Business Process Automation Using Intelligent Software Robots

João Diogo Castro

Thesis to obtain the Master of Science Degree in Engenharia Informática e de Computadores

Supervisors: Prof. José Manuel da Costa Alves Marques
Engr. Helena de Sousa Oliveira

Examination Committee
Chairperson: João António Madeiras Pereira
Supervisor: Prof. José Manuel da Costa Alves Marques
Members of the Committee: Prof. José Alberto Rodrigues Pereira Sardinha

November 2018
Abstract

Robotic process automation (RPA) technology may automate routine, low-skill, low value-adding but necessary work. Its' purpose, benefits and drawbacks were researched and compared to alternative business process automation technologies such as business process modelling suites (BPMS) and enterprise resource planning (ERP) systems. The problem is understanding how much and in which situations is RPA viable. To find out, research was done on the available RPA tools, the characteristics of business processes fit for RPA and how common features found in RPA tools can help deliver these benefits. A real case business process was selected for carrying out an implementation project of RPA. Metrics were collected regarding robot time and quality performance and compared to human time performance. The results are mixed, showing business process productivity increase in some situations and productivity reduction in others. Conclusions were reached regarding design patterns and challenges inherent to RPA, which help guide future projects.

Keywords: Application Integration with RPA, Tool Features, Financial Services, RPA Use Case
**Resumo**

A tecnologia de automação robótica de processos (ARP) pode automatizar trabalho repetitivo, que acrescenta pouco valor, não requer muito conhecimento, mas que é necessário. O seu propósito, benefícios e desvantagens foram investigados e comparados com tecnologias alternativas de automatização de processos de negócio, tais como os business process modelling suites (BPMS) e sistemas de planeamento de recursos empresariais. O problema é perceber em que situações e o quão viável é a tecnologia de ARP. Para descobrir, investigaram-se as ferramentas de ARP disponíveis, as características dos processos de negócio propícios à ARP e como as características comuns dessas ferramentas permitem obter esses benefícios. Foi selecionado um processo de negócio de um caso real para fazer a implementação de um projeto com ARP. Registaram-se métricas relativas ao tempo e qualidade de desempenho do robot, que foram depois comparadas com o tempo de desempenho de um humano. Os resultados foram mistos, demonstrando aumento na produtividade do processo de negócio nalgumas situações e redução do desempenho noutras. Obteram-se conclusões em relação a padrões de desenho e dificuldades inerentes à ARP, que permitirão guiar projetos futuros.

**Palavras Chave:** Integração de Aplicações com ARP, Características de Ferramentas, Serviços Financeiros, Caso de Uso ARP
Table of Contents

1 Introduction........................................................................................................................................... 9
1.1 Objectives ....................................................................................................................................... 10

2 State of the Art...................................................................................................................................... 12
2.1 What is RPA ..................................................................................................................................... 12
2.2 RPA and Existing Business Process Automation Technologies......................................................... 13
2.3 Benefits of RPA Automation ........................................................................................................... 16
2.4 Disadvantages of RPA Automation ................................................................................................. 18
2.5 RPA Tool Features Explained – How They Deliver Benefits ............................................................. 19
2.6 UiPath Concepts ............................................................................................................................. 29
2.7 Business Processes and RPA ........................................................................................................... 33
2.8 RPA Vendor’s Product Main Strength and Preferable Usage Context ............................................. 37

3 Methods ............................................................................................................................................... 39
3.1 Prototype ......................................................................................................................................... 39
3.2 Project ............................................................................................................................................. 40
3.3 Design Patterns ............................................................................................................................... 46

4 Results .................................................................................................................................................. 69
4.1 Evaluation Methodology .................................................................................................................. 69
4.2 Process Automation Statistics ......................................................................................................... 69
4.3 Conclusions from the Results ........................................................................................................ 72

5 Discussion ........................................................................................................................................... 74
5.1 Implementation Challenging Problems and Solutions ........................................................................ 74

6 Conclusion .......................................................................................................................................... 80
6.1 Future Work .................................................................................................................................... 81

Acknowledgements ............................................................................................................................... 82

Bibliography .......................................................................................................................................... 83

7 Annex ................................................................................................................................................... 85
List of Figures

Fig. 1. Example of Web Page Button Selector for Google Chrome Browser Using UiPath ........................................... 22
Fig. 2. Activities Panel (left) and Properties Panel (right) in UiPath Studio ........................................................................ 30
Fig. 3. Sequence in UiPath Studio .................................................................................................................................. 30
Fig. 4. Example of “Click” activity contained inside “Retry Scope” activity and condition testing with “Element Exists” activity .................................................................................................................................................. 31
Fig. 5. UI Explorer Showing Different Panels: 1 – Current Selector; 2 – Selector Attributes for Editing; 3 – UI Elements Hierarchy ........................................................................................................................................... 32
Fig. 6. Business Process Model of “Checking in New Liability Investigation Requests from Web Portal” ................................................................. 41
Fig. 7. Model of “Checking in New Liability Investigation Requests from Email” ................................................................. 42
Fig. 8. Business Process Model of “Delivering the Liability Investigation Results” ........................................................................... 42
Fig. 9. RPA Solution Deployment Architecture ................................................................................................................ 45
Fig. 10. Failed Click (Left) Comparison with Successful Click (Right) after Adjusting Properties ............................................. 50
Fig. 11. Configuration of the “Click” Activity’s Properties .................................................................................................. 51
Fig. 12. UI Element Region Referenced by the Selector ........................................................................................................ 51
Fig. 13. Example of “Windows Security” User Login Prompt ................................................................................................ 52
Fig. 14. Virtual Keyboard that was Automated in the Project .................................................................................................. 53
Fig. 15. Virtual Keyboard with Different Characters in the Buttons (Another Instance) ....................................................... 53
Fig. 16. Window Prompt Presented by a Desktop Application for Saving a File ........................................................................ 54
Fig. 17. Two Tabs are Open at the Same Time by the Robot During the Execution of an Implemented Automation .......................................................................................................................... 55
Fig. 18. Different Ways to Program Data Table Extraction from Web Pages ........................................................................ 57
Fig. 19. Correct Order of Clicking to Select Columns When Building Data Table Template, Following the Data Scraping Wizard ............................................................................................................................................. 58
Fig. 20. Wrong Order of Clicking to Select Columns When Building Data Table Template, Following the Data Scraping Wizard ............................................................................................................................................. 58
Fig. 21. Example of Using the “Retry Scope” and “Element Exists” Activities to Ensure the Expected Results .......................................................................................................................................................... 60
Fig. 22. Confirmation Prompt Launched by “Save As” Windows Application – Overwriting a File Exception ..................................................................................................................................................... 61
Fig. 23. Click Event Prompt and Catch Exception if the Event did not Trigger ........................................................................... 62
Fig. 24. “Google OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time (Language: English; Scale: 2) ................................................................................................................. 64
Fig. 25. “Google OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time (Language: Portuguese; Scale: 1) ................................................................................................................. 65
Fig. 26. “Google OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time (Language: Portuguese; Scale: 3) ................................................................................................................. 65
Fig. 27. “Microsoft OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time (Language: English; Scale: 1) ................................................................................................................. 66
Fig. 28. “Microsoft OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time (Language: Portuguese; Scale: 1) ................................................................................................................. 66
Fig. 29. “Microsoft OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time (Language: Portuguese; Scale: 3) ................................................................................................................. 66
Fig. 30. Path Text Field Which is Selected After “F4” Key is Pressed ....................................................................................... 75
Fig. 31. Extracted DataTable in Each Page of the New Request Search Results ........................................................................ 76
Fig. 32. UI elements to navigate results table, without the option of going to “Next Page” (button not compatible with “Data Scraping” wizard) .................................................................................................................. 76
Fig. 33. Table from Which the Robot Had to Extract Data Before The Order of The Columns Was Changed ......................................................................................................................................................... 77
Fig. 34. Table From Which the Robot Had to Extract Data After The Order of The Columns Was Changed ......................................................................................................................................................... 77
Fig. 35. Debug Features in UiPath (“Debug”, “Breakpoints” and “Slow Step”) ............................................................................. 78
Fig. 36. “Connect” activity in UiPath Studio .......................................................................................................................... 85
Fig. 37. Database Connection Wizard Step 1 ......................................................................................................................... 85
Fig. 38. Database Connection Wizard Step 2 ......................................................................................................................... 85
Fig. 39. Database Connection Wizard Step 3 ......................................................................................................................... 86
Fig. 40. Scanned Document as .png File Targeted for OCR

88
List of Tables

Table 1. RPA Tool Features Score (Ordered From Highest to Lowest) Adapted From (Le Clair, 2017) .................................................................................................................................................. 38
Table 2. Mapping of Business Process Tasks and the Design Patterns used to Implement it ........ 47
Table 3. First Human Test Results (Checking in New Liability Investigation Requests from Web Portal) ........................................................................................................................................ 70
Table 4. Second Human Test Results (Checking in New Liability Investigation Requests from Web Portal) ........................................................................................................................................ 70
Table 5. Human Test Results – Delivering The Liability Investigation .................................. 71
Table 6. RPA Tools Feature Score by Category (Ordered From Highest to Lowest) Adapted From (Le Clair, 2017) .............................................................................................................................................. 87
List of Acronyms
Anti-money laundering (AML), 36
application programming interface (API), 14
Business Process Management Suites (BPMS), 9
Cognitive Intelligence (CI), 15
Customer Relationship Management (CRM), 37
Enterprise Resource Planning (ERP), 9
information technology (IT), 9
Intelligent Automation (IA), 15
know your customer (KYC), 36
optical character recognition (OCR), 22
Robotic Process Automation (RPA), 9
1 Introduction

Robotic Process Automation (RPA) is an automation methodology that programs and deploys (intelligent) software robots to interact with information systems. This includes desktop applications, web applications, remote desktop applications and databases. What sets these software robots apart from traditional integration is that these are designed to communicate through the presentation layer (user interface), not the business logic or data access layer. This means the robot mimics the way a human uses a desktop, interacting with the user interface. RPA implementation and deployment can be done without changing the information systems that support the automated business process. Another distinguishing characteristic is that it can be programmed by business users, with less programming knowledge than information technology (IT) users, although some programming knowledge is still required. However, the extent to which the business user can use an RPA tool to deploy an automation depends highly on the simplicity of the business process. Complex business processes will require technical IT knowledge.

One of the purposes of robotic process automation is automating structured, repetitive, rules-based tasks. Another is information systems integration, by bridging and consolidating data in different applications or systems.

Choosing a business process that is fit for automation using RPA is key in the success of an RPA deployment, since not every business process will benefit from RPA. The impact of RPA may be significant when applied in the appropriate situations. It can further reduce business process costs and increase their quality, for big organizations performing rote business processes with high volume of transactions and prone to errors. RPA makes it possible to integrate different information systems in business processes where extending Enterprise Resource Planning (ERP) systems would not be viable in terms of cost and time, because RPA is faster and cheaper to implement than ERP. There is also opportunity for further optimization in situations where Business Process Management Suites (BPMS) cannot optimize certain parts of the business processes or integrate the information systems that support the process.

More advanced and intelligent versions of the software robots should be capable of handling unstructured data, analyzing text, finding patterns and inferring the next step. (Lowes, et al., 2015) Looking further into the future, the concept of intelligent automation appears, in which the software robots are, in general, more autonomous, capable of learning, adapting and providing better insights.

In short, to deploy robotic process automation, a suitable, preferably well-structured and high-volume, business process is chosen. Then, a business or technical developer specifies the steps the robots must perform (the automation). The automation is assigned to one or more robots from the central control room. If any error occurs, a human is assigned to handle exceptions that the robot cannot deal with.

Typical tasks the robots perform are opening emails and attachments, logging into applications, scraping data, filling in forms and so on. (Lowes, et al., 2015)
The number of vendors providing RPA tools is significant and the technology is still growing, with feature updates and bug fixes being released consistently over time. A small overview of twelve different vendors is provided, each supplying their own set of tools.

To better understand RPA technology, UiPath’s solution was used to develop several automations that tested the more distinctive features of RPA, such as recorders, excel integration, email integration, OCR, data scraping, visual programming and robot orchestrator. These automations focused mostly on interacting with web applications, extracting information and saving information in the desktop, using image files, text files and excel files. Information and orders were also submitted to the web application by the robot.

The automations were developed and deployed with the intent to automate a few business processes. However, during development, it was necessary to create testing conditions for the automation. This led to the creation of automations that were not necessary to automate the business process in itself but were a more efficient way of transitioning the state machine of the web applications to a state that was required for the first step of the business processes. Since none of the web applications were accessed through their code, this transitioning of the state was done interacting with user applications, one of which was deployed in a remote desktop.

1.1 Objectives

This project aims to evaluate the usage and related benefits of RPA in business processes, implemented by large organizations, in which there are large volumes of non-value adding actions that are required to be performed by humans, with more focus on the financial services and insurance sectors.

The key objectives are:

- Selecting a business process fit for RPA automation
- Selecting an RPA tool fit for the project’s requirements
- Implementing and deploying an RPA solution
- Collecting data and verifying the benefits of the RPA deployment

The business process will be selected according to suitability to robotic process automation and limited to the pool of stakeholders who offer to partner in this project by providing a case scenario as close to real as possible.

The initial part of the project will require gathering information about the current state of the art in terms of robotic process automation tools. These tools will be analyzed according to their features and one that meets the requirements for evaluating the success of an RPA implementation will be chosen.

An RPA implementation and deployment are to be done. Implementation of the business process automation will focus on delivering process automation with resilience to change, reduced business process cycle time, reduced error rates (negative outcomes such as wrong information) and thus reduced cost per execution.

Finally, data will be gathered relating to the execution of the target business process with and without the use of software robots, to verify that robotic process automation does deliver the expected
benefits for business processes with non-value adding tasks: cost reduction, time reduction, quality increase, headcount reduction.
2 State of the Art

2.1 What is RPA

In short, RPA is a technology that aims to further optimize business processes by automating more or less complex, rules-based, structured business processes with very frequent executions. RPA providers make an effort to develop solutions that can be learned and used by business users and that are quick to implement. RPA is another option for business process automation, which is still developing, but positive results have been confirmed by organizations that already deployed this kind of solutions (Lacity & Willcocks, 2016).

General Architecture

Although there are exceptions, the architecture of RPA tools is generally composed by three components:

- Developer Tools
- Control Room
- Software Robots

Developer Tools

The developer tools look like diagraming tools. It serves to program the robot. The better the functionality provided by the developer tools, the faster the robot can be configured, implemented and deployed. Good developer tools deliver speed and flexibility to the automation of business processes.

From this component, the user defines jobs (or sequences or workflows, depending on the RPA tool). These jobs are no more than sequences of the building blocks that encapsulate the robot instructions. When these blocks are well put together, the robot knows how to carry out the work previously done by humans and will inform if any exceptions occur. The blocks provide functionalities such as if-else decisions, loops and exception handling to comply with business rules.

Usually, the user may expand or hide blocks and, may also reuse these blocks in other sequences. This is very important to manage project complexity, since business processes usually involve a lot of actions (which the person may not be aware of at first because it is so natural for us to use a computer). Blocks can be created or rearranged using drag-and-drop, which means the user clicks one block, holds the mouse button down and drags the block to its’ new position.

Control Room

This is the component from which an RPA system administrator can do tasks related to RPA system management. These tasks include task scheduling, workload management, auditing, troubleshooting (accessing the execution logs), defining prioritized work queues and getting access to analytics. This component is also called control room, control center, control tower, orchestrator, depending on the RPA tool.
In general, the control room is critical to achieve deployment flexibility, security, compliance, agility and scalability which keep operations running, thus contributing to business continuity, quality, speed and compliance. Here is why.

First, it works as a central repository. The developers use the tools to sequence the steps the robot has to follow and, once tested these jobs can be uploaded or published to the control room. This repository may also safely store and distribute encrypted credentials that the robots require for the business applications to perform the business processes.

Secondly, it is where the roles and user permissions are assigned, to allow to establish who can create, update, test, review, approve and deploy robots. Robot scheduling is done from this component too.

Thirdly, the RPA system administrator can easily find tasks that were not completed or failed by looking at the job lists and analytics dashboard. It is also possible to audit the operations of the business and the best tools make sure to include features that specifically try to make auditing effortless.

Fourthly, having a centralized remote platform from which one may deploy a variable number of robots and different automations into production allows responding to business needs and different workloads.

Lastly, this is where the work of robots is managed from. The robots can be ordered to execute the jobs at that instant or scheduled according to available criteria. Jobs may be carried out by groups of robots or single robots. The results produced by their work is monitored from this component, whether it succeeded or failed (with or without exceptions). Normally, dashboards are provided by the RPA tool here, allowing the system manager to analyze robot status and workload distribution.

**Software Robots**

Mostly, software robots just represent an installation in a desktop or virtual machine which understands the job specifications and enables the execution of the instructions defined using the developer tools. Robots may perform more than 600 different actions in some solutions (Lowes, et al., 2015) and these can be further extended by creating custom code.

