworks related to the orchestration of processes in an IT setting.

An orchestrator is a system which communicates with related systems and instructs them on what and when to perform certain actions. Orchestration helps optimise the efficiency and effectiveness of an underlying program by ensuring actions are triggered and prioritised correctly, resulting in minimal downtime. The difference with choreography is that choreography tracks the interactions between sources, whereas orchestration initiates execution. (Peltz, 2003)

When researching the implementation of orchestration in web services, Web Services Business Process Execution Language (WS-BPEL, BPEL or BPEL4WS) can be seen as a common framework for the orchestration of web services. However, since many of the sources and revisions date back over 15 years, its relevance can be doubted. (OASIS, 2007) (Juric, Sarang, & Mathew, 2006) Possible reasons for this can be that such forms of orchestration aren't popular anymore and this is performed at a different level. Due to the lack of a better alternative, BPEL will still be considered for this research.

Related to BPEL, BPMN 2.0 also exists as a more modern concept in the same field. However, BPMN 2.0 has a greater focus on the visualisation and modelling instead of the execution. External platforms like Camunda allow for execution but aren't part of the official standard, making it an insufficient framework when solely talking about the orchestration execution. (Jurišic, 2011) (Geiger, Harrer, Lenhard, & Wirtz, 2018)

BPEL is a standardised language utilised for the construction, specification, and execution of business processes involving web services. It facilitates the composition of these services into an integrated process that can be executed either sequentially or in parallel, depending on the specific requirements. BPEL processes define the order of service composition and include a range of functionalities such as conditional activities, loops, variable declarations, and fault handlers, enabling the creation of complex business processes through algorithmic means.

Graphical representations of these processes are often described using Unified Modelling Language (UML) activity diagrams. BPEL supports two primary paradigms: executable processes and abstract business protocols. Executable processes detail the intricate workings of business processes and are designed for execution by an orchestration engine. In contrast, abstract business protocols outline the public message exchanges between parties without including the internal details of process flows, following the choreography paradigm.

Several elements are essential to construct a BPEL process. These include the identification of business partners interacting with the process, the specifics of data exchange types between the process and these partners, and a

defined workflow that dictates the sequence of process execution, including service invocation and data mapping. A BPEL process necessitates the use of a Web Services Description Language (WSDL) file, which provides the necessary framework for creating an executable BPEL definition. The WSDL file comprises components such as the namespace, partner link types, operations, and messages essential for defining process activities. Additionally, namespaces must point to associated WSDL schema locations and other resources like XSL style sheets and XML files used in the process.

Building a BPEL process involves specifying the execution order, activities, and conditional behaviours within a BPEL source file. This process interacts with external services through the WSDL interface, ensuring a cohesive and executable business process. (Albreshne, Fuhrer, & Pasquier, 2009)

4. Results

The following results section will introduce the actions taken to address the research questions and their outcome. Starting with gathering the information required for understanding the current orchestration problems and their solutions. Continuing to distil this information into a generalized framework and validating this framework in multiple ways.

4.1. Survey

The field research is led by a survey distributed across channels frequented by RPA professionals. The survey forms the basis of the research as it allows for a high number of respondents. The findings from these responses can then be distilled into a framework and further refined using interviews with subject matter experts.

The survey asks a number of questions which aim to uncover critical insights related to the reliability and orchestration of RPA. These inquiries delve into the experiences, challenges, and expectations of RPA professionals. By examining aspects such as unreliable execution, orchestrator functionality, and multi-platform considerations, we seek to address fundamental questions regarding the effectiveness and robustness of RPA deployments. The responses collected from experts in the field will contribute significantly to understanding the nuances of RPA environments while informing about potential and desired improvements.

4.1.1. Survey design

Table 3 lists each question asked to survey respondents, alongside what research question it helps answer. It is not an exact reflection of the survey as certain questions were split up to allow for multiple-choice questions combined with free text answers, additional notes, or confirmation on terminology definitions used during the survey. The full survey including the logic and flow can be found in appendix A.

The survey will lead with an introduction to gain insights into the background of each respondent and their familiarity with and exposure to RPA. It continues asking questions regarding their current experiences with RPA reliability and how severely reliability issues affect them. After gathering information about the underlying problem in the reliability segment, questions are posed about how to address these issues by using RPA orchestration. The RPA orchestration section inquires about the desired features of the respondent and how they would resolve their issues. To address another issue, the requirement for platform agnosticism is assessed in the following section, verifying whether this is desirable and collecting users' experiences. Finally, the respondents are asked to participate in an interview to aid further research into this topic.

Questions S2.3 and S3.1 contained a section where users could select predefined answers and write their own responses. These predefined responses were provided to grant some context for what kind of answers could be expected in this section. The predefined responses were sourced from personal experience as no possible effects or improved features could be found in related literature. The suggested effects can be found in Figure 7, while the suggested features will be explained in '4.4 Feature Extraction'.

In Table 4, the justification for each question can be found, explaining how it contributes to the research and why it was included.

#	Question	Related RQ
Introduction		
S1.1	How often do you use RPA tools?	-
S1.2	Does your job involve RPA development?	-
S1.3	What is your job title?	-
S1.4	How many years have you worked with RPA?	-
S1.5	What RPA tools have you worked with?	-
RPA Reliability		
S2.1	Have you worked with RPA environments where process execution could be unreliable at times?	1
S2.2	Have you faced an increase in maintenance due to process orchestration or scheduling?	1

S2.3	What effects due to issues with process orchestration or scheduling have you encountered?	1
S2.4	How severely do such issues affect your day-to-day work and maintenance regarding RPA?	1
RPA Orchestration		
S3.1	What features should an RPA orchestrator contain?	2
S3.2	How would the features you mentioned or included in the list above help resolve the issues you're currently facing with RPA scheduling and orchestration?	4
Multi-platform		
S4.1	Have you worked with multi-vendor RPA deployments, running different RPA tools alongside each other?	2, PRQ
S4.2	What were your experiences regarding RPA scheduling and orchestration with multi-vendor RPA deployments?	1, 2, PRQ
Interview		
S5.1	Would you be available for an interview?	-

Table 3 - Survey questions combined with related research questions

#	Question justification
Introduction	
S1.1	This question serves as background to gather more information from the respondent and assess their expertise. Frequent use of RPA tools demonstrates expertise due to the high amount of exposure the user gets to the platform.
S1.2	This question serves as background to gather more information from the respondent and assess their expertise. Professional RPA development signifies their expertise and knowledge of larger scale RPA deployments.
S1.3	This question serves as background to gather more information from the respondent and assess their expertise. The respondent's job title reflects their seniority and specific task or technology exposure within the RPA ecosystem.
S1.4	This question serves as background to gather more information from the respondent and assess their expertise. The number of years the respondent has been involved with RPA tools shows the breadth of their knowledge, having seen the ecosystem evolve and a broad set of use cases, alongside directly relating to their expertise and seniority.
S1.5	This question serves as background to gather more information from the respondent and assess their expertise. Having used a multitude of tools shows the breadth of knowledge regarding features included and missing across a larger part of the RPA market.
RPA Reliability	
S2.1	This question tries to verify whether research question 1, an unreliable RPA environment and its consequences, is an actual issue for the respondent. Unreliable execution directly links to the orchestrator's functionality of starting and correctly stopping processes.
S2.2	This question tries to verify whether the unreliable RPA environment, mentioned in research question 1, relates to the orchestrator and affects day-to-day operations for the respondent. Maintenance is performed continuously where necessary for live processes, minimising this shows a well-functioning RPA environment which can focus on more value creation.
S2.3	This question tries to extract specific details regarding what issues the orchestrator may have caused to label the environment as unreliable at times. This directly goes to answer research question 1, exploring the consequences of an unreliable RPA environment.
S2.4	This question tries to assess the impact of issues which arise due to orchestration and cause an unreliable RPA environment, providing more context for the answer to research question 1.
RPA Orchestration	

S3.1	This question directly impacts research question 2 by sourcing the most requested features directly from RPA professionals. The broad experiences of all respondents allow for a wide coverage of potential features and use cases.
S3.2	This question directly impacts research question 4 by relating the suggested features to improvements in the respective RPA environment. This shows how reliability can be improved and what the goal is of the features.
Multi-platform	
S4.1	This question speaks to the requirement of vendor agnosticism mentioned in the primary research question. Vendor agnosticism indicates the tool can be used by any platform but also by multiple platforms simultaneously. It can also be attributed to the requirements for an orchestrator in research question 2.
S4.2	This question speaks to the experiences the respondent had with the deployment of multiple RPA platforms in a single environment. It can elaborate on the issues, the requirements for a successful deployment and the urgency of implementation, related to research question 1, research question 2 and the primary research question, respectively.
Interview	
S5.1	This question serves as a sourcing tool for the interviews which are a later step of the methodology. Combined with the background of each respondent it allows for the selection of experts in the field of RPA.

Table 4 - Justification of survey questions

4.1.2. Survey participants

43 valid responses were recorded, out of which 33 fully completed the survey.

Over 90% of the respondents indicated their job involved professional RPA development. Of the same group, 88% reported using RPA tools daily or multiple times per week, and 95% for professional developers. These individuals averaged nearly 5 years of experience with RPA. Their job titles ranged from many RPA developers to creators of RPA tools, developers of RPA orchestrators, and RPA teachers.

Figure 5 shows the familiarity with specific tools across all respondents where they could select multiple answers. Respondents could elaborate on their 'Others' response by providing the specific tool name, this mentioned several less popular tools once, except for 'Softomotive' which was mentioned twice. 'Softomotive' was acquired by Microsoft in 2020 to expand their RPA capabilities, thus currently being merged into the PowerAutomate product line. (Microsoft, 2020)

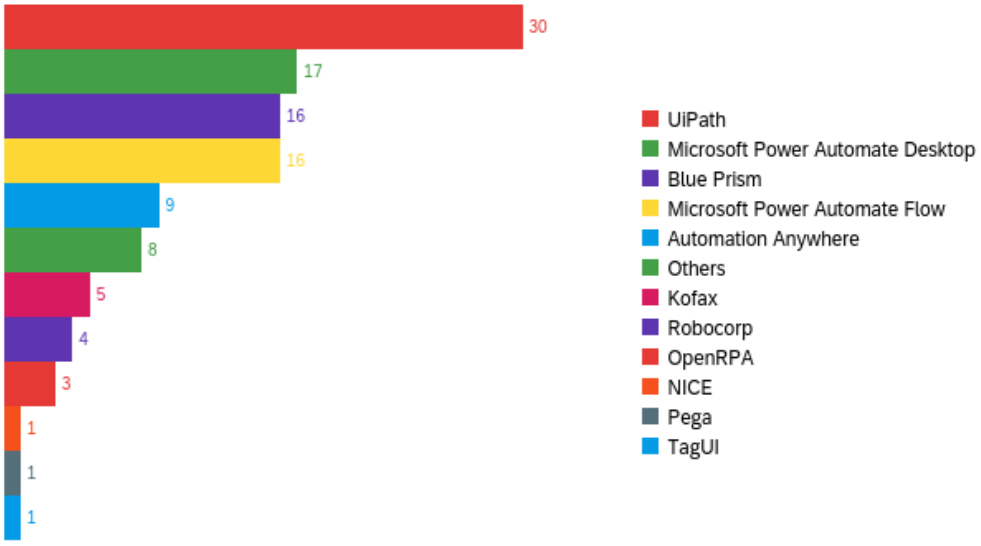

Figure 5 - Tool usage across survey respondents

4.1.3. Survey results

The features, specific issues and solutions related to an improved orchestrator from the survey and interviews have been aggregated and will be presented in '4.3.3. Interview results' and '4.4 Feature Extraction'. In this section, the results of the survey related to the respondents' background, problem recognition and issue identification will be shown.

One of the primary goals of the survey was to verify whether the problem was recognised throughout the RPA community. Out of the respondents, 77% have worked with an RPA environment where process execution could be unreliable. Due to the limited number of results and users working with multiple tools, this unreliability cannot be attributed to specific tools more than others.

Another validating number of the problem is that 60% of the respondents attributed an increase in maintenance due to problems with process orchestration or scheduling. When breaking down this response by tool usage of respondents in Figure 6, we can see UiPath being distanced from the other tools with less reported maintenance impact. This could indicate that UiPath has a more robust orchestration and scheduling mechanism than its competitors. However, these results are inconclusive due to the method used to collect this data.

