ISCTE IUL
Instituto Universitário de Lisboa

Department of Information Science and Technology

Demonstration-based Help for Interactive Systems

Pedro Miguel Freitas Rodrigues

A Dissertation presented in partial fulfillment of the Requirements for the Degree of Master in Computer Engineering

Supervisor:
PhD. José Luís Cardoso da Silva, Assistant professor,
ISCTE-IUL

Co-supervisor:
PhD. Rúben Filipe de Sousa Pereira, Assistant professor,
ISCTE-IUL

October, 2018

ISCTE IUL
Instituto Universitário de Lisboa

Department of Information Science and Technology

Demonstration-based Help for Interactive Systems

Pedro Miguel Freitas Rodrigues

A Dissertation presented in partial fulfillment of the Requirements for the Degree of Master in Computer Engineering

Supervisor:
PhD. José Luís Cardoso da Silva, Assistant professor,
ISCTE-IUL

Co-supervisor:
PhD. Rúben Filipe de Sousa Pereira, Assistant professor,
ISCTE-IUL

October 2018

ABSTRACT

The usability of day-to-day applications is of utmost importance. The lack of usability of these is one of the causes of frustration at work, as it creates barriers to the execution of tasks. The Change of applications by third parties, to increase usability, is difficult because it requires, usually, access to source codes and an increase on its complexity. This work proposes and implements a demonstration help tool that allows the improvement of tasks completion, decreases the time spent, and reduces the cost of learning. An analysis of work on aid tools is presented identifying positive aspects and research opportunities. The help tool developed allows the creation of automation through picture-driven computing, which makes it possible to develop help mechanisms independent from application source codes. Since the tool is image oriented, and tasks can involve multiple applications, it is also possible to develop help scripts that are not restricted to just one application. User studies were done with the objectives of validating the work developed and identifying platforms and tasks with usability problems in the business world. It was concluded that the work has positive effects in the accomplishment of tasks.

Keywords: Demonstration-based help; Picture-driven Computing; Automation; Usability.

This page was intentionally left in blank

RESUMO

A usabilidade das aplicações utilizadas no dia-a-dia é de extrema importância. A falta de usabilidade destas é um dos causadores de frustração no trabalho, pois cria barreiras à execução de tarefas. A alteração das aplicações por terceiros de forma a aumentar a usabilidade é difícil pois requer, usualmente, acesso aos códigos fonte e incremento da sua complexidade. Este trabalho propõe e implementa uma ferramenta de ajuda por demonstração que visa melhorar o sucesso na realização de tarefas, reduzir o tempo despendido, e diminuir o esforço de aprendizagem. Uma análise a trabalhos sobre ferramentas de ajuda é apresentada identificando aspetos positivos e oportunidades de investigação. A ferramenta de ajuda desenvolvida permite a criação de automações através de computação por imagem, que tornam possível o desenvolvimento de mecanismos de ajuda independentes dos códigos fonte das aplicações. Sendo que a ferramenta é orientada a imagem, e que as tarefas podem envolver múltiplas aplicações, torna-se também possível o desenvolvimento de scripts de ajuda não restringidos a apenas uma aplicação. Foram realizados estudos com utilizadores com os objetivos de validar o trabalho desenvolvido e identificar plataformas e tarefas com problemas de usabilidade no meio empresarial. Deste modo, concluiu-se que o trabalho tem efeitos positivos na realização de tarefas.

Palavras-chave: Ajuda por demonstração; Computação por imagem; Automação; Usabilidade.

This page was intentionally left in blank

ACKNOWLEDGEMENTS

I would like to express my appreciation to professors José Luís Silva and Rúben Pereira, my supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of the research work. I would also like to thank them for the assistance in keeping my progress on schedule and for always being available.

My special thanks to my girlfriend, that has been part of the entire process. Always willing to hear me and give her most honest opinions. For the support and encouragement on all stages of the dissertation.

I want to thank my parents and my sister for being the support for all my studies. For backing up all my decisions and being by my side on the good and less good moments.

I would also like to acknowledge the support from everyone working on the company where the user studies were made. For being great colleagues and always showing willingness to help.

This page was intentionally left in blank

INDEX

This page was intentionally left in blank

TABLES INDEX

This page was intentionally left in blank

FIGURES INDEX

LIST OF ABBREVIATIONS AND ACRONYMS

CHAIN	Contextual Help for Adaptive INterfaces
CTT	*ConcurTaskTrees*
DS	Design Science
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HILC	Help, It Looks Confusing
IDE	Integrated Development Environment
ISD	Instructional System Development
ISI	Interactive Systems Integration
IT	Information Technology
PDC	Picture-Driven Computing
RPA	Robotic Process Automation
UI	User-Interface

This page was intentionally left in blank

Chapter 1

INTRODUCTION

This chapter introduces the subject of the dissertation, the motivation for investigating and the questions and goals which are focused. The methodology adopted, and the structure of the dissertation is also presented.

1.1. OVERVIEW

The usability of information is vital for websites, products and services to achieve success. Managers and developers often recognize it but still leave it apart from the product-development process (Rosenbaum & Ramey, 2014; Winter, Rönkkö, & Rissanen, 2014). Although all advances on Human-Computer Interaction (HCI) some interfaces are still difficult to use, and these cause frustration to users (Lazar, Jones, & Shneiderman, 2006).

In addition to system design and respective User-Interfaces (UIs) failures, it is still verified a major difficulty in making people use help systems, either by not recognizing their existence, not wanting to pause their current tasks to search help or because they want to find the solution on their own (Dworman & Rosenbaum, 2004). Nowadays, with new approaches to aid, support may be more accessible and used. However, the design of more efficient user-centered help systems is still seen as a weak negotiator point by the business because it is faced as a secondary functionality (Dworman, 2007; Winter et al., 2014), see Figure 1.

To combat help systems resistance and still make users familiar with tools, especially the ones with some complexity, training programs focused on task modelling were proposed aiming to understand which are the key aspects that depend on the user (Célia Martinie, Palanque, Navarre, Winckler, & Poupart, 2011). This modelling has, yet, the potential to improve systems design if considered in the implementation process (Navarre, Palanque, Ladry, & Barboni, 2009). However, modelling has a certain level of complexity and some tools propose to overlap it by analyzing the task being accomplished and defining it (Machado, Lopes, Silva, & Silva, 2017).

Objectifying the improvement of aid systems, contextual help proved to be effective in learning user interfaces. Contextual-help defines the paradigm of help given in the context where the doubt has risen up, e.g. a certain tooltip associated to button (Akiki, 2018). This concept is then like the situated action theory which states the knowledge can only be interpreted in its context. From this theory help tools, like AIDE which through communication with the users discovers and solves their doubts, have emerged (Vouligny & Robert, 2005).

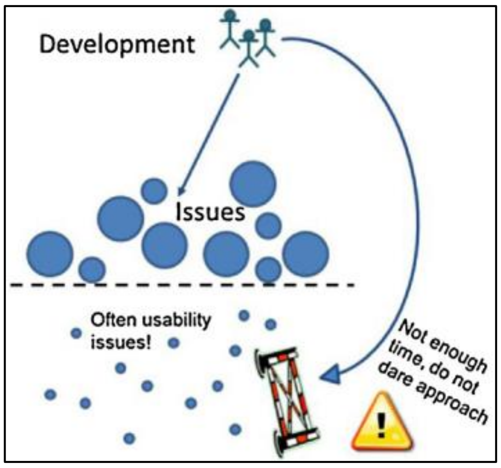


Figure 1 – Approach to usability issues. Study on a big company with usability set as one of its focus points (Winter et al., 2014).

Different help approaches have been developed. Either by using video, tutorials, functionalities as automation and picture-driven computing (PDC), between others (Grossman & Fitzmaurice, 2010; Pongnumkul et al., 2011; Vouligny & Robert, 2005; Yeh et al., 2011; Yeh, Chang, & Miller, 2009). Some also suggest approaches based on crowd learning, like help forums, arguing that people learn better when discussing subjects and that an one-to-many support is more efficient (Chilana, Ko, Wobbrock, & Grossman, 2013; Sun, Qiao, Chen, Xin, & Jiao, 2016).

Other interesting approach is the demonstration-based help that in addition to working in the context of the application demonstrates how a certain task is to be executed, e.g. through automation scripts (Silva, Ornelas, & Silva, 2016).

Part from this dissertation will be the development of a demonstration-based help tool. The tool will be developed and then tested on a technology company, called from now on Technological. Through an internship on this company, applications and tasks from their routines will be studied and, the most appropriated ones will be targeted to conduct user tests. This company has chosen not to reveal its identity.

1.2. MOTIVATION

Many of the current help systems are still based on user manuals with text and images or do not provide inter-application support. One of the major barriers to an overall improvement is the increase on the systems complexity (Vogel-Heuser et al., 2014) or the impossibility to access and change source codes.

With current technology it is possible to compute on a picture-driven (concept further explained on section 2.1. PICTURE-DRIVEN COMPUTING) way, which enables the interaction with pixels shown on screens, allowing inter and intra applications interaction and not accessing source code, or adding complexity to it.

Sikuli is an automation tool that uses image recognition to identify and control Graphical User Interface (GUI) components (‘Sikuli Script - Home’, n.d.). Using Sikuli, DemoHelp is a tool that allows real system interaction between different applications, enabling contextual help. The help is provided by the autonomous realization of the desired task, allowing the user to learn by observation. Additionally, DemoHelp can improve efficiency as the user is learning and the task is being completed (Ornelas, Silva, & Silva, 2016).

DemoHelp has many capabilities but has still space to be improved. Script generation is based on task modeling, which, as stated, is of some complexity and may be a barrier for unfamiliar users. This represents an important point of change since UIs are not static and scripts should be easily updated. Furthermore, image recognition has some challenges when pixel variation occurs, for example a simple change on the desktop theme, which is a barrier to script sharing (Ornelas et al., 2016).

When automating processes, most tools require users to have some level of experience and therefore companies must dedicate skilled manpower. When deciding what to automate, the greater good is considered and, in most cases, the automation is dedicated to complex corporate processes and single users and/or employees’ needs, different from one to another, are left behind.

Studies from 2006 denote that user frustration due to computer applications is a problem. On workplaces, users lose more than 40% of their time recovering from problems caused by computers and its applications (Lazar et al., 2006). Considering that some years have passed, it is thought that these values are smaller.

Researchers on the area have shown that a great portion of computer users suffer from negative affective reactions towards computers and, frustration, independently if in organizations or single individuals, can lead to maladaptive behaviors that can subsequently lower effective goal-oriented behavior (Ceaparu, Lazar, Bessiere, Robinson, & Shneiderman, 2004). Responding to frustration individuals can either adapt, solving the problem that is blocking the goal achievement and being constructive, or maladapt, making the frustration experience worst and creating additional problems (Shorkey & Crocker, 1981).

This frustration is highly related with lack on the usability of applications that are often designed with interfaces that are hard to use or features that are hard to find, despite managers and developers recognizing the role of information or content in overall usability of successful products and services (Lazar et al., 2006; Rosenbaum & Ramey, 2014).

On 2014 a study investigated the connections between usability efforts and organizational factors. The principal obstacles (see Figure 2 for a synthesis of the obstacles found) identified were (Winter et al., 2014):

- Unexpected changes to the interface design may be added;
- Different organization levels may have divergent interest regarding resources use;
- Usability issues are often considered minor problems;
- Complexity makes so that small changes have unforeseen effects on other parts of the product, so changes must be as little as possible;
- There is reluctance to allow users to influence the product;
- There is low understanding of what users really need;
- People who understand end-users are in positions without power to push changes;
- There is a wide range of end-user and it becomes difficult to understand how the real end-user is;
- There is insufficient knowledge of how the product is used in the end-user's context.

For the development of usable interfaces some of the challenges found on real projects pass through the need to raise help systems from secondary priorities, the necessity to properly plan how to resolve competing interests, the need to involve all stakeholders on a brainstorm even before the first mock ups, and the implementation of interactive designs (Dworman, 2007).

Regarding help systems, investigators find much resistance from users to use them. Some of the found reasons were that users appeared to not see help even when it is in front of them (cognitive blind spots), users are not willing to leave/pause their tasks to search help, users fear to leave task unfinished, they feel they can find a solution by them, users are more willing to access hints, tips, and quick-reference guides by avoid clicking on something names ‘help’ (Dworman & Rosenbaum, 2004).

