Platform-independent Specifications for Robotic Process Automation Applications

Carlos Jorge Martelo Correia[1] and Alberto Rodrigues da Silva[2]
[1]Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
[2]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
{martelo.correia, alberto.silva}@tecnico.ulisboa.pt

Keywords: Robotic Process Automation, Controlled Natural Language, Writing Styles, Software Robot, Domain-specific Modelling.

Abstract: Robotic Process Automation (RPA) is an emerged technology that uses software robots to automate human-intensive repetitive tasks, traditionally associated with bureaucratic and low-complexity processes. This article introduced a concrete scenario implemented with three popular RPA tools (i.e., UiPath, Robocorp, and Robot Framework), which supported the research. First, the paper introduces informally that scenario, implemented with the platform-specific languages of these tools, namely with keyword-based scripts and other visual models. The challenge addressed in this paper is how to describe RPA applications (or just soft robots) in a platform-independent way, close to natural language, and easily understandable by technical and non-technical people. Established on the preliminary results, we conclude that adopting controlled natural languages based on use cases and scenarios simplifies the specification and development of RPA scenarios, no matter the supported RPA tool. For future works, we intend to use the best notation to specify robots in a platform-independent way and implement transformation mechanisms for the proprietary format of popular RPA tools such as UiPath, Robot Framework, and Robocorp.

1 INTRODUCTION

Organisations need to adopt the latest technological improvements to remain competitive, particularly in their information technologies (IT), supporting their business processes. To achieve this goal, organisations must optimise their IT resources and alignment with their bureaucratic processes (Chakraborti et al., 2020). In this scope, Robotic Process Automation (RPA) is a technology that supports automation with software robots (bots) that replace back-office human tasks that are repetitive, tedious, or prone to errors by humans.

RPA is a technology that mainly uses “software (ro)bots” to automate the work done by humans. These software bots can execute workflow with multiple steps and interact with several applications. Examples of tasks performed by software bots are automated email query processing, data transfer between applications through screen scraping, updating a spreadsheet.

Cost increased productivity and time reduction is the main benefits of RPA applications(Houy et al., 2019; Jovanović et al., 2018; Romao et al., 2019). Software robots focus on specific tasks such as creating a monthly report in a few minutes, which humans manually could last several hours. Regarding the increase of productivity, RPA robots can complete the same volume of work doing more with a small number of resources. Furthermore, it improves accuracy because end-users are human, so there is a chance for mistakes. Finally, increase security through the RPA robot because there is no worry of information leakage from one part to another.

RPA tools are progressively adopted and used in the most competitive organisations. However, one problem when using distinct RPA tools is that there is no standard way of writing or specifying software robots and in a tool-independent way. Therefore, to discuss this issue, we informally introduce a concrete case study in which we have developed, i.e., the Navy Integrated Cataloguing System (NICS), implemented in three RPA tools (i.e., with UiPath, Robocorp, and Robot Framework).

The key goal of this paper is to research writing practices to describe software bots in an RPA platform-independent approach, close to natural language, so that they can be understandable by both technical and non-technical people. To support this research, we propose two types of writing software bots based on specification languages inspired by: use case scenarios (da Silva, 2021) and pseudocode specifications (Oda et al., 2016).

This paper is organised into six sections. Section 2 introduces the background of this research, including RPA technology and tools, and the relevance of textual notations used in this research. Section 3 describes the Case Study, presenting the informal requirements and a general overview of the leading business processes. Section 4 presents and discusses two types of writing RPA robots in a platform-independent way. Section 5 refers to these robots" implementation based on three popular RPA tools. Finally, Section 6 presents the conclusion.

2 BACKGROUND

This section introduces RPA tools and textual notations to describe soft robot-based applications in a high-level and platform-independent format.

2.1 RPA Tools

This section introduces the RPA tools used in this research: UiPath, Robot Framework, and Robocorp.

According to the Gartner report (July 2021) (Saikat Ray, 2021), the market leaders are UiPath, Automation Anywhere, and Blue Prism, shown in Figure 1. Gartner describes the RPA market for licensed software platforms used for developing scripts. RPA platforms program repetitive, rule-based, and liable tasks.