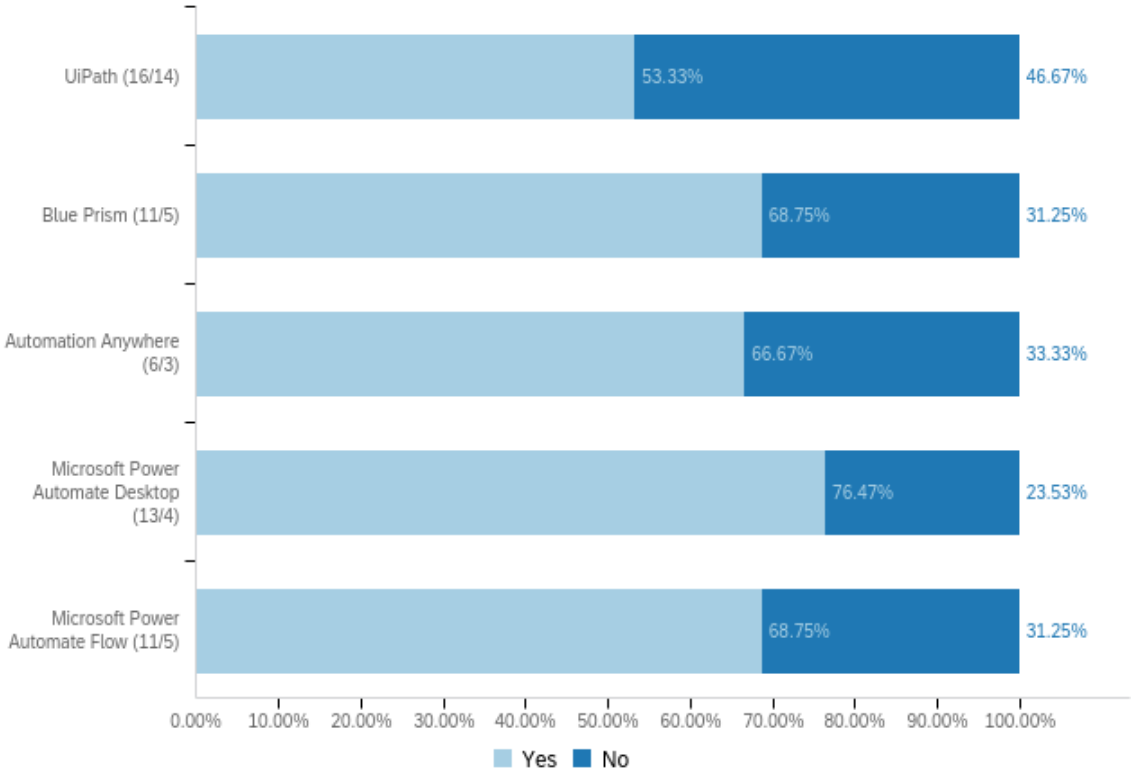

Figure 6 - Percentage of respondents per tool (multiple-choice) reporting an increase in maintenance due to orchestration or scheduling.

When provided with possible effects of orchestration issues many respondents found recognition in the requirement to rerun processes manually, accruing delays over a day, and not being able to fully utilise their license. In self-submitted responses users shared the following:

- Work had to be performed manually instead of a 'robot' doing it in an automated way.
- Loss of a competitive edge over competitors due to an increase in response times.
- Creation of more work due to requests or follow ups as a result of the script not completing in time.
- Client escalations and missed deadlines or service level agreements.
- A loss of credibility for the automation programme/RPA department throughout the rest of the organisation.
- A knock-on effect due to many interdependent, sequential automations or limited business hours and application uptimes.
- An impossibility of scheduling automations effectively during peak times or letting specific processes take priority.

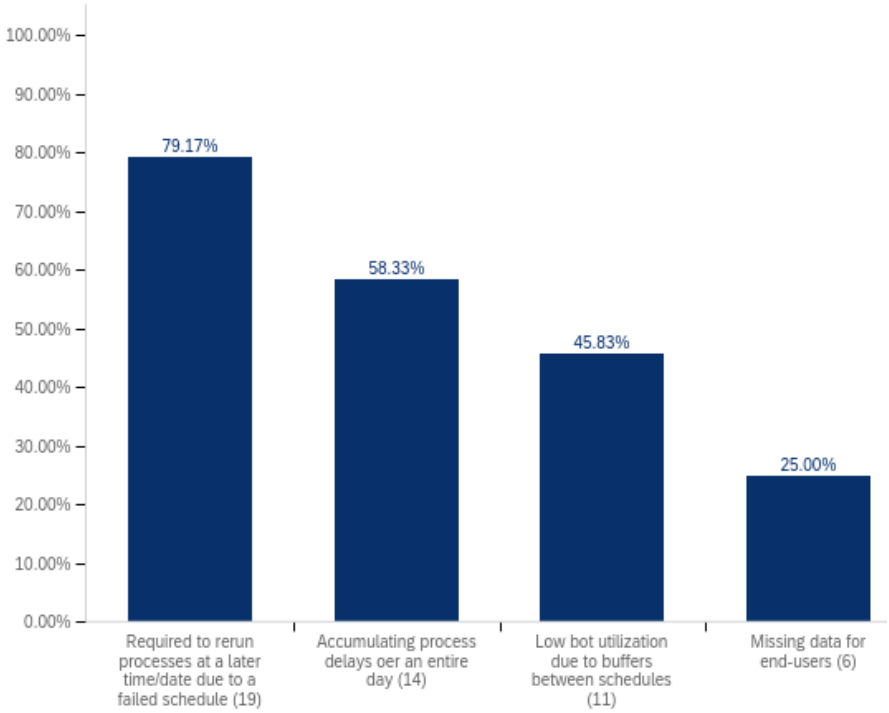

Figure 7 - Percentage of respondents encountering pre-selected issues as a result of insufficient orchestration capabilities in their environment.

When gauging how severely the orchestration issues affected respondents' day-to-day work and required maintenance in an RPA environment, they, on average, said that it had a moderate effect. The distribution of these responses can be found in Figure 8.

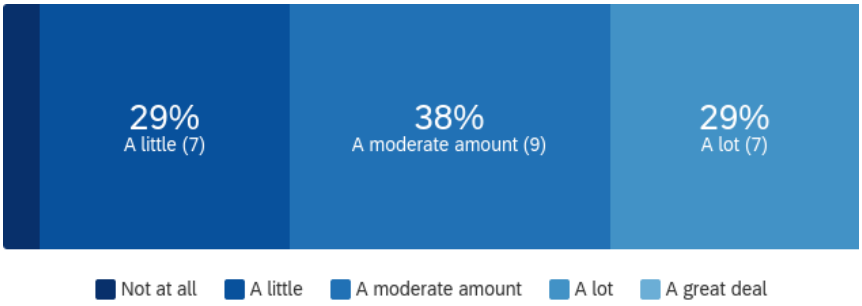

Figure 8 - Distribution of respondents indicating the severity of orchestration issues affecting their day-to-day and maintenance activities.

Multiplatform usage has mixed popularity and is not being embraced by many of the respondents, as per Figure 9. Respondents indicated that this mostly resulted in a confusing mix with multiple control rooms, which was hard to maintain. The respondents could see the benefits of it due to citizen development and centralised development using different tools or different capabilities per tool.

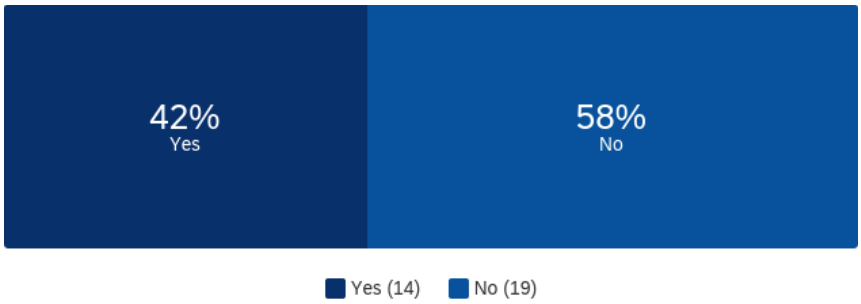

Figure 9 - Distribution of respondents indicating whether they have used multiple RPA platforms alongside each other.

Overall, the survey results show a strong recognition of the problem regardless of the tool in question. The impact of this problem is not insignificant, impacting day-to-day operations for a large percentage of the RPA deployments. A need for multi-platform integration can be observed by the number of users who have been in the scenario of using multiple tools in a single deployment.

4.2. Framework

The purpose of the framework is to provide a structure to plot features which can exist within an RPA orchestrator on. This should aid the structuring of an application to perform orchestration, provide a clear boundary as to what an orchestrator should contain, and serve as steppingstone to theorizing new possible orchestration features. The framework was created after the surveys were completed based on the responses and learnings gained during this process.

Keeping the aforementioned goals of the framework in mind, the process of devising the framework proceeded as follows. Going through the list of provided features, it can be seen that an orchestrator obviously needs to interact with the RPA environment. To provide a frame of reference for what can be interacted with, the layers of an RPA environment were critically assessed to see what it consists of. Starting from the point of view of the RPA tool, we can identify different elements in the environment. Firstly, at the lowest level, we have the infrastructure; the RPA tool must run somewhere on a machine containing a suitable operating system. Secondly, we can look at the purpose of the RPA tool, which is to run automated processes, becoming the middle level. Finally, we look at the real purpose of running these automated processes, which is manipulating data in some form; data is the top layer. This provides one of the axes for our framework.

The following axis can be found by asking what kind of interactions need to be performed as an RPA orchestrator. To arrive at these categories, the suggested features and common usage of other systems can be referenced. For the usage of other systems, the three-tier architecture was chosen as an example of what components are commonly found in an application. Based on the suggested features, this can be shaped into tiers that fit within an RPA environment. The three-tier architecture consists of presentation, logic, and data. These topics are frequently mentioned, as will be shown in ‘4.4 Feature Extraction’, in the desired features. The presentation tier can visualise elements on each layer to provide insights into the orchestrator. The logic tier can control the execution of processes and is the core of the orchestrator; this will be referenced as ‘Control’. The data tier is responsible for storing all related information and verifying the accuracy. As the actual data storage responsibilities are with the RPA tool, in an orchestrator this tier mostly focuses on governance, regarding exporting or logging data, managing inputs, and managing access.

By combining these axes, the framework pictured in Table 5 is constructed. The framework provides an intersection between the architecture of an RPA solution and the possible interactions with that environment, represented as the RPA layers and tiers, respectively. Due to this combination of what exists and what can be done with it, it should provide a matrix encompassing everything that can interact with each other and, therefore, contain all features.

RPA layers	Tiers: Presentation	Tiers: Control	Tiers: Governance	Description
Data	View or get alerted of what data is being processed in detail.	Manage what data is being processed and in what order	Keep records of what data was used and who accessed it	Data used by the process
Process	View or get alerted of the status of running, scheduled, or completed processes.	Manage when, where and what processes are being run.	Keep records of what processes ran and how they performed.	Processes developed in the RPA tool
Infrastructure	View or get alerted of the current status of infrastructure elements, such as server availability.	Manage infrastructure centrally, such as restarting servers or changing settings for an RPA platform.	Keep access records and manage RPA-related permissions per server.	RPA tools and servers
Description	From 3-tier architecture: Presentation	From 3-tier architecture: Logic	From 3-tier architecture: Data	

Table 5 - RPA Orchestrator feature framework

Each element on the axes referenced in the table above is further defined in Table 6.

Framework element	Definition
Infrastructure	The software and hardware required to execute automated processes in an RPA tool. The RPA tool itself is part of the infrastructure. A representation of what hardware can be used in an RPA environment can be found in Figure 3 and Figure 4.
Process	The developed script in the RPA tool which performs automated actions within certain applications on top of the infrastructure.
Data	The information regarding what transactions are to be completed by the process and the payload of relevant data it carries.
Presentation	The visualisation of information to provide insights to end-users.
Control	The manipulation of systems and data to achieve a desired way and order of working.
Governance	The storing, verifying, and reporting of information to service security and compliance.

Table 6 - Definitions of suggested RPA orchestrator feature framework axis categories.

As part of the interview, this framework was reviewed by selected experts in the RPA field. Being sent the framework ahead of time and entering into a discussion about whether they agree it could encompass all RPA orchestration features. The feedback will be discussed in '4.3.3. Interview results'.

4.3. Interviews

The interviews were used to stimulate in-depth discussions regarding RPA orchestration and reliability. It was therefore conducted in a semi-structured manner, allowing for the conversation to take a natural course to uncover additional insights. While the topics are similar to the survey, more detail and nuance can be provided by the selected experts. As a main difference, it also provided an opportunity for the validation of the framework created based on the survey results.

4.3.1. Interview design

#	Question	Related RQ
Background		
I1.1	Do you work with scheduling and maintaining processes in a production RPA environment?	-
I1.2	Do you use multi-purpose bots in your environment? (Bots performing multiple different processes over a day)	
I1.3	How would you judge the license utilisation in your environment? Are they being used efficiently and always in use?	1
RPA Reliability		
I2.1	How do you view the reliability of current popular RPA vendors in terms of scheduling and orchestration? Do they correctly resume in case of process instability?	1
I2.2	Is this one of the bigger problems when looking at overall RPA reliability in your environment?	1
RPA Orchestration		
I3.1	What are you missing when looking at current RPA orchestrators? Do they offer all the desired customizability and features?	2
I3.2	What are the strong and weak points of current RPA orchestrators?	2
Framework		
I4.1	What are your thoughts on the developed framework? Can it encompass all features related to RPA orchestration?	3

Table 7 - Semi-structured interview questions combined with related research questions

4.3.2. Interview participants

The interview questions were posed to 7 interviewees, of which 5 were sourced through the survey, and 2 had not completed the survey. The language spoken during the interview was split between English and Dutch, with 4 being conducted in the former and 3 in the latter. All participants held jobs within the RPA field, all possessing multiple years of experience. Among the interviewees were notable participants, namely the creator of a popular RPA tool and a developer for a commercial external RPA orchestrator. Apart from that, there were also participants with many years of experience as RPA consultants and in-house developers. These participants provide a diverse viewpoint on the implementations and uses of RPA throughout various industries.