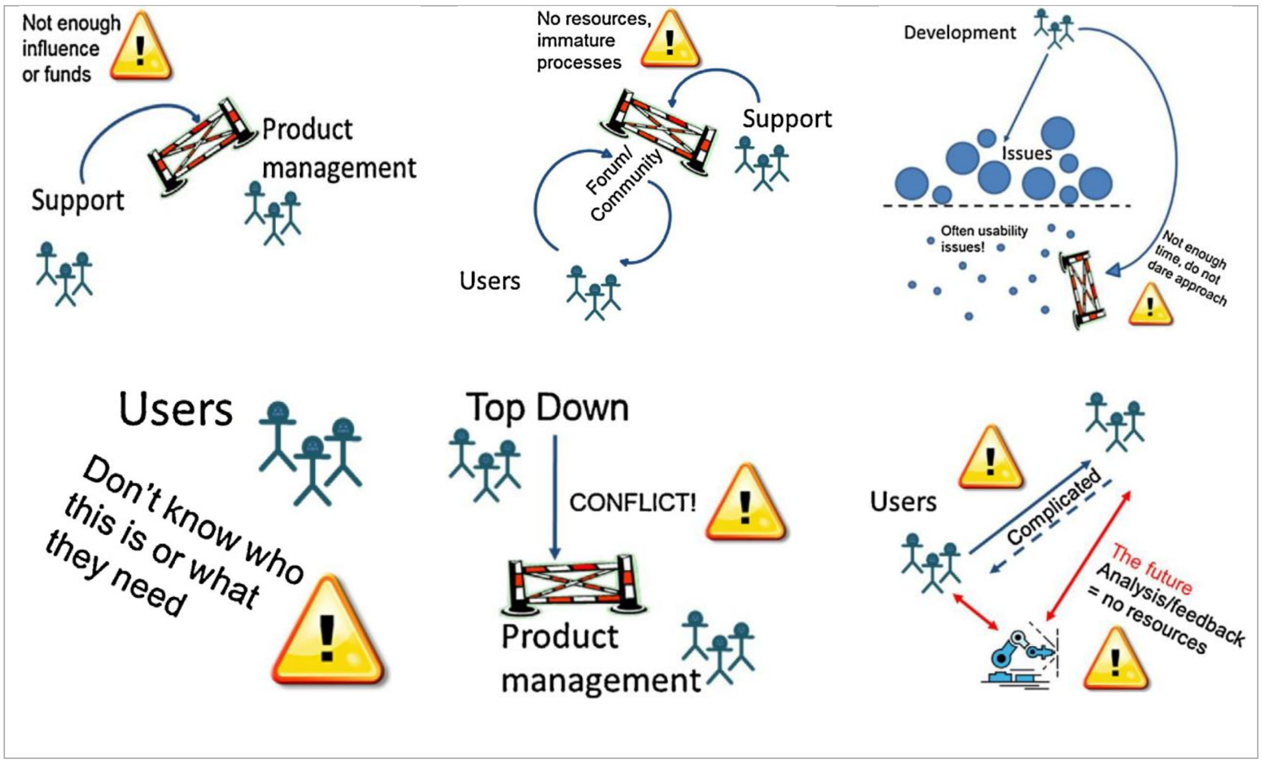

Not enough influence or funds
Product management
Support
No resources, immature processes
Support
Forum/ Community
Users
Development
Issues
Often usability issues!
Not enough time, do not dare approach
Users
Don't know who this is or what they need
Top Down
CONFLICT!
Product management
Users
Complicated
The future
Analysis/feedback = no resources

Figure 2 - Usability main obstacles on big companies (Winter et al., 2014)

1.3. GOALS AND RESEARCH QUESTIONS

In this work it is objectified the development of a contextual help tool. The tool takes advantage of Sikuli functionalities, especially the ability to automate via pixel recognition and template matching algorithms, and aims to improve DemoHelp, more concretely the way scripts are created and updated, generating them from a one-time execution of the task (one-shot learning).

Distributed and abstractive capabilities are also a high desired feature which requires improvements on script conception, since the environmental context must be as generic as possible. Reaching these goals, a script generated on one computer will run on any computer and the information related to the UI elements will be generic (e.g. the icons that drive the automation will have its background changed to transparent – theme independency).

With this, the creation of a tool that does not require its users to have experience on automation or script conception is expected, improving current help tools that work with object recognition and providing the closest to one-shot learning.

It is intended to evaluate the tool with users, for which study cases, web-based, desktop-based and mixed, will be created. These will be consistent with usual daily activities of employees of Technological and will enable inter/intra applications interaction. When assessing is expected a measurement of the learning process compared with previous help systems, verifying if it is improved. Usability tests are also planned, aiming at the development of a user-friendly tool.

It is expected that people make regular use of the tool and use it to create and share their own scripts.

Reaching these goals, script generation from a one-time execution of the task, generic scripting (independent from each computational context), proper evaluation of the contextual-help tool, and good usability, we aim to comprehend the impact of a demonstration-based help tool based on PDC and automation and have answer to the following question:

- Does a help tool based on demonstration, automation and PDC improve the success of task completion, time spent, and decreases the cost of learning?

1.4. METHODOLOGY

The work to be done will mainly follow the Design Science (DS) Research Methodology, see Figure 3, for Information Systems Researches presented in Peffers et al. (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). In it, it is stated that there are six steps in the DS process: problem identification and motivation, definition of objectives for a solution, design and development, demonstration, evaluation and communication.

Designing science consists on the creation and evaluation of the information technology (IT) artifacts and is intended to solve observed problems, make research contributions, evaluate designs, and communicate results. These artifacts may include constructs, models, methods, instantiations, or new properties of technical, social, and/or informational resources (Hevner & Chatterjee, 2010; Järvinen, 2007).

Problem identification, motivation, and goal setting for a solution have been introduced in previous sections. These emerged from a state-of-the-art review, properly presented on chapter two and three, which emphasizes a resistance on the usage of most classic help systems, mainly textbooks, and shows learning benefices on contextual-help tools.

In the development of these tools, independence from user computational skills and usability are sook and considering that the barrier to a solution that properly resolves the problem is complexity overload or denied accesses to the main applications source codes recent technologies like PDC pose as potential problem solvers.

For design and development, the tool will be developed based on picture-driven and automation tools. A proof of concept will be developed to validate the approach envisaged.

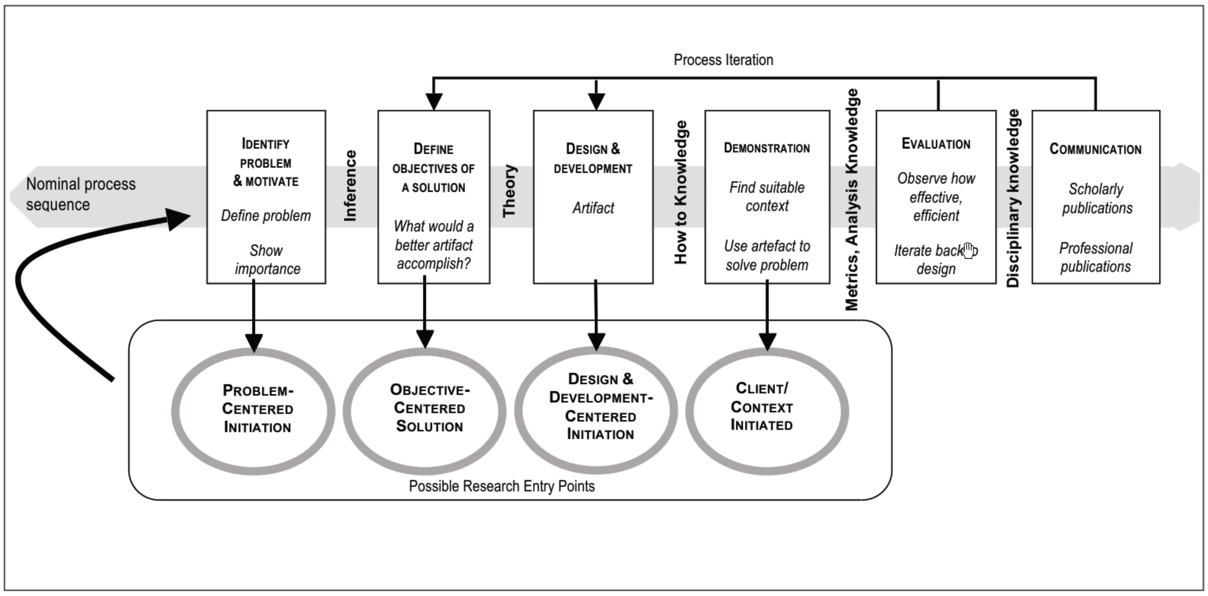

Process Iteration
Nominal process sequence
Identify problem & motivate
Define problem
Show importance
Inference
Define objectives of a solution
What would a better artifact accomplish?
Theory
Design & development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artefact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development-Centered Initiation
Client/Context Initiated
Possible Research Entry Points

Figure 3 - Design Science Research Methodology Process Model (Peffers et al., 2007)

The tool will consist on two main modules. The first aiming to simplify script generation, by gathering information of a task being accomplished and collecting user inputs, generating the correspondent script. The second a script runner, where we will incorporate Sikuli and take advantage of its previous stated capabilities.

To demonstrate the tool, interaction examples will be developed and distributed. With these it is expected that the capabilities of the tool will be understood. The ease of script creation, by executing only one time a task, and script execution, leading to automation and learning by observation, will be emphasized.

To evaluate the work, the assumptions made will be assessed, as well as its usefulness, satisfactions and ease of use will be measured, following usability measuring practices (Hornbæk, 2006). The evaluations will be conducted with employees from Technological. This evaluation will start with an interview process to collect which daily and periodic tasks and applications are the best candidates to be incorporated on our study.

The communication of the work developed is expected by the availability of this dissertation report and with the publications of scientific articles. With these, it is expected to contribute for a better daily usability on computer-based tools and to the acknowledgement of its importance by managers and software developers.

1.5. DISSERTATION STRUCTURE

This dissertation is structured in six chapters. In these the different work phases are described, beginning on this one with an introduction and ending with work discussion and conclusions.

The second chapter reflects the background. Related subjects such as PDC, automation and tasks models, and usability and frustration will be described. These are the topics considered important for the development of an automation and PDC based tool and its understanding. On the PDC field some market application will be presented with the goal of verifying the way automation and image recognition is handled.

In the third chapter, related work is presented, aiming to properly verify what approaches were taken and what the main improvement opportunities are. A review on related help solutions is presented sorting them into three main categories, scripts and automation-based, videos-based and GUI element-based. A summary table with the main characteristics of the works is presented and a critical conclusion made.

The fourth chapter is dedicated to the specification of the adopted approach when developing the tool. This chapter is divided on three sections. The first section described the proof of concept, a preliminary tool developed and tested with 10 users gathering good results in what concerns usability and learning through the approach being studied. This proof-of-concept has also resulted on the publication of a scientific article (Rodrigues, Silva, & Pereira, 2018). The second section provides the descriptions of the two main modules of the developed tool, the Recorder, which records scripts and shares them, and the Player, which shows script information to all users and allows their execution. On the last section the implementation is described. This description was divided into three sub-sections, the first explaining the way scrips are recorded, the second explaining how scripts are played and the last with information about how the interface was developed.

The fourth chapter is dedicated to the specification of the adopted approach when developing the tool. It starts with the presentation of a proof-of-concept and its results. This section is followed by the description of the implemented two main modules and then the description of the implementation, how it was done, and which resources were used.

The fifth chapter presents the evaluations and results of the practical cases in the company Technological. The first evaluation was through a questionnaire and identified platforms and tasks that most required a help tool. The second, consisted on the execution of multiple tasks by two different groups. The usability and viability of the developed tool and improvement on task performance are analyzed.

On the sixth chapter the discussion and the conclusions are drawn.

This page was intentionally left in blank

Chapter 2

BACKGROUND

This chapter presents a description of the topics most important for an adequate dissertation understanding. These are PDC, automation and task models.

This chapter presents the background information necessary to better understand the work presented. The first section presents PDC and gives special attention to Sikuli and to Robotic Process Automation (RPA) software. On the second section Automation and Task Models are presented.

2.1. PICTURE-DRIVEN COMPUTING

Kourousias & Bonfiglio present the visual information in a 2-dimensional spatial domain as nuclear data of the PDC paradigm. This visual information is what intends to be visible on a graphical output device (Kourousias & Bonfiglio, 2010). This paradigm is useful for task automation of systems already developed and/or without access to their source code (Silva et al., 2016).

2.1.1. Sikuli

PDC is implemented on a programming environment named Sikuli (Kourousias & Bonfiglio, 2010). This is an open-source research project developed at the User Interface Design at MIT. It was developed using python and uses the OpenCV package for finding images on the screen.

The tool allows for the standard programming functions (e.g. loops, conditions, variables) but with the addition of target patterns identification on the screen. This tool was developed to empower help designers in the general public (computer-skill independent) to create contextual help (Yeh et al., 2011).

At design time, the user can select a component from the interface by capturing its' screenshot and selecting the type of contextual help to add (available action to perform on specified UI elements) to that component, some examples are highlights, clicks, and type into (which presses the item and writes a specified message). When presenting the help to the user pixies on his screen are searched and the element is located, there the selected action is reproduced (Yeh et al., 2011).

To use the tool, users can take advantage of the SikuliX Integrated Development Environment (IDE) that is a basic script editor to load, edit, save and run scripts. On it, it is presented on a left bar the main image related capabilities sorted in categories like Sikuli settings, finding elements, mouse actions and keyboard actions. On the top bar shortcuts to take screenshots, insert images, create regions, run the script and on the center and bottom right a zone where the script commands are written. In what regards non-picture related functions users must have previous basic computer programming basis. On Figure 4 it is presented a screenshot of the IDE and on Figure 5 an example of a script developed on the IDE.