These reports assess relevant RPA tools based on the following criteria: (i) Enable citizen developers to build automation scripts; (ii) Integrate with enterprise applications, primarily via UI scraping; (iii) Have orchestration and administration capabilities, including configuration, monitoring, and security.

RPA tools offer advanced capabilities like intelligent document processing, process mining, and discovery. In addition, RPA tools also have emerging features and capabilities, such as a low-code user experience (UX) for building UI front ends for bots; and headless or serverless orchestration of automation workflows.

Open-Source RPA tools provide a solid groundwork to develop own customised robots without being connected with a commercial vendor, whose technology offers limited capabilities and high direct costs (Hüller et al., 2021). Open source solutions can go further into the RPA tools without a significant investment in software.

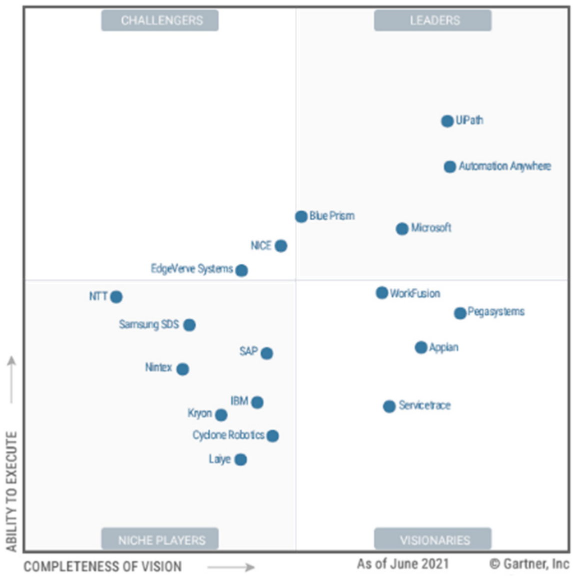

Figure 1: Magic Quadrant for RPA (Saikat Ray, 2021).

2.1.1 UiPaTh

UiPath (UP) Platform services offer governance features, a citizen-developer-friendly UiPath StudioX (UX). This profile is for business users looking to automate tasks. Ideal for users with limited or no experience writing code), a more complex profile, namely UiPath Studio (US) (For users looking to build complex unattended or testing automation. Ideal for users prior programming experience), enhanced computer vision, and cloud-orchestrated RPA (Saikat Ray, 2021).

UiPath is deployed worldwide and has the resources and partnerships to develop its platform by enabling end-to-end automation programs. At the beginning of 2021, UiPath decided to purchase Cloud Elements, an enterprise integration platform as a service vendor, which signals the importance of UI-based and API-based integration for scalable process automation initiatives.

UiPath has the following strengths: Strong brand position recognised among RPA technology. UX app for automation: UiPath's product portfolio includes a low-code UX app builder, Ui-Path Apps, which helps create business value by interfacing with various cloud and on-premises applications. Viability: UiPath demonstrates strong viability and strong revenue growth of 63% from 2019 to 2020. The vendor's community of more than 1 million users reflects its massive customer and partner ecosystem.

On the other hand, its weaknesses are: it still lacks a web-based RPA development environment; it has built a narrative around hyperactive automation and a range of complementary capabilities. However, many competitors that have entered the RPA market from adjacent software sectors provide capabilities that may surpass or match UiPath's offering, especially in terms of complex orchestration, decision automation, and case management. Pricing: UiPath's pricing and packaging strategy change, as evidenced by its introduction of developer persons and subsequent elimination of them within one year.

2.1.2 Robot Framework

Robot Framework (RF) is an open-source automation framework used for test and robotic process automation. The Nokia Networks developed the first version in 2005. RF is written in Python and is an open framework that can be virtually integrated with any other tool and is free to use without licensing costs (Roveda et al., 2017).

RF uses a straightforward syntax, applying human-readable keywords and, for that reason, does not require expert skills in coding. Moreover, it is possible to create new libraries implemented with Python, which allows the expansion of its resources (Hocenski & Stresnjak, 2011).