Although the user does not configure this component directly, it is important to keep it in mind, since one robot instance usually is the equivalent to one software license. Minimizing the number of robots needed to automate the business processes reduces costs.

This component is also responsible for registering logs of every action taken during automation execution. Logs may have different levels of detail. The user decides which level of detail is recorded during the executions.

### 2.2 RPA and Existing Business Process Automation Technologies

There are a few options when organizations already supported by information systems need to optimize business processes.

The first option is to extend the functionality of their current supporting information systems to integrate different information systems.
The second option is BPMS, which integrate organizational information systems, such as ERP systems. Like the previous option, BPMS implementations require changing the business logic layer and data access layer of the information systems. Usually, it has a smaller scope, costs less and there is less risk of failure to implement. The benefits it may bring are also smaller.

RPA is a more recent alternative, that aims to integrate information systems and to optimize the results of business processes by automating certain tasks. It should be even faster and less costly to implement than BPMS systems, because it requires less specialized programming knowledge (in some cases) and does not require changes to the underlying systems (the business logic and data access layer).

There isn’t one specific automation technology better than the other. It depends on the context in which it will be applied. A combination of the previous options may be applied as well.

**How RPA Differs From BPMS**

RPA is characterized by easy configuration, not necessarily requiring programming skills to develop automations. Unlike most BPMS solutions, which require programming skills and extensive specialized studying before implementing such projects, RPA tools can be learned by business users in a few weeks (some more recent BPMS solutions don’t require specialized programming skills and can also be learned by business users relatively fast). This is because RPA tools abstract the underlying system’s code. As well, the robot programming is more visual, with the instructions wrapped in visual blocks, which have different configuration options. Basic programming concepts are still required, such as basic data structures (arrays, strings, integers, collections), but the user doesn’t have to put so much effort digging through specific library functions and programming language documentations to achieve the desired result. In fact, some RPA tools are tailored and focus more on business users, for example, allowing the use of Visual Basic .NET expressions to configure the automation blocks (Lacity & Willcocks, 2016). This leads to lower development times which make RPA a good replacement for BPMS and ERP in projects where these cannot bring return on investment due to the long time it takes to finish them. However, it should be noted that this characteristic of RPA programming only applies to simple business processes, which involve user applications that are stable over time, which have very consistent user interface elements, features and behavior. For example, after clicking a button in the application, a new window will pop up always with the same pre-defined size and position on the screen, with buttons in the exact same place as in a previous instance of the business process execution.

Most important of all, RPA is meant to be non-invasive, which means it does not interact with the underlying application programming interface (API) of information systems. Instead it interacts directly through the user interface. There is no need to create, replace or extend the systems it sits on top of. If required, it is capable of safely logging in to applications. This allows RPA tools to patch the holes by integrating any applications that were not meant to be integrated, whether it is because the business processes changed, and they can’t support it from end to end or because it would be too expensive to build an information system with such a large scope (Lacity, et al., 2015). Sometimes, the information systems may not be integrated because the business process involves third party IT systems that do
not supply an integration interface via API. In other words, robotic process automation tools integrate using the presentation layer, while BPMS solutions integrate using the business logic and data access layer. RPA exposes “application user interfaces through reusable services.” (Chappel, 2010, p. 3)

**How RPA Differs from Scripting and Screen Scraping Tools**

Scripting and screen scraping tools, may be labeled as RPA, but lack to present a few of the characteristic features of RPA.

Scripting and screen scraping tools, older technologies, fall short from RPA tools because they cannot be deployed remotely. It must be done from the user’s desktops. RPA tools usually come with an orchestrator that allows remote deployment. It can also be deployed by the user in a local machine.

Screen scrapers function as record and replay. It only records the users moving fields across systems. These extract data based solely on the screen position of the field containing that data (the origin) and can then copy the data to another field, using the screen coordinates of this field (the target). If the coordinates of any of the fields change, the screen scraper automation no longer works, because it has no way to find out where the field was moved to, unless someone reprograms the screen scraper automation. On the other hand, RPA tools don’t depend on fixed data locations as they have other methods to locate the fields. One way is to associate the fields with their respective underlying representation, whether this representation can be found through HTML code (for web application fields) or Java Access Bridge (for Java applications) or surface automation in Citrix applications. Thus, changing the fields position does not break the automation (Lacity & Willcocks, 2016). In section 2.5, more in-depth information regarding these methods is provided.

RPA is designed for the enterprise. These tools fulfill enterprise security, scalability, auditability and change management requirements. RPA robots can be deployed, scheduled and monitored from a central “control room”. This ensures compliance with enterprise security standards and with business continuity plans, since the real-time information of the robots’ status and execution of business processes allow the system manager to know if any operations stopped. (Lacity, et al., 2015)

**Intelligent Automation**

RPA is capable of automating routine tasks (methodical, repetitive and rules-based). Nonroutine tasks, with more complex decision making, involving other types of intelligence, such as creative intelligence or even persuasion skills, and the ability to draw conclusions from patterns present a clear barrier to automation using RPA technology. These are left for Intelligent Automation (IA), also known as Cognitive Intelligence (CI). Typically, these tasks also have less volume of transactions and take longer to perform.

Cognitive technologies are becoming better at recognizing handwriting, identifying images and processing natural language. IA solutions combine robotic automation and artificial intelligence powered analytics. These may assist people with nonroutine tasks and possibly automate some of them entirely. One example of this kind of technology is IBM’s Watson.

IA solutions are more expensive than RPA solutions, since it requires longer configuration. It also requires machine learning specific to the subject it deals with and to the complex scenarios it is de-
signed for. One RPA solution is applicable to more situations than one IA solution and takes less time to implement.

RPA tools focus on reducing costs through automation of high volume, rules-based, routine tasks that are still necessary for business. IA tools aim to increase value delivery by improving non-routine tasks that require judgment.

Although there are exceptions, IA tools may enable businesses to expand their services, delivering more value to customers, while RPA tools partially or fully automate services, offering them with a lower cost and higher level of quality. (Lowes, et al., 2015)

2.3 Benefits of RPA Automation

Very shortly, RPA delivers better quality, compliance, increased efficiency, agility, data analytics, reduced costs, improved employee experience, customer experience and logistics. This means, less errors in business process outputs, better customer service, alignment with regulations and standards, faster business process execution, less overhead due to an adaptable workforce, data that feeds analytics to provide insight to the business performance and, finally, increased value added by employees since they focus on more complex tasks. As an extra, employee morale is boosted because they don’t have to do as many boring tasks. Since the overall results of the services provided are better, customer experience also benefits, especially due to increased speed in handling requests and less errors, which take time to handle. Better logistics are in place after RPA deployment, since the complexity of the communication channels is reduced (less people involved in the business process execution) and robot deployment may be centralized.

More detail is now provided about each benefit.

Improved Business Process Quality

Humans are more likely to introduce mistakes when performing the rote, repetitive tasks that RPA robots are designed for. People get bored, they lose focus and accidents happen. Software robots, on the other hand, are more consistent with these tasks. Robots still have to be tested, well programmed and managed. However, if these tasks are properly programmed, it is unlikely that the robots will make mistakes.

Increased Regulatory Compliance

RPA requires each step in the business process to be programmed inside the automated system, easing compliance checks and regulation audits. As such, costs in these obligatory processes are reduced and the results produced by these are of higher quality. This is especially applicable to industries with functional areas regulated by strict compliance guidelines, of which healthcare, banking and insurance are an example. The RPA deployments register detailed data about process execution, facilitating compliance with specific regulations.
**Increased Efficiency**

Unlike their human counterparts, software robots do not need time off, except for maintenance purposes. However, with proper IT service continuity methodologies and guidelines in place, the automation executions don’t stop, or at least the robot downtime is reduced. Processing times are reduced, given that machines are faster at executing transactions and routine tasks.

According to (Accenture, 2016), RPA pilots carried out by insurance companies resulted in a 40-80 percent reduction in processing times. Still in the financial services industry, companies such as Barclays and The Co-operative Group generated capacity of up to two hundred full-time employees through automated processes in operations. One insurer in the UK freed up to 34 percent of capacity in one business area after implementing an automation programme.

**Increased Agility**

Robots can be configured to run different pre-developed automations if required. The RPA system manager may assign robots to carry out one business process instead of another remotely and fast from the control room component. This allows the organizations which deployed RPA to respond faster to the needs of customers.

**Improved Data Analytics**

The tasks the robot executes produce data which can feed appropriate software components for their analysis. Most RPA tools provide these, plus integrations with third party tools for data analysis and data visualization. For example, UiPath’s RPA tool integrates with Kibana. This data analysis enables better decision making regarding the automated processes, allowing to answer questions like how many robots should be running at a given time of the day to handle the incoming workload. Detailed execution times of every step allow the organization to discover improvement points.

**Costs Savings**

Due to the scalability of software robots and workload adaptability, costs may be reduced. Only the necessary robots are active to respond to business demand.

If we take into account the costs of correcting an error introduced by a human employee, there are also costs saved in handling these problems, since well programmed robots make less mistakes.

**Higher Employee Productivity**

Because software robots perform the rote and boring part of the business processes, the employees can focus on higher value activities, ones that require social, problem solving, and decision-making skills. Robots and people complement each other, as robots perform the structured, rules-based tasks more efficiently than people and people can handle tasks outside the reach of automation technologies.

**Increased Customer Satisfaction**

Aided by the efficiency and quality of RPA, employees are freed from the less valuable tasks and they now have more energy and time to focus on the customer. There is higher probability the cus-
customer will be satisfied with their experience and less customers are lost which improves customer relationships. The client requests may be processed faster as well.

Improved Logistics

Since RPA opens the possibility to reduce the number of full-time employees and substitute off-shore work, the number of communication channels are reduced, taking away complexities related to the management of human resources. (Casale, et al., 2017)

2.4 Disadvantages of RPA Automation

Despite the many benefits, there are some drawbacks to RPA technology. Some of these disadvantages, as well as examples, will be pointed out in this subsection.

First, it requires extensive testing, because there may be many different scenarios and pre-conditions for the automations. Also, the robot may not perform the actions as expected, as sometimes it may not actually click a button, even when it was programmed to do so.

For example, not clicking a submit button to save the data from a submission form can change the pre-conditions for a later stage in the automation that relies on the data from the form that was supposed to be submitted. The robot will not have the necessary information available later and fail without indicating where it actually failed: at the time of clicking the button to submit the data. Instead it will raise an exception when it fails to retrieve the data that was supposed to have been submitted.

Another example is, when saving a file. If a file was previously saved to a different folder, the next time the file is saved, the path in the folder navigation window will be different. The robot may have unexpected behavior, such as not writing the file path of the file saving location. Then the file may be saved to another unexpected location and an automation that relies on that file may fail.

The system administrator is then required to directly access the machine on which the robot is working to understand what went wrong (which may be a bit hard since the robot may throw an error or an exception at a later stage of the workflow than when the error actually happens). The administrator will have to go through the workflow and must have understanding of that workflow in order to solve the problem. The better RPA providers make an effort to deliver comprehensive logging features which help solving these problems faster.

Using UiPath, one way to mitigate this issue, is by isolating each task by creating a queue item. When the robot fails to execute the task, this is registered in the orchestrator. The system administrator can then access the machine in which the robot is working and tell the robot to retry the queue item task through orchestrator's transactions manager and see where the robot performs a bad step. However, some changes to the automation might be required, so that the robot performs the steps slow enough for the administrator to understand what is being done wrong. Examples of these changes are configuring the steps to be done without the simulation option or the send message option. When these options are active, the robot omits the mouse pointer movements and speeds up the execution of the process.

Another disadvantage of RPA can be the visual programming, depending on the size of the project. It may become hard to manage the code if the automation is too complex and involves many cases
and decisions. Very good structuring of the whole automation is required. This is particularly true for UiPath’s tool in which, for example, the “If” activity expands the visual flow of the program horizontally, this is, widens the window of UiPath Studio, making navigation of the implementation harder. A situation with three or more branches (nested “If” activities) can make a simple sequence harder to read and understand.

A particular disadvantage is the fact that it is hard to estimate implementation times, since it requires knowing the underlying information systems with which the robot will have to interact with, from the viewpoint of the user. It requires domain knowledge of the business process, since the programmer should be familiar with the behavior of the systems that the robot has to interact with. However, the same can be said for traditional programming, since the developer has to be acquainted with the API’s and libraries that the program relies on and communicates with. Because RPA has to interact with user interfaces, the interface’s behavior can be a little bit less defined (not as well documented) than with traditional programming. In traditional programming, the developer can get more defined information about the system the program communicates with and how it communicates with it, whereas in RPA it is more abstract due to the visual nature of the communication.

Another disadvantage that was found when using UiPath’s solution is the fact that some instructions which are prevalent in traditional programming languages are not available. This requires the programmer to solve the problem in a more cumbersome ways, which may be more complex or not be the usual solutions used in traditional programming. An example is the “continue” instruction, which stops executing the current iteration of a loop and moves on to the next iteration. This shows the lack of maturity of the RPA tool.

2.5 RPA Tool Features Explained – How They Deliver Benefits

This section presents a general analysis of RPA tool features and relates them to the benefits associated with RPA automations, according to a number of categories: Usability; Implementation; Deployment Flexibility; Security; Scalability; Extensibility.

Note on RPA Nomenclature

Advisors have reported the occurrence of “RPA washing”, which is when companies put more effort into advertising and marketing than developing the automation tools. Some claim to provide more RPA capabilities than they actually do (Lacity & Willcocks, 2016).

RPA tools from different vendors may have different names for similar components and a lot of different terms are introduced by the RPA vendors. There is even a different term used for RPA, such as SPA, meaning smart process automation. When it comes to RPA tool features and capabilities, different terms refer to similar concepts. For example, UiPath calls its’ central control room component “UiPath Orchestrator” and, WorkFusion calls their own “Control Tower”.

UiPath’s solution names the developer tools “UiPath Studio”, the control room is called UiPath Orchestrator and the software robot is called UiPath Robot.
RPA Tool Interface Usability

In general, better usability contributes to speeding up the RPA robot development process. It also increases the quality of the produced automations, easing the management of the automated processes.

From the current learning experience, using UiPath’s RPA tool, as a robot automation becomes more complex and bigger, it gets harder to manage and understand the already defined job steps. Errors are harder to pinpoint and fix. Navigating the automation is more confusing. A good user interface contributes to reducing these problems impacting automation development quality and speed.

Interactive Diagrams

A good visual representation of the steps of an automation will help with understanding what is programmed for the robot to do. Unlike traditional programming, in which more effort is required to remember where “that part of the code that implements a certain business logic” is, RPA tools with a good user interface will reduce this strain from the programmer. The developer can focus on the more important part, which is designing the automation in such a way to optimize the business process and thus deliver business efficiency.

The previous visual representations of the automation steps should also encapsulate the underlying code to an appropriate degree of abstraction, facilitating the use of the tool to business users (as can be seen in Fig. 41). Also, the abilities to zoom in, zoom out, hide and expand diagram steps help visualizing and editing the automation step, for when changes are required because the robot failed.

Drag-and-Drop

Drag-and-drop is a feature that allows users of the development tools to click an action or step from a menu or in the automation, then, while holding the mouse button, drag the selected action to the desired location and drop it by releasing the mouse button. This functionality enables visual programming, which makes RPA tools more accessible to business users with basic programming knowledge. It is valuable that the business users can program the RPA automations themselves, since efficient automation of business processes requires subject matter expertise. The people exposed to those business processes for a longer time are more likely to find improvement points in the business process, which makes the RPA automation more efficient.

Configuration Assistance

Configuration assistance reduces the learning curve. This includes simple things like displaying information about what an action does when the action is selected in the menu or small texts that pop up when the mouse is hovered above the configuration options of a step in the automation. Also, displaying the type of input or output a step produces, improves the configuration of steps (actions). (Brain, 2016)

Expression completion is another feature that makes RPA tools more accessible to the less technical users, by suggesting valid expressions to finish what the user started to write.
Wizards

Configuration wizards for certain robot actions, provided in tools such as UiPath’s, guide the user in configuring the robot action. Some actions may be more complex and require more configuration settings than others. The wizards guide the user through these settings, which contributes to lower the learning curve and the user is less prone to making errors. For example, one of UiPath’s wizards helps with data extraction from web pages.

Debugging Information

Well-presented debugging information increases quality, since errors will be identified more easily and fixed faster. (Brain, 2016) Real time compiler warnings help prevent errors during development and other warnings make it easier to identify errors when using the tools.

Implementation

It is important to produce RPA implementations fast and without errors to reduce costs and deliver benefits to business earlier. The following features aid implementation efforts.

Recorder

The recorder assists implementation by capturing a sequence of user actions, from which it builds a basic skeleton for the overall process automation. The recorder isn’t always reliable but further refinement can be done, since the recording introduces steps to the business process automation diagram that the programmer can later configure. Recorders increase implementation speed, which reduces costs.