2.1.2. Robotic Process Automation

"Robotic Process Automation software automates repetitive, rule-based processes usually performed by people sitting in front of computers". The interaction with computers is done by mimicking the interaction between humans and computers and therefore robots can open email attachments, complete e-forms, and record and replay data (Schatsky, Muraskin, & Iyengar, 2016).

In order to mimic user interactions, current robots operate in two main modes. Either they are familiar with applications, usually through communication plugins, and can interact directly with all UI elements (e.g. buttons, text boxes, drop menus, combo boxes), or they operate through PDC using also screenshots and image recognition techniques.

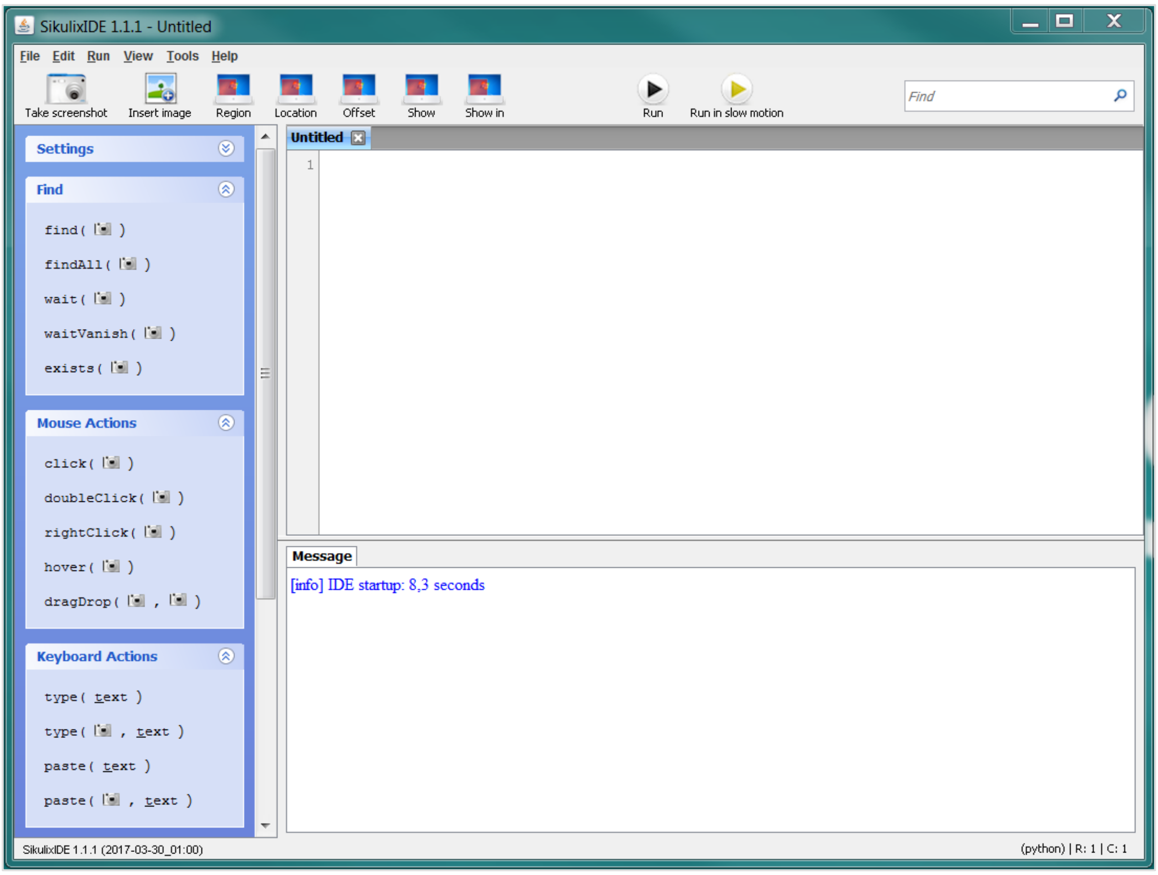
Most common RPA automations are designed for companies which recognize the existence of tasks that pose as good candidates to be automated. Part from the study done on the Technological was the definition of these kinds of tasks and the exploration of the available tools, focusing the automation through PDC to understand how are these softwares implementing it.

On Figure 6 a synthesis of RPA goals and its description is presented. This information was gathered from one of the market RPA applications. The goals emphasized are Increased Customer Satisfaction, Better Resource Utilization, Increased Accuracy and Improved Productivity.

The following two subsections present the applications used on Technological while investigating the capabilities of RPA applications. The way PDC is implemented was the focus of the application testing. Next UIPath ('UiPath Robotic Process Automation', 2005) and Blue Prism ('Blue Prism - Robotic Process Automation', n.d.) are presented.

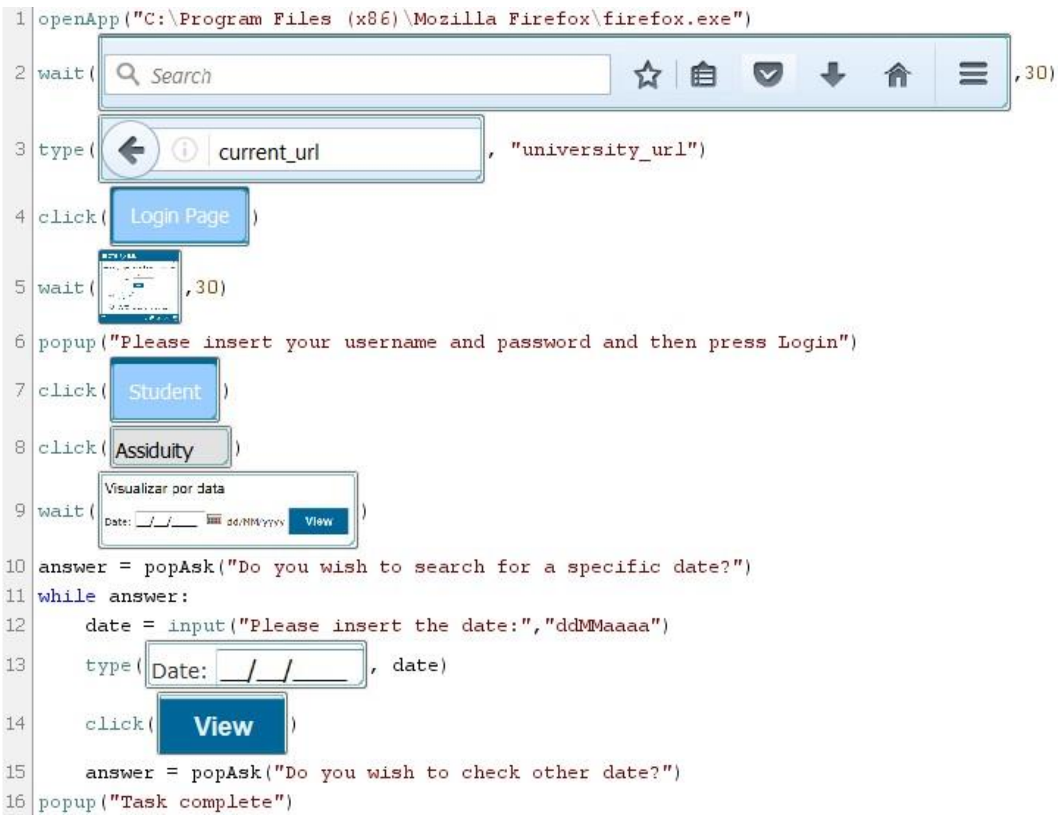
2.1.2.1. UIPath

UIPath was the first RPA software used on Technological. It has been developed with the goal to design and deliver the most advanced automation platform to business of all sized, business process outsourcing providers and shared service organizations and under the mission that human work should be creative and inspiring ('UiPath Robotic Process Automation', 2005).



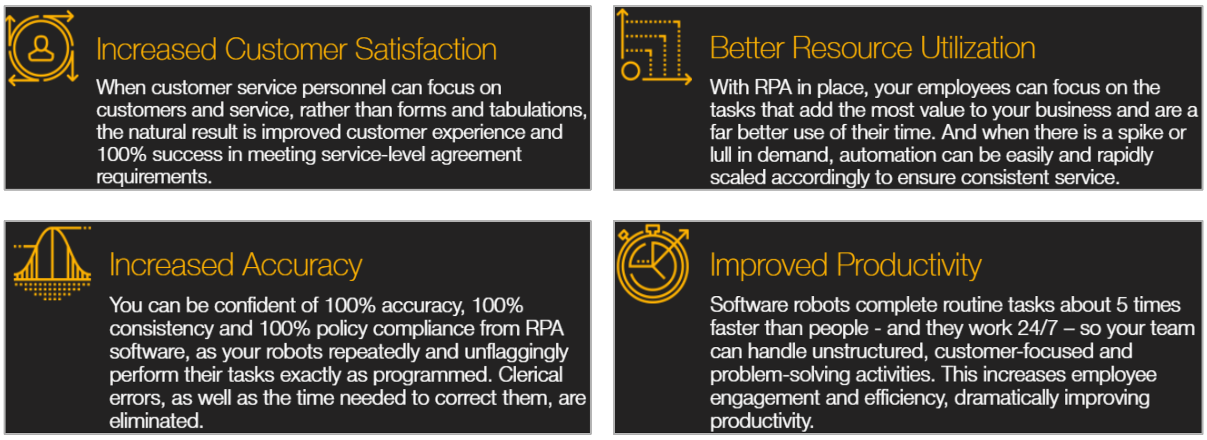
```
[info] IDE startup: 8,3 seconds
```

Figure 4 - Sikuli IDE



```
openApp("C:\Program Files (x86)\Mozilla Firefox\firefox.exe")
wait( ,30)
type( , "university_url")
click( )
wait( ,30)
popup("Please insert your username and password and then press Login")
click( )
click( )
wait( )
answer = popAsk("Do you wish to search for a specific date?")
while answer:
    date = input("Please insert the date:","ddMMaaaa")
    type( , date)
    click( )
    answer = popAsk("Do you wish to check other date?")
popup("Task complete")
```

Figure 5 - Example of a Sikuli script



Increased Customer Satisfaction

When customer service personnel can focus on customers and service, rather than forms and tabulations, the natural result is improved customer experience and 100% success in meeting service-level agreement requirements.

Better Resource Utilization

With RPA in place, your employees can focus on the tasks that add the most value to your business and are a far better use of their time. And when there is a spike or lull in demand, automation can be easily and rapidly scaled accordingly to ensure consistent service.

Increased Accuracy

You can be confident of 100% accuracy, 100% consistency and 100% policy compliance from RPA software, as your robots repeatedly and unflaggingly perform their tasks exactly as programmed. Clerical errors, as well as the time needed to correct them, are eliminated.

Improved Productivity

Software robots complete routine tasks about 5 times faster than people - and they work 24/7 – so your team can handle unstructured, customer-focused and problem-solving activities. This increases employee engagement and efficiency, dramatically improving productivity.

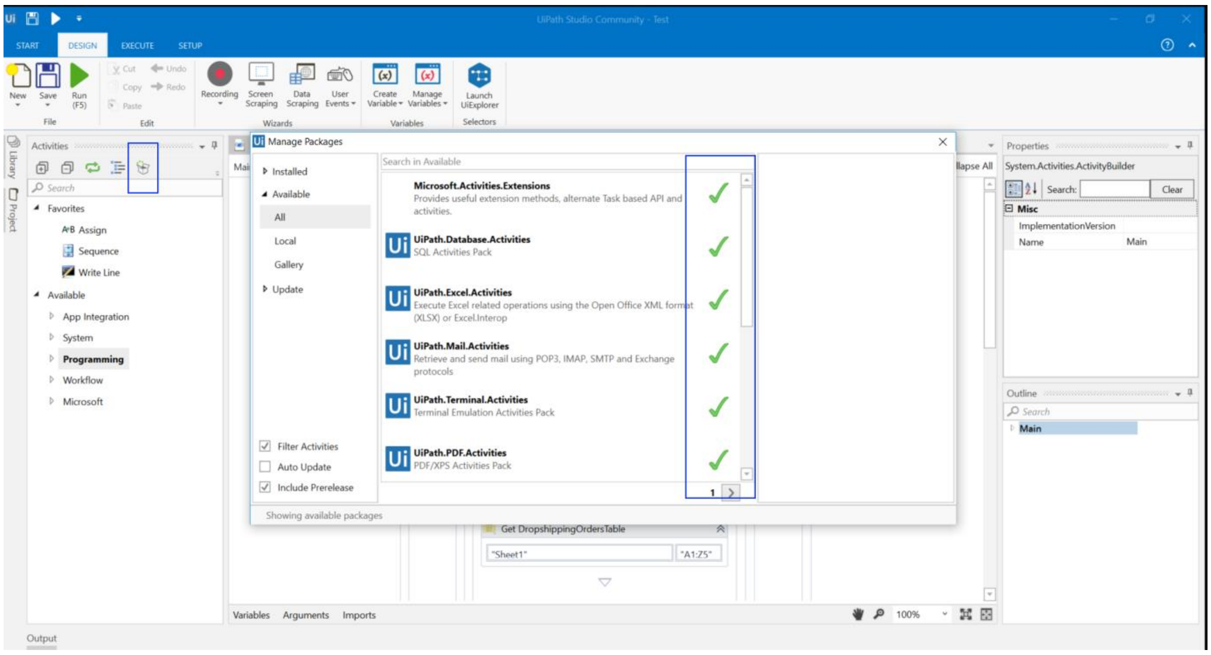
Figure 6 - Synthesis of RPA goals[1]

This software can interact with Office applications (Excel and Word), CSV formatted files, databases, terminals, desktop (interacting with files and folders), Web (particularly with Internet Explorer, by its ability to make HTTP and SOAP Requests), and PDF. This is done through packages that are imported to the IDE. Additionally, it works with PDC.