RF code describes test cases format, written using keyword-based scripts in a tabular format and written in plain text or tab-separated values.

The benefits of RF are: (Hocenski & Stresnjak, 2011) it is an open-source tool; easy to understand and more intuitive due to keywords, and it is possible to develop scripts in Python and Java.

The disadvantages of RF are that its installation requires all packages, drivers and library separately installed; do not provide debugging capability (meaning that is not available the option to use breakpoints); its IDE shows some issues, e.g., it crashes when using the tabular and "text editor" mode.

2.1.3 Robocorp

Robocorp (RC) is a tool for creating software robots based on the Robot Framework (RF), the automation mentioned above (Robocorp, 2021).

In RC, it is possible to build software robots using RF, Python, or both. RC is a virtual Python environment based on Conda, an open-source package management system and environment management system. RC has similar advantages and disadvantages with the above-mentioned for RF.

2.2 Controlled Natural Languages

This section briefly introduces controlled natural languages for writing use case scenarios and workflows based on pseudocode notations.

A controlled natural language (CNL) is a straightforward way to communicate with a constrained version of a natural language. That includes a constrained vocabulary, grammar syntax, and writing styles (da Silva, 2017), (da Silva & Savić, 2021). CNLs can improve communication among humans, mainly for non-native speakers of the corresponding natural language. In addition, the constraints on a natural language make it easier for computers to analyse such texts to improve computer-aided, semi-automatic, or automatic translations into other languages.

The advantages to adopting CNLs are that their sentences are easy to understand, are semantically correct, and can be computationally manipulated.

Concerning the writing of **use cases** and **use case scenarios**, da Silva discusses several linguistic patterns and guidelines to help engineers write them rigorously and systematically (da Silva, 2021). Spec.1 shows a partial specification of a use case scenario as discussed in (da Silva, 2021).

```
UseCase uc_1_ManageInvoices
[…]
0. Scenario MainScenario (Main):
1. System: Shows a list of Invoices and
available actions, namely CreateInvoice,
UpdateInvoice, ConfirmPayment, SendInvoices, and
PrintInvoice. In addition, there are actions to
Close the interaction space, Select/Unselect
Invoices, Search Invoices, and Filter Invoices.
2. Actor: Browses the list of Invoices and
consult Invoices
3. Actor: Selects the option Close.
```

Spec. 1: Partial specification of a use case scenario [from (da Silva, 2021).].

Pseudocode is a popular technique to describe an informal and high-level computer program or algorithm (Oda et al., 2016). Pseudocode writes in symbolic code translated into a programming language before being executed.

Pseudocode aims to be more accessible for people to understand than conventional programming languages code, and it shows in a platform-independent format the main principles of such algorithms. For instance, Spec. 2 presents the description of an algorithm in pseudocode written in English. It is usually used in textbooks and scientific publications to document algorithms and plan software and other algorithms (Roy, 2006).

```
Define the function fizzbuzz with an argument n.
  if n is not an integer value,
    throw a TypeError exception with a message…
  if n is divisible by 3,
    if n is divisible by 5
      return 'fizzbuzz'
    else
      'fizz'
  else if n is divisible by 5,
    return the string 'buzz'.
  Otherwise,
    return the string representation of n.
```

Spec. 2: Example of pseudocode written in English [from (Oda et al., 2016).].

3 CASE STUDY

The NICS (Navy Integrated Cataloguing System) describes a fictional scenario of an application used to manage the parts supply of the navy ships.

The creation of this application occurred in 2019 with the main task of cataloguing articles/parts on navy ships. Articles refer to parts of the navy ship's equipment. The application allows the search for articles/parts in three different ways, namely: "by free research", "by equipment research", or "by article/part research". Each article/part has detailed information about its history and associated documents. It is worth mentioning that the entire history of articles from 2011 onwards was loaded into the application's database, considering that all associated documentation was in digital support.

However, articles/parts documentation before 2011was handwritten, and, for that reason, they were not possible to be loaded. A back-office operator was in charge of digitising all this missing documentation, approximately 78.000 articles/parts documents, and providing the IT team to upload it into the system. After two years daily of work dedicated to this task, this employee only processed around 10000 documents.