Element Identification and Screen Scraping

RPA robots have to identify user interface elements, such as buttons, data input fields and information (preferably structured). More developed tools require less configuration to achieve correct element identification, which speeds up RPA implementation. It also supports robot resilience to changes in the business process. This means that if an application, which the robot has to interact with in the business process automation, is updated and elements containing the data manipulated by the robot change position in the screen, the robot still knows where to get the data from. Robot resilience to change reduces automation maintenance costs.

The term screen-scraping refers to the ability of a computer program or software robot to extract information from the screen, at the presentation layer level, meant to be read by humans. This includes identifying a button by its image or copying a string of information by identifying the string that is adjacent to it, the latter also called text scraping (example of string format for text scraping: “some label: desired information”).

There are several techniques to achieve screen scraping and element identification, in different situations. These can be split into two main categories. The first category relies on code (generally faster and more efficient) and the second relies on image.

Starting with the first category, one technique is when the RPA tool associates a screen element with a name that supposedly never changes or changes very few times. In web applications, this hap-
pens through parsing HTML code, possibly generated by a UI framework, which means the HTML has a predictable structure, so information can be found consistently. In this situation, even if the UI element changes position in the screen, the RPA tool is able to find it again, because it is still associated with the same HTML name. This name can be a unique identifier or a set of attributes that identify the element uniquely.

Fig. 1. Example of Web Page Button Selector for Google Chrome Browser Using UiPath

Another is using RPA tool-specific connectors and commands that communicate with commonly used desktop applications, such as Microsoft Excel. These connectors are abstracted by the set of actions that can be drag and dropped into the automation sequence, in the development tools. In Blue Prism, these connectors are called a Visual Business Object (Chappel, 2010). In UiPath they are called activities, which can be downloaded in packages.

Now to describe the second category, which relies on image. This is most often used when it is not possible to acquire the data through the previous methods, since it tends to be slower and less resilient to change. The first technique may be called surface automation. A region of the screen, containing the desired user interface element or information, is associated with a name by the RPA tool. Then the robot knows which UI element to interact with. If the image changes location, then the automation no longer works.

To solve this, another technique called advanced image recognition can be used to dynamically search for and identify the image contained in the previously selected region of the screen.

If the information is not accessible through text, but as an image, then optical character recognition (OCR) is required (for example, a .jpeg extension image of a paragraph or a copy-protected pdf file). OCR compares bitmap data from the screen with previously processed information to understand what text content is written and converts that content to a string that can be processed by the robot. (Brain, 2016)

Image recognition also has to be used when the robot needs to interact with virtualized desktops, using remote desktop protocols, or Citrix, because the robot cannot access the application code. Alternatively, the installation of software on the server side of Citrix or RDP may provide access to the code, but it requires this extra work. (Lowes, et al., 2015)

Release Management and Version Control

Release management and version control are features that support implementation speed and prevent quality issues. For example, if the robot is executing an outdated automation version with older business rules, it may insert errors into the information systems it interacts with, affecting quality.

Blue Prism’s Process Studio allows comparing the old version and current version of the process being edited side-by-side (Chappel, 2010).

UiPath does not have a built-in release management and version control system but it provides integration with third party tools, such as Team Foundation Server and GIT.
Debugger

Debuggers that provide clear and insightful information about what is happening during process execution accelerates the implementation, since bugs will get fixed faster. The developer does not have to bother printing variable content one by one, since this information will be displayed in the debugger.

The more advanced RPA development tools may have a “slow step” functionality, like UiPath. This allows the programmer to step through the automation steps, watching where the robot is clicking and other actions it is performing. It is a useful feature, considering robots may perform actions in less than one second.

Step through, breakpoints and viewing the contents of variables are common functionalities, included in the debuggers. This is included in UiPath’s and Blue Prism’s RPA tools (Chappel, 2010).

Blue Prism, unlike UiPath, allows the programmer to make changes to the code dynamically, while executing the automation in debug mode. This is a useful feature since a lot of time is spent stopping the execution, resetting the initial conditions for the execution and testing of the automation (for example data that is expected to be found in a certain stage of the user application) and running the automation again to find the bugs, which may not show up consistently in every execution of the automation. (Lacken, 2017)

Deployment Flexibility

The robots can be deployed on the cloud or in premises. They can also be deployed to work in attended mode, in which the robot is essentially working side by side with the human and requires some monitoring. Alternatively, it can be deployed in unattended mode, in which the robot works independently of the human worker and minimal human intervention is required to execute the tasks.

The ability to deploy RPA on the cloud, in-premises, attended or unattended have impact on costs, agility and efficiency. More detail is provided about each deployment model.

Cloud or In-Premises Deployment

Most RPA solutions can be deployed to the cloud. This is advantageous because it enables scalability, as each robot often requires at least one machine or virtual machine to work with. Virtual machines can be deployed when an extra robot is necessary. However, some tools do not require a dedicated desktop or virtual machine for the robot to work. One such tool is Kofax Kapow.

Also, deployment to the cloud allows fast deployment of Virtual Machines configured in the same way, with same applications installed, same windows properties and so on. This is important because if the robots are not working on similar desktops, then when a change to a business process happens, it is likely that more than one automation will have to be changed.

Attended Deployment

Deploying a robot to a machine that is being used simultaneously by that robot and a human employee is called attended deployment. In some cases, it is possible to have the robot working without interfering the human, since some tools like UiPath’s allow the robot to work in the background, if compatible actions are involved. But in general, the actions performed by the robot will require win-
dows to open, hence temporarily impeding the human from carrying on his work. Nonetheless, the robot is still faster than the human doing certain tasks, so the interruption is not hurting overall efficiency. In general, the employee benefits from interacting with the software robot while performing work.

Attended deployment is a good option when the business process is too complex to automate, when it has too many rules or involves handling information that the robot is not capable of interpreting. Instead of implementing all the business rules or tasks, part of them are carried out by the human employee while the tasks that are easily automated by RPA technology are done by the software robot.

Consider the scenario in which a customer goes to a bank for a loan. Because the robot is able to provide consolidated information faster and with less effort than the human worker, the clerk asks the robot to aggregate all the data relevant to determine if the client is likely to repay the loan or not. This improves the quality and speed of deciding to grant the loan or not. Also, while the robot is collecting the information, the clerk may focus on doing tasks the robot cannot do, such as interacting with the customer to acquire the remaining information needed to complete the process.

Some tools, like UiPath, allow implementing user events into the automation, such as the human clicking a UI element. These events trigger the robot to perform a specified sequence of actions when the conditions of the event are met.

Unattended Deployment

Robots have their own dedicated machine to produce work in unattended deployments. No human intervention is required during the execution of the business processes, except when unexpected errors or situations occur. In such situations, the robot may be programmed to flag the exception or error to be handled by a human employee. Exception management features, such as alerts, logs or redirecting the pending business process to a human are key to enable unattended deployment.

The RPA system manager is able to see the results of the business process executions from the control room component. If there is any problem with the robots, the system manager must investigate and solve the problem so that the robot is operational again.

Unlike attended deployment, unattended RPA deployment only requires human intervention when exceptions are found and flagged to be resolved by human workers. (Brain, 2016)

Security

Since RPA brings a higher return on investment when applied to business processes with a high volume of transactions, it is important that the data and messages exchanged are secure, encrypted. Security is essential for compliance with regulations in most industries. Leaked data can also lead to loss of concerned customers, so security should increase customer satisfaction. Data can also be tampered with when being transacted so security measures help keep the quality of the automated services.
**Safe Credential Storage**

Safer RPA tools provide a safe credential storage. This feature contains the secure credentials used in the automated business processes. It distributes the login information only to the robots that require it to execute business processes. Some RPA tools allow to configure rules to handle the stored passwords. One example is changing the password every time it is used by generating a new password that complies with the organization’s security standards. This adds a dynamic layer of security to the business process execution.

This is a very important feature, so the RPA tools that do not provide credential stores may simply store them encrypted in a database. For example, UiPath stores the credentials in the Orchestrator with 256-bit encryption (AES_CBC_256 encryption algorithm) (UiPath, 2018). They may also provide the ability to integrate with a third-party credential manager. Ideally, the credential storage is part of the RPA tool itself to access the features provided by credential managers faster. (Brain, 2017)

**Textual Action Logging**

Enterprise-level RPA tools keep activity logs of every action, if the feature is activated. It allows auditing the system and the business process executions. These logs contain the actions carried out by the robots and their timestamps. The logs are multipurpose, as they can also be used for analytics, so it is possible to configure their structure. It is also possible to define which information is shown in these logs, as some of the data being processed by the robots may be sensitive and access may only be provided to certain people.

**Visual Action Logging**

Besides textual formatted logs, it is possible to save visual logs, in other words, screenshots of the robots’ activity. These images are a nice complement for compliance validation and can also be used for handling exceptions and debugging (Brain, 2017).

UiPath has the “Save Image” activity, which is useful for when exceptions occur and these need to be reported to a human to resolve, since the robot might not be able to capture data from the elements or screen, which can be due to UI complexity or UI animations that make it hard for the RPA software to capture data displayed in a more interactive or animated way. For example, graphs designed for business intelligence, which display information in an interactive way, may use elements from which data is hard to grab.

**Role-based Access**

To reduce the possibility of an internal attack, safeguarding continuity of business operations, roles are assigned to the employees by the RPA system administrator. Each person in the RPA team is attributed login credentials which may restrict his actions in the RPA system.

For example, one or more people may be responsible for one business process, so one environment can be created for that business process and only that person or group of people have access to that environment. With an environment configured, only authorized automations and robots can be deployed to that environment. (Brain, 2017)
Another example is the creation of tenants for access to the control room component of the RPA tools.

**Scalability**

Scalability features provide service efficiency and quality, also increasing customer experience since the requests are answered faster and with less errors. Cost reduction can also be achieved through proper resource management.

**Scheduling**

The control room component of RPA tools allows scheduling certain jobs or automations to be executed according to defined triggers. It is common to be able to set defined, repeatable time spans against a variable number of resources. More advanced tools allow the user to set more complex triggers, based on events.

For example, using UiPath, the user can create new schedules, tailored according to process, environment, trigger, repetition time cycle and stopping condition. The trigger is a rule that decides when a robot should start executing a process automation given a time and date set by the orchestrator (control room component) user. But it doesn’t allow more complex events to trigger the automations. Nonetheless, more complex rules to stop an automation can be programmed in the developer tools component of the RPA product.

The repetition time cycle is how often the robot repeats execution of the process automation, for example, every three days. The stopping condition is also a date and time for the robot to either cancel or terminate. There is also a tab called “execution target” which allows the user to choose which robots perform the automation or the number of robots that perform the automation, in which case the orchestrator assigns the execution tasks to the robots automatically. If a robot is already executing a job when the trigger is activated, then the job that is scheduled has to wait for the robot to finish its’ current executing job, by being placed in a queue (this information is related to UiPath’s RPA solution).

In business, especially customer service, where timely response to a customer could be the difference between customer retention and customer loss, certain tasks have precedence. Scheduling alone cannot respond to these more urgent needs, so work prioritization features are necessary to properly manage queues and respond to the business needs with agility. Work prioritization allows to classify and order incoming tasks depending on their importance to the business. Hence, work prioritization features contribute to customer satisfaction and business agility (responsiveness).

**Queuing Workload**

RPA tools frequently provide queuing methods to manage workloads. Items can be inserted to the queues with different levels of priority, which the queue system uses to sort the execution of business process instances or related tasks, delegating resources to finish important ones first. The queue can be manually ordered by classifying the inserted items, using tags and labels with a certain hierarchy, so that the robots know which ones to execute first when deployed.
The best RPA tools allow the user to do this from the control room component or through the programing of automations in the developer tools component. With the latter case, the robot can sift the incoming workload and tag the tasks for the next process execution based on programmed rules. For example, the user can configure it to sort the tasks by due dates, price, or quantity.

**Resource Grouping**

There is yet another feature to help handling urgent tasks. Resource grouping assigns a queue of work to multiple robots at the same time, having them work together to finish it faster. The work is done in parallel by a group of robots instead of sequentially by a single robot. Resource grouping also helps in responding to peaks in workload at a certain moment. (Brain, 2017)

**Load Balancing**

Process load balancing is not available in every RPA tool, but it enables those with such capabilities to dynamically control the usage of the available robots. When configured, the automation can adapt to different levels of demand. This avoids unused resources as one robot typically requires one virtual or physical machine to work.

**Extensibility**

RPA tools often provide integration with third party tools to make up for weaknesses and lack of features, such as text analysis or security tools. This extends the tools’ capabilities. Also, for robots to be more efficient the tools include integration with most commonly used applications, such as Microsoft Excel, databases and email.

These integrations are implemented by steps that the user can drag and drop to the automation sequence. This way, the user doesn’t have to develop a robot that does common tasks, for example, looking in the desktop for an Excel file, opening the file, then clicking a cell and inserting data into that cell. More detail is provided about Excel interaction capabilities below.

It is also possible to extend the RPA tool by programming packages of steps or downloading packages programmed by other people into the tool. For example, in UiPath these packages are managed from the developer studio. Extensibility features such as these contribute to lowering robot implementation time, increases the quality of the implementations and makes it easier to manage the automations.

**Code Extension**

There are a few ways to extend RPA tool capabilities using code.

First, creating custom activities, which are then embedded into the developer studio and can be used in the automations just like any other activity. To exemplify, in UiPath, a NuGet package must be built outside of UiPath’s studio, then that package must be installed from within UiPath Studio’s package manager and the activities that were programmed in that package are available in the studio.

Second, directly programming in the automation. UiPath does this with the “Invoke Code” activity, which opens a small window where the user writes VB.Net code that is then executed by the robot.
when the automation is running. UiPath only allows expressions to be written in Visual Basic .Net. No other programming language is available for now, except if an external script or program is invoked.

Lastly, invoking external code, such as PowerShell commands or applications. In UiPath this is done through the “Invoke Power Shell” and “Start Process” activities. The latter requires the user to specify the executable path and the arguments to be passed to the application. (Brain, 2016)

**Excel**

In UiPath, it is possible to create dataTables and insert them into a specific cell from an excel sheet. The way the robot works is by opening the sheet (unless it is already open), it then finds the specified starting cell and pastes the dataTable to that cell. Any data that was in the cells that are covered by the dataTable starting at the initial cell is overwritten. No rows or columns are inserted, the cells are modified in-place and the file is immediately saved.

There are several operations available, such as filtering and sorting a table, executing excel macros, reading cell formulas and others.

**Email**

In UiPath, it is possible to send mail messages using SMTP, getting POP3 mail messages. UiPath provides integration with Outlook mail application. When it is configured to send an Outlook email, it opens the Outlook application, authenticates with the provided credentials, sends the email and then closes the Outlook application instance. It is also possible to save email attachments to a specified folder and to save a mail message.

**Database**

Using the activities for database integration in UiPath, it is possible to connect to a Database, execute update, insert, delete commands and queries. The user can also disconnect from the database. The “Connect” activity provides a wizard which helps the user in establishing a connection string.

Illustrative images of the wizard can be found in the annex, **Fig. 37, Fig. 38** and **Fig. 39**.

**PDF**

UiPath allows two operations to be performed on PDF files. One is reading the text from the PDF file and another is processing the PDF file with OCR to extract its’ contents, so that it can be processed in the automation.

**Google Text Analysis**

For natural language recognition, UiPath provides the ability to connect to Google Text Analysis services. As input, this activity step requires a key and a service url to access the service. Also, the input text must be passed as an argument to the activity. Currently, this service only supports English text analysis, so if the project requires analyzing text in other languages, the text first has to be translated into English.
2.6 UiPath Concepts

UiPath wraps basic programming concepts into more abstract, higher-level concepts, easier to understand and to apply by business users. This abstraction is exemplified by Fig. 3, which shows how instructions programmed for the robot to follow are represented in UiPath Studio. It also benefits technical users, allowing them to focus on the overall solution, instead of having to constantly deal with more detailed code, contributing to the efficiency of the automation and to the speed of implementation.

Therefore, it is important to understand the particular concepts of UiPath to better understand how RPA is implemented and deployed. An explanation of the most important concepts is provided in this subsection.

Activity

An activity corresponds to an instruction. It programs the robot to perform a certain action, for example, clicking a button, creating a folder, opening a file and many others. A “sequence” is the name given to a succession of activities, which are executed by the robot (Fig. 3 shows a sequence in UiPath Studio).

An activity belongs to a package and has a set of properties. The set of properties varies from activity to activity. The set of properties is further divided into subsets of properties, which can have names such as: “common”; “input”; “output”; “misc”; “options”. The activities do not necessarily have the same subsets of properties, as some activities may not have the “output” subset of properties, for example. This means the activity does not produce any output value to be used by the programmer. Fig. 2 shows the activity panel and the properties of an activity, in particular the subset “input”.

Some properties have a default value. The value of a property can be of many different types. For example, the value of an input property can be of the type “string” or it can be of the type “int”.