4.3.3. Interview results

The interviews have been coded in combination with the survey results to extract information regarding:

- current problems with orchestration,
- the consequences of limited orchestration,
- desired features,
- expected improvements and its impact,
- multivendor usage,
- and framework feedback.

The codes used and the frequency of their occurrence within the interviews are shown in appendix B.

When examining the problems interviewees were facing in the orchestration of processes, they often mentioned the lack of reliability and flexibility when scheduling processes. Schedules may not always trigger properly without further alert to the user. Alongside this the possibilities to trigger a schedule can be limited and mostly consist of time-based triggers instead of event-based triggers. Interviewees indicated this negatively impacted their license utilisation due to the requirement for buffers between processes or other limitations. The schedules would also require manual intervention or constant monitoring, increasing the required headcount and limiting scalability. Orchestration solutions do exist from RPA vendors or external parties; however, the interviewees mentioned that these were excessively expensive or had unclear pricing, which could easily inflate with high usage. Most interviewees were well

aware of the issues, and those primarily working with UiPath reported being less affected by this due to the features offered by UiPath's base tool and their add-on orchestration suite.

The primary consequences of such issues were reported to be an increase in maintenance required to make sure all processes ran correctly. Some organisations addressed this by having large teams dedicated to triggering and monitoring the processes non-stop, with reports of companies having 20 people offshore only monitoring the orchestration. Increasing headcount so significantly isn't possible for every organisation, which then came with other consequences of missed deadlines, the process needing to be completed manually, reputation damage or loss of credibility. Poor license utilisation can also be a consequence due to dedicated licenses remaining on standby for long periods of time, periodically checking whether there was data to process. Organisations also chose to only run processes during standard business hours as stability could not be guaranteed overnight. When uncertainty exists regarding the length of a process due to a fluctuating amount of data, large buffers may be created between processes so they won't interfere with each other's schedules due to a lack of processes queueing after each other.

Features to solve these issues will be explained in detail in '4.4 Feature Extraction', common themes existed surrounding improving triggering of processes, improving the loading of data to a queue, and improving the presentation of data and metrics which can be retrieved from the environment.

The expected improvements to come with these features include increased reliability and consistency of execution, reducing the need for constant monitoring. With this consistency comes an increased trust from the business in the capabilities of the automation division and enables scaling up development. Apart from gaining trust in the business, responsibilities can also be shared with them by giving them live insights into and control over the execution. This control over execution can be extended due to the enhanced flexibility of the orchestration, for instance, by queueing processes or letting priority processes interrupt others, which optimises the license utilisation. Improvements can also be found in clarifying what actions are required from stakeholders by providing additional information regarding statuses and ongoing issues. Moreover, helping spot future issues due to better schedule planning and dependency monitoring.

The benefits of multivendor usage were recognized by survey respondents and interviewees. Different tools have different strengths, they can be priced differently, have different functionality, or be more user friendly for citizen developers in a democratized development model. The problems are clear with orchestration and oversight becoming unclear and there being limited collaboration between the tools.

All interviewees responded positively to the framework found in Table 5, mostly saying it seems complete and could contain all features. However, there were two separate suggestions for including an integration layer representing third-party integrations with external applications. This would be positioned within the RPA layers axis.

4.4. Feature Extraction

During the interviews and surveys one of the main priorities was gathering an array of features to design the framework around and use within the framework. Table 8 will show the extracted features from both the interviews and surveys. The table provides a short feature name, a short description, the framework section it belongs to, and the frequency it was mentioned. The frequency indicates how many unique respondents or interviewees suggested this feature, this is different from the frequency the code was used, which can be found in the coding table in appendix B.

During the survey prior to contributing their own features, the respondents got a list of predefined features. This was included to give them a reference for what potential orchestration features could look like. The predefined features had to be categorized in a MoSCoW classification, allowing respondents to indicate what was important for them. These features have been included in the feature list and will be marked with a '(PD)' meaning predefined in the feature name. Respondents or interviewees may have still suggested this feature separately from this question.

#	Feature	Description	Quadrant	Freq.
1	Access Control (PD)	Provide access to orchestrator functionality for stakeholders and team members in a granular way.	IG - Infrastructure \ Governance	4
2	External Viewing Access (PD)	Provide access to stakeholders not involved on development to view process progress, data, errors, statistics and process schedules.	IP - Infrastructure \ Presentation	3
3	Change process configuration variables	Access config files with hardcoded values and switch between configurations, for example to choose development, testing and production settings.	PC - Process \ Control	4
4	Credential management	Keep credentials in a secure environment accessible by specific processes, machines, and permissioned users.	DG - Data \ Governance	4
5	Encryption of process data	Encrypt queue data for sensitive processes so it cannot be viewed in logs or from outside the process.	DG - Data \ Governance	1
6	Group machines by server capabilities	Create pools of machines to run processes on based on the installed software (versions) or user preference.	IC - Infrastructure \ Control	5
7	Stop/Defer schedules based on dependency unavailability (PD)	Process schedules can be skipped based on the availability of dependencies on a server. The availability can be ticked on/off manually, it could be based on a work window or by a connection to the dependency indicating its availability.	PC - Process \ Control	5
8	Monitor dependency availability	Poll related systems to see whether it is live and accessible for the process.	IC - Infrastructure \ Control	1
9	Log environment changes	Log changes made to the environment by users and the system	IG - Infrastructure \ Governance	4
10	Process dashboarding (PD)	A dashboard showing process execution, process statistics, and live run conditions.	PP - Process \ Presentation	11
11	Environment dashboarding	A dashboard showing configured process schedules, machine and process availability, and license utilisation.	IP - Infrastructure \ Presentation	12
12	Data dashboarding	A dashboard showing what (non-sensitive) data has been processed by the processes. For example, grouped by business unit or department.	DP - Data \ Presentation	8
13	Error logs	Keeping a log of all errors and exceptions encountered by processes, opening it up for further analysis.	PG - Process \ Governance	5
14	Log exports	Exporting logs (automatically) to analyse them in external tools or for compliance reasons.	PG - Process \ Governance	2
15	Machine recording	Recording or screenshotting the system at the time of error within processes.	PP - Process \ Presentation	3
16	Data exports	Exporting queue data (automatically) to keep logs before it is overwritten by the next batch.	DG - Data \ Governance	2
17	Flexible server usage	Ability to play processes on any available machine instead of assigning processes to a specific machine in a schedule.	IC - Infrastructure \ Control	1
18	Environment notifications (PD)	Ability to send out notifications to stakeholders in case of certain events such as failed processes, machine availability, etc.	IP - Infrastructure \ Presentation	2
19	External Queue Item Creation (PD)	Create queue items from external sources through an API call.	DC - Data \ Control	3
20	Start process on queue addition	Start or queue a process once items are added to the queue.	PC - Process \ Control	3
21	External process start (PD)	Ability to receive external request to start or queue a process.	PC - Process \ Control	10
22	Conditional process start	A checklist before starting a process, such as other processes needing to have finished beforehand.	PC - Process \ Control	3
23	Priority process triggering	Safely stop running processes and start a higher priority process.	PC - Process \ Control	5

24	Dynamic load triggering	Trigger multiple instances of a process on different machines to absorb loads above a certain threshold or meet service-level agreements.	PC - Process \ Control	6
25	Queue processes (PD)	If process time and a trigger overlap, the process should be queued and started after the previous process is finished.	PC - Process \ Control	4
26	Software Version Control	Ability to verify and change software versions on machines.	IC - Infrastructure \ Control	7
27	Restart Machines (PD)	Ability to automatically restart machines in case of certain errors as an attempt to resolve machine-related issues.	IC - Infrastructure \ Control	3
28	Data Input Validation	Ability to validate provided queue data according to schema before loading.	PG - Process \ Governance	2
29	Process Input Validation (PD)	Ability to choose from predefined process launch parameters and validate the correctness of entered parameters	PG - Process \ Governance	1
30	Log process steps	Provide a detailed overview of what specific steps a process took.	PG - Process \ Governance	6
31	Runtime statistics	Provide statistics regarding processing times, their standard deviations, and other metrics to provide more insights into processes.	PP - Process \ Presentation	6
32	Process log visualisation	Use process mining to visualize the process logs and have the ability to view what route the process commonly takes, how long it takes and where errors may be more frequent.	PP - Process \ Presentation	3
33	Connected queues	Multiple queues allowing you to push forward between them where validation can be performed that all items are transferred between them.	DG - Data \ Governance	3
34	Cross-platform queues	The ability to have connected queues between different platforms	DC - Data \ Control	1
35	Queue payload adjustment	The ability to adjust queue payloads outside of processes to aid with testing and provide humans with the ability to fix data-related issues.	DC - Data \ Control	1
36	Forecast runtimes	The ability to forecast process runtimes before the process starts and live, allowing for better planning and scheduling.	PP - Process \ Presentation	6
37	Quarantine servers	In case processes fail on a certain server or maintenance needs to be performed, quarantine the server and run any scheduled processes on a different compatible server.	IG - Infrastructure \ Governance	2
38	Process queueing verification (PD)	An extension on the external process start, where users are prevented from trying to start the process too many times and filling up the queue.	PG - Process \ Governance	0
39	Multi-platform overview (PD)	A combined overview of the operations of multiple different RPA platforms.	IP - Infrastructure \ Presentation	3

Table 8 - All identified features including the frequency it was mentioned

Table 9 provides a combination of Table 5 and Table 8, mapping out each of the features onto the framework. Additionally, Table 10 shows the frequency of how many times the features in each category were mentioned. The left-hand value in each cell relates to the feature number, while the right-hand value is the frequency it was mentioned. The shades of blue indicate a more frequently mentioned feature, with darker shades being mentioned more often. Besides this the features have been fit into the areas of the framework, with differing cell sizes, these roughly correlate to the frequency but aren't consistent.

This allows for the analysis of trends and hotspots within the features gathered during this research. Hotspots regarding the number of features or frequency they were mentioned can indicate the current lack of orchestration in concentrated areas.

RPA layers	Tiers: Presentation	Tiers: Control	Tiers: Governance
Data	12 - Data dashboarding	19 - External Queue Item Creation (PD) 34 - Cross-platform queues 35 - Queue payload adjustment	4 - Credential management 5 - Encryption of process data 16 - Data exports 33 - Connected queues
Process	10 - Process dashboarding (PD) 15 - Machine recording 31 - Runtime statistics 32 - Process log visualisation 36 - Forecast runtimes	3 - Change process configuration variables 7 - Stop/Defer schedules based on dependency unavailability (PD) 20 - Start process on queue addition 21 - External process start (PD) 22 - Conditional process start 23 - Priority process triggering 24 - Dynamic load triggering 25 - Queue processes (PD)	13 - Error logs 14 - Log exports 28 - Data Input Validation 29 - Process Input Validation (PD) 30 - Log process steps 38 - Process queueing verification (PD)
Infrastructure	2 - External Viewing Access (PD) 11 - Environment dashboarding 18 - Environment notifications (PD) 39 - Multi-platform overview (PD)	6 - Group machines by server capabilities 8 - Monitor dependency availability 17 - Flexible server usage 26 - Software Version Control 27 - Restart Machines (PD)	1 - Access Control (PD) 9 - Log environment changes 37 - Quarantine servers

Table 9 - All features mapped in the the respective framework areas

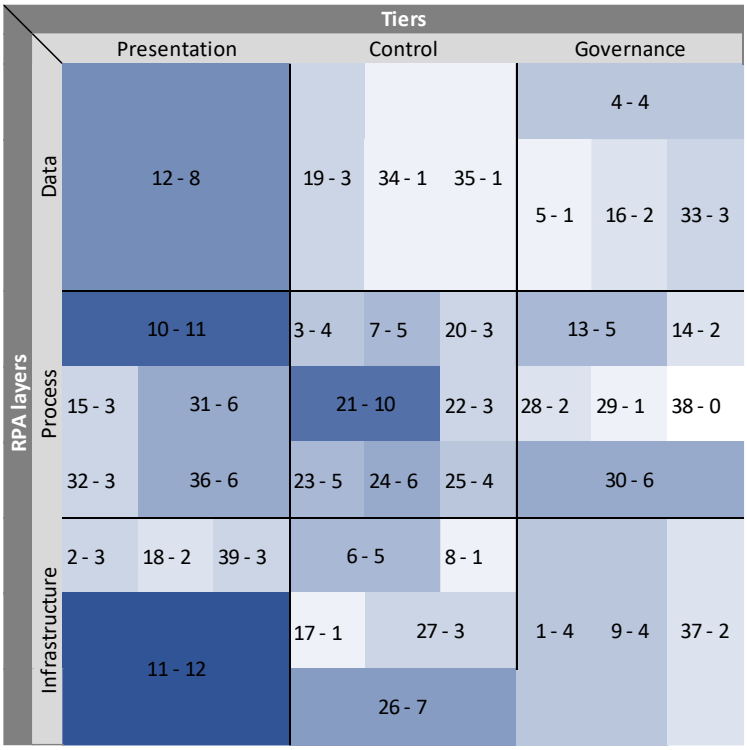

Table 10 - All features mapped and coloured by frequency compared to the whole (displayed as feature number - frequency)

5. Prototype

The prototype section is a continuation of the results section showing the steps performed to validate the framework through a prototype. This is the second step of validation after being reviewed by experts during the interviews.