The following text (Spec. 3) illustrates the informal requirements of the NICS application. With the purpose of legibility, this text draws attention to the following text fragments: candidate actors dashed underline text; data entities are **bold** and use cases marked as underlined text.

NICS is the short name for the "Navy Cataloguing Information System", which allows users to search for articles/parts. Articles refer to parts of the navy ship's equipment. The application allows the search for articles/parts in three different ways, namely: "by free research", "by equipment research", or "by article/part research". Each article/part has detailed information about its history and associated documents.
A user has a **user profile**, namely as ITManager, backOfficeOperator, or organizationalEntity.
An ITManager registers and manages users […].
An organizationalEntity corresponds to a navy department responsible for creating Cataloguing documents.

[…].

Spec. 3: Partial informal requirements of the NICS.

Figure 2 indicates the domain model of the NICS application with a simplified UML class diagram, and Figure 3 illustrates the UML use case diagram. Finally, section 4 presents the equivalent specification for the use case "uc_3_SoftRobot".

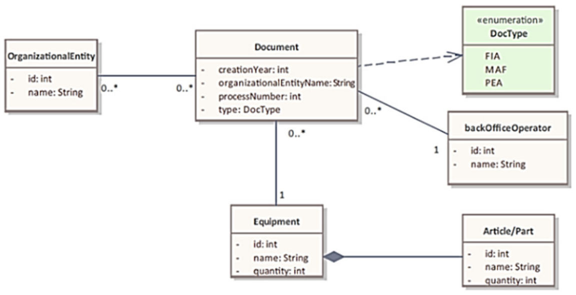

Figure 2: Domain model of the NICS (UML class diagram).

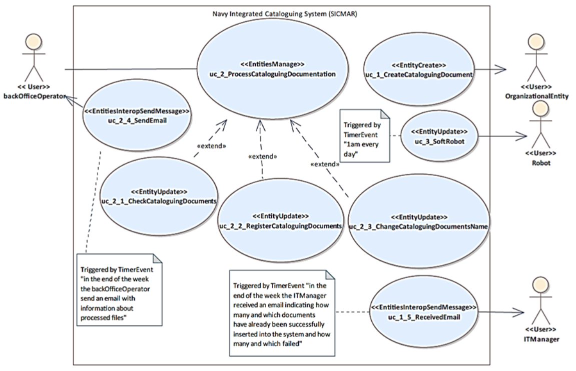
Navy Integrated Cataloguing System (SICMAR)
<< User>>
backOfficeOperator
<<EntitiesManage>>
uc_2_ProcessCataloguingDocumentation
<<EntityCreate>>
uc_1_CreateCataloguingDocument
<<User>>
OrganizationalEntity
<<EntitiesInteropSendMessage>>
uc_2_4_SendEmail
Triggered by TimerEvent "1am every day"
<<EntityUpdate>>
uc_3_SoftRobot
<<User>>
Robot
«extend»
«extend»
«extend»
<<EntityUpdate>>
uc_2_1_CheckCataloguingDocuments
<<EntityUpdate>>
uc_2_2_RegisterCataloguingDocuments
<<EntityUpdate>>
uc_2_3_ChangeCataloguingDocumentsName
Triggered by TimerEvent "in the end of the week the backOfficeOperator send an email with information about processed files"
Triggered by TimerEvent "in the end of the week the ITManager received an email indicating how many and which documents have already been successfully inserted into the system and how many and which failed"
<<EntitiesInteropSendMessage>>
uc_1_5_ReceivedEmail
<<User>>
ITManager

Figure 3: Use case model of the NICS (UML notation).

4 ROBOTS SPECIFICATION

This section discusses how to write RPA robots in controlled natural languages that shall be understandable by both technical and non-technical stakeholders. In particular, it discusses two distinct writing styles: based on use case scenarios and based on pseudocode.