By default, the different activities available to the RPA engineer are displayed in UiPath Studio, inside a panel on the left side of the screen, and the properties are displayed inside a panel on the right side of the screen (after an activity is selected).

So, activities are chosen from a package that contains somewhat related activities, which tell the robot to perform a certain action, according to the configuration of the properties of the activity. For example, there is the package UiPath.Excel.Activities.

A sequence of activities implements a robotic process automation.
Some activities are straightforward, and it is possible to understand what is the behavior of the activity according to its’ name, however some activities such as “Retry Scope” require some explanation. Now follow brief descriptions of such activities.

“Element Exists” Activity
This activity will verify if the UI Element indicated by a selector exists on the user screen or not. It will return a Boolean value, true if the UI Element can be “seen” on the screen by the robot and false otherwise.

This activity is not so easy to configure if the user is not familiar with UiPath’s selectors and the way UiPath identifies UI elements. A useful feature for learning about selectors and understanding them is UI Explorer (more detail about this feature below).

“Retry Scope” Activity
This activity will retry the sequence of activities that it contains until a certain condition is met (as seen in Fig. 4). To program the condition, the RPA developer must use an activity that produces a
Boolean value as output. The number of times the contained sequenced is retried is specified by the programmer by configuring the activity’s properties.

![Retry scope activity](image)

**Fig. 4.** Example of “Click” activity contained inside “Retry Scope” activity and condition testing with “Element Exists” activity

**Selector**

A selector is a string with an XML format used by the UiPath software. It allows the robot to identify which UI elements to interact with, since it provides information such as the application in which the UI element is presented or such as unique webpage element identifiers (for example the id attribute of an HTML element).

Selectors are used in most activities since the core functionality of UiPath’s software is interacting with UI elements. However, some activities do not require a selector, such as activities that only interact with the UiPath Orchestrator component. One example of those activities is the “Get Transaction Item” activity.

There is a wildcard character “*” that can be used to match one or more characters. This allows for implementing automations that are resilient to changes and which work successfully for different instances of the same business process. For example, the selector (1) will only work for the business process instance corresponding to the request registered in “PERIT/2018/303”.

```html
<html title='EDOCDEMO - Distribuição PERIT/2018/303' />
<webctrl aaname='Terminar' parentid='actionsContainer' tag='BUTTON' />
```

(1)

If instead we replace the previous string with the wild character “*”, resulting in the selector (2), then the robot will be able to process any other request with a different identifier that is registered on the web application.

```html
<html title='EDOCDEMO - Distribuição *'/>
<webctrl aaname='Terminar' parentid='actionsContainer' tag='BUTTON' />
```

(2)

**UI Explorer**

UI Explorer is a feature that allows the user to click any element on the screen and see the corresponding UI element selector in its’ XML representation. Using UI Explorer, it is possible to edit selec-
tors’ attributes, so that the robot is more resilient to changes. The tool also allows viewing the hierarchy of UI Elements displayed at any moment. This feature enables more advanced programming of the robot for situations in which the automatically generated selectors are not so reliable. A lot of experience is required for understanding and taking proper advantage of this feature. **Fig. 5** demonstrates the information that can be visualized regarding selectors and UI Elements.

![Image](image.png)

**Fig. 5.** UI Explorer Showing Different Panels: 1 – Current Selector; 2 – Selector Attributes for Editing; 3 – UI Elements Hierarchy.

**Queue**

Queues contain queue items and transaction items. These are key concepts for managing workloads and monitoring the activity of robots. Proper usage of these methods in the automations will enable efficiency, agility, quality and compliance. It enables efficiency and quality because if an instance of the business process results in an error, it can quickly be retried and executed again from the orchestrator component. It enables agility and compliance due to the fact that the RPA system administrator can quickly examine how many business process instances failed and manually order the robot to retry them from the orchestrator component.

**Queue Items**

Queue items are a data structure with relevant attributes for managing workloads and pending tasks, which can store a collection of information (a string dictionary) defined by the RPA programmer. It is possible to create a queue item and add it to a queue in the Orchestration component by using the activity “Add Queue Item”. Among other properties of this activity, the most relevant are:

- **Deadline** – the date limit for which the item can be processed;
- **Priority** – a value from the set: low; normal; high;
- **QueueName** – the name of the queue in the orchestrator in which the item is added;
• Reference – a reference to the item. It is possible to choose in the queue configuration in the orchestrator if all of the items’ reference in the queue is unique or not.
• Item Information – the property which receives the string dictionary.

When a queue item is first added to the queue in the orchestrator, using the previously mentioned activity, the item’s status is “New”.

Transactions (or Transaction Items)

Transaction items is the name given to queue items which are already “In Progress” or which have any other status different than “New”. The status that a transaction item can be in are:

• In Progress – a queue item is assigned this status when it starts being processed;
• Failed or Successful – the programmer defines when an item failed or succeeded by using the activity “Set Transaction Status” to change it to one or the other;
• Abandoned – the automation may fail, and the queue item is no longer being processed, so after a certain time its’ status changes to abandoned;
• Retried – if an application exception occurs, the robot tries to execute the automation sequence again, so the status of the queue item is changed;
• Deleted – the status attributed to an item after being removed manually from a queue, using UiPath’s Orchestrator.

(UiPath, 2018)

Screen Scraping and Data Scraping

UiPath distinguishes screen scraping from data scraping. The screen scraping wizard is designed to extract text in UI Elements which are not data tables and the data scraping wizard is designed to extract data contained in data tables. So, screen scraping refers to when the text contained in a single UI Element is extracted and data scraping refers to when the text contained in a data table (a group of smaller UI Elements, the cells) is extracted. The previously mentioned wizards guide the user in programming the robot for extracting data, going step by step to configure it.

2.7 Business Processes and RPA

There are a few factors related to the business process which will be automated by RPA technology that make it more likely for RPA to deliver benefits.

RPA is better applied to rule based well defined processes instead of judgement-based processes. This means that the logic behind the decisions should be simple and straightforward. The rules must not be ambiguous.

The data handled during the business process should be as structured as possible. Semi-structured data can still be manipulated by the robots, but unstructured data is simply out of reach. If processing unstructured data is necessary, it must first be parsed by other tools and then passed to the robot. Spreadsheets, databases, CSV files are examples of structured data. Invoices, purchase orders and
contracts contain semi-structured data. Examples of unstructured data can be images, video and voice.

Processes with a lot of exceptions will increase the time to implement the automations, so it is preferable to choose processes that are standardized. Also, to keep implementation time low and reduce maintenance costs, the chosen process should be stable and change as little as possible over time, since changes to the process are likely to introduce the need to reprogram the robots. Changes related to the sequence of tasks to be performed and structural changes to the systems the robot interacts with require reprogramming of the robot.

The benefits delivered by the automations are higher if the current overall executions of the target process tend to result in a lot of human errors, since robots are expected to perform the tasks always in the same way. In other words, if a business process often results in error and has to be executed again, then it is a good target for robotic process automation.

Since RPA can centralize the execution of business processes through cloud deployment and centralized management of the robots, there will be cost reductions if the business process is currently decentralized. (Capgemini Consulting, 2016)

A very important detail about the business process is how often it is executed. The effort in automating the process should be worthwhile and the higher the frequency of execution of the business process, the greater will be the return on investment. This is because the robots should take less time to execute the business process than it currently takes.

**Framework for Choosing the First Target Business Processes**

At the beginning of an automation project that uses RPA, the technology should be applied to simple tasks and gradually moved on to more complex tasks. The latter will involve more rules and exception handling, increasing the automation’s scope (Kroll, et al., 2016). If possible, and to showcase the benefits of robotic process automation, the chosen business process should involve large volumes of execution.

One simple framework for choosing target business processes for robotic process automation is identifying which business processes have been automated previously, in other situations. One way to find these is through case studies. If a similar business process has been automated before, there will be more chances of success and of finding support and help if necessary. This will also immediately identify a few possible processes within the environment at hand.

Next, evaluate the business process being considered for automation according to the factors presented previously. It is also worth considering which kind of applications the robot will have to interact with, since it is more efficient to interact with some applications than others.

More precisely, virtualized applications are generally harder to automate, because the robot cannot identify the underlying elements in the user interface through code, so it will have to rely a lot on image recognition or fixed positions of the UI elements. This means, for example, that if the image changes location or the text to be extracted changes to a location with a different background, the robot must be reprogrammed.
Web applications follow up next, since these involve accounting for page loading and refresh times. Also, web applications may have more interactions than desktop applications. A higher number of interactions or user interface animations generate a higher number of scenarios. The robot will have to be prepared to recognize different elements of the user interface for each scenario.

Processes susceptible to change are not a good initial target either, because if the process changes, the instructions given to the robot require modification. It may also hinder the learning process of the robot developer.

**Process Types and Useful Features**

Following are some characteristics of business processes that probably require or benefit from the corresponding RPA tool features.

**Simple (Standardized)**

Small number of steps and little exceptions. These processes should be straightforward. Using a recorder is a good feature to speed up the implementation and deliver benefits with less effort. Automation of simple process tasks that are performed very often is worth the little effort enabled by recorders.

**Complex**

Big number of steps and few exceptions. To benefit from automating such processes, a low-volume of transactions is enough. Exception handling features, such as alerts sent from the control room are beneficial to ensure business continuity and quality of service. Features such as transactions that allow retrying a certain part of the automation are also very handy, since only part of a long process execution has to be repeated.

**Highly Regulated**

Processes in financial services sectors for example, or where different authority levels may be required for accessing information and applications. These processes benefit from RPA features such as action logging, auditability features, extensive access management controls to make sure no one is modifying robots that are not their responsibility, safe credential storage and encrypted use of credentials.

**Only Native Applications**

Process involves only usage of applications in the desktop. These processes are fairly compatible with RPA tools, since it is easy for the tool to access the underlying elements of the user interface. Still, features like integration with excel sheets can be beneficial to these projects, speeding up the implementation.

**Involving External Web Applications**

Process requires interaction with web applications. Automating these processes is a bit harder than automating processes that involve native applications only, especially if the web pages that the RPA
tool is working with are very interactive. Compatibility with web browsers is a must, otherwise the automation implementation will be much harder, since with web applications the analyst has to be concerned about identifying the web page structure and understanding it, in order to develop automations that are resilient to changes. OCR and data scraping features are important, just like the number of methods the RPA tool provides for the robot to identify the elements (image recognition, element selection and anchors).

Data Migration

Processes where it is necessary to access databases and provide credentials to access information. For these processes, it is helpful if the RPA tool comes packed with database integration and with credential management features or if it provides integration with third-party credential managers.

Involving Virtualized Desktop

Processes in which a virtualized desktop is used to execute the business process. These processes are the hardest to automate, since RPA tools are not able to recognize the elements inside the virtual machine. So, it must use image recognition features in order to know where to click. Citrix integration is useful for these situations.

Financial Services

Current Situation and Challenges

The financial services industry face many obstacles. There are regulations such as taxation, Foreign Trade Policy and others, which if not complied may bring very high fines amounting up to billions.

Anti-money laundering (AML) and know your customer (KYC) business processes are very costly. Organizations in the industry must ensure compliance with AML. There are huge fines associated with these. According to (Thomson Reuters, 2016) financial firms spend 60 million American dollars per year on KYC, customer due diligence and client onboarding business processes. KYC compliance teams can go from one hundred and fifty to a thousand full-time employees.

The business generates and must handle heterogeneous semi-structured data corresponding to invoices, bills, declarations, certificates and letters.

Each day, there are around thousands of transactions to complete, which means high-volume processing is required. Some of these tasks have very short timespans, having to be completed within the day and demanding efficiency from the business operations.

Moreover, some of the processes are complex, involving delayed collections, loans, different banks and different rules for different countries. Multiple large-scale and expensive information systems are in place to support the financial institutions, deterring system integration efforts. There is lack of automation, which makes the financial services a prime target for RPA solutions.

RPA Automatable Tasks in KYC

Some tasks in KYC are good targets for RPA technology:
• Setting Up Customer Data – consists of manually entering data into a Customer Relationship Management (CRM) system. The necessary information comes from a government-issued identification document. The document format is standardized, varying only for different locations. RPA could identify the customer and enter his or her information into the CRM.

• Validating Existing Customer Information – RPA could validate customer information by accessing databases, extracting information from documents, social media, merging data from different sources and filling in forms.

• Customer Information Collection – Information regarding the customers’ credit worthiness, business he or she may be involved in and identity is required to be collected by regulations. It must also be updated regularly and managing this is challenging. The information can be available from received papers and pdfs or other public sources, such as databases. RPA could be used to perform this search, extraction, aggregation and storage of information.

2.8 RPA Vendor’s Product Main Strength and Preferable Usage Context

This section highlights the main strengths of each vendor’s product and the contexts in which they are more appropriate. The information is adapted from a report on RPA products (Le Clair, 2017).

Leading Vendors

Automation Anywhere is best suited for shared services and back-office processes. The product stands out for overall ease, duration, and cost of deployment. However, it is not so accessible to users without programming knowledge and skills.

Blue Prism focuses on the regulated industry, providing strong features related to security and the control room, such as workload management, encryption and auditing. Not having a recorder to aid implementation and not so polished usability are a couple of drawbacks.

UiPath provides an open platform, which makes it a good option for first testing RPA technology. The control room functionalities it provides are strong enough to support more than just initial proofs of concept, allowing any organization working with UiPath to deploy robots on a larger scale, efficiently automating business processes. The wide range of possible integrations with third party tools, such as Kibana, CyberArk, Google Text Analysis, Microsoft OCR and others, constitute a versatile architecture. One drawback is that robot licenses still must be purchased for commercial use.

Other Vendors

WorkFusion focuses on the financial services industry. They provide another open platform, with its’ RPA Express product, which can be used for commercial purposes. It is strong when it comes to RPA analytics features.

Kofax’s Kapow is adequate for data aggregation using web interfaces. It is slightly different from the rest because of its’ architecture, which does not require robots to be installed on a desktop or virtual machine. The robots are placed in a single centrally managed virtual machine, without need to update individual robots. However, its’ overall capabilities score is one of the weakest. (Le Clair, 2017)
In this project, more attention has been given to: Blue Prism due to its’ focus in the regulated industry; WorkFusion because of its’ focus on the financial services industry; UiPath since it is more accessible and provides extensive learning resources, along with good capabilities. That is why not all the vendors are summarized in this section, but their scores in terms of capabilities are presented to provide context.

Table 1. RPA Tool Features Score (Ordered From Highest to Lowest) Adapted From (Le Clair, 2017)

<table>
<thead>
<tr>
<th>Robot Development</th>
<th>Control Room, System Management, Reporting and Resilience</th>
<th>Deployment, Governance and Security</th>
<th>Cognitive Intelligence</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>UiPath</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Automation Anywhere</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>NICE</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>EdgeVerve Systems</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Blue Prism</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Kryon Systems</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Pegasystems</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>WorkFusion</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Redwood Software</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Contextor</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Softomotive</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Kofax</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Color Code: Weak: 1, Average: 2, Strong: 3, 4, 5, 6, 7
3 Methods

3.1 Prototype

During the study to understand RPA, a prototype was developed using the UiPath tool. This allowed to better understand the capabilities and limitations of the tool. It helped learning how to use some of its' features and which ones stand out RPA from traditional programming. In this section, a brief description of the prototype is provided, along with some of the lessons learned.

Prototype Automations

The following tasks were automated:

1. Login to web application;
2. Login to web application, check if a UI element exists and send email depending on previous check result;
3. Read text file from desktop's folder, process string from text file, enter data fields and submit form on web application;
4. Login to web application, extract data using data scraping and OCR, create data table from the extracted data and saving the data table to an excel sheet;
5. Login to web application, enter data fields, submit form, take screenshot and save to desktop's folder.

The first task, a simple login, was fast to implement using UiPath's recorder. The recorder generated a sequence of steps which was further configured. This method was later used in the project to implement another simple login.

The second task reused the automation developed in the first task for logging in, by drag and dropping the automation file from the project folder to the automation diagram sequence, just as if it were any other activity. This is very good for reusing already implemented automation components but passing arguments to the invoked automation is not so simple and adds complexity to the automation that is currently being implemented. This knowledge was later useful for organizing the project in a modular way, avoiding passing arguments from one automation sequence to another.

Sending email is done through integration activities, that allow sending and reading email from applications like Outlook or sending by SMTP protocol and reading through POP3 protocol. One issue that arose was hiding the password, which I did not manage to do with the Outlook activity. Being aware of this issue was important for deciding if Outlook would be used to read emails in the project.

The third task presented no challenges, however it contributed to learning how to implement web application automations that submit forms, or which perform data entry. The project involved similar tasks to this one, such as applying search filters and copying data from one web application to another. The recorder feature was used for this and proved to be a relatively fast way to automate such tasks.
The fourth task was challenging due to data scraping and saving that data in the excel sheet. The data scraping wizard provided by UiPath Studio is quite flexible. It allows extracting data spanning multiple pages and it allows selecting three different extraction methods: native, fullText and OCR, each with different characteristics (UiPath, 2017). The user can also choose the format of the data table to where the information is extracted to. These features helped implement the project as it required extracting data from tables in several pages and using data from an excel sheet which had to be downloaded from a web page.