On Figure 7 a screenshot of UIPath's interface is presented. The top bar implements the functions regarding the creation, saving, and running of the developed workflow, wizards that allow the user to easily record the interaction with the interface (there are four modes, basic, desktop, Citrix and web), data scraping and some user events (e.g. keyboard interactions with the system). Also, some variable related functions as an UI explorer that allows the user to see how the elements are classified by UIPath is available, see Figure 8, where it is presented the classification for the search button on Google's main page[2]. On the left bar there is the activities bar. An activity is a function that is already in UIPath (e.g. 'Message Box', creates an information box that is displayed to the user). On the center, users create their workflow that can be structures in sequences or flowcharts. The right bar allows configuring the activities parameters and provides a structure of the developed workflow.

Focusing the way that PDC is implemented, there are two main ways to program the workflow: using the activities on the activities bar or using Citrix record mode (script generated from a one-time execution of the activities). In both the generated workflows and their activities are similar, the difference is that on Citrix mode it is created an abstraction from the IDE unlike when using the activities from the activities bar on the IDE. On Figure 8 it is show the main categories for UI Automation where the activities are sorted and on Figure 10 it is presented the displayed GUI when using the Citrix mode.

[1] Retrieved on January 2018, from https://www.nice.com/websites/rpa/index.html#whats_new
[2] Accessed on March 2018, www.google.com



Microsoft.Activities.Extensions
Provides useful extension methods, alternate Task based API and activities.

UiPath.Database.Activities
SQL Activities Pack

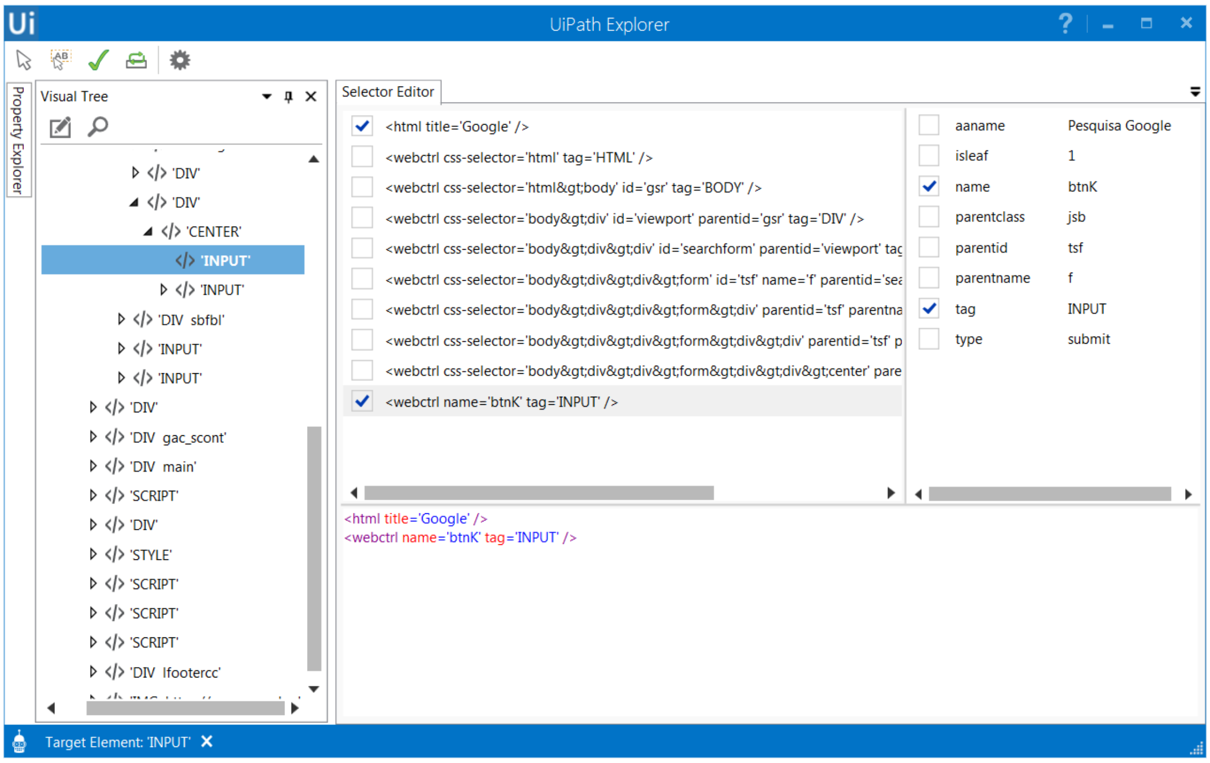
UiPath.Excel.Activities
Execute Excel related operations using the Open Office XML format (XLSX) or Excel.Interop

UiPath.Mail.Activities
Retrieve and send mail using POP3, IMAP, SMTP and Exchange protocols

UiPath.Terminal.Activities
Terminal Emulation Activities Pack

UiPath.PDF.Activities
PDF/XPS Activities Pack

Figure 7 - UIPath GUI



```
[✓] <html title='Google' />
[ ] <webctrl css-selector='html' tag='HTML' />
[ ] <webctrl css-selector='html&gt;body' id='gsr' tag='BODY' />
[ ] <webctrl css-selector='body&gt;div' id='viewport' parentid='gsr' tag='DIV' />
[ ] <webctrl css-selector='body&gt;div&gt;div' id='searchform' parentid='viewport' tag
[ ] <webctrl css-selector='body&gt;div&gt;div&gt;form' id='tsf' name='f' parentid='sea
[ ] <webctrl css-selector='body&gt;div&gt;div&gt;form&gt;div' parentid='tsf' parentna
[ ] <webctrl css-selector='body&gt;div&gt;div&gt;form&gt;div&gt;div' parentid='tsf' p
[ ] <webctrl css-selector='body&gt;div&gt;div&gt;form&gt;div&gt;div&gt;center' pare
[✓] <webctrl name='btnK' tag='INPUT' />
```

[ ]	aaname	Pesquisa Google
[ ]	isleaf	1
[✓]	name	btnK
[ ]	parentclass	jsb
[ ]	parentid	tsf
[ ]	parentname	f
[✓]	tag	INPUT
[ ]	type	submit

```
<html title='Google' />
<webctrl name='btnK' tag='INPUT' />
```

Figure 8 - UIPath element characterization

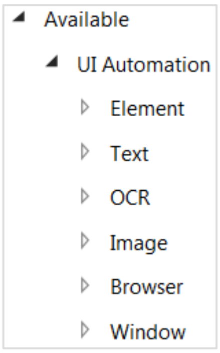


Figure 9 – Activities on the activities bar related to UI Automation

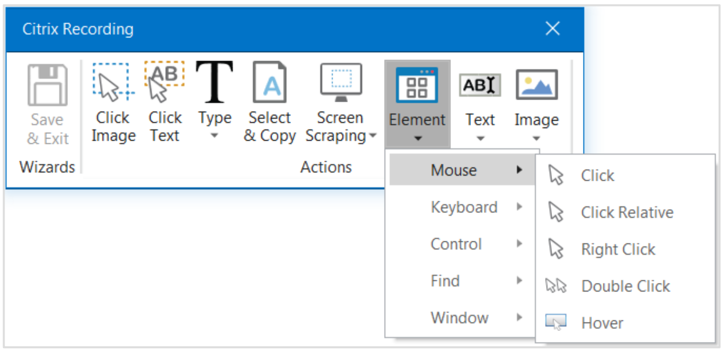


Figure 10 - GUI when using the Citrix mode

In UIPath, image selection is made by the user, he is the one who cuts around the desired UI element to interact. To make the identification as complete as possible, when selecting, UIPath gives the opportunity to add extra information like where in the image the user wants the click to be made and gives the opportunity to indicate a reference point (UI element). This reference point will then work as an anchor, and the element to be clicked will be described by its distance to the anchor (see Figure 11). When inserting text, UIPath also presents a menu (see Figure 12) that easily allows the user to insert special keys, to make the text secure (by covering the inserted characters, useful for passwords) and to ensure that the input text is empty.

On Figure 13 it is presented an example of an image driven script. In the sequence it will insert text on an input box and simulated the enter key, and then it selects one of the results by clicking. The script is applicable to google motor engine using Internet Explorer.

This approach requires an interaction with the interface almost atomic, users must assign to every action the proper image and does not guaranty reliability when screen resolutions change.

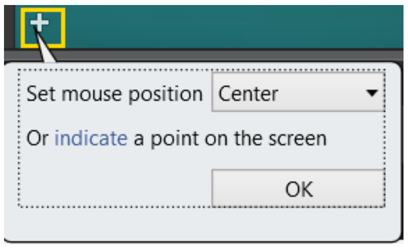


Figure 11 - Image identification GUI

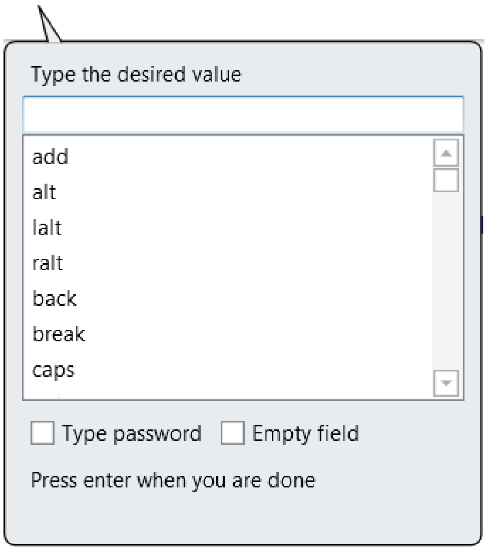
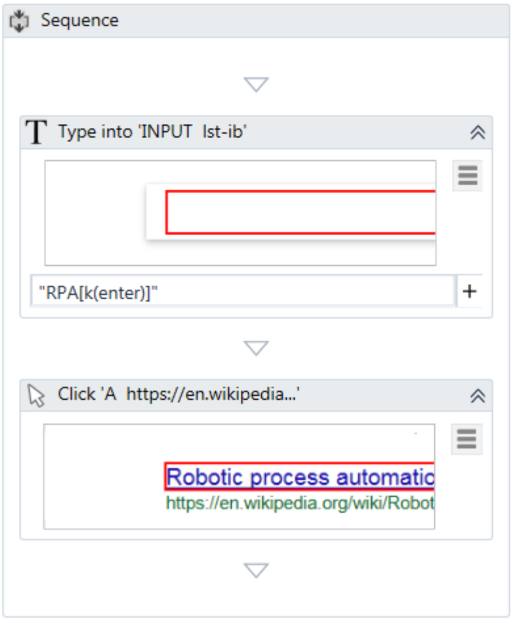


Figure 12 - Text insertion GUI


Sequence

Type into 'INPUT Ist-ib'

"RPA[k(enter)]"

Click 'A https://en.wikipedia...'

Robotic process automatic
https://en.wikipedia.org/wiki/Robot

Figure 13 - Example of script made with UIPath

2.1.2.2. Blue Prism

BluePrism is also software for RPA. With their software they aim to enable business operations to be agile and cost efficient through the automation of manual, rule based office administrative processes, reducing costs and improving accuracy ('Blue Prism - Robotic Process Automation', n.d.).

Like UiPath, BluePrism allows automation either by accessing applications, through connection plugins (on desktop apps), and HTML (on web-based apps – limited to Internet Explorer) or by PDC. On Figure 14 it is presented the BluePrism's main GUI, which is a development studio. The main functions are located either on the top or the right bar. On the right bar the main function blocks can be dragged into the development grid, on the center, and on the top bar some control commands (for example to run the model, to adjust zoom, save the developed model, etc.) are available. This software functions on a model driven way, being developed a flowchart like the UML ones and on each box the functions are then specified. On Figure 15 it is presented an example of the GUI to add events to a flowchart component.

Focusing the way PDC is implemented, on Figure 16 it is presented the GUI that allows selecting visual elements. On the center the page is presented and on the top bar the user has tools that enable him to select the region of interest, some additional tools are also available (e.g. cutting part of the image, copying it, selecting another element to focus). On the right bar the user names the piece of image selected and sees its position relatively to the selected component, the full GUI. When reproducing the PDC script, BluePrism finds the main GUI, the container, and searches the region with the coordinates corresponding to the selected image.