4.1 Based on Use Case Scenarios

Spec. 4 describes the NICS robot with the use case scenario writing style as discussed by (da Silva, 2021).

```
UseCase uc_3_NICS_SoftRobot
[…]
Scenario MainScenario (Main):
s1. Robot: Get the list of digitalised documents
from a specific location folder (in a “pdf”
format), with file names outside the standard
format.
S2. Robot: Read and convert each document from
“pdf” into “txt” format, using OCR technology
with a scale of zero.
S3. Robot: Browses the list of Documents and
opens each one of them in “txt” format.
S4. Robot: For each document, extract and insert
in an excel file the following text fragments:
document type, organisational entity
(responsible for document creation), process
number, and creation year.
S5. Robot: In addiction, insert into the excel
file the path where is the document.
S6. Robot: Get the document filename in the
correct standard format from the Excel file and
change the filename of each one.
S7. Robot: If the document filename is in the
correct format, upload the file into the data
store of the NICS application and move it to the
Processed Document Folder.
S8. Robot: If the document file has an incorrect
format filename, move it to the Failed Document
Folder.
S9. Robot: Send an email to IT Manager
specifying successful documents inserted into
the system and how many have failed.
```

Spec. 4: Use cases scenario specification.

4.2 Based on Pseudocode

Spec. 5 describes the NICS robot’s main steps based on the discussed pseudocode notation.

First, the partial text corresponds to the declaration and initialisation of variables.

Second, it is possible to visualise the reading and conversion of each document from “pdf” into “txt” format, using OCR technology with a scale of zero.

Third, due to the problem of files having different designations, for the same information (i.e., for the document number parameter, in a document we have <Nº: 12345> and in others <Number: 12345>), it is necessary to standardise the expressions and to give the same expressions to all documents, in this case, whenever “Number:” appears, it becomes “Nº:”.

Fourth, define regular expressions to extract the desired data, e.g., <strType>_<strEntity>_<strProcessNumber>_<strCreationYear>.

Fifth, extracte and insert data into the Excel file.

Sixth, obtains the new filename of the document file in the correct standard format in the excel file.

Seventh, if the filename is in the correct format, then the filename is changed. After that, the document file shall be uploaded to the system and moved to the processed documents folder. Otherwise, if the filename has an incorrect format, it is moved to the Failed Documents folder.

Eighth, send an email to IT Manager specifying successful documents inserted into the system and how many have failed.

```
Soft Robot NICS:
//1: declaration of variables
  pdfPath = Environment.CurrentDirectory
  pdfFiles = Directory.GetFiles(pdfPath,"*.pdf")
  totalNumberOfPdfFiles = 0
  numberOfFinishedPdfFiles = 0
  numberOfFailedPdfFiles = 0
  extratedText, strtype, strNumber, strEntity,
strCreationYear, strOC, newPdfFileName,
oldPdfFileName

//2: read and convert document file from
"pdf"//into "txt" format
begin
  FOR each pdfFile In pdfFiles
   […]
    READ pdfFile with OmniPage OCR (SCALE(0))
    WRITE extractedText
  END IF

//3: replace wrong format information in the
//document file
ExtratecdText = strText.Replace("Nº:","Number:")
ExtratecdText = strText.Replace("From.","From:")
ExtratecdText = strText.Replace("'","")

//4: extract information from the document file
//using regular expressions
IF String.IsNullOrEmpty(strCreationYear)
  StrCreationYear =
System.Text.RegularExpressions.Regex.Match(strTe
xt,"(?i)(?<=Data:\s)(\d{2}.\d{2}.\d{4})").Value
END IF

//5: write extracted information from the
//document file into the excel file
    WRITE strType
    WRITE strEntity
    WRITE strProcessNumber
    WRITE strCreationYear
//6: read new filename in the excel file
    READ     newPdfFileName
  IF cell.Length < 17 OR cell.Length > 19 OR
cell.Contains("/") OR String.IsNullOrEmpty(cell)

//7: change document filename and move it to a
//specific folder
   MOVE pdfFile INTO FailedPdfsFolder
      numberOfFailedPdfFiles =
numberOfFailedPdfFiles + 1
      totalNumberOfPdfFiles =
totalNumberOfPdfFiles+1
```

```
  ELSE
    RENAME (oldPdfFileName, newPdfFileName)
    move pdfFile INTO FinishedPdfsFolder
      numberOfFinishedPdfFiles =
numberOfFinishedPdfFiles + 1
      totalNumberOfPdfFiles =
totalNumberOfPdfFiles + 1
  END IF
 END FOR

//8: send an email to the IT Manager
 SEND EMAIL
end.
```

Spec. 5: Specification-based on Pseudocode.