Implementing the last task of the prototype proved useful during the project implementation when it came to debugging and fixing errors. Saving a screenshot when an exception is thrown provides insight into what the robot did wrong if the robot is working unattended. The screenshot is taken using the “Save Image” activity.

3.2 Project

Target Business Process
For this project, an insurance business process from a liability investigation company was selected for robotic process automation. This company investigates liabilities on request from an insurance company. There are two important entities participating in this process: the insurance company (external) and the liability investigation company (now designated as L.I. company since its’ business process will be automated). This business process was taken from a real case.

The insurance company owns a web portal in which it delegates liability investigation requests to its numerous suppliers, one of which is the L.I. company. The L.I. company also uses this web application to communicate the investigation result to the insurance company, completing its’ service. However, some of the more urgent requests are communicated to the L.I. company by email. To support this business process, the L.I. company has a database in which it stores specific liability investigation data and Edoclink, which is a workflow web application that works as a document repository and supports the processing of the documents.

Parts of the Process to be Automated
The business process was split into three parts for better implementation modularity and for enabling increased agility when managing the work of the robots from the orchestrator component. These are the parts:

1. Checking in New Liability Investigation Requests from Web Portal
2. Checking in Liability Investigation Requests from Email
3. Delivering the Liability Investigation Results

Checking in New Liability Investigation Requests from Web Portal
The process, as it is, requires one employee to check regularly the insurance company’s web portal for new liability investigation requests. If new requests exist, the employee proceeds to create a new
process, manually copying the data regarding the request contained in the web application of the insurance company to the internal information system of the L.I. company. The employee downloads any documents available on the web application connected to the request and attaches them to the new process in the Edoclink web application. **Fig. 6** models this part of the process.

**Fig. 6. Business Process Model of “Checking in New Liability Investigation Requests from Web Portal”**

**Checking in Liability Investigation Requests from Email**

Exceptionally, some urgent investigation requests are sent to the L.I. company’s email. Again, an employee from the company must check the email every now and then, create a new process in the internal information system and manually copy the details from the email, downloading any email attachments and uploading them to Edoclink.

During the process, new documents may be sent to the L.I. company’s email to attach to currently active processes. An employee must verify if there are new “update” emails in the company’s email address. If such emails have been received, the employee must then identify which process the attached documents belong to, download the attached documents and upload them to the respective process in the company’s internal information system. The model of this part of the process is presented in **Fig. 7**.
Delivering the Liability Investigation Results

To finish the liability investigation service, an employee from the L.I. company logs in to the insurance company's web application, finds the corresponding request, fills in the necessary data by manually copying from the internal information system to the web application, uploads the liability investigation report from the internal information system and submits. These tasks are modelled in Fig. 8.

Fig. 7. Model of “Checking in New Liability Investigation Requests from Email”

Fig. 8. Business Process Model of “Delivering the Liability Investigation Results”
Why Apply RPA to This Business Process and Expected Benefits

This is a suitable business process for understanding the capabilities of RPA tools and the benefits it delivers, because this is one of those typical business processes targeted for small quick wins. In other words, it would be too costly to get a return on investment from using BPMS tools to achieve system integration but implementing RPA may bring initial benefits with an implementation and deployment duration of three to eight months. Also, one of the systems is external to the L.I. company, so it would be very unlikely that the insurance company would agree on doing a dedicated system integration on the data access level nor does it have an API publicly available for integration.

An RPA integration should be faster, cheaper and more effective, achieving:

- Increased business efficiency;
- Cost reduction and value added (by having the employee doing other more value-added tasks);
- Higher employee satisfaction since he or she will not have to do the previously described tedious and repetitious tasks anymore;
- Greater quality, since the employee could introduce errors during the data transition from one system to the other.

Because the business process is digitized, information is expected to be somewhat structured and it will allow using a wide range of RPA tool features, while delivering benefits to the L.I. company.

Expected Challenges and Tasks Executed by the Robots

Before starting the project, some challenges were expected. This section describes these challenges and helps understand why this business process is a good case for learning more about RPA and to clear some questions regarding the technology. It also justifies carrying out this study.

The business process is fairly complex, because the robot will have to interact with five different systems: two web applications, email, database and desktop.

Another challenge that was foreseen is the fact that the request information contained in the emails may not be standardized, not following a template. This required programming the robot for different cases and exception handling.

The tasks may seem simple from a human and practical viewpoint, but for the robot, the tasks have to be broken down into much more detail, which will require understanding the tasks and careful analysis to sequence the robot steps in a more efficient way. The order of actions that humans take may not be the same as for the robot, since one may be more efficient for the human but less efficient for the robot. For example, the human may alternate in between windows to copy field 1 from a pdf file and paste field 1 to a web application field, while it would be more efficient for the robot to identify every relevant field from the pdf file, place it into a list variable, and then sequentially fill in the web application form.

The chosen business process will allow the implemented RPA automation to use a wide array of features and demonstrate RPA capabilities including:
• Login Using Encrypted Credentials – logging in to web applications (Edoclink and insurance company portal), email and database;
• Safe Credential Storage – providing the credentials for logins;
• Web Browser Interaction – interacting with internet explorer for performing transactions to and from insurance web portal;
• OCR – needed for understanding and extracting information from both web applications or email attachments;
• Screen Scraping – extracting information from both web applications;
• Database Integration – exchanging data between the robot and the database;
• Email Integration – accessing L.I. company email, filtering the emails and extracting information from emails (email text and attachments);
• Desktop File Saving – saving email attachments, uploading files to Edoclink web application and uploading report results to insurance web portal;
• Desktop Folder Navigation – accessing the files required to upload to Edoclink and insurance web portal;
• Regular Expressions – deciding if email is urgent or not and for decision making regarding saving relevant information to database and Edoclink web application;
• Centralized Scheduling / Orchestration – optimizing the whole automation, parsing requests on time and delivering reports on time;
• Prioritized Work Queues – handling urgent requests before normal requests;
• Assets Management – deploying robots to automate the target business process;
• Analytics – retrieving robot performance metrics to evaluate if the benefits were delivered;
• Exception Handling – for notifying the people in charge of any exceptions that the robot cannot process;
• Alerts – for warning the business process owner or system manager that the robot stopped working unexpectedly.

The orchestrator will be used to schedule the robot. A diagram of the solution architecture is provided in Fig. 9. RPA Solution Deployment Architecture, with application components in red, infrastructure components in blue and business objects in grey.
RPA Tool Choice

For a 7 months project, looking to verify the benefits associated with RPA automation, higher importance is given to the following aspects of the RPA tools:

1. Robot Development
2. Deployment, Governance and Security
3. Control Room, System Management, Reporting and Resilience

Robot Development

Robot Development is the most important aspect to ensure fast and successful implementation of the RPA solution. A faster implementation will free more time for experimenting different deployment configurations and for covering more exception cases that may happen during processing through RPA exception handling steps. For example, using one or two robots to process the workload. This has significant impact on the business process performance and in maintenance costs.

Deployment

Deployment is the second most important aspect, to ensure a problem-free transition from the development and testing environment to the operational environment. If the deployment isn’t successful, it will not be possible to collect metrics from the reporting features and verifying if RPA automation entails benefits.
These two previous aspects should ensure a higher probability of a successful automation of the business process.

Control Room, System Management, Reporting and Resilience

This is important to verify that the automation of the processes using RPA technology did indeed deliver the expected benefits, as read in the literature.

Most RPA tools provide dashboards in the control room, which provide an overview of the business process execution, showing how many instances of the business process were executed, among other metrics. The control room also allows access to logs that tell how long the robots took to execute the tasks.

The better the reporting features provided by the chosen tool, the more metrics can be collected, and more granular optimizations can be done to the automation.

Features that contribute to automation resilience to business process changes and unexpected results from the systems the robot will interact with, will allow testing RPA technology in more advanced use cases. An example of an advanced use case is one in which the process is subject to small changes and the deployed solution is able to respond to those changes without intervention from humans or with very short intervention, in other words, no need for reprogramming the robot.

Choice Rationale

The chosen RPA tool for carrying out this project is UiPath. The annex contains Table 6. RPA Tools Feature Score by Category (Ordered From Highest to Lowest) Adapted From , page 87, where it is possible to see that UiPath is a good fit according to the criteria previously explained, as it consistently ranks among the top four tools.

3.3 Design Patterns

The implementation of robotic process automations depends highly of the applications with which the robot must interact. From details, such as the selectors of a web page UI element and their attributes, to more persistent pre-conditions such as having to interact with a virtual application, several details can impact on the development time and automation efficiency.

The development time is impacted since the better structured the interface elements’ selectors are, the faster it is to develop an automation that interacts with the right UI elements, in an expected way. This is because the selectors will have an XML representation with evident patterns, extensive and explicit attributes that allow for detailed indication of the UI element targeted by the activities of UiPath. Less structured and not so explicit XML representations usually require more thinking and operations to achieve the desired actions from the robot.

The automation efficiency also suffers when worse structured interface elements’ selectors are present, as it is necessary to program the robot to perform extra actions that ensure that pre-conditions for each step are consistent across different automation executions. This means it is required to program the robot to perform activities such as “Retry Scope”, “Try Catch”, “Element Exists”, “If”, which
can identify and/or handle exceptions. The exceptions happen more often when the selectors are not reliable or properly selected, configured and programmed.

Here is an example of a pre-condition for the execution of a step. Imagine the following scenario: in step 1, click a search button that loads the results of a table which contains data; in step 2, extract the data from the table. So, the pre-condition for step 2 is the existence of a table with data. If the robot clicked the search button too fast (for example the webpage did not finish loading yet or is a bit slow in response) and the webpage does not show a table when the search button is clicked, then the data extraction step (step 2) will fail, although the error happened in a previous step. In this situation and similar ones, it is necessary to perform extra actions, such as using the “Element Exists” activity to check if the search button is ready to be clicked, ensuring the table with data will load before attempting to extract the data from the table.

However, certain tasks such as application logins and file downloads, among others, can be quite similar when applied to different processes. Hence, patterns appear, and some best practices or even modular components can be reused.

The following table shows which design patterns were applied in the different tasks of the business process that was automated.

<table>
<thead>
<tr>
<th>Process Task</th>
<th>Design Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login to Insurance Company’s Web Portal</td>
<td>- Application Logins (Virtual Keyboard Example)</td>
</tr>
<tr>
<td></td>
<td>- Clicking UI Elements</td>
</tr>
<tr>
<td>Search and List New Requests</td>
<td>- Clicking UI Elements;</td>
</tr>
<tr>
<td></td>
<td>- Creating and Deleting Desktop Folders;</td>
</tr>
<tr>
<td></td>
<td>- Data Table Scraping;</td>
</tr>
<tr>
<td></td>
<td>- Downloading Files from a Web Browser Application;</td>
</tr>
<tr>
<td></td>
<td>- Reading Excel Files;</td>
</tr>
<tr>
<td></td>
<td>- Using Queues for Managing Robotic Work</td>
</tr>
<tr>
<td>Register Requests</td>
<td>- Clicking UI Elements;</td>
</tr>
<tr>
<td></td>
<td>- Data Scraping;</td>
</tr>
<tr>
<td></td>
<td>- Opening and Closing Web Browser Tabs;</td>
</tr>
<tr>
<td></td>
<td>- Using Queues for Managing Robotic Work</td>
</tr>
<tr>
<td>Task Description</td>
<td>Design Patterns Implemented</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Download Attachments</td>
<td>- Creating and Deleting Desktop Folders;</td>
</tr>
<tr>
<td></td>
<td>- Downloading Files from a Web Browser Application</td>
</tr>
<tr>
<td>Create New Process in Database</td>
<td>- Database Integration</td>
</tr>
<tr>
<td>Copy Data from New Request to New Process in Database</td>
<td>- Database Integration;</td>
</tr>
<tr>
<td></td>
<td>- Using Queues for Managing Robotic Work</td>
</tr>
<tr>
<td>Create New Process in Edoclink</td>
<td>- Clicking UI Elements;</td>
</tr>
<tr>
<td></td>
<td>- Opening and Closing Web Browser Tabs;</td>
</tr>
<tr>
<td>Copy Data from New Request to New Process in Edoclink</td>
<td>- Using Queues for Managing Robotic Work</td>
</tr>
<tr>
<td>Upload Attachments</td>
<td>- Uploading Files to a Web Browser Application</td>
</tr>
<tr>
<td>Login to Edoclink</td>
<td>- Application Logins (Windows Prompt Example)</td>
</tr>
<tr>
<td>Find Corresponding Process</td>
<td>- Clicking UI Elements;</td>
</tr>
<tr>
<td>Download Investigation Liability Report</td>
<td>- Clicking UI Elements;</td>
</tr>
<tr>
<td></td>
<td>- Downloading Files from a Web Browser Application</td>
</tr>
<tr>
<td>Copy Data From Edoclink Process to Insurance Company’s Web Portal</td>
<td>- Creating and Deleting Desktop Folders;</td>
</tr>
<tr>
<td></td>
<td>- Data Scraping;</td>
</tr>
<tr>
<td></td>
<td>- Downloading Files from a Web Browser Application</td>
</tr>
<tr>
<td>Upload Investigation Liability Report</td>
<td>- Uploading Files to a Web Browser Application</td>
</tr>
<tr>
<td>Login to Outlook Email</td>
<td>- Manipulating Emails Through Outlook (Reading Emails Example)</td>
</tr>
<tr>
<td>Check for New Request Keywords</td>
<td>- Manipulating Emails Through Outlook</td>
</tr>
</tbody>
</table>
Table 2. Mapping of Business Process Tasks and the Design Patterns used to Implement it (Continuation)

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Design Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move to Other Email Folder</td>
<td>- Manipulating Emails Through Outlook (Moving Emails to Another Folder Example)</td>
</tr>
<tr>
<td>Update Request in Edoclink</td>
<td>- Clicking UI Elements; - Opening and Closing Web Browser Tabs; - Using Queues for Managing Robotic Work; - Uploading Files to a Web Browser Application</td>
</tr>
<tr>
<td>Inspect Email</td>
<td>- Manipulating Emails Through Outlook (Saving Email Attachments Example)</td>
</tr>
<tr>
<td>Send “Request Accepted Email”</td>
<td>- Manipulating Emails Through Outlook (Sending Email)</td>
</tr>
</tbody>
</table>

Now follow descriptions of the design patterns. The design patterns are ordered by complexity and grouped by related topics. First, the more simple and general design patterns are introduced.

**Clicking UI Elements**

To click a UI element, the RPA engineer should use the “Click” activity.

First, the RPA engineer, uses the option to “indicate on screen” the UI element. This will automatically configure the activity with the selector string of the UI element that was clicked.

Second, the selector that was automatically configured in the “Click” activity should be inspected, for the reasons explained in the *Inspecting Selectors Example*.

Lastly, the click should be tested. The *Missing to Click the UI Element Example* demonstrates why.

Not following this design pattern will increase implementation times. If the selectors are not properly inspected, it will be necessary to execute the automation more times to find out where the automation faults and stops working. It can be time consuming to execute long automations just to pinpoint a specific error or fault.

*Inspecting Selectors Example*

The string that represents the selector may contain information (red flags) that the automation will not work across every instance execution, but only for a particular execution. Consider the following selector (3).
This selector will work for the instance with unique identifier “11TESTE/2018/769” but will not work for another instance with a different identifier. In this case, the unique identifier is associated with a data register for a service request. For another service request, it would no longer click the desired UI element.

**Missing to Click the UI Element Example**

The “Click” activity must be properly configured and tested, since, sometimes, the selector is unreliable, and the robot clicks an unexpected UI element.

During the development of the automation there was a particular instance in which the robot failed to click the desired button (from a dropdown menu of options), as shown in Fig. 10.

![Fig. 10. Failed Click (Left) Comparison with Successful Click (Right) after Adjusting Properties](image)

One way to fix this problem is by taking a closer look at the properties of the “Click” activity (Fig. 11). If the UiPath Studio user expands the “CursorPosition” option, three input fields will show, which allow tweaking where the robot will click, relative to a fixed position of the UI element region that is referenced by the selector (Fig. 12 shows this region).
Application Logins

Login prompts vary, but typically involve writing a username and a password into two different text fields on a website or window. Automating this task is often straightforward.

Before programming the login automation, as a best practice, the RPA architect should design the system in such way that the username and password strings (user credentials) are all stored in the same location and safely using adequate methods such as encryption. Using UiPath, this is done by logging in to the UiPath Orchestrator component, clicking the “Assets” tab from which a list of assets is shown. Now it is possible to create a new asset with the type “Credential”, which saves the username and password strings in the orchestrator component.