After selecting the piece of image of interest and naming it the user gets back to the GUI which allows event specification (Figure 15) and adds the action to be performed on that image that can be a left or right click, single or double click, etc. When ready the flow is presented on the main GUI as a flowchart.

2.2. AUTOMATION AND TASK MODELS

When automating it is important to properly understand what is being done. One way to understand how a task is, is to design it in detailed. The following two topics present the concept of automating and a notation that provides a proper task analysis.

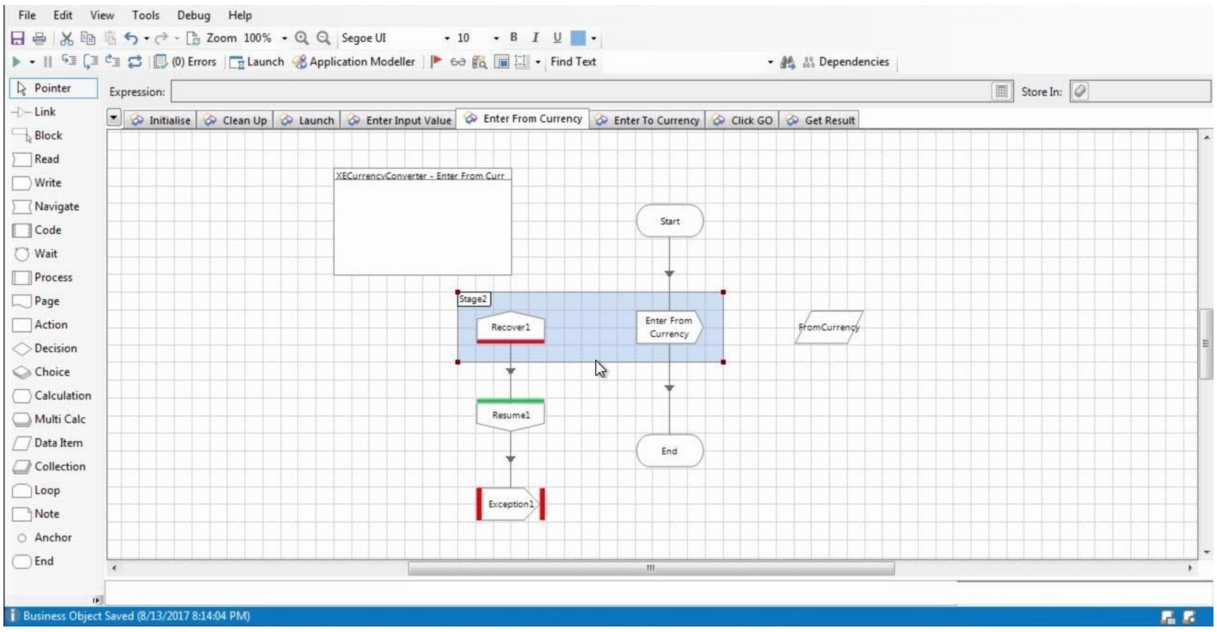


Figure 14 - Blueprism GUI

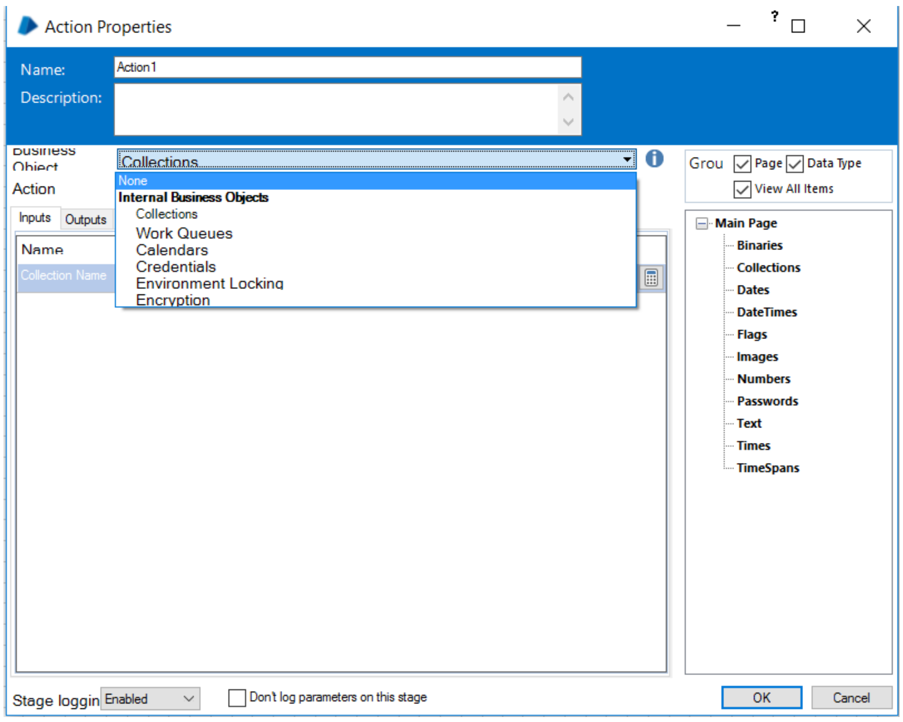
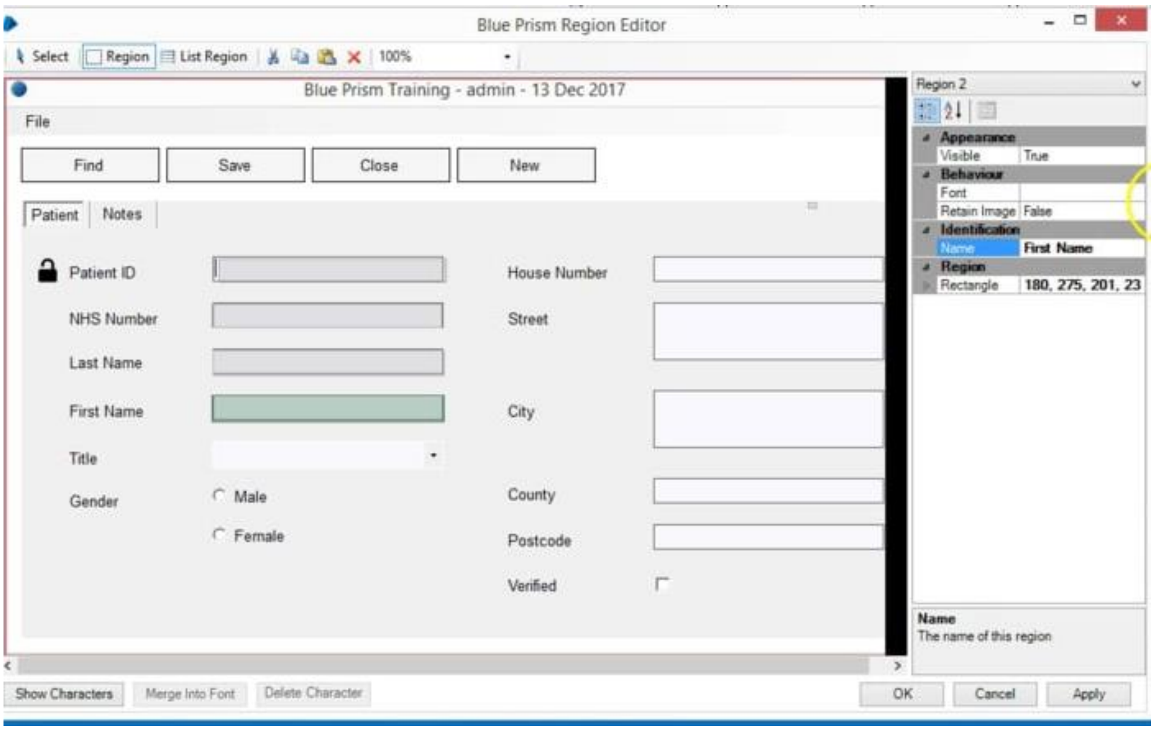


Figure 15 – Specification of the actions to occur on a block from the flowchart



Visible	True
Font	
Retain Image	False
Name	First Name
Rectangle	180, 275, 201, 23

Name
The name of this region

Figure 16 - GUI to automate via PDC

2.2.1. Automation

Automation consists on the transformation of functions of systems to automatic processes, in order to affect it following a specific target (Vogel-Heuser et al., 2014). Some basic concepts associated to automation are: *business process* that consist of sequences of activities; an activity that is a discrete process step performed either by machines or humans (activities may consist of one or more tasks); a set of tasks to be performed by a user in a workflow system is called a worklist; a workflow is the automation of a business process in whole or in part, during which documents, information or tasks are passed through participants following a set of rules (Shi, Lee, & Kuruku, 2008; Stohr & Zhao, 2001).

The level of automation of a system corresponds to the proportion of automatic functions to the entire set of functions of a system (Vogel-Heuser et al., 2014). In general, there are three action types, full automation, semi-automation or manual. On automated actions, these are fully conducted by a computer and all possible outcomes are predicted. Semi-automated tasks require an interaction between the computer and a human that can validate, compare, or add information to the workflow. Manual actions are normally complex decision-making tasks that require comprehensive knowledge and skills to take the necessary decisions (Shi et al., 2008). Other way to describe automation level is a scale from one to ten in which one represents business process on which the computer offers no assistance and humans take all decisions and actions and on level ten the computer decides everything, acting autonomously and ignoring human inputs (C. Martinie, Palanque, Barboni, & Ragosta, 2011).

On critical workflows a semi-automated approach is best suited making the system work on a multi-agent basis, an human agent interacting with a software agent, and on this cases it is important that the human agent is aware of what the other agent has done, what he is doing now and for how long, why he is doing it, and what is going to be done next and when (Boy, 1998).

Automation, however, is seen by some researchers as having a negative impact on the workplace, fragmenting work, deskilling workers, and destroying morale. Others see it as an enabler to the elimination of drudgery and possibility to richer and more fulfilling jobs (Stohr & Zhao, 2001). When automating some problems and concerns are:

- When the automation requires user to input some variables to set the session the automation can be put at risk, either because the input parameters are not the expected or even because the responsible might forget that the automation program was waiting for the parameter (Vogel-Heuser et al., 2014);
- The fact that people cannot recognize repetitive tasks worthy of automating leading to bad automation task choosing. User interviews revealed that tasks were identified as repetitive if performed numerous times in a row manually, or if complex but infrequent (Amershi, Mahmud, Nichols, Lau, & Ruiz, 2013).

Automation, then, requires an accurate process definition similar to when planning training sessions. This definition has been studied for some years and task models are a suggested approach.

2.2.2. Task models

Instructional System Development (ISD) developed a set of guidelines concerning the design, setup, evaluation and maintenance of educational or training programs. It is called ADDIE, an acronym for its systematical phases: Analysis, Design, Development, Implementation and Evaluation. The first phase, analysis, consists on a proper definition of a task ('Instructional Systems Development (ISD)', n.d.; Célia Martinie et al., 2011).

Powerful analytical tools, *ConcurTaskTrees* (CTT) for example, provide a thorough process definition and are needed to formally analyze workflow processes for correctness and consistency. One of the most valuable aspect of this modulation is the decomposition of tasks in its sub-tasks properly understanding the required steps to accomplish it (Célia Martinie et al., 2011).

Focusing people training some major difficulties arise. These might be, defining what is a “reasonable” coverage of the states of the systems, since in these days the number of states is almost infinite; identification of which states are the most important; how to implement in the training unlikely events that cause failure; How to assess the level of training of someone and decide what is yet to be better trained. Despite these, computer-based training have benefits, it allows cost-savings and ensures a better training program availability (Célia Martinie et al., 2011).

When defining a task-model notation it is important to support capabilities such as hierarchical logical structure, allowing for different abstraction level and process refinement, a wide variety of temporal relationships, supporting interactive behaviors, and representation of relevant relationship. To define a task, it is necessary to identify the objects and the actions that allow the communication. Objects consist on the entities that are manipulated to perform tasks (e.g. menus, icons, state requests of some application, …) (Paterno, Mancini, & Meniconi, 1997).

CTT and HAMSTERS implement task models on a similar way. On them it is possible to specify if a task is a user task, application tasks, interactive tasks or abstract tasks. User tasks are performed entirely by the user, application tasks are completely executed by the systems, interactive tasks performed by user interaction with the system and abstract tasks which require complex actions and their performance does not properly fit the others. To express temporal relationship among tasks LOTOS (Bolognesi & Brinksma, 1987) concurrent notation is used. The possibility to represent concurrent tasks makes CTT different from the only sequential relationship available on GOMS (Card, Moran, & Newell, 2008), despite their similarity on hierarchical tasks disposition (iCS, 2010; C. Martinie et al., 2011; Paternò, 2004). On Figure 17 a task model developed with HAMSTERS is presented as an example. It was presented on the context of a board flight software.