5.1. Feature selection

As validation for the developed framework, a selection of the features will be developed and modelled. To limit the scope and ensure the feasibility of providing a proof of concept in a timely manner only 4 features will be developed and modelled, while another 2 features will only be modelled.

The selection of features is made to provide a broad view of aspects within the framework. All tiers and layers will be included in at least one feature. To further narrow down the selection, the popularity of each feature will be taken into account, preferring commonly mentioned features. The proof of concept will be deployed within PPHE's RPA department, therefore, based on an informal discussion with PPHE, a higher priority will be given to features valuable in their environment to demonstrate the effectiveness of improved orchestration.

#	Feature	Section	Action	Justification
18	Environment notifications	IP	Develop	Covers presentation section, fits with PPHE needs, high rating in MoSCoW
19	External queue item creation	DC	Develop	High rating in MoSCoW, fits with PPHE needs, feasible scope, covers less populated Data section
21	External Process Start	PC	Develop	Highly requested, fits with PPHE needs, covers most requested PC region
25	Queue Processes	PC	Develop	Fits with PPHE needs, very high rating in MoSCoW, covers most requested PC region
14	Export logs	PG	Model	Covers governance, feasible scope
27	Restart machines	IC	Model	Covers infrastructure control, showcase feasibility of control outside of RPA application

Table 11 - Selected features for UML modelling and prototype development

The selected features can also be seen as part of a theoretical RPA operations story with the following plot points:

1. A process is started externally.
2. This process is queued as the server is still in use.
3. Once the process starts running, there are more queue items which need to be added; these are created externally.
4. An error that cannot be resolved occurs during the process, resulting in the process getting stuck and providing a 'Warning' state to the Blue Prism control panel.
5. A notification is sent out alerting stakeholders of the current warning state.
6. To resolve this issue the machine is being restarted for Blue Prism to regain control over the virtual machine and potentially resolve the cause of the error.
7. The process logs are exported automatically so these can be analysed.

5.2. Feature modelling

To model the features, UML will be used in two different diagram forms. Each feature will be modelled in a class diagram and a sequence diagram. The class diagram will show the structure of the components and how they interact with each other. The sequence diagram will show in what order this interaction happens to have the function operate correctly. Finally, a combined class diagram will show how all features are combined into a single system.

Feature 14 – Export logs

The exporting of logs contains four classes: the request being made from the front end, the orchestrator handling the request, the RPA interface providing the log, and the file system storing the log. Figure 10 and Figure 11 show the class and sequence diagram for this action, which will be further explained below.

The request will allow for specific processes and time ranges to be exported in the desired file format using its input variables. Zero to many requests can be made to the orchestrator, which will always remain a single system. The request will be the first element triggering the exporting action.

The orchestrator is at the centre of the action, facilitating the request. It has environment variables stating what formats are available for export and where the log will be saved. It connects to the RPA interface as a single system being able to interface with one or more RPA interfaces. For the file system, a one-to-one connection is made. The orchestrator is able to receive the requests and start the appropriate follow-up actions. First, it will get the previously ran processes to be able to select which logs need to be exported. Second, the session log for these processes will be retrieved, showing the detailed steps the process went through. The orchestrator now possesses all required information and will format and save the log, with the file being output to the file system.

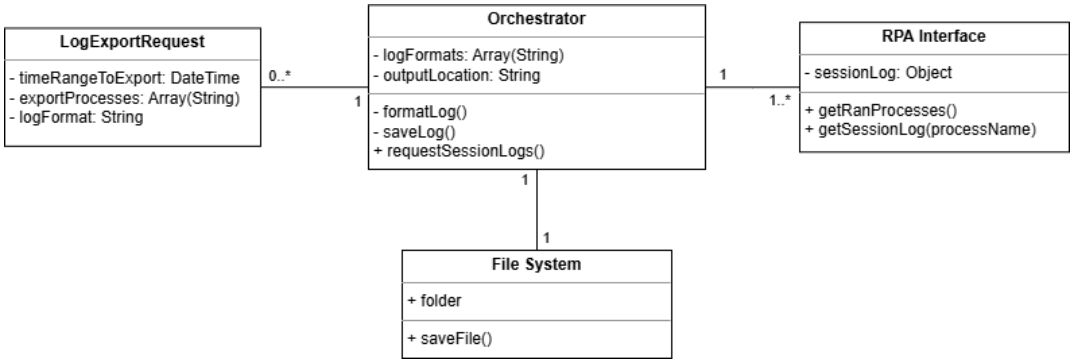

Figure 10 - UML class diagram for export logs feature

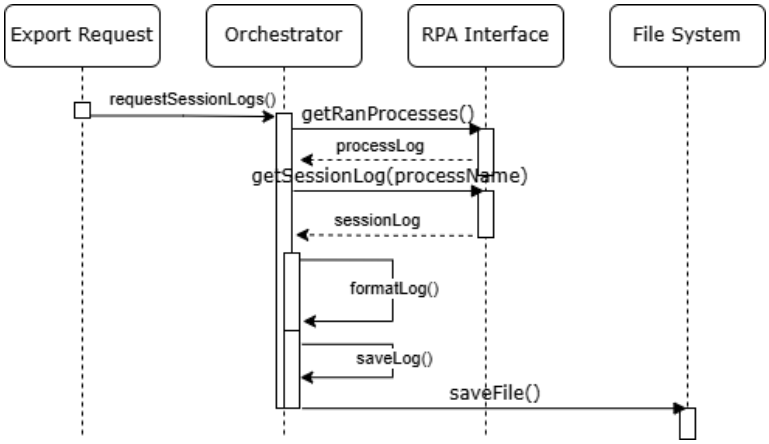

Figure 11 - UML sequence diagram for export logs feature

Feature 18 – Environment notifications

The environment notifications feature contains 3 classes, the orchestrator checking for statuses and triggering the notifications, the RPA interface providing the statuses, and the notification system for delivering the notifications to the end users. Figure 12 and Figure 13 show the class and sequence diagram for this action, which will be further explained below.

The feature is not based on requests, but when it's enabled, the orchestrator will periodically check for process and server statuses. It contains environment variables defining what statuses are accepted for both the processes and servers and after what time threshold of a status being visible a notification should be sent. The orchestrator, as the single system, can interact with one or more RPA interfaces and has a single notification system distributing notifications across multiple channels. For sending process-related notifications, the orchestrator will first get the ran processes from the RPA interface. Afterwards, it will compare the statuses from these processes to the list of accepted

process statuses. In case the state of a process is not in the list, the send notification action in the notification system is triggered. Server status notifications are sent by periodically retrieving the server statuses. The server status for each selected server is compared to the acceptable statuses; when not included in this list, the process waits and keeps checking. If the status doesn't change before the unavailability time threshold expires, a notification will be sent.

The notification system contains environment variables defining the text to send for certain statuses, the channels to publish the notification in, and the recipients of the notifications.

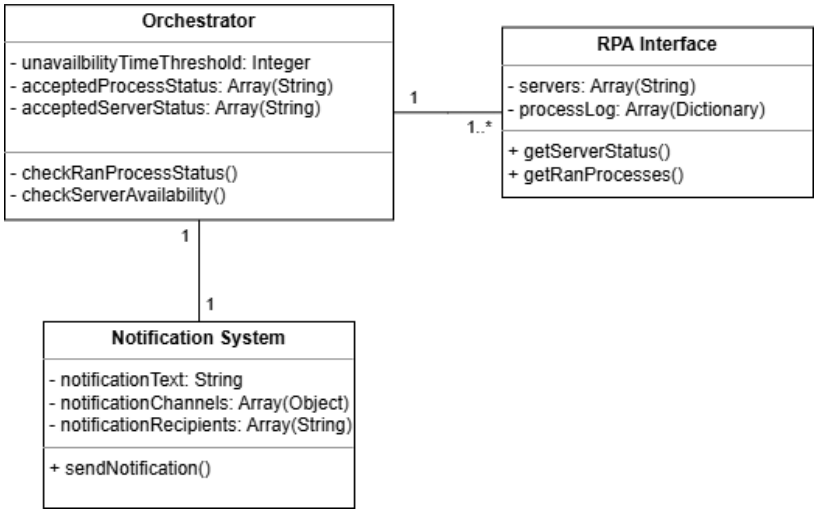

Figure 12 - UML class diagram for the environment notifications feature

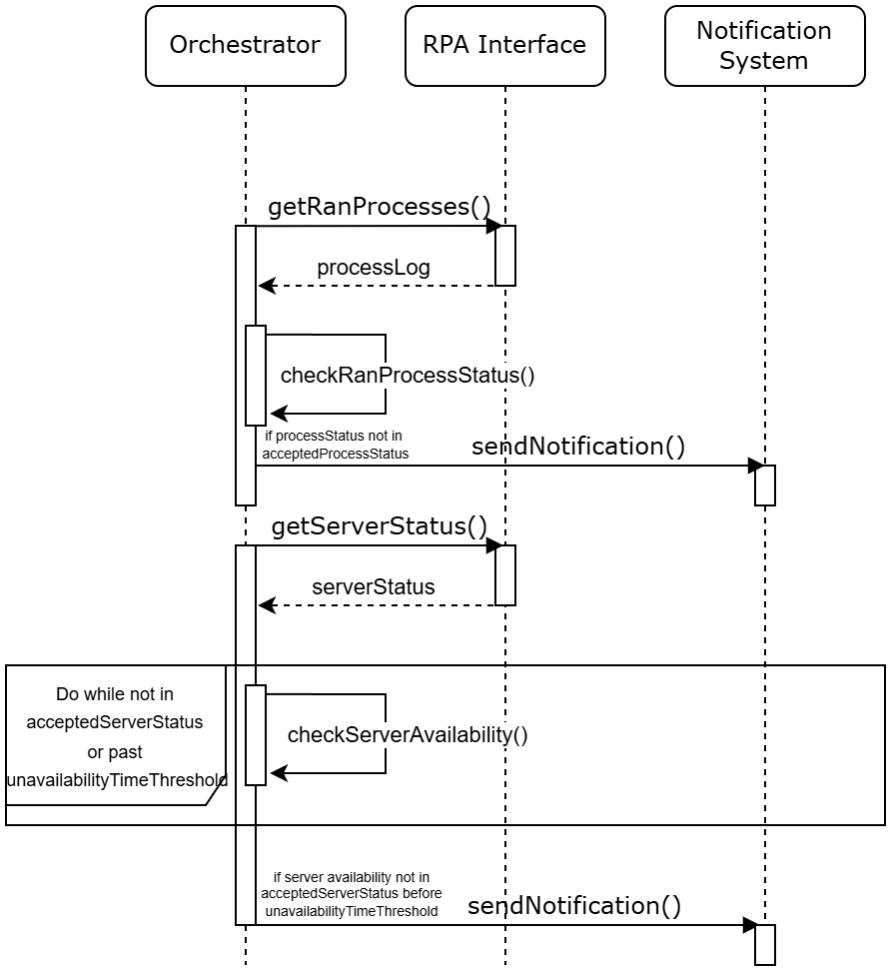

Figure 13 - UML sequence diagram for the environment notifications feature

Feature 19 – External queue item creation

The external queue item creation feature contains four classes: the request triggering the function, the queue item forming the payload to add to the queue, the orchestrator connecting to the RPA interface, and the RPA interface where the data needs to be loaded. Figure 14 and Figure 15 show the class and sequence diagram for this action, which will be further explained below.

The request will indicate to which queue and RPA interface what data should be added. The data payload consists of one or more queue items as part of a single queue request. The queue item contains a numeric value for the priority with which it needs to be processed, as well as the data associated with the item. Zero or more requests are made to the orchestrator.

The orchestrator is capable of receiving the request and triggering the add to queue function in the RPA interface. It has one or more connections to an RPA interface. When the request is received the orchestrator directly passes it onto the RPA interface.

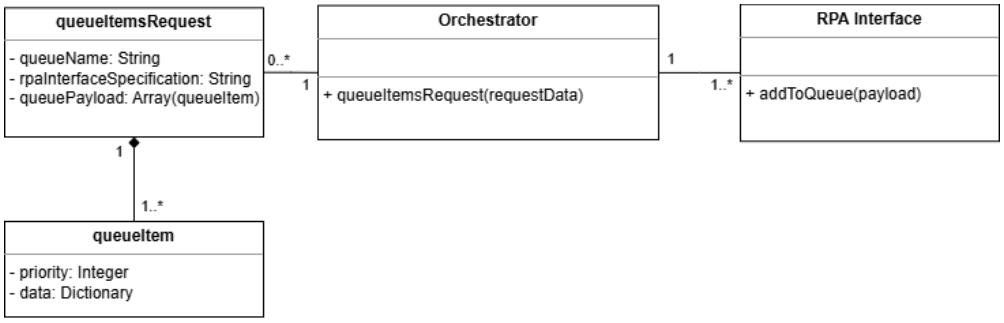

Figure 14 - UML class diagram for the external queue item creation feature

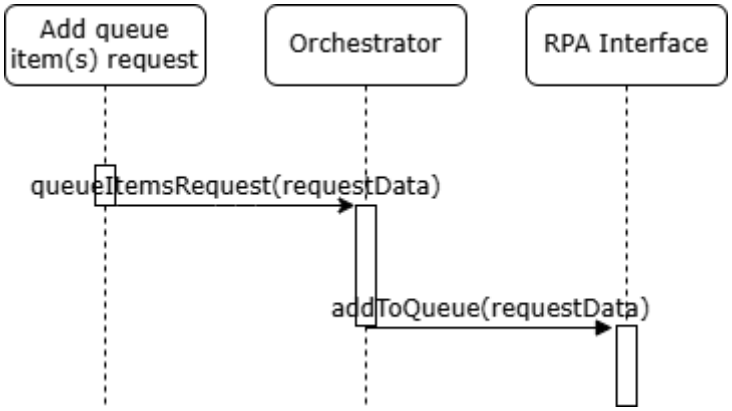

Figure 15 - UML sequence diagram for the external queue item creation feature

Feature 21 and 25 – External process start and queue processes

The external process start and queue process diagrams combine two different features, as the features can much better be modelled together due to their co-dependency. The features contain four classes: the request triggering the function, the orchestrator checking server availability and managing the queue, the queue maintaining the request order, and the interface reporting availability and starting processes. Figure 16 and Figure 17 show the class and sequence diagram for this action, which will be further explained below.