When it comes to programming the automation, first, the robot must attach to the window displaying the login prompt (either webpage or desktop window) using the “Attach Window” or “Attach Browser” activity.

Second, the robot needs to retrieve the user credential strings from the central repository, instead of retrieving them from a variable coded in the automation. The robot can then access the credential using the “Get Credential” activity, while executing a business process.
Lastly, two "Type Into" steps are defined, the first receiving the username string and the second the password string as input.

This is a general, simple description of a login. However, variants may occur to which the design pattern should be adapted. Examples are provided below.

**Windows Prompt Example**

The application may open a dedicated desktop application window for doing the login, as seen in Fig. 13. In this variant, the only difference from the generic implementation for the login is that the RPA engineer must first attach to the new window that popped up in the screen, instead of attaching to the web browser application. The selector for the username and the selector for the password must be configured.

As an example, the selector used to write into the password field was the following:

```
<wnd app='iexplore.exe' cls='Credential Dialog Xaml Host' title='Windows Security' />
<ctrl automationid='PasswordField_2' />
```

**(4)**

**Virtual Keyboard Example**

A virtual keyboard does not allow to type into the password field of the login prompt (example in Fig. 14). Instead, the robot must click visual keys on the screen to insert a character in the password field. The only way for the robot to know where to click, is by reading every character in the buttons and then associating the button selector with the respective character in a dataTable, since the position of a character is always random in different execution instances. Then, when entering the password, the robot reads each character of the password, checks the dataTable to find which selector indicates where that character is located and clicks the button in the virtual keyboard.

The RPA engineer should first try to find a pattern in the selector strings corresponding to the buttons. Without a pattern, it may be impossible to program the robot to map each character available with its’ corresponding selector, since the robot will not know which UI elements to interact with. If not
impossible, it will probably require more work to implement the recognition of the characters and their position or UI element in the screen. The best way to find a pattern in the selectors of the UI elements, is to look at the selector of the first virtual keyboard button and at the last virtual keyboard button.

A concrete example of an existent pattern is when the first button has the selector (5) and the last button has the selector (6), from a total of 70 buttons.

```
<webctrl id='btnkey0' tag='INPUT' />  (5)
<webctrl id='btnkey69' tag='INPUT' />  (6)
```

From one webpage load to another, the selectors are always the same for a button in the same position, although the character in that button may change, as can be seen when comparing Fig. 14 and Fig. 15.

The existence of such patterns in the selector strings simplify the implementation, since the selectors can be generated with string manipulation and in each execution of the automation, the robot just has to map the character in each button to the selector. The automation is then more readable and less complex because the selectors are generated in one loop by string manipulation, instead of having one activity for each button to find its’ corresponding selector. The automation is also more efficient when executing, as the robot does not have to rely on image position and image recognition features (the selectors are always the same for a given button position). This is typically the least efficient variant in login automation, since the robot must perform additional actions to achieve the same result.

![Fig. 14. Virtual Keyboard that was Automated in the Project](image1)

![Fig. 15. Virtual Keyboard with Different Characters in the Buttons (Another Instance)](image2)
Creating and Deleting Desktop Folders

To create a folder, the “Create Directory” activity is used. It requires the full path of the directory to be created as input, which is configured as a string.

To delete a folder or a file, the “Delete” activity is used. It is similar to the “Create Directory” activity, as it receives the full path of the directory or file to be deleted as input, configured as a string.

Downloading Files from a Web Browser Application (Using Internet Explorer)

Once a download button is clicked, the steps needed to save a file are similar, since downloading any file in the Internet Explorer web browser will trigger the same program flow. The robot must be programmed to interact with this flow, since UiPath does not provide an activity for downloading a file to a specified file path. Once the several steps are programmed in the automation sequence, it can be reused for other business processes. For example, these other business processes may require downloading files from a different web page.

As best practice, in order to have greater control and compliance over the automatic operations, the files should be saved to a specific folder. Therefore, the robot should create a directory before saving the files. How the folders are created and organized depends on the business logic and the business process at hand. In this case, for each service request, a folder is created, in which the downloaded files are saved to (see Creating and Deleting Desktop Folders).

The program flow that is triggered is now described and should be automated as follows. A small window from a desktop application will appear, which looks like the one in Fig. 16.

The robot should click the down arrow (seen in Fig. 16) to open up the dropdown menu, with more options other than just “Save”. Now, the robot should click “Save as”. This will bring up a new window prompting the user (in the case of an automation, the robot) with the location to save the file. There is a shortcut that the average user may not be aware of, since it is cumbersome to remember every folder path in a user’s desktop, which the robot can take advantage of. If the user writes the full path to the folder appended by the name of the file to be saved (example: C:\Users\joao.castro\Documents\SomeFile.pdf) in the file name field of the prompt window and then clicks the “Save” button, then the file will be saved to the specified location, with the specified name. So, to perform less actions and improve efficiency, the robot can just write the full path and name of the file to be saved, instead of having to browse the destination folder (where the file is to be saved).

Fig. 16. Window Prompt Presented by a Desktop Application for Saving a File

Uploading Files to a Web Browser Application (Using Internet Explorer)

Uploading files to a web browser application is similar to downloading files, after the appropriate button for uploading a file is clicked on the web application.
Opening and Closing Web Browser Tabs

Each time a new web browser application tab is opened, it should be closed when the robot no longer needs it to perform business operations. Ignoring this aspect of the implemented automations will lead to overusing the machine’s processing resources, on which the robot is working. This will eventually halt the machine and stop the robot from producing work.

Not following this design pattern will:

- Increase implementation times – this design pattern presents a consistent way to deal with the issue of freeing the machine resources before starting the execution of a new instance of the automation (or a new transaction item);
- Hurt business process automation efficiency - the robot will have processing breaks more often and this will require troubleshooting the problem in each machine the automation was deployed to;
- Increase costs – the downtime of the automation system, the deployed robots, will be higher. Also, extra costs in maintaining and solving the errors will be incurred.

When first understanding the execution of the business process to automate, the RPA engineer must identify when and how many new tabs are open by whoever is executing the business process. Then, the engineer should identify when the tab is no longer needed. At this point of the automation, the tab should be closed by the robot.

Depending on the flow of the program, one variable of the type "UiPath.Core.Browser" for each tab might be required in order to later close the tab, when it is no longer needed. Typically, the number of variables of the previous type required are equal to the number of tabs open, at the same time, in one web browser. However, there may be business processes in which this does not hold true. It is up to the RPA engineer to analyze the business process and decide how many variables are adequate for the situation.

For example, for the automation interacting with the tabs in Fig. 17, two variables are required to save the tab information, so that these can be closed later, when no longer needed.

![Fig. 17. Two Tabs are Open at the Same Time by the Robot During the Execution of an Implemented Automation](image-url)

To perform actions on a browser tab, usually it is required to use the “Attach Browser” activity. This activity provides an optional output field called “UiBrowser”, which returns a value stored in a variable of type “UiPath.Core.Browser”. This variable references the tab in which the robot will now perform actions.

It is worth mentioning that a new browser tab will also be open when the activity “Open Browser” is programmed in the automation and this activity also provides an optional output field called “UiBrowser” with similar behavior to the one that can be found in the “Attach Browser” activity.

The RPA engineer should write in the corresponding variable name (of type “UiPath.Core.Browser”) in each activity that opens or attaches to a new web browser tab. This will save a reference to the web
browser tab, which will be later used by the “Close Tab” activity when the web browser tab is no longer needed.

The “Close Tab” activity has an input field named “Browser” which receives a variable of type “UiPath.Core.Browser” and will close the tab to which the input variable corresponds to. If only one tab is open, the web browser application will be closed.

Summing up the steps that should be followed when implementing this design pattern:

1. Identify when a new web browser tab is open;
2. Count the number of web browser tabs open at the same time at any given time during the automation and create a number of variables of the type “UiPath.Core.Browser” equal to the maximum count open web browser tabs;
3. Identify when a web browser tab is no longer needed by the robot to perform business operations;
4. Create and name as many “UiPath.Core.Browser” variables as needed;
5. Configure the activities “Attach Browser” or “Open Browser” to save the “UiPath.Core.Browser” variable value, accordingly;
6. Program the robot to use the “Close Tab” activity when the web browser tab is no longer needed.

Screen Scraping

This design pattern is more appropriate for situations in which the information of a single UI Element needs to be extracted. There are two options to program screen scraping. The first option is better for new UiPath Studio users.

The first is to use the wizard in UiPath Studio called “Screen Scraping”. The wizard will wait for the programmer to click the UI Element containing the information. Once clicked, it displays a preview of the extracted data, as well as the time it took to perform the data extraction. The programmer can then select one of three different methods to extract the data: “FullText”, “Native” or “OCR”. As a best practice, the different methods should be checked to see how long it takes to extract the data and which one is more accurate. Finally, an activity is created with the appropriate configuration.

The second is to directly drag and drop a data extraction activity into the sequence and configuring it. Some experience is required to choose the most adequate activity. Different activities come with different properties to configure.

Out of the three different methods available for extracting text (FullText, Native, OCR), FullText is the fastest, Native comes second in speed and OCR is the slowest, according to UiPath’s documentation (UiPath, 2016).

Data Table Scraping

This design pattern should be followed whenever data has to be extracted from a data table being displayed in a web page. UiPath provides a wizard for obtaining this data which can be programmed to extract the data with different methods. However, this feature is not always reliable, and the programmer may need to try other more creative ways to get the data.
To begin with, the programmer needs to start the wizard and create a template of the data table to extract the data from. This template will allow the robot to extract data from the table, regardless of how many lines the data table contains. However, the columns and the order of the columns must remain the same across different business process execution instances.

A new window will pop up, asking the programmer to click on a cell from the data table. When the first cell is clicked, the programmer has the option to extract the whole table (option one) or to select only a part of the data table to extract (option two). **Fig. 18** models the flow of the data scraping wizard.

If the first choice is taken, then UiPath automatically extracts the entire table, which may bring undesired data (sometimes empty columns). The wizard will show a preview of the data table being extracted. If the format of this data table does not match what is required for automating the business process, then the programmer has to restart the wizard and follow the second option.

Following the second option, another cell, with the same column as the first cell that was clicked, should be clicked. This allows the robot to identify one column of the data table template. Now the programmer can add as many columns as desired from the data table to further define the template.

**Fig. 18. Different Ways to Program Data Table Extraction from Web Pages**

The lines of the data table should be consistent. For example, when defining the first column to be extracted, if the programmer clicks a cell on line one and then a cell on line two, when defining the second column to be extracted, the programmer should click a cell on line one and then a cell on line two, as seen in **Fig. 19**. If another clicking order is chosen for the same set of cells as in the image, the robot will fail to find a pattern and will fail to define a template of the data table. The same will happen if a cell from another line is clicked (as seen in **Fig. 20**).
Fig. 19. Correct Order of Clicking to Select Columns When Building Data Table Template, Following the Data Scraping Wizard

<table>
<thead>
<tr>
<th>Etiqueta Código</th>
<th>Assunto</th>
<th>Estado</th>
<th>Origen</th>
<th>Último Envio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1st Click</strong></td>
<td>3rd Click</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2nd Click</strong></td>
<td>4th Click</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 20. Wrong Order of Clicking to Select Columns When Building Data Table Template, Following the Data Scraping Wizard

<table>
<thead>
<tr>
<th>Etiqueta Código</th>
<th>Assunto</th>
<th>Estado</th>
<th>Origen</th>
<th>Último Envio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1st Click</strong></td>
<td>3rd Click</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2nd Click</strong></td>
<td>4th Click</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The third option requires analytical skills to understand the underlying structure of the web page and the data table. Significantly more work is required to extract data from tables which do not have a structure consistent enough to use options one or two.

After the template for the data table is defined, the programmer can also program the robot to click the button that goes to the next page, containing more data. This way, it will automatically go through all the pages and extract the data contained in the data table, regardless of how many pages there are, until a maximum number of rows is reached. This number of rows is configured by the programmer. So, for example, if there is only one page of data, it extracts without trying to go to the next page. But, if there are three pages, the robot will extract the data in each of the three pages. However, this feature is not always reliable, and the programmer has to manually implement some logic for the robot to extract all the data correctly.

In the project, the second option was used, without a reliable button for the next page of the data table available. A more detailed example is provided in Data Scraping from Web Pages Without Reliable Selectors, page 75.
Unreliable Sequence Sections

The RPA programmer should execute automations several times during implementation to identify failure points. The robot may produce inconsistent results as explained in Debugging - Understanding What the Robot is Doing Wrong, in page 78. When these failure points are identified, extra measures should be taken to ensure the expected results and to ensure it does not fail. This provides business process reliability and quality.

To do that, the activities “Retry Scope” and “Element Exists” can be used.

The “Retry Scope” activity is more appropriate for when the robot does not throw any exception but also does not produce the expected results. This activity will retry an action until a condition is met. For example, when a robot clicks a button too fast and the button click does not open a window that should be open when the button is clicked, then the robot can be programmed to check if the window is open. If the window is not open, the robot will click the button again for the number of times specified by the programmer. If the window still is not open, an exception is thrown, else if it is open, the automation continues to the next steps.

The “Element Exists” activity is used in the “Retry Scope” activity to check if the condition is met. In the example provided previously, the “Element Exists” activity can be used to check if the new window is displayed on the screen.

This design pattern is illustrated in Fig. 21. Not following the design pattern will result in lower business process quality, since more errors will occur.
Exception and Confirmation Prompts (Events That May or Not Happen)

When the user is interacting with applications, sometimes an exception prompt or confirmation prompt pops up. A prompt is a new window that is open by the application being used, providing some information regarding an issue and asking the user what to do. An example of a confirmation prompt (Fig. 22. Confirmation Prompt Launched by “Save As” Windows Application – Overwriting a File Exception Fig. 22) is when the user tries to save a file, using a file name that is already being used by another file in the location where the user wants to save the new file. The windows operative system will open up a new window informing the user that there is already a file with that name. It then waits for the input from the user, deciding what to do next.
There are two options to handle these event situations, when using the UiPath tool.

One is to check if the element exists (with the “Element Exists” activity) and if positive, then click the “Yes” button. Otherwise perform no activity and continue the automation execution.

The other, is trying to click the confirmation button even if no window pops up asking to confirm to overwrite. This “Click” step is configured to continue to the next step even if an exception is thrown (in particular the “SelectorNotFoundException”, which is thrown when an expected UI element is not found) by setting the property “ContinueOnError” to true.

Because activities are retried by default for 30 seconds until an exception occurs, this value (timeout value) must be changed to make the automation more efficient. If it is not changed, the activity will wait 30 seconds, an exception will be thrown, the exception will be ignored, and it will continue to the next step of the automation. The timeout parameter is then configured to 3000 milliseconds (3 seconds), by changing the “TimeOut” property in the “Click” activity.

The following Fig. 23 depicts the implementation of this design pattern following the first option.
Using Queues for Managing Robotic Work

Two things should be considered when creating queues and starting/ending transactions.

One, queues can be used strategically for covering unexpected exceptions, such as application exceptions. Application exceptions are exceptions that occur during execution and are related to the code execution itself. Business exceptions are the other type, which is related to business logic exceptions, for example a missing information field. Whenever an application exception occurs while a transaction is being processed, the orchestrator component will send a message to the robot to retry the transaction for the number of times configured in a parameter of the queue to which the transaction belongs.

External applications are the applications with which the robot must interact during the business process execution. One example of such an application in this project is the Insurance Company's Web Portal.

To avoid having to implement an automation to roll back the state of an external application and for when doing this is not possible, the RPA programmer should analyze the business process before starting the automation implementation and identify some key points in the business process when the state of the external applications changes in a persistent manner. Once identified, the robot should be programmed to use the activity "Set Transaction Status" to change the state of a transaction to successful, immediately after a persistent change is submitted to the external application.

For better understanding this concept, see Identifying External Applications’ State Changes Example.

Two, the queues can also be used to organize the transactions that the robot has to process and to store information related to the transaction, so, in a way, it can be used to pass arguments between
different sequences of the automation. For more detail, check the Organizing the Transactions and Passing Arguments Example.

As a good practice, the RPA programmer should only allow unique queue items to be part of the queue to prevent the robots from doing the same work twice. Also, the queue item’s status should be changed to “Successful” using the “Set Transaction Status” activity, when the requirements of the business process are complete. These two preventive measures ensure business process quality and efficiency, by avoiding that the robot processes the same transaction more than once.

Identifying External Applications’ State Changes Example

For example, consider the project’s business process part named Checking in New Liability Investigation Requests from Web Portal (described in page 40).

In this project’s example, a new process instance register is created in the Edoclink web application. The application persistently stores the information of the process instance in the register. Then a file must be uploaded and associated to this new process instance register. When the file is uploaded and submitted the application’s state changes again. Imagine there is an exception after the process is created and before the file is uploaded. If the robot retries the automation from the beginning, a duplicate process will be created. Instead, if we place a transaction end point right after the process is created and we place a transaction start point before trying to upload the file, then the robot will only retry to upload the file and a duplicate process instance is not created in the Edoclink web application.