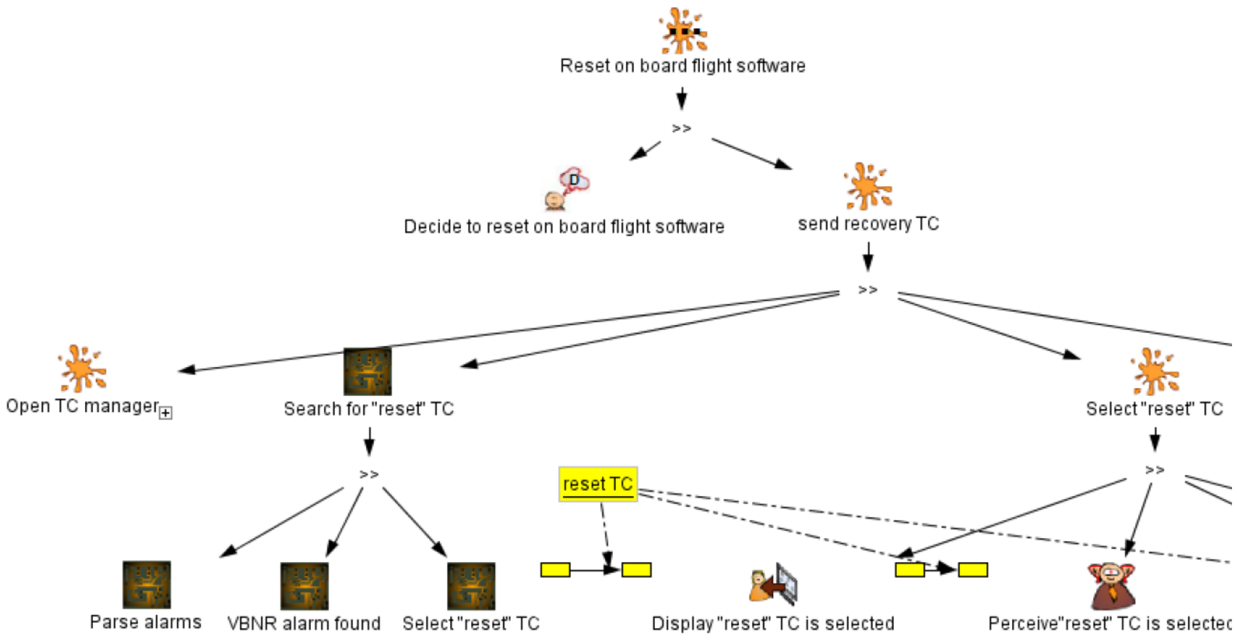

Reset on board flight software
>>
Decide to reset on board flight software
send recovery TC
>>
Open TC manager
Search for "reset" TC
Select "reset" TC
>>
reset TC
>>
Parse alarms
VBNR alarm found
Select "reset" TC
Display "reset" TC is selected
Perceive"reset" TC is selectec

Figure 17 - Task model developed with HAMSTERS (C. Martinie et al., 2011)

However, task models pose as a barrier to the adoption of task automation mainly because of the increased effort to create them (Amershi et al., 2013; Stohr & Zhao, 2001). Trying to reverse the effort required on the definition of task models, Machado et al. presented a tool that can define a task models from a task one-time execution. It is based on picture-driven computing and automation and the produced task models follow the CTT notation. The developed tool is then able to reproduce scripts posing as an UI testing tool (Machado et al., 2017).

Chapter 3

LITERATURE REVIEW

This chapter presents a literature review on the topics most important for the dissertation. From this review the points open to contributions were identified and selected.

On this chapter help and automation tools which aim to improve daily interactions between users and computers are described. Literature presents a lot of work, and with its revision lessons learned are retained as well as possible contribution points identified.

3.1. METHODOLOGY FOR TOOL SELECTION

The tool research was made through digital repositories. The repositories from which most information was taken are ACM Digital Library, Springer, IEEE Xplore and Elsevier. Additionally, information from AAAI Digital Library, Wiley Online Library, Taylor and Francis Online and Scientific Research was also taken.

When researching, the main time period was between 2010 and 2018. This choice was due to the identification of a review made by Grossman and Fitzmaurice (Grossman & Fitzmaurice, 2010) that presents help tools based on Video and Animated Documentation, Contextual Assistance, and Contextual Video Assistance previous to 2010. There are two works that do not belong to this range, being one from 2004 and the other from 2005. These automation tools are presented because they describe interesting approaches which with the addition of current technology capabilities would pose as good candidates on the help-tools umbrella.

The keywords used on the research were ‘action repetition’, ‘contextual help’, ‘crowdsourced help’, ‘demonstration help’, ‘GUI automation’, ‘knowledge sharing’, ‘model-driven help’, ‘programming by demonstration’, ‘software support’, ‘workflow automation’, ‘’picture-driven computing’, ‘tutorial’, ‘user support’. There were also papers found when reading other papers, for example, when tools were developed having some relation with others, or in cases where the paper also makes a presentation of similar research works. In this research, a systematic review of the literature was not done.

Another defined goal for the research was to gather information that would enable the creation of a help-tool that would meet users’ needs. For the last years help is given through different means, varying between automation, video tutorials, tool clips, etc. For these different help mechanisms many tests with real-users were conducted and from them help and automation tools presented. Some automation tools are presented below, and despite not aiming primarily to help users, provide interesting information useful on the development of a help tool.

The following sections group help-tools that work under the same basis, through scripts providing automation, videos, GUI Elements and, at the end, some conclusions are drawn.

3.2. SCRIPTS AND AUTOMATION-BASED

Sheepdog (Lau, Bergman, Castelli, & Oblinger, 2004) is a programming-by-demonstration system that works in different phases. First, it learns from multiple experts, each performing the same procedure (set of steps to achieve a goal) directly on a Windows desktop. With the recording, Sheepdog models the procedure allowing different configurations. These models can be reproduced on an end-user device via an executable file. When generating the models, the system, Sheepdog, abstracts from low-level Windows events into clean high-level representations and uses Input/Output Hidden Markov Models to determine the odds of occurrence of the next node. It is believed that the interface where the system automation is mapped must be collaborative, allowing the user (expert) to add extra-input. When replaying the script step by step the system takes a snapshot and from it determinates and indicates what the next step is.

CoScripter, previously called Koala, (Leshed, Haber, Matthews, & Lau, 2008), is a collaborative scripting environment for recording, automating, and sharing web-based processes. It has two main modules, a plug-in that allows users to record and play actions, and a repository where users can share their scripts, rate and comment other scripts. The actions recorded are recognized and replayable via HTML interpretation.

ActionShot (Li, Nichols, Lau, Drews, & Cypher, 2010) is an extension to Firefox web browser built on top of the CoScripter web recording/playback platform. Unlike CoScripter, ActionShot records continuously users' browser history grouping actions by pages' host name. It also provides a visual interface that allows users to use or share any action they have ever performed. The description of each module is based on text taken from the HTML code, usually the text displayed on widgets.

CoCo (Lau et al., 2010) is a system that automates web tasks. It takes leverage of a repository with previous web scripts and users' personal web browsing history. People ask questions via Twitter and the system can either respond asking for more information or with an answer. The system can cover each user needs.

TaskTracer (Dragunov et al., 2005) is able to track users behavior when using applications like Microsoft Office, e-mail, and Internet Explorer (an extra add-in has to be installed) being able to recover the full context of the task if desired. To classify the task, the user must indicate its beginning and ending. Authors acknowledge it as an extra burden. They aim at the ability to restore all applications associated with a process, the documents used and even the indication of the last changes.

Smart Web Tutor (Sun et al., 2016) proposes an approach to enhance Massive Open Online Courses (MOOCs) with education resources development for instructors and students, allowing real-time collaborative learning for students. As for the recording phase, it is based on the DOM structure, common on web-sites, and works on three main phases, 1) recording, 2) optimizing, and 3) replaying. A similarity formula that considers platforms variability is also presented. Different system may require different steps to accomplish a task. For the real-time collaborative learning, authors propose a communication protocol to send each step on a synchronous way, making possible to a group of people to watch a procedure executing on real-time.

3.3. VIDEOS-BASED

Ambient Help (Matejka, Grossman, & Fitzmaurice, 2011) works on a secondary monitor. While people are performing a task on the primary monitor AmbientHelp is automatically displaying videos or/and textual information relevant to the process. Whenever people need help or want to see AmbientHelp suggestions they focus on the secondary monitor and search the desired help-content.

Pause-and-Play (Pongnumkul et al., 2011) proposes a learning improvement via video tutorials. The improvement consists on receiving a video-tutorial and recognizing the main steps on it, with use of computer-vision. Then, with the addition of plugins to applications, to trace user behavior, it is possible to pause and play the video tutorial according to the user performance on the application. In addition, they also deployed a functionality which allows easy navigation through the most important video parts.

ToolClips (Grossman & Fitzmaurice, 2010) is a contextual-based help system that aims to replace the tooltips associated to UI elements, functioning on a similar way. Instead of the standard tooltips (e.g. with the icons' name), when the cursor dwells over an UI element the user can access video or text with information about the elements' functionalities as form of use. With this approach the user is presented with extra-information inside the application.

3.4. GUI ELEMENTS-BASED

LemonAid (Chilana, Ko, & Wobbrock, 2012) (Chilana et al., 2013) is a crowdsourced contextual help system. It is an approach to provide technical help based on the selection of GUI elements (a label, widget, link, and image) instead of keywords. Technically, it works as a layer above a web application UI, independent from the back and frontend implementations. People get help by selecting an UI element and, from there, LemonAid presents the five most asked questions by the community. Users can use the search bar to enter keywords and refine the questions shown, can make new questions, or can see the answers to the existing questions. They feel that an investment in one-to-many support is more efficient and provides greater cost savings.

DemoHelp (Ornelas et al., 2016) is an help system that takes advantage of PDC and automation, with Sikuli, and is to be used side by side with an application. Users select an action to execute in the help system and the procedure is triggered on the application. To make Sikuli scripts users must define CTT task models in an enriched way, with applications' UI elements, and define the different possible scenarios.

Interactive systems Integration Tool (ISI) (Silva et al., 2016) is a tool that works with enriched task models (CTT) and scenario selection (with a tool from the Human Interfaces In Information Systems Laboratory[3]) to automatically create Sikuli scripts. This way they developed a tool that removes complexity from selected procedures, presenting a simplified GUI with simple labels that indicate what tasks are automated and ready to be reproduced. The tool can merge multiple interfaces, from multiple applications, that must be used to complete a task in only one interface. After choosing the task, all steps are executed by Sikuli, on a virtual machine being presented to the user only the result. In this approach procedures are not limited to any application.

Help, It Looks Confusing (HILC) (Intharah, Turmukhambetov, & Brostow, 2017) is a tool that relies on Computer Vision techniques rather than accessibility APIs and, therefore, is domain independent. Its' focus is on personal automation and script generation from non-programmer users. It is a tool that automates procedures by demonstration. When demonstrating, it produces the needed screenshots and their corresponding mouse-keyboard events. Scripts can be made from video records, with specialized screen casting software, or a sniffer program. Inputs are converted to log files that pass through a classifier algorithm. It is stated by the authors that pure programming-by-demonstration is still unrealistic, and they verify that for small procedures the classifier algorithm has not great performance. Adding capabilities to HLIC Intharah et. al developed a tool named RecurBot (Intharah, Firman, & Brostow, 2018) that uses HILC to retrieve the human interaction with the interface (clicks, and images) and focus on the loop identification aiming to automate the repetitive task removing the effort from users. They do it by analyzing multiple task execution, discovering

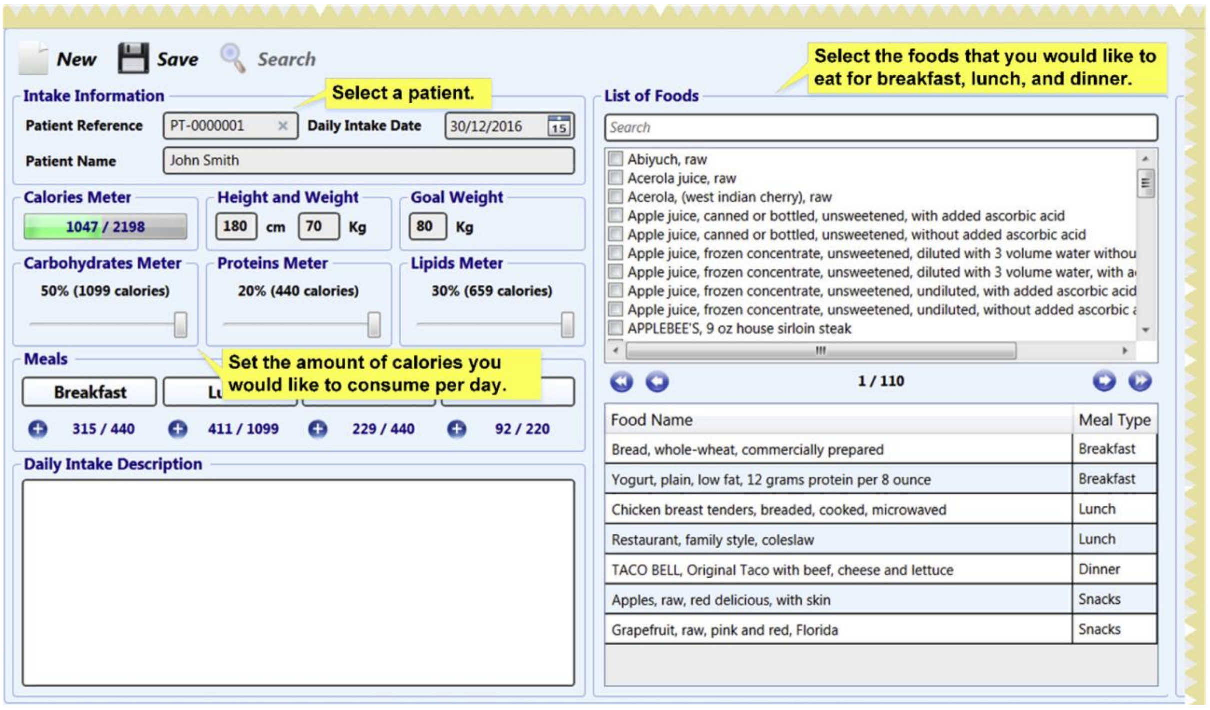
[3] http://hiis.isti.cnr.it/lab/home, accessed on: 07-10-2018

the pattern and making suggestions to verify they're results. When the task is figured RecurBot can automate the loop.