4.3 Discussion

This analysis suggests that both notations could be suitable for the purpose. In particular, the use case scenario notation is simple but has limitations in describing processes and lacks vocabulary, whereas the pseudocode-based is more appropriate for describing algorithms and consequently translated into code. Moreover, the use case scenario notation offered the advantage of ensuring that stakeholders communicate in the same language, as most are non-technical.

Table 1 compares the two writing styles to describe software robots according to suitability, expressiveness, and overall rating criteria (scores according to the following criteria (1=weakest; 6=strongest).

Table 1: Comparative summary for the two notations.

Writing styles	Criteria		
	Suitability	Expressiveness	Overall Rating
Use case scenarios	5	5	5
Pseudocode	3	2	3

5 ROBOT IMPLEMENTATION

This section has two parts, the first refers to some aspects of the NICS robot implementation, and the second part presents an initial comparative analysis.

We decided to start this research by implementing the NICS robot on these three RPA platforms (i.e., on UiPath, Robot Framework, and Robocorp) in a reverse engineering perspective, as well as to understand the difficulties experienced by users in creating a specific scenario on more than one RPA platform.

Figure 4 and Specs. 6 and 7 partially show the implementation of the code on these three platforms. (For more details on these implementation issues, the reader may consult our repository at https://github.com/Martelo39/PIS_RPA).

The focus of this work is not on the comparison of RPA tools but to demonstrate that each one has its differences and specificities. Above all, in some cases, they lack in-depth knowledge of the programming language used by the RPA tool in question.

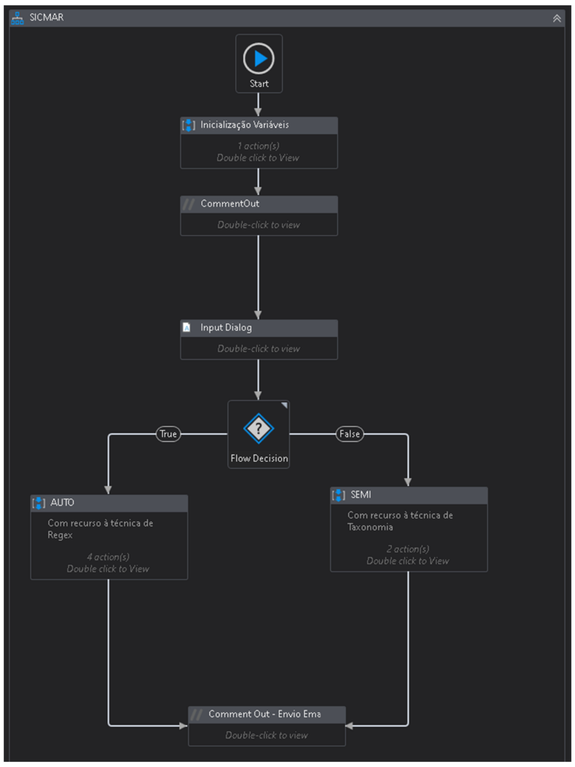

Figure 4: Robot NICS defined in UiPath.

```
*** Settings ***
Library           ocr.py
Library           OperatingSystem
Library           String
*** Variables ***
[…]
*** Test Cases ***
NICS
ocr.OcrTesseract    ./Output/Images
  FOR    ${fileTxt} IN    @{fileNamesTxt}
   ${strText} Get File ./Output/txt/${fileTxt}
  […]
  @{strDate} Get Regexp Matches    ${strText}
(Data: ..-..-..)
  […]
  ${newPdfFileName}    Set Variable
${type}${underscore}${entity}${underscore}${numb
er}${underscore}${creationDate}.pdf
  Move File    ./PDFs/${filePdf}
./Output/CompletedFiles/${newPdfFileName}
  […]
  END
```