The request will detail the process name, server and interface to be run on, and input parameters for the process. Zero or more requests can be sent to the orchestrator instructing it to start a process. The same cardinality exists between the request and the queue, where the queue consists of zero or multiple process requests.

The orchestrator can receive the request and trigger it at the appropriate time. After receiving the request it will add the process to the queue directly to which it has a one-to-one relation. If items exist in the queue, the RPA interface will be checked to see whether it is ready to receive a process based on its server status. In case the server is busy, it

will wait and check again later based on the environment variable's wait time. When the server is not busy anymore, the orchestrator will retrieve the next process from the queue and instruct the RPA interface to start it. The orchestrator can be connected to one or more RPA interfaces.

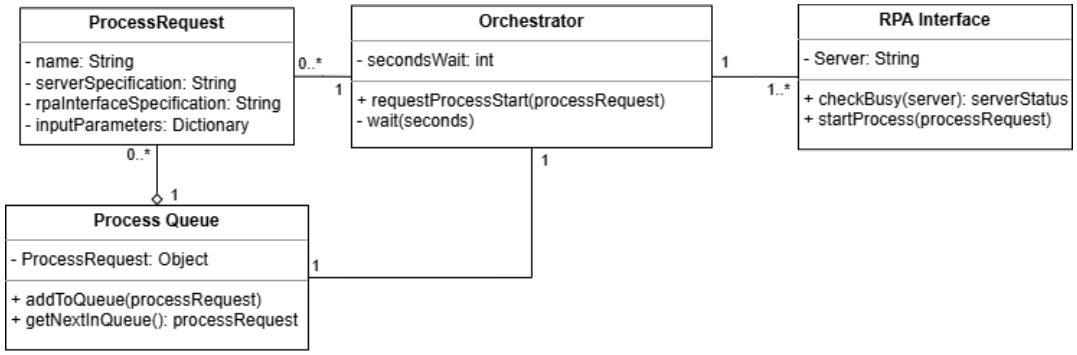

Figure 16 - UML class diagram for the external process start and queue process features

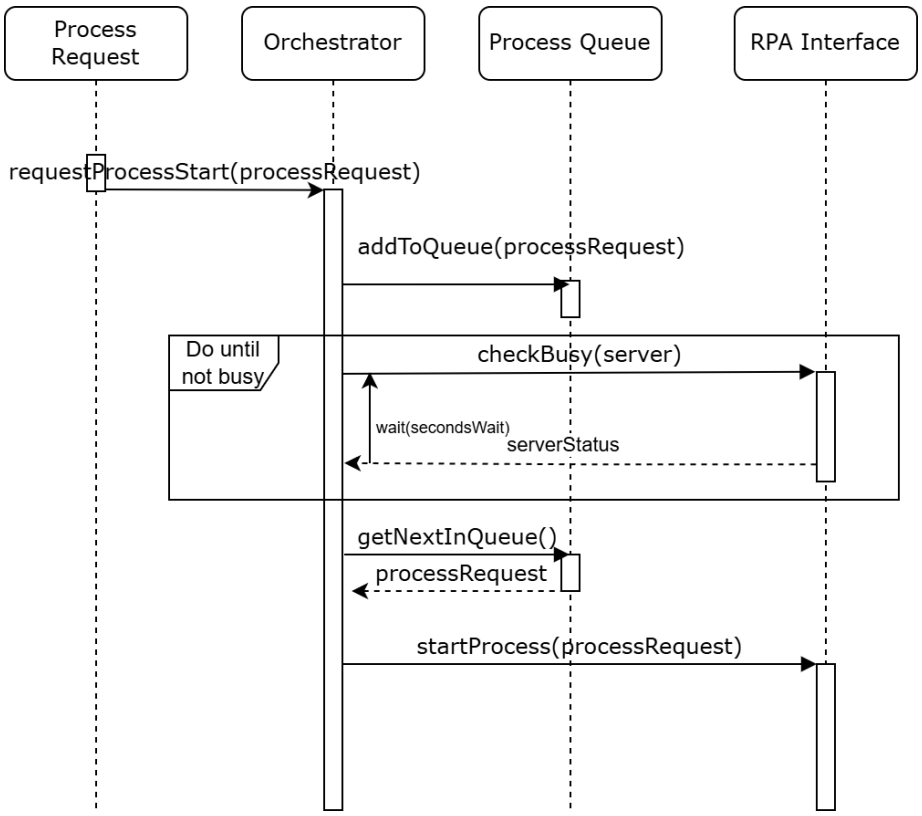

Figure 17 - UML sequence diagram for the external process start and queue process features

Feature 27 – Restart machines

The restart machines feature contains 3 classes, the request triggering the function, the orchestrator executing the command and the server which can be restarted. Figure 18 and Figure 19 show the class and sequence diagram for this action, which will be further explained below.

The request will indicate what server needs to be restarted, sending zero or more requests to the orchestrator.

The orchestrator contains an environment variable detailing how the restart should be performed corresponding to the windows shutdown command. It can receive the restart request after which it sends a restart over the network to a specific server. The server of which there can be one or more, will receive this restart via the network and perform accordingly.

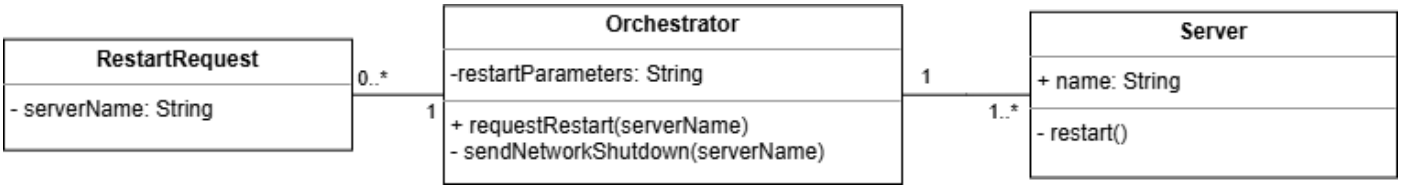

Figure 18 - UML class diagram for the restart machines feature

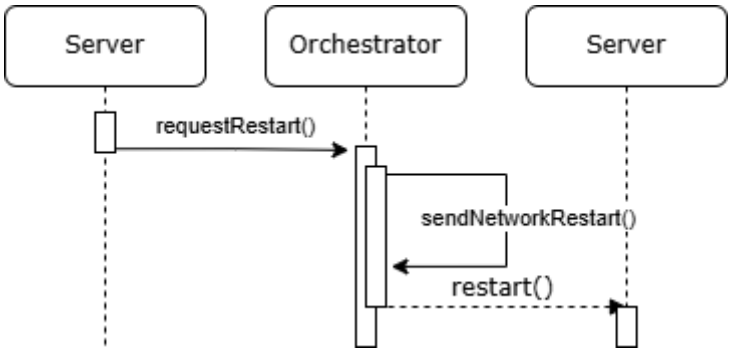

Figure 19 - UML sequence diagram for the restart machines feature

Combined features

The combined class overview in Figure 20 shows the overlap between certain features and their classes. The orchestrator being at the centre processing requests and triggering functions in the connected systems. Particularly the orchestrator and RPA interface show the number of connections and different features included. The diagram shows the initial requests on the left side, the orchestrator in the middle, and the connected systems on the right side.

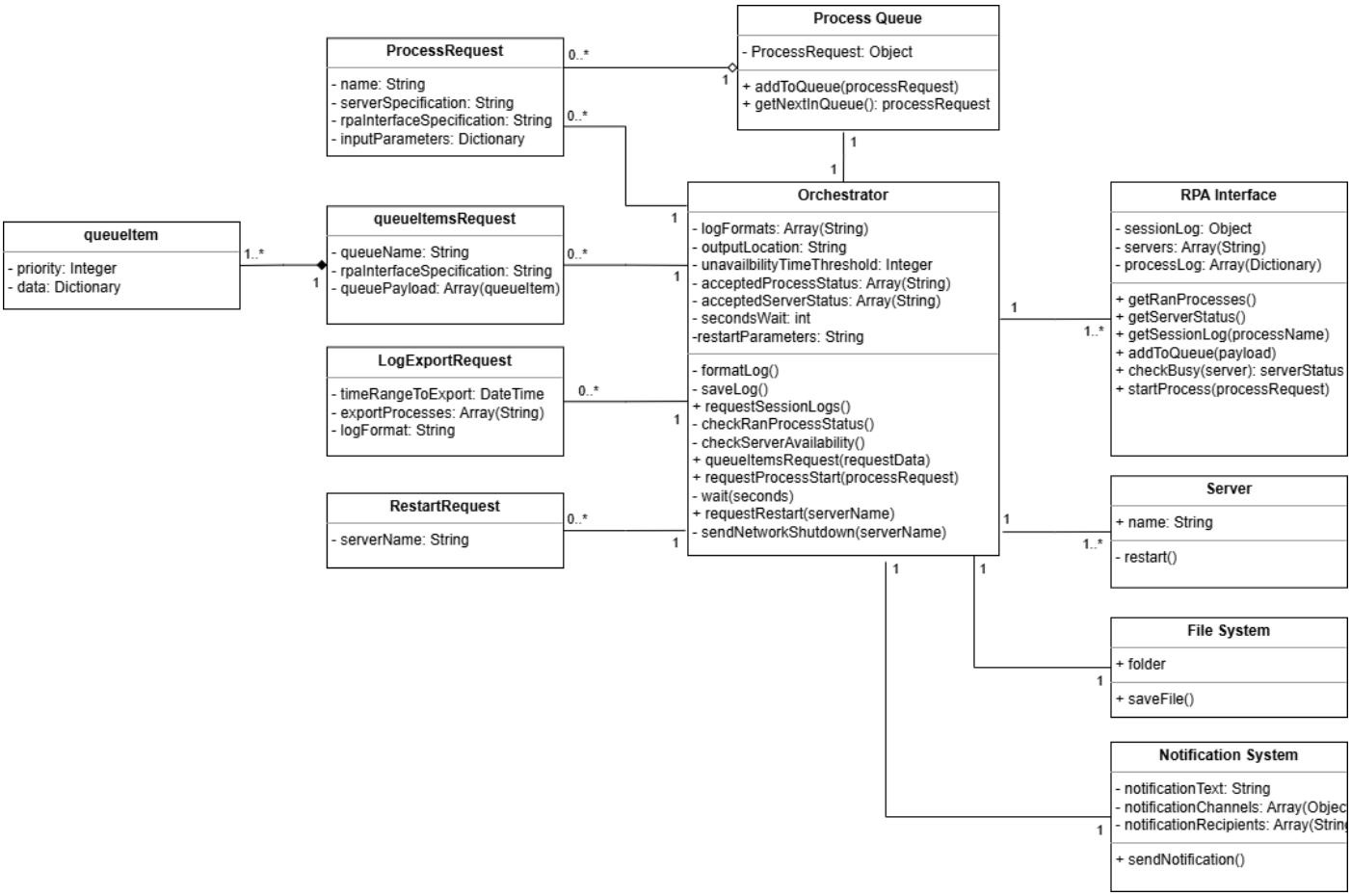

Figure 20 - UML class diagram for all combined modelled features

5.3. Prototype development

Based on the UML above, the environment notifications, external queue item creation, process queueing, and external process start features will be prototyped. The goal of the prototype is showcasing the possibility of implementing these improved features outside of the standard RPA application. Therefore, the final output is the proof of implementation being feasible and not the underlying code. The effort to make this production ready code with proper error handling has not been made as it's not the purpose.

Due to the different ways of implementation and connecting to the RPA tools the UML provides a generalisation of in which way functions are used. While these functions are generally used, more layers exist beneath them to facilitate the connections and commands.

5.3.1. Architecture

The prototype will include all three layers of the three-tier architecture: the presentation, application, and data tiers. The initial decision stage is the selection of platforms and applications to develop these tiers. Based on the specific requirements for each tier, the appropriate tools for a prototype were selected.