Contextual Help for Adaptive INterfaces (CHAIN) (Akiki, 2018) was developed to provide model driven contextual help to adaptive user interfaces, interfaces that change their presentation at runtime. The adaptation to change is made by an association between concrete elements and scene elements. Concrete elements are defined by specific parameters like the concrete id on the code and scene elements are associated to them (e.g. a menu is a concrete element and its 'inner' options are the scene elements). CHAIN was developed using JavaScript for the web, C# with .Net Framework for the desktop and C# with Xamarin for mobile. When source-code is not available it is able to use the PDC paradigm, interacting in different ways depending on where the application is running. For web application it uses Selenium capabilities and for desktop application it makes use of an UI Automation Framework ('UI Automation Overview', 2017). The help is given by the definition of a model on Cedar Studio (Akiki, Bandara, & Yu, 2013), which also allows for the importation of Selenium scripts. The focus of CHAIN is in providing proper annotations on the target interface, being also able to insert some values, see Figure 18. From their studies they state that one-shot learning is something appreciated, that help across several applications was something that they did not reach but is of value, as that crowdsourcing would be an interesting trait.

On Table 1 the works previously stated are summed up and sorted into categories. All works were sorted on three main groups, automation, automation/help, and help. With these groups we aimed to label the goal of the work explaining if the solution was conceived to improve processes through automation or, to help users. The points of interest of the research were the application domain, how the interaction with the real application was made (interaction core), if it works on a crowdsourced way, implements PDC on its interaction, if it can "learn" by user demonstration and if with user interaction it learns and improves.

The highlighted points emerged from an interconnection between our goals and the identified works. From these it is possible to identify which solutions are dependent on application source codes and if not, how they interact with it, to see if continuous learning is implementable and viable, to realize the challenges of the interaction with the elements of interfaces through image and, to check if there has been an investment on providing help through the crowd.


New
Save
Search
Select a patient.
Select the foods that you would like to eat for breakfast, lunch, and dinner.
Set the amount of calories you would like to consume per day.

Intake Information

Patient Reference: PT-0000001 Daily Intake Date: 30/12/2016

Patient Name: John Smith

Calories Meter: 1047 / 2198

Height and Weight: 180 cm 70 Kg

Goal Weight: 80 Kg

Carbohydrates Meter: 50% (1099 calories)

Proteins Meter: 20% (440 calories)

Lipids Meter: 30% (659 calories)

Meals: Breakfast | Lu

315 / 440 · 411 / 1099 · 229 / 440 · 92 / 220

Daily Intake Description

List of Foods

Search

- Abiyuch, raw
- Acerola juice, raw
- Acerola, (west indian cherry), raw
- Apple juice, canned or bottled, unsweetened, with added ascorbic acid
- Apple juice, canned or bottled, unsweetened, without added ascorbic acid
- Apple juice, frozen concentrate, unsweetened, diluted with 3 volume water withou
- Apple juice, frozen concentrate, unsweetened, diluted with 3 volume water, with a
- Apple juice, frozen concentrate, unsweetened, undiluted, with added ascorbic acid
- Apple juice, frozen concentrate, unsweetened, undiluted, without added ascorbic a
- APPLEBEE'S, 9 oz house sirloin steak

1 / 110

Food Name	Meal Type
Bread, whole-wheat, commercially prepared	Breakfast
Yogurt, plain, low fat, 12 grams protein per 8 ounce	Breakfast
Chicken breast tenders, breaded, cooked, microwaved	Lunch
Restaurant, family style, coleslaw	Lunch
TACO BELL, Original Taco with beef, cheese and lettuce	Dinner
Apples, raw, red delicious, with skin	Snacks
Grapefruit, raw, pink and red, Florida	Snacks

Figure 18 - Example of an interface with annotations made with CHAIN (Akiki, 2018)

Beyond the previous works with similar goals to the one to be developed in this work, on the following paragraphs three works that take interesting approaches for GUI (re)definition and user interaction are presented. On them, valuable content is also taken to our design phase.

Prefab (Dixon, Nied, & Fogarty, 2014) presents a toolkit for pixel-based enhancements. This is a system that is independent from source codes and authors acknowledge that interpreting an interface from its raw pixel values is a large a multi-faceted problem and that writing sophisticated code is not enough to successfully interpret most interfaces citing Dixon (Dixon, Fogarty, & Wobbrock, 2012) who reports that clickable "target" are often ambiguous and cannot be reverse engineered without human intervention. Prefab can redefine an interface based on programmers' input. To do this, it first captures a source window bitmap and interprets it, then a modified interface is provided, the source is mapped based on the modified interface provided, and finally the desired output is displayed on the target.

Studying capture-replay techniques Sun et al. (Sun, Chen, Xin, & Jiao, 2015) presented a tool designed for automation on web-browsers. Their tool provides an interface where users that capture the interaction can make changes to the recorded procedure, adjusting the process to be replayed. On their work it is also presented important efficiency aspects to consider on capture-replay technique implementations. In order to achieve executable task models, PLOW (Allen et al., 2007), takes a different approach, making the capture process step-by-step adding extra-input to the steps while recording. These extra-inputs range from users' language

Table 1 – Grouping of studied works by categories and specification of their features.

Main goal	Domain	Interaction core	Crowdsourced	PDC	One-shot learning	Continuous learning	Name
Automation	Web	Automation scripts	Yes	No		No	CoScripter
	Web	Automation scripts	Yes	No		No	ActionShot
	Web	Automation scripts and question-answer analysis	Yes	No		Yes	CoCo
	Plugin dependent	Automation scripts	--	No		No	TaskTracer
	Independent	GUI Elements	No	Yes		Yes	RecurBot
Automation /Help	Independent	Learn - replay procedure	--	No	Yes	Yes	Sheepdog
	Independent	Enriched task models and PDC	No	No	No	No	ISI
Help	Web	Dialogues with users	No	No	No	Yes	AIDE
	Web	Scripts - Web' DOMs	Yes	No	Yes	No	Smart Web Tutor
	--	Videos or/and textual help	No	No	No	No	ToolClips
	Plugin	Videos	No	No	No	No	Pause-and-Play

	dependent						
	Plugin dependent	GUI Elements	Yes	No	No	No	LemonAid
	Independent	Videos or/and textual help	No	No	No	No	Ambient Help
	Independent	Enriched task models and PDC	No	Yes	No	No	DemoHelp
	Independent	GUI Elements	No	Yes	Yes	Yes	HILC
	Independent	Interaction models (through Cesar Studio)	No	Yes*	No	No	CHAIN

Notes:
'*' PDC is implemented through Selenium in web applications and through an UI Automation Framework in Desktop Applications.
One-shot learning aspects was only considered for works who aim at helping people

processing (speech and keyboard) analysis, loop indication (start and end points), and correct parameterization. The produced automation works for web applications.

3.5. CONCLUSIONS

From the presented research it is concluded that most help system focus on web applications, where DOM structure is available. Despite the objective not to be automation, works with this purpose were analyzed to verify the way the capture-replay process is designed.

Most authors also emphasize the importance of users being able to add more information to automation, specifying for example parameters variability, icon identification corrections, or loop indication.

It is also usual the inclusion of classification algorithms to identify tasks, however, for short procedures it is still something unstable.

It is clearly identified as a challenge domain independency, either from source codes and/or extra-plugins. Not requiring the user to install extra-plugins or to restrict the automations/demonstrations to a set of applications is something that brings many benefits like intra-application help possibilities. To achieve this independency a system that does not depend of application code or structure is a must; PDC has the capabilities required.

Icon or widget identification is also a major barrier. Developing a system that would run on a single computer/machine would be an advantage, considering that there would be no theme or context variability. By theme variability it is considered the size and color of icons and widgets, for example, and by context variability it is considered the widgets surrounding which can be a great asset for proper identification.

Improving the way people get help, making it easy to accomplish tasks is a major advantage either for corporate workers in their daily activities where they must be productive as for singular computer users that many times feel that they can’t execute some tasks because of their lack of skills instead of recognizing the fault on software usability.

Chapter 4

APPROACH

This chapter describes the approach taken on the development of the tool. This approach contemplated interaction with users to carry out a proof of concept, the sketching and programming of the final presented tool.

The proof of concept led to the publication of a scientific article (Rodrigues et al., 2018), part of the text written in the first section is replicated from there.

With literature review the following important aspects were identified: crowdsourced capabilities, PDC oriented and scripts made from one-shot demonstrations. These were the pillars for the foreseen tool since currently there is no identified tool focused on helping users with these traits.

Having investigated PDC, with the domain and source-code independence on a strong side and the difficulty to a stable image identification on the weak side, before initiating the final tool development a proof of concept and a user study were conducted.

These were executed with the objective of validating the approach, automation-oriented help with image recognition. Since these are new concepts, the development and testing of a preliminary tool is useful for validating the idea and to observe user reactions.

On the first section the proof of concept and its results are described. From it positive results were indicated. The second section describes the modules of the developed tool and on the third section the way the modules were implemented are described.

4.1. FEL TOOL – PROOF OF CONCEPT

To conduct a first evaluation, a tool called FEL was developed aiming to verify the acceptance and usability of a help tool that would run automation scripts through PDC.

This tool was only capable of playing Sikuli scripts, presenting an interface which allowed the selection of scripts to play (see Figure 19).

4.1.1. Participants

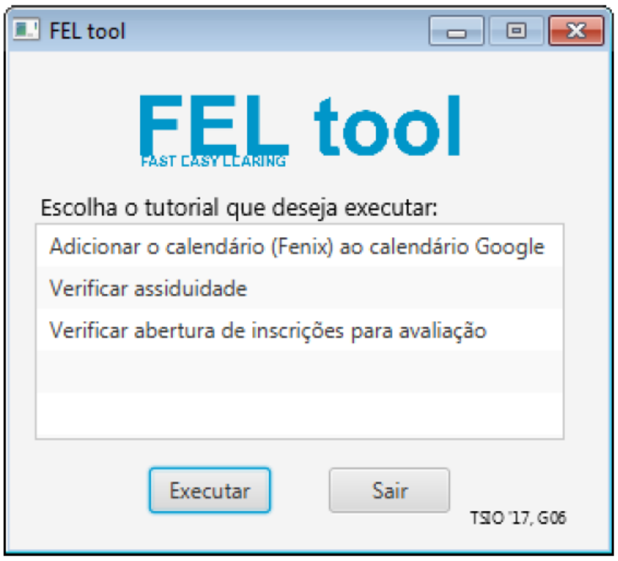
Ten users participated on the experiment. There were three females and the age range went from fifteen to fifty-four. They were all familiar with the use of computers, using them at least three times per week, but without previous interaction with the university online platform (where the tasks had to be performed – see section 4.1.2.).

4.1.2. Procedure

Users were divided into two groups, at the end they had to answer an adapted version of the USE questionnaire[4] regarding the help system.

Both groups had to perform twice the three tasks (detailed below). First, tasks had to be carried out using a help tool and then without any help. Group one used the current university platform guidebook and group two used FEL.

[4] Retrieved from: http://garyperlman.com/quest/quest.cgi?form=USE, on October 2017



Escolha o tutorial que deseja executar:

Adicionar o calendário (Fenix) ao calendário Google
Verificar assiduidade
Verificar abertura de inscrições para avaliação

Figure 19 - FEL gui.