Spec. 6: Robot NICS defined in Robot Framework.

```
*** Settings ***
Library         RPA.core.notebook
Library         OperatingSystem
Library         String
Library         ocr.py
*** Variables ***
[…]
*** Keywords ***
Process all digitizing documents
  ocr_tesseract  ./Output/Images
  FOR    ${fileTxt}    IN    @{fileNamesTxt}
    ${strText}=  Get File
./Output/txt/${fileTxt}
    @{strDate}=    Get Regexp Matches
${strText}    (Data: ..-..-..)
    […]
  END
*** Tasks ***
  Process all digitising documents
```

Spec. 7: Robot NICS defined in Robocorp.

Table 2: Comparison of the three used RPA tools.

RPA Tool	Criteria			
	Suitability	Programming Skills	Time Behaviour	Overall Rating
UP	6	5	5	6
RF	3	2	4	3
RC	4	2	3	3

After the assessment, it is possible to analyse the results, summarised in Table 2 (scores according to the following criteria: 1=weakest; 6=strongest). Regarding the criteria Programming Skills: 1 means nice-to-have, and 6 does not require programming skills.

We verify that UiPath is the most suitable tool for beginners because it does not require programming knowledge, and it provides a visual paradigm that is easy to read and write simple models. UiPath also provides a vast number of features, such as the capture-and-play feature that allows recording end-user actions and mimics them in the same manner. However, UiPath is more expensive than the other tools. (For instance, purchasing UiPath for the first time costs an extra $3k for the development environment (Studio) and $6k/year per robot).

Robot Framework and Robocorp can be good alternatives because they are open sources, offer code ownership, and cost-effectively scale without additional overheads. However, it is necessary to know to program, and, for that reason, they are more complex for beginners.

6 CONCLUSION

At the beginning of this project report, we describe RPA technology, Controlled Natural Language based on CNL-B, and Pseudocode notations. First, however, the organisations should identify their appropriate processes for this technology since not all need or shall be automated. The most suitable processes (to use RPA) are repetitive, rule-based, low-complexity, and a high volume of tasks. One of the essential advantages of RPA implementation is the cost and time reduction achieved by the organisations.

The controlled natural language like CNL-B provides basic terms necessary to communicate and shows some limitations in the provided vocabulary, grammar syntax, and verb forms (da Silva, 2021).

Specifications based on pseudocode notations are popular to describe informal and high-level computer programs or algorithms (Oda et al., 2016). Pseudocode writes in symbolic code and translates into a programming language before being executed.

Both CNL-B and Pseudocode allow describing software algorithms, supporting the development of the program, and the maintenance of business processes. In this way, it is possible to put the stakeholders in the same direction, thus ensuring that they communicate in the same language and share a shared vision.

The research discussed in the paper uses a case study that involved the concrete implementation of an RPA scenario in the scope of the NICS (Navy Integration Cataloguing System) application. That scenario was defined in the CNL-B and Pseudocode to decide which of the two notations fits better for describing the RPA scenario. We concluded with this article that using both specifications to describe in an agnostic way the RPA NICS scenario would simplify the development of scenarios, no matter what of the RPA tool used. Furthermore, controlled natural languages allow writing more systematically and consistently and, above all, more straightforward than the specification approach based on pseudocode, a more technical writing approach. Given the above, the main objective is to ensure that stakeholders communicate in the same language, especially non-technical ones. Therefore, we considered a specification approach based on controlled natural languages (CNL-B) as the most adequate.

For future work, we intend to research the following challenges. First, use one of the notations discussed in this article to specify robots and explore transformation mechanisms for proprietary formats (i.e., UiPath, Robot Framework, Robocorp). Second, extend the ASL language (Gamito & da Silva, 2020) to support the rigorous specification of RPA robots based on the Xtext technology (Bettini, 2016; Fowler, 2010). Third, research and develop transformation mechanisms for proprietary formats of UiPath, Robot Framework, Robocorp. Fourth, research how to test RPA robots on top of our recent work on model based testing (Estivill-Castro et al., 2018; Silva et al., 2018; Maciel et al., 2019). Fifth, use and compare other RPA tools, like Blue Prism or Automation Anywhere.