Organizing the Transactions and Passing Arguments Example

For the part of the business process described in Checking in New Liability Investigation Requests from Web Portal (page 40), the robot lists all the new service requests by inserting their unique service number into a queue in the orchestrator component. Then, the robot extracts the items from this queue one by one, starting a transaction, and uses the service number stored in the queue item to search for the page containing more information about the service request. It then collects more detailed information about the service request, stores it in a queue item inside a different queue. Finally, the robot processes the items one by one, copying their information (service number, process number, request date and more) into the Edoclink web application.

OCR Text Extraction from Image File

UiPath provides integration with three different engines for executing OCR: Microsoft OCR; Google OCR; Abbyy OCR. The RPA programmer can use activities that execute OCR locally in the machine or activities that connect to the cloud services to perform the OCR operation.

The results may not be satisfactory at first, so different engines should be tested, using several configurations for better data extraction results (see Fig. 24). Also, changing the properties of the activities changes the results of the extracted text.

In order to solve problems with the output of the OCR operation, assuming the files are more or less consistent in format and visual, the RPA engineer should:

- Change the language property of the activity to the language found in the files;
• Change the scale property to a higher number for smaller images;
• Use Google Tesseract (used by “Google OCR” activity) for small area images;
• Use Microsoft MODI (used by “Microsoft OCR” activity) for large area images;
• Test several times and with different cases to ensure quality in the business process automation;

(UiPath, 2016)
In this project, Microsoft and Google engines were tested. The examples are provided below.

**PDF File OCR Example**
This example describes the procedure followed during the project to extract information using the Microsoft OCR engine which executes locally in the desktop where UiPath Studio is installed, this is in which the robot performs its' work.

The emails sent may contain file attachments, from which the robot must extract data. These file attachments were either image files or pdf files. To extract the information, the robot must perform Optical Character Recognition (OCR) on the file.

Fig. 40, in the annex, shows the document that was scanned into a pdf file and was targeted by the OCR activities.

Fig. 24. “Google OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time
(Language: English; Scale: 2)

First, since the image being targeted by OCR contains text in Portuguese, the language was changed from English (“eng”) to Portuguese (“por”). To do this, an additional file had to be downloaded, which contains training data for “Google OCR” to be able to recognize Portuguese words. This file was placed into a folder which contains files with training data for the several languages supported by the tesseract OCR engine by Google. The “Google OCR” activity also had to be configured accordingly, by changing the language property from “eng” to “por”.

Second, the scale property was changed to different values. The scale property of the “Google OCR” activity allows enlarging the image or UI Element that is being targeted by the OCR engine.
Enlarging the image may improve the read result of the activity. This is the reason why several executions were performed using different values for this field, since in certain situations, further configuring the automation can greatly improve the results of the automation (see Fig. 25 and Fig. 26).

Finally, because the results still did not allow correct extraction of the necessary data, another OCR engine was tested, Microsoft OCR. See Fig. 27, Fig. 28 and Fig. 29 for the results, with different configurations, for comparison with Google OCR. Note that in Fig. 27 and Fig. 28 the execution time is not displayed, meaning it took under one second.
Fig. 27. “Microsoft OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time

(Language: English; Scale: 1)

Fig. 28. “Microsoft OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time

(Language: Portuguese; Scale: 1)

Fig. 29. “Microsoft OCR” Activity Configuration on the Left and Extracted Text on the Right with Execution Time

(Language: Portuguese; Scale: 3)
Database Integration

For the robot to insert data in a database, the programmer should use the activities from the package “UiPath.Database”. The activities from this package that are mostly used in this design pattern are “Connect”, “Insert”, and “Disconnect”.

First, the programmer should create a new data table with the same structure as the destination database table, using the “Build Data Table” activity. After this, all the rows should be added to the previously created data table, using the “Add Data Row” activity.

Second, a new connection needs to be established to the database, which is done with the “Connect” activity. A connection string is passed as an argument to this activity for opening the new connection to the desired database. The activity provides a connection wizard, shown in Fig. 37, Fig. 38 and Fig. 39 in the annex.

Third, the robot is programmed to insert the created data table into the database through the “Insert” activity. The connection variable created in the previous step, the destination database table name and the data table variable are passed to this activity as arguments.

Finally, the connection to the database is closed resorting to the “Disconnect” activity.

Reading Excel Files

Whenever it is required to read or edit data contained in an excel file, the activities from the package “UiPath.Excel” should be used, since many common tasks are implemented by these.

First, the “Excel Application Scope” activity must be created and configured. This activity provides the ability to use other excel related activities. It opens the excel file that needs to be processed. This file is specified using its’ full file path as input to the activity.

Lastly, the robot is programmed to use the activity “Read Range”. The programmer must specify the input for the activity, which consists of two inputs: “Range” of the sheet to read and “Sheet Name” which is the sheet inside the excel file containing the desired information. The data is then outputted to a dataTable variable that the programmer can now use in the automation.

Not following this design pattern would require the programmer to have the robot click the windows start menu, open an excel application and then open a file just like a normal user. This design pattern shortcuts all these actions.

Reading Request Information from the Excel File Example

In the project implementation, the robot reads the range of cells containing the information of the new requests and outputs this range into a “dataTable” type variable. The request information starts always on the first row and ends on the row equal to the number of requests plus one. So, the “Read Range” activity receives the following input for the “Range” to read:

"A1:F" + (Convert.ToInt32(requests) + 1).ToString

(7)

The “Sheet Name” input receives the string “PesquisaServicos”.

Manipulating Emails Through Outlook

Tasks related to handling emails, such as reading, writing, moving to folders are automated using activities from the package "UiPath.Mail". This package contains activities that work with different mail protocols, like SMTP. In particular, there is a set of activities that allow integration with Outlook and it automates the interaction with this desktop application.

The programmer just has to search the activities panel in UiPath Studio to find an activity that performs what is needed. Some actions may not be available and have to be programmed by writing code, checking the .Net documentation for functions that do what is required. For more basic cases, such as the ones described in the following examples, there are activities available, which save implementation time.

Not following this design pattern, in other words, not using these activities, would require programming the robot to open up the Outlook desktop application on the Windows environment and clicking the interface of Outlook to perform the required actions.

Reading Emails Example

For reading emails through the outlook application, the programmer should use the “Get Outlook Mail Messages” activity. This activity logs into the Outlook account and outputs a list of emails starting with the most recent email. All it requires is the account name (email) and the mail folder from which to extract the mails. It does not require the programmer to give the password. However, the account must have been previously logged into and the credentials saved, using the Outlook desktop application.

In addition, the programmer specifies the maximum size of the list and may choose to mark the emails as read. It is also possible to choose if only unread emails are extracted from the mail inbox in outlook.

Moving Emails to Another Folder Example

Again, to move emails to another folder only one activity is needed, called "Move Outlook Mail Message". This activity behaves in similar fashion to the one described in the previous example. It logs in to Outlook and moves the email passed as input to the folder name, also configured as input to the activity. No password configuration in the activity is required.

Saving Email Attachments Example

Saving email attachments is done with the “Save Attachments” activity. This activity behaves different from the previous ones. It does not require an account name. It only requires the file path of the folder to where the email is to be saved and the email variable in which the attachments are located.

Sending Email Example

Sending an email is done with the “Send Outlook Mail Message” activity. As with the previous activities, the user specifies which account is used to perform the action. Then, the email body string and subject string are configured in the activity. This is the content that will be sent in the email. Additionally, the programmer may configure the activity to send an attachment.
4 Results

4.1 Evaluation Methodology

To evaluate the benefits delivered by the RPA automation the following statistics will be collected:

- The amount of time it takes an employee or a person to perform the tasks the robot will replace;
- The amount of time it takes the robot to perform the automated tasks;

The amount of time it takes an employee or person to perform the tasks the robot will replace will be measured by starting a smartphone timer when the human volunteer starts the first step of the automation and stopping the smartphone timer.

The amount of time it takes the robot to perform the automated tasks will be acquired from the execution logs of the UiPath tool, for the corresponding automation.

- The percentage of times the robot fails and introduces errors

The robot may fail due to an unexpected case, for which it was not programmed to handle, such as an email with different information structure. It may introduce errors if the robot programming logic is flawed. It is very unlikely to introduce errors otherwise, since the RPA tool is expected to deliver the basic operations without producing wrong output.

The calculation of this percentage done following the equation (8). For this calculation, one failure is considered one output error.

\[
\text{robotErrors} = \frac{\text{outputErrors}}{\text{processExecutions}} \quad (8)
\]

With this information we can evaluate the quality of the business process after RPA deployment and estimate how many hours will be required from a human worker to process the requests that the robot could not process.

4.2 Process Automation Statistics

This section presents the results of executing the different parts of the business process in terms of time. The number of errors made by the robot is also shown. A comparison can then be done between human execution and robot execution.

Checking in New Liability Investigation Requests from Web Portal

The initial conditions, before executing the tests, were created by a robot. To publish a new request service to the web portal, a human has to enter data into an application located in a remote desktop. Instead of entering this data by hand, an automation was developed to create these requests. Still, these tests did not include downloading any documents attached to the new service requests, since it was not possible to automate the attachment of documents to the new service requests.

The tests consisted of logging into the insurance company’s web application, accepting the service requests and copying their information from this web application to the Edoclink web application, where a new process was created for each new service request.
**Human Processing Results**

Two tests were performed. The first test included accepting 27 new service requests and copying them from one web application to another. However, these were not inserted in the database, which means it took a little less time to execute the business process. Hence, a second test was carried out on 4 new service requests and it also included copying the new service requests to the database. The results are presented in the following tables.

**Table 3. First Human Test Results (Checking in New Liability Investigation Requests from Web Portal)**

<table>
<thead>
<tr>
<th>Execution Instance</th>
<th>Time</th>
<th>Execution Instance</th>
<th>Time</th>
<th>Execution Instance</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:03:37</td>
<td>11</td>
<td>00:02:41</td>
<td>21</td>
<td>00:02:11</td>
</tr>
<tr>
<td>2</td>
<td>00:03:00</td>
<td>12</td>
<td>00:02:14</td>
<td>22</td>
<td>00:02:05</td>
</tr>
<tr>
<td>3</td>
<td>00:02:43</td>
<td>13</td>
<td>00:02:24</td>
<td>23</td>
<td>00:02:04</td>
</tr>
<tr>
<td>4</td>
<td>00:02:40</td>
<td>14</td>
<td>00:02:10</td>
<td>24</td>
<td>00:01:52</td>
</tr>
<tr>
<td>5</td>
<td>00:02:27</td>
<td>15</td>
<td>00:01:52</td>
<td>25</td>
<td>00:02:04</td>
</tr>
<tr>
<td>6</td>
<td>00:02:14</td>
<td>16</td>
<td>00:02:08</td>
<td>26</td>
<td>00:02:07</td>
</tr>
<tr>
<td>7</td>
<td>00:02:10</td>
<td>17</td>
<td>00:02:08</td>
<td>27</td>
<td>00:02:10</td>
</tr>
<tr>
<td>8</td>
<td>00:02:17</td>
<td>18</td>
<td>00:02:09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>00:03:47</td>
<td>19</td>
<td>00:02:07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>00:02:16</td>
<td>20</td>
<td>00:02:08</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>00:58:02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>00:02:19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4. Second Human Test Results (Checking in New Liability Investigation Requests from Web Portal)**

<table>
<thead>
<tr>
<th>Execution Instance</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:06:36</td>
</tr>
<tr>
<td>2</td>
<td>00:03:59</td>
</tr>
<tr>
<td>3</td>
<td>00:03:56</td>
</tr>
<tr>
<td>4</td>
<td>00:03:55</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>00:18:26</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>00:04:37</td>
</tr>
</tbody>
</table>

**Robot Processing Results**

The robot had to process 100 new service requests. Four requests failed to be processed due to an application exception and 96 requests were processed successfully. The robot performed with 96% success.
The robot took a total of one hour, 6 minutes and 11 seconds to process the requests. That resulted in an average processing time of 39.71 seconds per request.

The errors reported by the robot upon failure to process the transaction items were of the type “Application Error”, more concretely, the robot failed to find the UI element targeted by a selector (SelectorNotFoundException), which means it was not on the screen. To better understand how the robot reports the errors, the annex shows images of the orchestrator component displaying information regarding the processing of the transactions, in Fig. 42 and Fig. 43.

**Checking in Liability Investigation Requests from Email**

**Human Processing Results**

The human was able to process 4 new request emails and 4 update request emails in 12 minutes and 10 seconds.

The human performed the same tasks the robot performed as is described in the following subsection.

**Robot Processing Results**

The robot was able to process 4 new request emails and 4 update request emails in one minute and 40 seconds.

It successfully created new processes in Edoclink, however it failed to update Edoclink with the requests containing attachments in the email.

It successfully classified the emails into three different cases: non-related to service requests; new service request; update service request. It moved the non-related emails to a different folder from the inbox and marked all processed emails as read. It also flagged exceptions for emails missing required data related to the service request.

**Delivering the Liability Investigation Results**

**Human Processing Results**

One test was performed in which 20 request services were processed. The results are presented in the next table.

<table>
<thead>
<tr>
<th>Execution Instance</th>
<th>Time</th>
<th>Execution Instance</th>
<th>Time</th>
<th>Execution Instance</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:03:48</td>
<td>2</td>
<td>00:01:53</td>
<td>3</td>
<td>00:01:44</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td>9</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>00:02:19</td>
<td></td>
<td>00:01:29</td>
<td></td>
<td>00:01:50</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td>16</td>
<td></td>
<td>17</td>
</tr>
</tbody>
</table>
Robot Processing Results

The robot had to process 100 service requests located in the Edoclink web application. One request failed to be processed, but all others were successful. This means the robot performed with 99% success.

The robot completed the test in a total of 6 hours, 12 minutes and 7 seconds. So, on average, the robot required three minutes and 43 seconds per request, approximately.

4.3 Conclusions from the Results

Checking in New Liability Investigation Requests from Web Portal

Considering the results of the second human test, the robot was faster than the human by three minutes and 58 seconds, approximately, when considering the average time for processing a request. This means the robot can process 6 requests, approximately (6,976 requests, almost 7 requests), during the time it takes the human to process one service request.

The success percentage of the robot is high. Considering the human requires an average of 4 minutes and 37 seconds to process one request, we may estimate that a human processes 104 processes in 8 hours of work, approximately. If the robot replaced the human in the 8-hour work day, it would produce on average 4 errors, which means a human has to process 4 service requests. Then, 18 minutes and 27 seconds per day (approximately) of human work would be required to process the requests that the robot could not handle.

Checking in New Liability Investigation Requests from Email

The automation implementation for the email requires some improvement and minor bug fixes so that the robot can correctly update the already existing requests in Edoclink with the attachments contained in the email.

The human performed the same actions as the robot so that a reasonably reliable comparison can be made between the performance of the human and the robot.

The robot was 10 minutes and 30 seconds faster than the human. In a real-life scenario, this value would depend on the number and type of emails received, since one type can be faster to process than the other. This value allows the reader to have an estimate of the impact of the automation on this part of the business process.

Delivering the Liability Investigation Results
Unlike in the first part of the business process (Checking in New Liability Investigation Requests from Web Portal), the robot takes significantly longer than the human to process the requests.

Considering the average time for processing a request, the human was faster than the robot by two minutes and two seconds. This means the human can process two requests, approximately (2,208 requests), in the time it takes the robot to process one service request.

The robot is slower than the human because it pauses when processing the requests for unknown reasons. It is not possible to know if with more time it would be possible to identify the reason and to implement the automation in a different way, one that does not pause, or if this pause is due to incompatibility in the way the UiPath robot interacts with the web application (lack of maturity and versatility of the RPA tool).

Deploying the automation in the current state would not be beneficial if the business process requires a fast processing time but could be beneficial if it does not require fast processing time.
5 Discussion

5.1 Implementation Challenging Problems and Solutions

This section presents challenging moments faced during the implementation of the project and the solution to those problems, if there is one. This will allow the reader to understand some drawbacks of implementing RPA as well as problems that may arise. These drawbacks and problems may impact the benefits of carrying out and deploying an RPA project. This knowledge will help estimate if there is a return on investment associated with potential RPA projects and may help identify possible pitfalls during the execution of these projects.

Retrieving and Using Login Credentials Safely

As for the second problem, the login using the virtual keyboard, while protecting against key logging attacks by third parties, is not as safe as desired, since the implementation has to convert the password from the type “secureString” to “string”. Securestring types provide passwords are stored with the type SecureString, which stores the password in an encrypted way. But, since the robot has to know where to click in the virtual keyboard, it is required that the password is cast into the String type, so that a match between the password and the button which has to be pressed can be done. This conversion, to the string type, leaves the operation vulnerable to password interception by malicious hackers, since the runtime code can be accessed and analyzed to discover the string that represents the password.