Firstly, the presentation tier provides a platform for end-users to interact with the features. The full range of identified features reflects functionality and data that are of interest to multiple groups of people, including RPA specialists and process owners. Therefore, the presentation layer should be widely available, easy to access, and able to identify or authenticate users to handle roles and permissions. As the presentation layer isn't directly required for the features selected for development to function, the development should also be simple and not too time-consuming. Looking at the ecosystem where these features will be trialled, similar to other professional settings, most users have access to the Microsoft suite of applications. Considering these requirements, Microsoft PowerApps was chosen for the front end of the prototype, providing authentication via Microsoft, browser access, and low code development.

Secondly, the application layer executes all of the developed features. Looking back at the combined features UML in Figure 20, the primary interactions exist between the orchestrator and the RPA interface. Interaction between the application layer, the core of the orchestrator, and the RPA interface should therefore be easy. Considering the common architecture of RPA platforms illustrated in Figure 4 in the literature review and the identified features, a connection to the individual servers and database is required. This connection is made most simply by setting up the orchestrator inside the same network. Therefore, the application will run on a server in the same network as the RPA application, which consists of a library of Python scripts.

Thirdly, the data layer stores and exchanges information between the application and presentation layers. As the presentation and application layers have been set up in different environments, a connection between the two must exist. As we're working within Microsoft's environment, they provide a data gateway allowing access to the file system and databases inside the network where it's set up. This allows the application layer to store data in CSV and JSON files to simplify the setup and portability of the orchestrator. As it can also connect to databases, this option would have also been valid but require more setup; it does allow for the connection to the Blue Prism database directly from the presentation layer.

Finally, the RPA interface on which the orchestrator will be tested should be identified. This is SS&C's Blue Prism version 7.1.2 without the API, hereafter referred to as Blue Prism, due to the experience with and access to this platform. Besides this, it's one of the market leaders per Figure 2 and among the first modern RPA tools released.

The presentation layer and data layer primarily exist to facilitate the application layer, which performs the actual orchestration functions. Additionally, it contains the most development via Python, justifying the most detailed explanation of the three. As mentioned before, the primary interaction for the orchestrator is with the RPA interface, the platform which we're planning to orchestrate. Multiple methods of interacting between the two exist depending on the platform. For Blue Prism, this consists of an optional API that can be installed, a command line utility, and direct modifications to its well-structured database. The API has a wide and growing range of functions which can be called but requires a complex server setup, including authentication servers and message brokers. (Blue Prism, 2024) (Blue Prism, 2024) Blue Prism also offers a command line which can be used on servers where Blue Prism is installed, offering

similar but more limited capabilities compared to the API. (Blue Prism, 2024) The database allows any change to be made but requires extreme care so as not to impact the regular system operation.

5.3.2. Orchestrator code

The orchestrator's core exists of multiple Python scripts interacting with each other to achieve the desired effect. The structure and function of the code per feature will be explained below. For a more detailed view, the source code can be found on GitHub: https://github.com/djoprel/RPA_Orchestrator

5.3.2.1. Components

The orchestrator code consists of multiple components separated by function, which will be explained below. Each script has been made specifically for the interaction with Blue Prism in mind, made clear with the 'bp' file prefix. In case of combining multiple RPA platforms, the calls will be directed to the proper scripts from the interface.

The primary script is the 'bp_mainRunner' file, which should constantly run for the orchestrator to function. This will retrieve the commands which were sent from the presentation layer and trigger the relevant functions in the related scripts. Other runners are opened in a separate session to function concurrently while normal commands are triggered sequentially due to the effect they may have on each other and simplicity during the prototyping phase. When all commands have been processed it will wait for the following commands to arrive.

The functions which are called from the main runner exist within the 'bp_mainFunctions' file. These functions contain the high-level actions which need to be performed. The available functions are 'addToQueue', 'requestProcessStart', 'startNotifications', and 'stopNotifications', relating to feature 19, 21/25, 18, and 18, respectively. This commonly consists of gathering the required information and starting the function or runner interacting with the RPA interface.

In order to collect the information required to run certain functions, connections have been made to the database in the 'bp_queries' file. This script contains a collection of queries which are used throughout the different scripts to get the relevant data from the database. The query is passed to the 'bp_InteractDatabase' file, which handles the actual database interaction by opening the connection, querying the database, and closing the connection.

Besides getting data the queries can also insert data which is required for feature 19, adding external queue items. Via the main function 'addToQueue' the required data is collected from the database, after which the provided queue items are reformatted to match with the database structure using the 'bp_formatQueueData' script. After the data has been transformed, it is then inserted into the database in the same way BluePrism would do this.

Besides the database another method of retrieving or updating data is via the command line interface. In the 'bp_interactCommandLine' script the possibility exists to extract the previously ran processes in the environment or trigger a new process.

Triggering and getting previously ran processes is useful for the external process start and process queueing functions, features 21 and 25. Started from the main function 'requestProcessStart', the 'bp_processQueue' script is activated, which manages the queue, including adding to the queue, marking processes as triggered, and archiving triggered processes. The queue is then used by the 'bp_queueRunner' which is started in a separate instance. While there are processes left to run, this will remain active, checking the server status where the process needs to run and starting the process as soon as the server is available. After a process is started, the next process will be taken from the queue and it waits for the server to become available again.

The server status is also important for feature 18, sending environment notifications, also started as a runner from the main functions. 'bp_notificationRunner' remains active until a stop request is sent, constantly checking the server and process states against a list of acceptable states. In case a process has an invalid state, the script directly sends a notification request to the notification system. For invalid server states, the script will wait for a specific amount of time, checking if it moves into an acceptable state during that time. Otherwise, it sends a notification request to the notification system.

5.3.2.2. Configuration

The scripts come with a number of config files, primarily including environment-dependent variables, which are required to run.

The connection details for the database and command line interface can be found in the 'bp_DB.config' and 'bp_CLI.config', respectively.

The 'bp_servers.config' file contains variables stating the names of the servers in an environment, the state at which servers are ready to receive process triggers, and the states in which a notification should be sent for a server or process.

The 'bp_runners.config' contains the variables for all runner scripts. For the queue runner, this includes whether the runner is active and wait times between status checks or after triggering a process. The notification runner section contains variables mentioning whether the runner is active, is requested to stop, the wait time between checks, the time a server should be in a specific state to send a notification, and the trigger URL for the notification system. Finally, the main runner section has a variable stating the wait time between checking for new commands and the location of the shared data folders to exchange data with the presentation layer.

5.3.3. Data Structure

The shared data files are used as a persistent storage and to exchange data between the presentation and application layers.

The 'bp_orchestratorCommands.csv' passes commands from the presentation layer to the application layer. It consists of the date and time the request was made, which function should be triggered, the parameters to be used, who requested it and what the status is. The status will be amended by the application layer so the end-users can see their command has been completed or is in progress from the presentation layer.

The 'bp_processQueue.csv' contains the processes which are currently queued to be triggered on one of the servers. It contains an id for the request, the name of the process, the time it was requested and the time it was triggered. Periodically it will be moved to the 'bp_processQueueArchive.csv'.

The 'bp_notificationLogRecent.csv' contains recently created notifications, which are periodically archived to 'bp_notificationLogArchive.csv'. It contains the type of notification such as a process or server, the origin stating a process or server name, the time the notification was requested, the status of the process or server, and the session id of the process.

These files and some of the database tables are also synced to Microsoft Dataverse via Power Query, which allows them to be visible in the front end.

5.3.4. PowerApps Logic

To instigate a refresh of the front-end using Power Query a Power Automate flow is created which can be triggered from the front-end. The files are also altered by Power Automate when a command is sent from the front-end to the orchestrator by updating the commands file.

Besides dealing with the files, Power Automate is also set up as the notification system, being able to deliver notifications via Teams, Email, their app or trigger other services. It is triggered using an HTTP request from the orchestrator.

5.3.5. PowerApps Presentation

The front-end exists within PowerApps allowing for low-code creation of a front-end tying together with other Microsoft services. Separate pages exist for each function specifying the required inputs, grouped by their category in the framework. Additionally, an about and past commands page exists for information about the app and sent commands.

The past commands and environment notifications pages solely show information about previous commands or notifications and their current status. These pages don't require any further interaction.

The request process start page shows all processes which can be triggered and a button to trigger them. While the add to queue page shows all available queues and an area to input the queue items as a csv text.

5.3.6. Testing

All except for the add-to-queue functions have been tested through the UI to make sure they work within PPHE's Blue Prism test environment. This full integration test was not possible for the add-to-queue functionality as this required elevated rights in the database, which weren't present by default for the test environment. As the queue items need to be inserted, write access to the tables is required. The code has been tested on a separate system with a local Blue Prism installation to ensure the code functions properly. The presentation layer doesn't add much more complexity, especially when the other functions are proven to be working.

Besides the testing phase the prototype hasn't been put to use yet in PPHE's environment. Although, the solution works well, responding quickly to commands and being accessible from anywhere. Due to it being a prototype the code is not ready for production use and will therefore require further development before being utilized.

5.3.7. Expansion

The prototype has been designed in a modular way, having primary functions execute functions within libraries related to an interface or task. This allows for simple expansion and customizability, additional features can easily be added and triggered. The UI has also been designed for this, where the data presented on the surface has an extra layer which allows the orchestrator to correctly route the action to the correct RPA interface.

Besides the modularity, reusability has also been kept in mind with the function libraries being able to be used from different functions with varying parameters. In the prototype this reusability can primarily be seen for the queries to the database which supply many of the different functions with data.

Modularity and reusability lie at the core of the framework as it tries grouping closely related functions. However, drawing relations between the currently prototyped functions is limited due to the minimal number of features which have been implemented. When looking at different features within the same cell, row or column in the framework in Table 9, examples of other functions can be found that should closely resemble each other.

For example, looking at feature 34, cross-platform queues, the functions and interaction will be in part similar to that of the management of queued processes in the way it controls the flow of data between the interface and persistent storage. Feature 36, forecast runtimes, has similar aspects to the environment notifications in the way it retrieves data and makes this available in the front end. Feature 33, connected queues will update queue data in the same way the external queue item creation does. Feature 37, quarantining servers, will judge what needs to happen to a server based on process and server status, similar to the environment notifications.

When examining the relation between the prototype and the UML models, the functions in the models can all be found in the prototype. The structure of the functions is the same but, in most cases, has additional parts added to aid configurability, constant running, and processing and retrieving related data.

6. Discussion

As illustrated in the literature review, a gap of knowledge exists surrounding orchestration of robotic process automation processes. The effectiveness of built-in solutions, the required features, and potential improvement possibilities have not been extensively researched.

The literature showed us the wide range of RPA tools which are available and commonly used. Additionally, the infrastructure of RPA tools could be seen as a web of many different machines and stakeholders. Proper orchestration mechanisms exist within other aspects of software development but haven't been applied and tailored to RPA.

6.1. Problem identification

The results section aims to address these findings by starting with a survey among RPA professionals investigating their satisfaction with current RPA orchestrators.

The severity and prevalence of issues with the orchestration of processes automated using RPA can be clearly identified from the survey and interview results. A majority of the respondents have seen issues regarding orchestration impact their RPA deployment, and it consistently impacts their day-to-day.

The same experiences were shared across different tools, illustrating an issue across the entire market for RPA tools. The number of respondents recognising such issues ranged from half to three-quarters depending on the tool. Nearly half of the respondents also indicated having used multiple platforms at one time, requiring multiple interfaces to orchestrate all processes.

The literature also showed us a significant difference in core orchestration features when comparing the market leaders among RPA tools. Potentially highlighting a lack of knowledge regarding user desires and absolute requirements for an orchestrator.

The nature of RPA means processes are completely self-developed and can be categorised as software development. With the current state of RPA tools, it's possible to create such custom processes but due to the lacking orchestration, control and management of these processes can be problematic. Based on the results and literature, it has been established that the problem is present.

6.2. Effects of ineffective orchestration

Having illustrated the problem exists, the effects of orchestration issues have been investigated.

The interviews and surveys showed inadequate orchestration can result in serious issues affecting not only the RPA department and its team members but the entire organisation. Possible side effects were mentioned to be that work had to be done manually to make up for the lack of timely processing. Additionally, this untimeliness was mentioned to result in a loss of competitive advantage due to a slower response time than competitors. The delayed processing will eventually lead to a vicious cycle and knock-on effect, leading to a diminished capacity triggering issues for other processes.

The increased maintenance to deal with these issues can increase costs and reduce the ROI of automation projects. Moreover, it can reduce the trust in the capability of RPA to complete processes with accuracy and timeliness.

Dealing with sometimes business-critical processes, one can argue that the effects of ineffective orchestration are severe.

6.3. Orchestration requirements

Once the issues have been identified, solutions to these problems can be theorised. The orchestrator will require an extended feature set to properly mitigate the inadequacy of the base orchestrator.

In the interviews and surveys, a broad desire could be seen for additional features, with many ideas coming from the RPA professionals. Many of the professionals were in agreement regarding what features were important and clearly missing in some if not all tools. This again validates the shared experience of inadequate orchestration which is ripe for improvement.

Even the features which are only mentioned once in the interviews or survey can be seen as an interesting case. Each RPA solution is utilised in a very different manner by the professionals using them.

You could argue that this forms one of the key takeaways in this research; with such a variance in the desired features, no orchestrator can be perfect. Every organisation deals with different process flows and priorities within their operations. The RPA vendors can't tailor the orchestration to everyone's needs, even when improving the base feature set by adding commonly requested features. A strong desire exists for customizability and broadening the interoperability of orchestration.