ACKNOWLEDGEMENTS

Work partially supported by funds under FCT UIDB/50021/2020 and 02/SAICT/2017/29360.

REFERENCES

Bettini, L. (2016). *Implementing domain-specific languages with Xtext and Xtend*. Packt Publishing Ltd.

Chakraborti, T., Isahagian, V., Khalaf, R., Khazaeni, Y., Muthusamy, V., Rizk, Y., & Unuvar, M. (2020). From Robotic Process Automation to Intelligent Process Automation. *International Conference on Business Process Management*, 215–228.

da Silva, A. R. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, *43*, 139–155.

da Silva, A. R. (2017). Linguistic Patterns and Linguistic Styles for Requirements Specification (I): An Application Case with the Rigorous RSL/Business-Level Language. *Proceedings of the 22nd European Conference on Pattern Languages of Programs*. ACM.

da Silva, A.R., Paiva, A.C.R., Silva, V.E.R. (2018). Towards a test specification language for information systems: focus on data entity and state machine tests. Proceedings of MODELSWARD'2018.

da Silva, A. R., & Savić, D. (2021). Linguistic Patterns and Linguistic Styles for Requirements Specification: Focus on Data Entities. *Applied Sciences,* vol. 11, no. 9.

da Silva, A. R. (2021). Linguistic Patterns, Styles, and Guidelines for Writing Requirements Specifications: Focus on Use Cases and Scenarios. *IEEE Access, vol. 9, pp. 143506-143530.*

Estivill-Castro, V., Hexel, R., & Lusty, C. (2018). Continuous Integration for Testing Full Robotic Behaviours in a GUI-stripped Simulation. In MODELS Workshops.

Fowler, M. (2010). *Domain-specific languages*. Pearson Education.

Gamito, I., & da Silva, A. R. (2020). From Rigorous Requirements and User Interfaces Specifications into Software Business Applications. *International Conference on the Quality of Information and Communications Technology*, Springer.

Hocenski, Z., & Stresnjak, S. (2011). *Usage of Robot Framework in Automation of Functional Test Regression*.

Houy, C., Hamberg, M., & Fettke, P. (2019). Robotic process automation in public administrations. *Digitalisierung von Staat Und Verwaltung*.

Hüller, L., Jenß, K. E., Speh, S., Woelki, D., Völker, M., & Weske, M. (2021). *Ark Automate—an Open-Source Platform for Robotic Process Automation*.

Jovanović, S. Z., urić, J. S., & Šibalija, T. v. (2018). Robotic process automation: overview and opportunities. *Int. J. Adv. Qual*, *46*(3–4), 34–39.

Maciel, D., Paiva, A. C., & Da Silva, A. R. (2019). From Requirements to Automated Acceptance Tests of Interactive Apps: An Integrated Model-based Testing Approach. In ENASE'2019.

Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., & Nakamura, S. (2016). Learning to generate pseudocode from source code using statistical machine translation. *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, 574–584. https://doi.org/10.1109/ASE.2015.36

Robocorp. (2021, November 15). *Open Source RPA Solutions*. Https://Robocorp.Com/Solutions.

Romao, M., Costa, J., & Costa, C. J. (2019). Robotic process automation: A case study in the banking industry. *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6.

Roveda, L., Ghidoni, S., Cotecchia, S., Pagello, E., & Pedrocchi, N. (2017). *EURECA H2020 CleanSky 2: a Multi-Robot Framework to Enhance the Fourth Industrial Revolution in the Aerospace Industry CNR-ITIA Calibration View project EURECA H2020 CleanSky 2: a Multi-Robot Framework to Enhance the Fourth Industrial Revolution in the Aerospace Industry*.

Roy, G. G. (2006). Designing and Explaining Programs with a Literate Pseudocode. *J. Educ. Resour. Comput.*, *6*(1), 1–es. https://doi.org/10.1145/1217862.1217863

Saikat Ray, A. V. N. R. P. V. K. G. M. A. (2021, November 18). *Magic Quadrant for Robotic Process Automation*. https://www.gartner.com