“One instance of the System.String class is both immutable and, when no longer needed, cannot be programatically scheduled for garbage collection; that is, the instance is read-only after it is created, and it is not possible to predict when the instance will be deleted from computer memory. Because System.String instances are immutable, operations that appear to modify an existing instance actually create a copy of it to manipulate. Consequently, if a String object contains sensitive information such as a password, credit card number, or personal data, there is a risk the information could be revealed after it is used because your application cannot delete the data from computer memory.” (Microsoft, 2018)

One may think generating a new random password each time the robot logs in would work too, but that would require the password to be randomly generated using the String type, converted to SecureString, stored to the orchestrator and sent to the system administrator, in case manual logging to the web application is required. This is not a safe solution either.

Instead, a notification may be sent to the system administrator each time the robot finishes its’ tasks. Then the administrator is responsible for logging into the web application, changing the password and also changing the corresponding asset in UiPath’s orchestrator. This solution would make the system slightly safer, since the attackers would have less time to find the password.
Writing to Text Field in Download Window Prompt

When downloading a file from a web page, the robot has to interact with a “Save as” window that pops up, as described in page 54.

This automation sequence executed correctly several times, yet when there was a minor change to the default path in the “Save as” dialog, the robot had unexpected behavior. It was no longer typing the correct path into the text field. Instead, it selected the field in which to write the string by clicking it (just like before), but because the default path was longer than the previous default path, it did not clear the text and did not write the string it was supposed to write. It then moves on to the next step, without throwing any error in the actual step where it faults.

To solve this problem, I had to think of new ways to interact with the “Save As” window to save the file and manually test these ways. I found out that pressing the “F4” key selected the path text field, then pressing the “End” key selected the text and pressing the “Back” key emptied the text field (deletes the text).

![Fig. 30. Path Text Field Which is Selected After “F4” Key is Pressed](image_url)

So, the robot was programmed to press the three keys sequentially, using the “Send Hotkey” activity, which simulates the robot pressing a key of the keyboard.

Then, the robot was instructed to write the desired file path, without selecting any UI Element, using the “Type Into” activity, which is not configured with a selector. This means the robot simulates writing and the writing message is sent to whichever UI Element was selected in previous steps of the automation (in this case, the text field).

Data Scraping from Web Pages Without Reliable Selectors

One solution did not work because the data scraping wizard did not extract the information of the data tables as expected. Therefore, another solution had to be implemented, although it is less optimal than the first one. The more optimal solution is now described.

There were two main issues why the more optimal solution did not extract the data tables correctly. First, an empty row added to the datatable extracted from each page in the web portal. Second, extraction of all data tables in different pages required navigating to the next page without a button to go to the next page compatible with the “Data Scraping” wizard (as seen on Fig. 32).

With each extracted datatable there was an empty row, which had to be removed in the end, before generating and adding the queue items to the queue “New Requests – CA Web Portal”, as seen in Fig. 31. Extracted DataTable in Each Page of the New Request Search Results.
Fig. 31. Extracted DataTable in Each Page of the New Request Search Results

Because the web application does not have an arrow to show the next page of results, it was necessary to generate the selectors for each button, with one button corresponding to one page, so that the robot could go through all the pages and extract the table shown in each page.

<table>
<thead>
<tr>
<th>Nº Serviço</th>
<th>Nº Processo</th>
<th>Rep Fornecedor</th>
<th>Data Pedido</th>
<th>Tipo</th>
<th>Estado</th>
</tr>
</thead>
<tbody>
<tr>
<td>180400557003</td>
<td>P217H000900</td>
<td>12-02-2018 17:13</td>
<td>Patrimônial</td>
<td>Pendente para aceitação</td>
<td></td>
</tr>
<tr>
<td>180400557013</td>
<td>P217H000900</td>
<td>12-02-2018 17:13</td>
<td>Patrimônial</td>
<td>Pendente para aceitação</td>
<td></td>
</tr>
<tr>
<td>180400557023</td>
<td>P217H000900</td>
<td>12-02-2018 17:13</td>
<td>Patrimônial</td>
<td>Pendente para aceitação</td>
<td></td>
</tr>
<tr>
<td>180400557033</td>
<td>P217H000900</td>
<td>12-02-2018 17:13</td>
<td>Patrimônial</td>
<td>Pendente para aceitação</td>
<td></td>
</tr>
<tr>
<td>180400557043</td>
<td>P217H000900</td>
<td>12-02-2018 17:13</td>
<td>Patrimônial</td>
<td>Pendente para aceitação</td>
<td></td>
</tr>
<tr>
<td>180400557053</td>
<td>P217H000900</td>
<td>12-02-2018 17:13</td>
<td>Patrimônial</td>
<td>Pendente para aceitação</td>
<td></td>
</tr>
<tr>
<td>180400557063</td>
<td>P217H000900</td>
<td>12-02-2018 17:13</td>
<td>Patrimônial</td>
<td>Pendente para aceitação</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 32. UI elements to navigate results table, without the option of going to “Next Page” (button not compatible with “Data Scraping” wizard)

If an arrow, or any other button to change to the next page, was present, then the robot could be easily configured to extract data from the first page and then from the following pages, using the “Data Scraping” wizard for data spanning multiple pages. This solution did not extract the last page and extracted the first page twice. Also, removing the last row of each datatable was not working consistently.

That is why a less optimal solution was implemented. This solution is now explained.

After the filter is submitted, a table with requests is presented. The robot scrapes the number of requests found in the search, using the “Get Full Text” activity. This activity uses the FullText method (following the design pattern Screen Scraping - page 56).
Then, based on the number of requests, the robot makes a decision on how to extract the request information. If the number of requests is zero, no information is extracted. If the number of requests is ten or less, then the robot extracts the data contained in the table.

Otherwise, the robot exports the data to an excel file and then retrieves the data by interacting with the excel file. The robot clicks the “export” button and saves the excel file using the design pattern described in the subsection **Downloading Files from a Web Browser Application (Using Internet Explorer)** (page 54). The information contained in the excel file is accessed using the design pattern **Reading Excel Files** (page 67).

**Automating Application Interactions Without Stable Interfaces Over Time**

During development, the automation was being tested on a shared environment. The Edoclink application platform which the robot was interacting with was being used by other people, not just by the robot. Other people using the Edoclink web application could make persistent changes to the interface layout, which meant that automations which were already working properly, could stop working and would require reprogramming or changing the interface layout back to its previous state. This was problematic for parts of the automation that required the robot to extract data from a table in the Edoclink web application.

For example, the robot required and extracted data from the column “Estado” in the table shown in the next image.

![Fig. 33. Table from Which the Robot Had to Extract Data Before The Order of The Columns Was Changed](image)

However, after some changes were made to the data table’s column order by other users of the system, the robot could no longer automate the business process properly, since it was extracting unexpected data which caused the automation to fail. The data table with a different column order is shown in the next figure.

![Fig. 34. Table From Which the Robot Had to Extract Data After The Order of The Columns Was Changed](image)

This is because the robot identified which column to extract data from through the following selector:
The selector has no reference to the column name, only to the index of the column to extract (idx='3'), so when the order of the columns was changed, the automation was no longer working. The robot had to be reprogrammed and the selector changed to:

<webctrl class=' e-cell' tag='td' idx='4' />

This example shows the importance of applying RPA to business processes which interact with applications and interfaces that are stable over time. It also shows that ideally the robot should have a dedicated environment for performing its work, to avoid unexpected changes to the system and to better understand what goes wrong. It confirms what can be found in the literature, as read in page 33.

**Domain Knowledge**

When filling in the search filters to find new requests in the insurance company’s web application, I found that the web application would reset the value of the field “Estado” (state of the request), so the order in which the robot typed in the values for the filter had to be changed. It was important to test the implemented automation several times to find this nuance about the application’s behavior.

Such intricate problems are the reason why domain knowledge and knowing how to use the applications with which the robot must interact is so important and a big part of the implementation process. The RPA developer must learn how the applications work and have hands on experience in executing the target process manually.

**Debugging - Understanding What the Robot is Doing Wrong**

During development, automation execution results were inconsistent, with errors happening, for example, 50% of the time.

To solve this problem, UiPath’s “Debug”, “Breakpoints” and “Slow Step” features were used. In “Debug” mode, instead of normal “Run” mode, UiPath Studio displays more information about the value of variables and goes through the automation sequence slower, highlighting which activity is currently being processed by the robot.

![Debug Features in UiPath](image)

**Fig. 35.** Debug Features in UiPath (“Debug”, “Breakpoints” and “Slow Step”)

These features allow quickly going through the problematic automation several times to understand what the robot is doing wrong and well, speeding up implementation time. Without “Slow Step” turned
on, the robot performs the tasks too fast to understand what is happening. Very fast mouse clicks on consecutive specific UI buttons, which then open a new window is an example of a sequence of actions that make it hard to see where the robot clicked.

First, the activity which was throwing the error was found. Then, a breakpoint was inserted and the execution started by clicking the “Debug” button, with “Slow Step” turned off. When the program stopped at the breakpoint, the “Slow Step” was activated to examine how the robot was interacting with the UI.

For example, in one instance, this allowed to conclude that the robot was not performing the expected operations, because it did not recognize the filepath field of the desktop “Save As” window as a text input field, despite being one, at least visually (as explained in Writing to Text Field in Download Window Prompt, page 75).
6 Conclusion

Robotic process automation is growing. There is potential in the capabilities RPA may one day provide, when combined with advances in artificial intelligence. If so, robots will replace a reasonable amount of functions carried out by people today in large organizations. But even small organizations can benefit from RPA when the target business process is a good fit. For now, RPA is low-code visual programming applied to routine, rules-based business processes that add little to no value to the services that organizations provide but which are still essential to their operations. It is the product of progress in software applications development, since RPA makes use of already implemented functionalities and adds a layer of abstraction to those functionalities, so that automations can be implemented by users with less programming knowledge than the average software engineer.

From the project experience and results, I could make the following conclusions.

After implementing the automations for the tasks, the robot was tested several times, including working unattended overnight. This led to interesting findings, since automations that had been executed successfully more than once would still fail every now and then, due to errors thrown by the robot. It is very important that automations are tested extensively, and errors are handled appropriately, either by retrying the automation more than once or by sending alerts to the person in charge of the operation of the business process. In business processes handling sensitive data, such as monetary transactions, debt/payment related information or account balances, an error from the robot could have bad consequences (such as due bills not being registered appropriately and later incurring in fines for not paying them on time, for example) and the error could go unnoticed if the exceptions are not registered.

Exception handling and proper use of selectors is key to developing a resilient automation and this took the most time to implement in the project.

When a business process is well selected, RPA may bring rewards and achieve different business goals, such as costs reduction or efficiency increase. However, selecting the business processes is not a straightforward task, since minor details related to the applications which the robot integrates may have significant impact in the success of the implementation and therefore on the success of the project. Different business process characteristics impact the RPA deployment in different ways. It may impact the maintenance of the project or the implementation. An automation that produces a few errors will impact the maintenance costs, while an automation that is more complex to implement but which does not produce errors will impact the implementation costs more. It is necessary to consider these aspects in the long term to decide if the project is viable or not.

Regarding the objectives of the project, it is possible to conclude that a business process fit for RPA was successfully chosen, since the part of the business process named Checking in New Liability Investigation Requests from Web Portal clearly yields benefits when automated with RPA technology.

The project was implemented and deployed, although with a couple of shortcomings. The data was successfully collected, analyzed and allowed verifying the main benefits of RPA that were to be clarified (efficiency and quality). The costs reduction benefit of RPA could be analyzed better with more data regarding the time it took to automate each part of the process.
Also, the chosen RPA tool (UiPath), proved to be a good choice, since it allowed analyzing the results of the automation and it was considered the leading RPA tool in the market in the paper (Clair, 2018).

6.1 Future Work

The system currently executes basic decision making by searching keywords in the emails that it processes. This decision-making component can be further enhanced by using third party text analytics engines, which can integrate with UiPath, to identify Portuguese names in the body text of emails. This component can also be further developed to better process emails with less structured body text and subject.

There is an issue with the robot pausing when processing the “Delivering the Liability Investigation Results” part of the process. Identifying the issue could help to determine if the tool lacks maturity or if it is a minor bug introduced in the implementation of the automation.

Another possibility is experimenting with and implementing assisted automations for application in business processes with complex business rules but with repetitive tasks, in order to aid, improve and make more efficient the work of non-technical employees within a company. This will allow evaluating the benefits of this type of automations and verifying which business processes are more prone to reaping a return on investment in RPA technology. This would help determine if it is rewarding to apply RPA to short business processes.

From the experience attained in this project, workload distribution among different robots is not so straightforward and requires previous planning and architecting the entire business process automation carefully to enable flexibility in the deployment of several robots to automate the same business process in parallel. It would be valuable to understand how the “Parallel” activity works in UiPath so that an automation can be tuned even further to increase efficiency in the processing of business transactions, as well as understanding orchestrator features which may enable parallel processing of the business transactions.
Acknowledgements

I am forever grateful to my parents, João Manuel and Manuela Castro, for providing me with all the resources required for my academic and personal growth. Their never-ending support allows me to focus on developing myself in every aspect of life, to achieve better results and to stay on the right path. I admire, and I am grateful for the sacrifices that my parents did for me.

I would also like to thank Agnieszka Kuš for caring and being such an amazing person.

I would like to thank my team, project manager Helena Oliveira and senior software developer Luís Francisco, for helping me keep the project on track and for all the extra knowledge they shared with me.

Finally, my gratitude to my thesis supervisor and CEO of Link Consulting Professor José Alves Marques for giving me access to a real-world scenario and resources to accomplish this project, as well as teaching me how to manage pressure situations, meetings and presentations.
Bibliography


7 Annex

Database Connection Wizard

Fig. 36. “Connect” activity in UiPath Studio

Fig. 37. Database Connection Wizard Step 1

Fig. 38. Database Connection Wizard Step 2
Fig. 39. Database Connection Wizard Step 3
### RPA Tool Choice Rationale – Ordering of Tools to Help Decision Making According to Feature Categories

Table 6. RPA Tools Feature Score by Category (Ordered From Highest to Lowest) Adapted From (Le Clair, 2017)

<table>
<thead>
<tr>
<th>Robot Development</th>
<th>Deployment, Governance and Security</th>
<th>Control Room, System Management, Reporting and Resilience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation Anywhere</td>
<td>NICE</td>
<td>Blue Prism</td>
</tr>
<tr>
<td>UiPath</td>
<td>Blue Prism</td>
<td>UiPath</td>
</tr>
<tr>
<td>Pegasystems</td>
<td>Automation Anywhere</td>
<td>NICE</td>
</tr>
<tr>
<td>NICE</td>
<td>UiPath</td>
<td>WorkFusion</td>
</tr>
<tr>
<td>Kryon Systems</td>
<td>WorkFusion</td>
<td>Kryon Systems</td>
</tr>
<tr>
<td>EdgeVerve Systems</td>
<td>Pegasystems</td>
<td>EdgeVerve Systems</td>
</tr>
<tr>
<td>Contextor</td>
<td>Kryon Systems</td>
<td>Redwood Software</td>
</tr>
<tr>
<td>Softomotive</td>
<td>EdgeVerve Systems</td>
<td>Automation Anywhere</td>
</tr>
<tr>
<td>Blue Prism</td>
<td>Redwood Software</td>
<td>Kofax</td>
</tr>
<tr>
<td>WorkFusion</td>
<td>Contextor</td>
<td>Softomotive</td>
</tr>
<tr>
<td>Redwood Software</td>
<td>Kofax</td>
<td>Pegasystems</td>
</tr>
<tr>
<td>Kofax</td>
<td>Softomotive</td>
<td>Contextor</td>
</tr>
</tbody>
</table>

Color Code: Weak | Average | Strong
---|---|---
1 | 2 | 3
4 | 5 | 6
7

87
REQUISIÇÃO DE SERVIÇOS

Lisboa, 31-12-2008 10:53

Apólice nº 58200269459
Processo nº 6015825208

Vimos por este meio requisitar os seguintes serviços:

Produto: Habitação

Peritagem e avaliação ☑

Constatação de Danos sem Regularização ☐
  de Prejuízos

Constatação de Danos com Regularização ☐
  de Prejuízos
  ☐
  ☐

Junto Anexamos:

Condições Particulares ☑

Participação de Sinistro ☑

Orçamento ☑

Certidão das autoridades ☐

Observações:

Para qualquer esclarecimento poderão contactar-nos pelo Tel. 21 380 ☑ ou Fax 21 367 ☑

Gestão de Patrimoniais e Resp. Civil

Fig. 40. Scanned Document as .png File Targeted for OCR
Fig. 41. Visual Programming in UiPath Studio
Fig. 42. Errors Reported by the Robot During Testing of the “Checking in New Liability Investigation Requests from Web Portal” Automation
Fig. 43. Detailed Information of One Error Reported by the Robot During Testing of the “Checking in New Liability Investigation Requests from Web Portal” Automation