6.4. Framework

To provide guidelines around the identified features, a framework to plot these features has been developed. Such a framework expands the knowledge of what features are available to the RPA community. Apart from that it also allows for further theorizing within a set boundary to come up with additional features by looking at the intersections on the framework. It provides a new perspective on RPA orchestration and what it can do.

Besides the categorization of features, it can also serve as a guideline for development indicating the required connections and interoperability between components. Many of the features within one of the aspects or cells of the framework will utilize similar methods for performing actions or gathering the information. A modular component library allows for simple expansion and customisation of the orchestrator.

6.5. Feasibility of improved orchestration

To facilitate the aforementioned customizability and interoperability, orchestrators should be partially open and provide for easy integration. An argument could be made that an effort from RPA vendors may be required to open up their platforms and expand their library of API's. However, this can also be approached from the other side by RPA professionals building their own orchestration layer.

The feasibility of building such an improved orchestration layer has been illustrated in the results section by developing and modelling a set of features. These features were commonly requested and useful for the company, which trialled them and provided input regarding the development. To adhere to the framework these features were built in a modular manner, trying to maximise code reuse and modularity while limited to the development window for a prototype.

The prototype has shown the feasibility by correctly implementing certain features which integrate into the chosen RPA tool in multiple ways. By using command line utilities, direct database connections or application programming interfaces, most actions the built-in orchestrator performs can be duplicated or altered.

It has also shown that to develop such an orchestrator, developers don't necessarily need to start from scratch and can use a combination of other tools to their advantage. In this case authorization, communication, and the user interface were facilitated by Microsoft's Power suite of applications. Certain interactions with the RPA tool had also already been made available through the command line utility they provide. This makes the development more accessible for smaller organisations and less experienced developers.

Therefore, it is completely feasible in the current state of RPA to take control over the execution and management of self-developed processes in a centralised manner.

6.6. Effects of improved orchestration

As mentioned before, the interviews and surveys showed the same problems are experienced across different tools. The improved orchestration can also relieve these issues across tools while consolidating the orchestration into a single point of access. Increasing clarity when working on the operational side of RPA.

RPA professionals have indicated they expect such an orchestrator to lead to increased reliability and consistency of execution. This will reduce the manpower required for monitoring and maintaining the operations of processes, while increasing the trust in the solution.

Besides relieving current pains, more functionalities can be added to further include the business in the execution of their processes, relieving even more work from the RPA operations team. This will allow end-users or process owners to directly manage and interact with their own process and gain live insights. The RPA team will be able to allocate more time to development and scale their department at a faster pace.

As shown in the literature, with growing automations there's also a significant growth in maintenance for RPA projects. It could be argued that the improved orchestrator aids in localizing where maintenance is required, provides tools to make rapid changes in the environment to make sure the rest of automations can remain operational, and therefore reduces the overall maintenance effort.

When discussing the feasibility of improved orchestration, the ability for RPA vendors to make an increased effort was mentioned, such as providing extra API endpoints. While one can argue that this is of great help to the RPA professional orchestrating their processes, a script or tool to connect and trigger the API is still required. Therefore, an increased involvement from RPA vendors still does not invalidate the requirement for a customisable orchestrator. Creating separate scripts to interact with these APIs and perform small tasks will create a chaotic web of components trying to interact with the same system or execution timeline from different areas. The centralisation of these scripts, reusing components and facilitating proper interaction creates the core of the orchestrator.

6.7. Limitations and future research

One of the first clear limitations was the availability of prior research and relevant sources for this research. The RPA field isn't researched much being a niche in between business and IT. The available sources do talk about the reliability, best practices and limitations of RPA but don't specifically focus on one of its large operational areas, orchestration. To fill this gap in knowledge, a lot of focus has been put on the interviews and surveys performed with RPA professionals.

While the quality of the responses in the interviews and surveys is high, the quantity and diversity are somewhat low. Seven interviews were performed with RPA professionals from Western Europe and 33 surveys were fully completed globally. As certain RPA tools might be more common in certain geographic areas this could skew the results based on the experiences with a single platform.

To guide respondents, elements in the survey already contained suggested answers and a space to include their own observations. This could be leading them and increasing the relevance or response rate of suggested answers and stop further observations from being made. This felt like a necessary step to increase the quality of their other observations by providing a guideline. The suggested answers have been highlighted in the research so readers can take this into account.

Other external commercial orchestrators which are available have not been considered as solutions to the orchestration problem. These orchestrators may solve many of the issues posed in the research. This research first set out to prove the relevance of such external tools and, afterwards, the feasibility of integration. Therefore, other orchestrators being available doesn't invalidate the results but it can be seen as a future research topic to assess their capabilities.

Besides external orchestrators, the improvements RPA vendors are making to their own platforms have not been considered. In future research the improvements and whether it will reduce the need for enhanced orchestration can be studied.

The problem of orchestration is, in many cases, a problem of limited resources. Therefore, there can also be other solutions to some of these issues. The required resources are licenses and servers; by using an open-source RPA tool and considering a horizontally scalable server architecture, orchestration can be simplified. This may invalidate the need for certain advanced orchestration features by adjusting the architecture. The availability and maturity of such tools, along with the feasibility of this architecture, could be a point of future research.

With the rise of artificial intelligence and large language models creating agentic automation, the next chapter for software automation has arrived. This new development allows AI to interact with applications and data based on prompts and removes the need to develop a script fully. Future research could investigate the relevance of the

developed orchestration framework for agentic automation. Early signs point to this still being relevant as primarily the development is revolutionised, and the orchestration remains similar. Therefore, this framework can also be considered relevant while RPA evolves.

7. Conclusion

The purpose of this research was to find a way to identify whether an issue exists with the current state of RPA orchestration and how this could be improved.

The research question posed to address these questions in a structured manner is divided into four subcategories. Namely, the consequences of an unreliable RPA environment, the requirements for an improved RPA orchestrator, translating the requirements into a high-level framework for RPA orchestration, and how this improves the reliability of an RPA deployment. These questions come together to form the primary research question: 'How can RPA reliability be improved by a vendor-agnostic improved process orchestrator?'

Using interviews and surveys with experts in the RPA field, the sub-questions could be answered. The consequences were identified and quantified, resulting in a broad recognition of the problem and desire for a solution. Many different requirements were introduced depending on the specific use cases for the RPA professionals, while they indicated how this would improve the reliability of RPA deployments. These results were translated into a high-level framework guiding the layout of an RPA orchestrator. The framework was tested by presenting it to RPA experts for feedback in an interview, with unanimously positive responses. This comes together to answer the primary research question: RPA reliability can be improved by a vendor-agnostic improved process orchestrator due to its extended feature set and customizability. The creation of the orchestrator can be achieved by using the framework as a starting point for the requirements analysis and design.

The reasoning for this approach is that due to a lack of available literature on this topic, most of the information had to be acquired from the community of RPA experts and professionals. As the prevalence of this issue hadn't been researched before and only stemmed from an observation within PPHE's RPA deployment, the presence of this problem had to be identified first. Then, by gathering requirements and the related improvements, the definition of an improved orchestrator was formed. As the loose list of requirements wasn't extensive, it needed to be reduced to a generalised framework. To finally show the validity of the framework, it was posed to experts, and a prototype was developed based on the features and framework.

As the researched topic stemmed from a personal observation, the current overall outcome was expected. However, the results gave a deeper understanding and more nuance to the initially observed problem. Primarily showing how an orchestrator will never be perfect for every user and that therefore the customization was one of its most important features. Besides this in the interviews it also showed an alternative approach to the problem, by using open-source tools the resource limitation on licenses and servers could be minimized and the orchestration issue reduced.

The scientific contribution is a framework providing a structured way to start addressing the issue of orchestration by cataloguing possible solutions and showing its relation to the platform. This can be used as a jumping-off point for the design of orchestrators or the investigation of current capabilities and alternatives.

8. References

Albreshne, A., Fuhrer, P., & Pasquier, J. (2009). Web services orchestration and composition. *Hewlett-Packard's Dev. Resour. Organ*, pp. 46-52. Retrieved from https://www.unifr.ch/inf/softeng/en/assets/public/files/research/publications/pdf/WP09-03.pdf

Automation Anywhere. (2021, October 19). *Using Control Room*. Retrieved from Automation Anywhere Documentation version 11.3: https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/enterprise/topics/control-room/getting-started/using-control-room.html

Blue Prism. (2024, June 25). *Blue Prism API 7.1.2*. Retrieved from Blue Prism API Documentation: https://docs.blueprism.com/en-US/bundle/blue_prism_API_7_1_2_OpenAPI/page/index.html

Blue Prism. (2024, September 23). *Blue Prism API Installation*. Retrieved from Blue Prism Enterprise 7.1 Documentation: https://docs.blueprism.com/en-US/bundle/blue-prism-enterprise-7-1/page/Guides/bp-api/api-introduction.htm

Blue Prism. (2024, June 25). *Blue Prism architecture overview*. Retrieved from Blue Prism Enterprise 7.4 Documentation: https://docs.blueprism.com/en-US/bundle/blue-prism-enterprise-7-4/page/Guides/infrastructure-reference/architecture-overview.htm

Blue Prism. (2024, June 25). *Blue Prism Command Line Options*. Retrieved from Blue Prism Enterprise 7.1 Documentation: https://docs.blueprism.com/en-US/bundle/blue-prism-enterprise-7-1/page/helpCommandLine.htm

Blue Prism. (2024, June 25). *Control Room*. Retrieved from Blue Prism Enterprise 7.1 Documentation: https://docs.blueprism.com/en-US/bundle/blue-prism-enterprise-7-1/page/frmControlRoom.htm?tocpath=Interface%7CControl%7C_____0

Davenport, T. H., & Brain, D. (2018, June 13). Before Automating Your Company's Processes, Find Ways to Improve Them. *Harvard Business Review*. Retrieved from Harvard Business Review: https://hbr.org/2018/06/before-automating-your-companys-processes-find-ways-to-improve-them

Deloitte. (2017). *The Robots are ready. Are you? Untapped advantage in your digital workforce.* Deloitte. Retrieved from https://www2.deloitte.com/cn/en/pages/strategy-operations/articles/the-robots-are-ready.html

Drost, N., Spaaks, J. H., Andela, B., Veen, L., Zwaan, J. M., Verhoeven, S., . . . Borgdorff, J. (2020, September 9). *Best practices for software development.* Netherlands eScience Center. Retrieved from Netherlands eScience Center Guide: https://zenodo.org/record/4020622

Fortune Business Insights. (2024, December 16). *RPA Market Size*. Retrieved from Fortune Business Insights: https://www.fortunebusinessinsights.com/robotic-process-automation-rpa-market-102042

Fung, H. P. (2014). Criteria, Use Cases and Effects of Information Technology Process Automation (ITPA). *Advances in Robotics & Automation, 3*. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2588999

Gartner. (2024). *Gartner Magic Quadrant for Robotic Process Automation.* Gartner. Retrieved from https://www.gartner.com/en/documents/4016876

Geiger, M., Harrer, S., Lenhard, J., & Wirtz, G. (2018, March). BPMN 2.0: The state of support and implementation. *Future Generation Computer Systems*, pp. 250-262. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S0167739X17300250

Ivančić, L., Suša Vugec, D., & Bosilj Vukšić, V. (2019, August 26). Robotic Process Automation: Systematic Literature Review. *Business Process Management: Blockchain and Central and Eastern Europe Forum, 361*, pp. 280-295. doi:https://doi.org/10.1007/978-3-030-30429-4_19

Jensen, A. (n.d.). *I Love Automation*. Retrieved 2023, from LinkedIn, Discord: https://www.linkedin.com/groups/12755028/

Johannesson, P., & Perjons, E. (2014). *An Introduction to Design Science.* Springer Cham. doi:https://doi.org/10.1007/978-3-319-10632-8

Juric, M., Sarang, P., & Mathew, B. (2006). *Business Process Execution Language for Web Services.* Birmingham: Packt Publishing Ltd.

Jurišic, M. (2011, December). Transition between process models (BPMN) and service models (WS-BPEL and other standards): A systematic review. *Journal of Information and Organizational Sciences*, pp. 163-171. Retrieved from https://www.researchgate.net/publication/289158237_Transition_between_process_models_BPMN_and_service_models_WS-BPEL_and_other_standards_A_systematic_review

Keymark. (2023). *RPA Software Leaders Comparision Matrix*. Retrieved from Keymark Inc: https://www.keymarkinc.com/rpa-tools-comparison/

Kim, S.-H. (2023). Development of Evaluation Criteria for Robotic Process Automation (RPA) Solution Selection. *Electronics, 12*(4), 986. Retrieved from https://www.mdpi.com/2079-9292/12/4/986

Kregel, I., Koch, J., & Plattfaut, R. (2021). Beyond the Hype: Robotic Process Automation's Public Perception Over Time. *Journal of Organizational Computing and Electronic Commerce*, 1-21. Retrieved from https://www.researchgate.net/publication/351167879_Beyond_the_Hype_Robotic_Process_Automation's_Public_Perception_Over_Time

Microsoft. (2020, May 19). *Microsoft acquires Softomotive to expand low-code robotic process automation capabilities in Microsoft Power Automate*. Retrieved from Microsoft Power Automate Blog: https://www.microsoft.com/en-us/power-platform/blog/power-automate/microsoft-acquires-softomotive-to-expand-low-code-robotic-process-automation-capabilities-in-microsoft-power-automate/

Microsoft. (2023, 05 23). *Get started with Copilot*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/power-automate/get-started-with-copilot

Microsoft. (2024, June 25). *Triggering Desktop Flows*. Retrieved from Power Automate Documentation: https://learn.microsoft.com/en-us/power-automate/desktop-flows/trigger-desktop-flows

Noppen, P., Beerepoot, I., Weerd, I. v., Jonker, M., & Reijers, H. A. (2020). How to Keep RPA Maintainable? *Business Process Management*, 453-470. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-58666-9_26

OASIS. (2007, April 11). *Web Services Business Process Execution Language Version 2.0*. Retrieved from OASIS Documentation: https://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

Patri, P. (2020). Robotic Process Automation: Challenges and Solutions for the Banking Sector. *International Journal of Management, 11*(12), 322-333. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3785775

Peltz, C. (2003). Web services orchestration and choreography. *Computer, 36*(10), 46-52.

Schuler, J., & Gehring, F. (2018). *Implementing Robust and Low-Maintenance Robotic Process Automation (RPA) Solutions in Large Organisations.* SSRN. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3298036#references-widget

UiPath. (2024, October 17). *Orchestration User Guide*. Retrieved from UiPath Documentation version 2022.10: https://docs.uipath.com/orchestrator/automation-suite/2022.10/user-guide/introduction

Willcocks, L. P., Lacity, M., & Craig, A. (2015, October). The IT function and robotic process automation. *The Outsourcing Unit Working Research Paper Series*. Retrieved from http://eprints.lse.ac.uk/64519/

Appendix A: Full survey

Q1.1 Hello! My name is Diederik Oprel and I am an ICT in Business master's student at Leiden University in the Netherlands. I am conducting a research study on improving Robotic Process Automation (RPA) reliability by designing a vendor-agnostic RPA process orchestrator and I would like to invite you to participate in this survey. The purpose of this study is to identify the current issues with RPA orchestration, identify orchestration features to resolve these issues, implement these features in a framework and build an open-source working prototype with a subset of these features. Your participation in this survey is voluntary and your responses will be kept confidential. The survey will take approximately 5-10 minutes to complete. The survey consists of 3 parts, an introduction, RPA reliability and RPA orchestration. While answering the complete survey is appreciated, feel free to skip questions you don't feel like answering in too much detail. Thank you for your time and participation! If you have any questions or concerns about this survey, please contact me at d.j.oprel@umail.leidenuniv.nl. By completing this survey, you are giving your consent for the use of your responses for research purposes only.

End of Block: Consent

Start of Block: Introduction

Q2.1 Part 1: Background information This part will ask for background information regarding your experiences with RPA. It will be used to group results by experience per platform, company role, and seniority.

Q2.2 How often do you use RPA tools?

- ○ Daily (1)
- ○ Multiple times per week (2)
- ○ Weekly (3)
- ○ Monthly (4)
- ○ A few times a year (5)
- ○ Never (6)

Q2.3 Does your job involve RPA development?

- ○ Yes (1)
- ○ No (2)

Display This Question:

If Q2.3 = Yes

Q2.4 What is your job title?

_____

Display This Question:

If Q2.3 = Yes

Q2.5 How many years have you worked with RPA?

_____

Display This Question:

If Q2.3 = Yes

Q2.6 What RPA tools have you worked with?

- [ ] UiPath (1)
- [ ] Blue Prism (2)
- [ ] Automation Anywhere (3)
- [ ] Microsoft Power Automate Desktop (4)
- [ ] Microsoft Power Automate Flow (5)
- [ ] NICE (6)
- [ ] Pega (7)
- [ ] Kofax (8)
- [ ] OpenRPA (9)
- [ ] Robocorp (10)
- [ ] TagUI (11)
- [ ] Others (12)

Display This Question:

If Q2.6 = Others

Q2.7 Please mention any RPA tools you have worked with which were not mentioned in the list above

_____

Q2.8 Please use this space for any background information regarding your work with RPA, if necessary.

_____

_____

End of Block: Introduction

Start of Block: Reliability

Q3.1 Part 2: Reliability This part will explore the reliability issues which can arise when scheduling processes to run unattended. We call the system which performs this scheduling an orchestrator, you can find a formal definition for an orchestrator below. Definition: *An orchestrator is a system which communicates with related systems and instructs them on what and when to perform certain actions.*

Q3.2 Have you worked with RPA environments where process execution could be unreliable at times?

- Yes (1)
- No (2)

Q3.3 Have you faced an increase in maintenance due to problems with process orchestration or scheduling?

- Yes (1)
- No (2)

Display This Question:

If Q3.3 = Yes

Q3.4 What effects due to issues with process orchestration or scheduling have you encountered? This question is designed to show some examples of what issues can arise due to scheduling issues. Please provide your own experiences in the text box in the following question.

- Required to rerun processes at a later time/date due to a failed schedule (1)
- Missing data for end-users (2)
- Accumulating process delays over an entire day (3)
- Low bot utilization due to buffers between schedules (4)
- Other (5)

Display This Question:

If Q3.3 = Yes

Q3.5 Please mention any side effects due to orchestration or scheduling issues which were not mentioned in the list above

_____

_____

Display This Question:

If Q3.3 = Yes

Q3.6 How severely do such issues affect your day-to-day work and maintenance regarding RPA?

- ○ Not at all (1)
- ○ A little (2)
- ○ A moderate amount (3)
- ○ A lot (4)
- ○ A great deal (5)

Q3.7 Please use this space for any comments you would like to share regarding RPA reliability, if necessary.

_____

_____

End of Block: Reliability

Start of Block: Orchestration

Q4.1 Part 3: Orchestration This final part is designed to identify what features an orchestrator should contain to address the issues you're facing with scheduling and orchestration and what current features you like. Some tools may have advanced orchestration already, but the goal is to identify what is required for an independent platform agnostic orchestrator. So features for an orchestrator don't have to be unique, but rather a combination of existing orchestrators and commonly missing features. 'RPA Orchestration' can be defined as follows: *The RPA orchestrator is the part of an RPA environment which communicates with related systems and instructs them on what and when to perform certain processes.*

Q4.2 Are you familiar and in agreement with the term 'RPA orchestration' as described above?

- ○ Yes (1)
- ○ No (2)

Display This Question:

If Q4.2 = No

Q4.3 What does the term 'RPA orchestration' mean to you?

_____

_____

Q4.4 From the following list of example features, please group the ones relevant to you by importance for an RPA orchestrator. This question should provide you with an idea of how to categorize features, the following question will allow you to provide your own feature ideas.

Must have	Should have	Could have	Will not have
_____ Queue processes instead of fixed time schedules (1)	_____ Queue processes instead of fixed time schedules (1)	_____ Queue processes instead of fixed time schedules (1)	_____ Queue processes instead of fixed time schedules (1)
_____ Processes can be added to queue on demand by process owners (2)	_____ Processes can be added to queue on demand by process owners (2)	_____ Processes can be added to queue on demand by process owners (2)	_____ Processes can be added to queue on demand by process owners (2)
_____ Roles and permissions (Process owners should only be able to access their own processes) (3)	_____ Roles and permissions (Process owners should only be able to access their own processes) (3)	_____ Roles and permissions (Process owners should only be able to access their own processes) (3)	_____ Roles and permissions (Process owners should only be able to access their own processes) (3)
_____ Notify stakeholders in case of delays, failures or high number of exceptions. (4)	_____ Notify stakeholders in case of delays, failures or high number of exceptions. (4)	_____ Notify stakeholders in case of delays, failures or high number of exceptions. (4)	_____ Notify stakeholders in case of delays, failures or high number of exceptions. (4)
_____ Realtime process statistics for process owners (processed items, time to completion) (5)	_____ Realtime process statistics for process owners (processed items, time to completion) (5)	_____ Realtime process statistics for process owners (processed items, time to completion) (5)	_____ Realtime process statistics for process owners (processed items, time to completion) (5)
_____ Defer processes in case it returns a specific error (applicable when applications or data sources are not available) (6)	_____ Defer processes in case it returns a specific error (applicable when applications or data sources are not available) (6)	_____ Defer processes in case it returns a specific error (applicable when applications or data sources are not available) (6)	_____ Defer processes in case it returns a specific error (applicable when applications or data sources are not available) (6)
_____ Queue verification, to avoid a process from being added too many times (7)	_____ Queue verification, to avoid a process from being added too many times (7)	_____ Queue verification, to avoid a process from being added too many times (7)	_____ Queue verification, to avoid a process from being added too many times (7)
_____ Remotely restart the RPA environment in case it is not reachable (8)	_____ Remotely restart the RPA environment in case it is not reachable (8)	_____ Remotely restart the RPA environment in case it is not reachable (8)	_____ Remotely restart the RPA environment in case it is not reachable (8)
_____ Provide a combined process overview of multiple RPA tools (9)	_____ Provide a combined process overview of multiple RPA tools (9)	_____ Provide a combined process overview of multiple RPA tools (9)	_____ Provide a combined process overview of multiple RPA tools (9)
_____ Process added to queue upon data availability (email/API trigger) (10)	_____ Process added to queue upon data availability (email/API trigger) (10)	_____ Process added to queue upon data availability (email/API trigger) (10)	_____ Process added to queue upon data availability (email/API trigger) (10)
_____ Overview of previously ran processes (11)	_____ Overview of previously ran processes (11)	_____ Overview of previously ran processes (11)	_____ Overview of previously ran processes (11)
_____ Fill process parameters from predefined list (12)	_____ Fill process parameters from predefined list (12)	_____ Fill process parameters from predefined list (12)	_____ Fill process parameters from predefined list (12)

Q4.5 Are there other features which were not included in the list above which are relevant for RPA orchestration? This is the most important question, please provide an answer.

_____

_____

Q4.6 How would the features mentioned by you or included in the list above help resolve the issues you're currently facing with RPA scheduling and orchestration?

_____

_____

Q4.7 Have you worked with multi-vendor RPA deployments, running different RPA tools alongside each other?

- ○ Yes (1)
- ○ No (2)

Display This Question:

If Q4.7 = Yes

Q4.8 What were your experiences regarding RPA scheduling and orchestration with multi-vendor RPA deployments?

_____

_____

Q4.9 Please use this space for any comments you would like to share regarding RPA orchestration, if necessary.

_____

_____

End of Block: Orchestration

Start of Block: Interview

Q5.1 Would you be available for an interview to further explore this topic and your answers?

- ○ Yes (1)
- ○ No (2)

Q5.2 Would you like to receive a copy of the research once finished?

- ○ Yes (1)
- ○ No (2)

Display This Question:

If Q5.1 = Yes

Or Q5.2 = Yes

Q5.3 Please provide an email address where I can reach you:

_____

End of Block: Interview

Appendix B – Coding frequency surveys and interviews

Code Tree	Frequency
Framework	17
Suggestions	3
suggestion-area-overlap	1
suggestion-dimension-integrations	2
framework-feedback-positive	11
framework-feedback-suggestion	3
Multi-Vendor	21
Reasons	11
problem-multiple-channels	5
reason-different-tool-strengths	5
reason-tool-democratization	1
multivendor-usage	10
Orchestration	391
Current offerings	62
Problems	33
problem-currentoffer-excessive-cost	6
problem-currentoffer-unclear-pricing	2
problem-license-utilization	3
problem-manual-intervention-required	9
problem-process-scheduling	12
problem-scalability	1
orchestration-currentoffer-cost	10
orchestration-currentoffer-problemrecognition	19
Features	277
Improvement	32
improvement-business-involvement	3
improvement-clarify-action-required	5
improvement-compliance	2
improvement-cost-effectiveness	2
improvement-incident-prevention	1
improvement-infrastructure-setup	2
improvement-process-continuity	5
improvement-reduced-maintenance	2
improvement-resource-capacity	3
improvement-scalability	2
improvement-usability	5
Feature	132
feature-advanced-access-control	4
feature-control-process-variables	6
feature-data-encryption	1
feature-dependency-based-server-usage	7
feature-dependency-diagnostics	2
feature-environment-logging	4
feature-environment-presentation	16
feature-error-localization	7
feature-exporting-data	3
feature-flexible-server-usage	4

feature-improved-alerts	2
feature-improved-queue-loading	6
feature-improved-triggers	19
feature-infrastructure-control	5
feature-input-validation	2
feature-load-balancing	3
feature-priority-process-switching	5
feature-process-logging	6
feature-process-queueing	4
feature-process-statistics	6
feature-process-visualization-process-mining	3
feature-queue-connections	3
feature-queue-data-adjustment	1
feature-runtime-forecasting	6
feature-server-quarantine	2
feature-software-version-control	5
orchestration-feature-desired	77
orchestration-feature-impact	23
orchestration-feature-strength	13
Issues	52
Issue	30
issue-excessive-resources-monitoring	4
issue-increased-maintenance-required	13
issue-missed-deadlines	3
issue-poor-license-utilization	6
issue-process-handed-back-to-business	2
issue-reputation-damage	2
orchestration-issue-consequences	22