The evaluation consisted on the execution of three tasks, executed on the online platform of a university. These were chosen as means of example to illustrate the proposed approach. The three tasks were:

- Check attendance (“Verificar assiduidade”) – allows users to quickly check their attendance (overall view or day search). For the specific date check, users must insert the date on a dialog box on a pre-defined format. This task was chosen because it is a useful feature that is used by all students. This way, we are automating a frequent task and providing a learning opportunity via contextual help for those new to the interaction.
- Check evaluations sign up (“Verificar abertura de inscrições para avaliação”) – directs users to an online page from the university platform where all the evaluations are registered and some of them may require inscription. With the goal of making all tasks complete, beyond directing to the accurate page it also checks the page for any evaluation where the inscription is still not made notifying the user whether there is any sign up. This one was chosen because it is a new feature and most of the users (students) still need to discover it.
- Synchronizing scholar calendar with Google Calendar (“Adicionar calendário ao Google”) – has as challenge the interaction between multiple-applications (inter-systems task). With this task, it is possible to incorporate the university calendar with the personal calendar of each user. To perform this task, users not using FEL must check university online platform guidebook where they

took the first steps, and then had to check Google help books to finalize the synchronization.

4.1.3. Measures

User's interaction was recorded and then analyzed. Notes were taken during the studies. When analyzing, special attention was given to the time needed to complete tasks, and the number of errors and wait times superior to five seconds (considered as indicative of hesitation) on the second time performing a task.

Participant characterization was made, and users filled a questionnaire that addresses four aspects: Usefulness, Ease of Use, Ease of Learning and Satisfaction (as defined in the standard USE questionnaire). Answers were on a 7-point Likert scale with values from 1 (strongly disagree) to 7 (strongly agree). The questionnaire included open question on the strengths and weaknesses of the tool.

4.1.4. Results

On Table 2, it is presented the characterization of people that were involved on the experiment.

Table 2 - Demographic information of users participating on the proof of concept study.

		Number of participants	**Percentage**
		10	100%
Gender	Male	7	70%
	Female	3	30%
Educational Qualifications	Highschool	2	20%
	University degree	8	80%
Average days on computer (per week)	Between 3 and 5	3	30%
	More that 5	7	70%

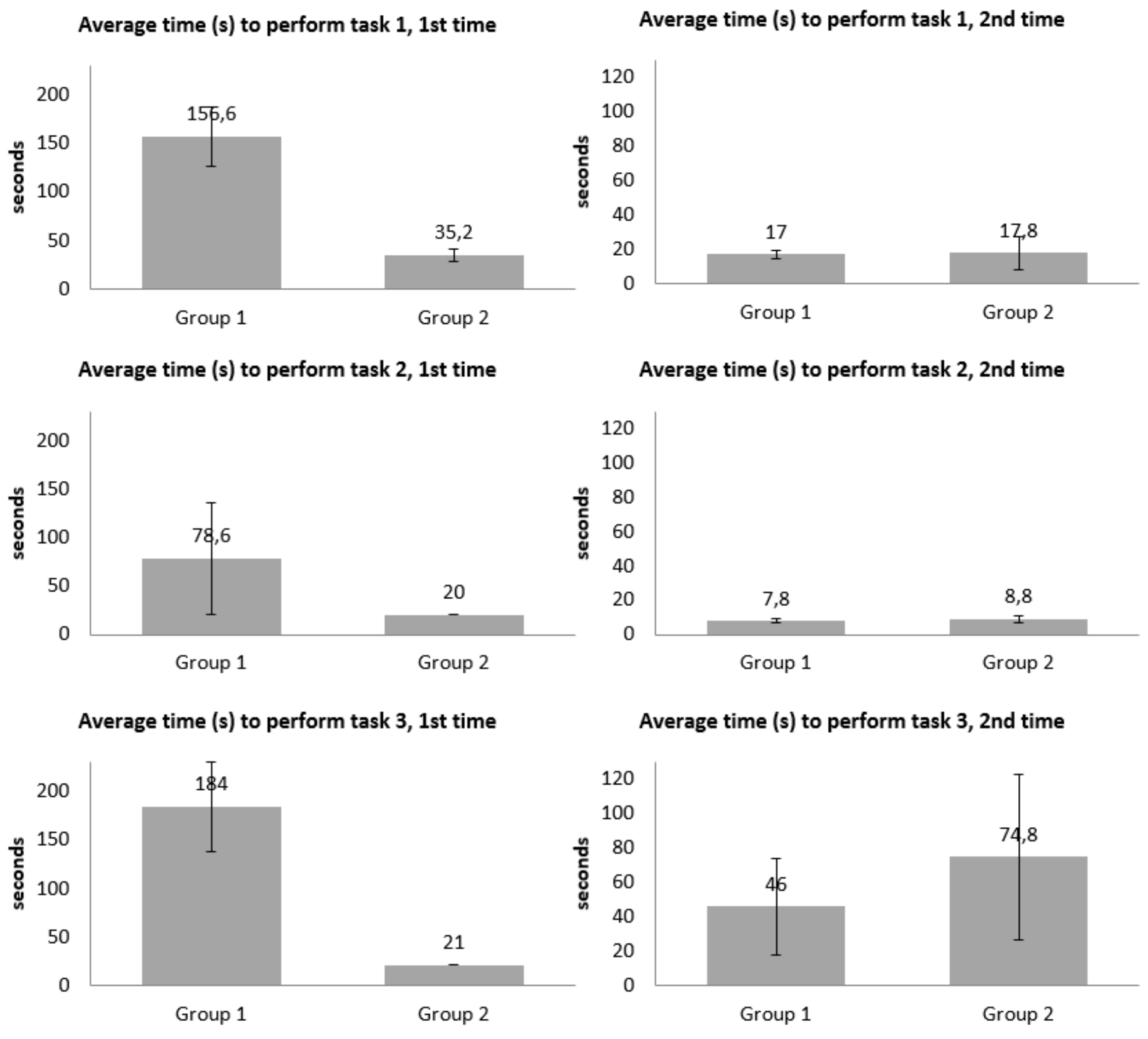


Figure 20 - Average time and standard deviation when executing study tasks.

On Figure 20 is presented a comparison between the average times on performing the tasks. The left ones represent the first time performing each task, in which the task was performed with a help system (current guidebook or FEL depending on the group users were sorted into) and the ones on the right represent the second time performing the task, recurring to no kind of help. From these results, it can be seen that using FEL (users from group two), on a first-time interaction with the system, tasks are performed much faster (5,71 times faster) and performing tasks on a second time, without any help system, users from group one (without FEL) performed only 1,26 times faster (average from the three tasks).

Regarding errors and wait times superior to five seconds (indicating potential hesitations), on the Figure 21 is presented the difference between the number of wrong clicks and wait times superior to five seconds when performing tasks on a second time, being on group two, and the same numbers regarding users sorted on group one. To calculate the difference, average values from the experiments were used; regarding wrong clicks group 1 µ

= 0,2 σ = 0,56 and group 2 μ = 0,67 σ = 1,05; for wait times group 1 μ = 0,33 σ = 0,90 and group 2 μ = 0,47 σ = 1,06. As for the wait times superior to five seconds, the occurrences are very similar whether in the group one or two. Concerning wrong clicks performing a task, group two presented an average of 0,46 more wrong clicks than group one.

One of the hypothesized differences between both cases is that users learning from observation (users from group two) are exposed to more distractions. Whilst having to execute the task while reading a page of text and viewing images is done entirely by the user, when observing an automated interaction with little intervention the attention required is diminished.

On Figure 22 the results from the USE questionnaires are summed up. It is shown that in all aspects the FEL tool was scored higher than the current guidebook. Little variation was shown on the Ease of Learning category but as for Usefulness and Satisfaction they are higher for more than 1 point. It was emphasized by group one users that there were some text inconsistencies and that the reading was too extensive. As for group two users, one user referred that some automated steps were too quick.

4.1.5. Conclusions

The work described on this section resulted on a scientific publication named "Demonstration-Based Help: A Case Study" (Rodrigues et al., 2018). Part of the text written in this section is replicated from there.

The presented results must be interpreted carefully due to the reduced number of participants. Nevertheless, the results seem to indicate that the approach has a tendency that enables users to perform tasks easier and faster without compromising much the learning (as users tend to learn better by doing instead of just watching). With regards to acceptability, very good results were obtained but user satisfaction (5 out of 7) can still be improved. Based on the comments collected we believe this can be further improved by reducing (partially) the speed of tasks execution. Overall, the results seem to indicate benefits of the approach over traditional help.

It is considered that the study done was beneficial. Positive results were collected. People were able to interact well with the prototype and were receptive to the approach, help was given properly causing good results on first time task executions. The proof-of-concept led to the development of a new version with the features described below.

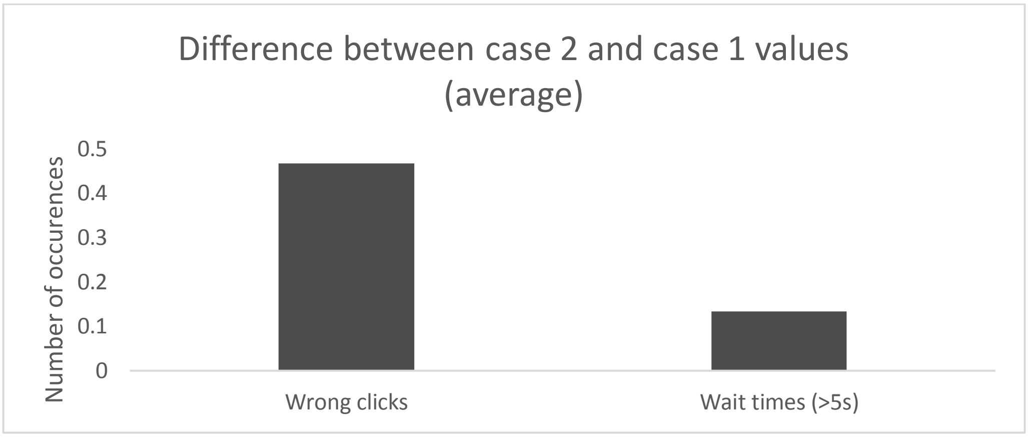


Figure 21 - Difference between the number of miss-clicks and long wait time between group 2 and group 1 when executing tasks for a second time.

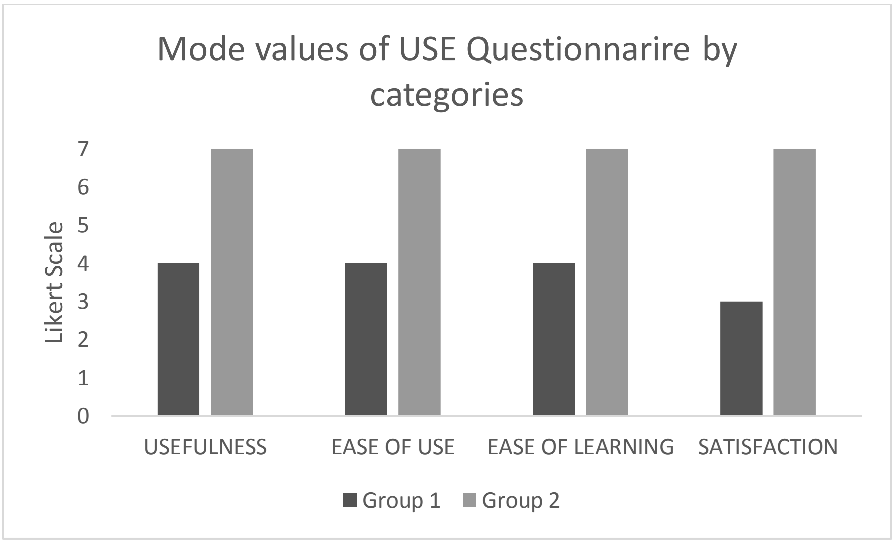


Figure 22 - Mode value of the proof of concept USE questionnaire sorted by categories

4.2. MODULES DESCRIPTION

The help tool consists of two main modules. One allows the execution of the available scripts (Player) and the other allows the creation and sharing of new scripts (Recorder).

The description of these two modules is divided on the following two sections. The first describes the Recorder and the second the Player.

4.2.1. Recorder

When developing the recorder, the experience gained when exploring other automation tools with image recognition resources (Sikuli IDE, and RPA tools) was an asset. Tasks executed on a computer are mostly composed by keyboard and mouse inputs, and by the images shown on the screen, so these are the main points to be captured to allow for an automatic task reproduction.

Through keyboard people can insert all kinds of text with different goals. It might be a set of characters that allows browsing a site, it can be item description, a password, among many others. Mouse allows mostly the interaction with buttons or the selection of a certain item. Image on the screen is what identifies components. All items of a GUI are defined by pixels, being shown to the user as images.

A picture-driven script consists on a set of sequential images which the user has interacted with. To allow replication of a task, the association between images and the actions performed (with the mouse or keyboard) is required.

When starting a recording people must specify a title for the task they are about to perform, they must describe the task and indicate which applications their task will interact with. This way all necessary metadata is gathered, and people must think about the task they are going to demonstrate. All this information is then available for people that wish to (re)play the task.

Briefly, the recorder consists on the capture of the keyboard inputs, mouse inputs and image on the screen and the orchestration of all the relations between the three on a sequential way. This had to be completely developed in order to be possible to create easily accessible instructions to be used by the Player and passed to the Sikuli. The technical development is detailed on “4.3.1. Recording scripts”.

4.2.2. Player

When considering the automations player, it had to be like a script gallery. In order to develop a crowdsourced tool, people had to have all the information about a script to be then able to choose the right one.

Designing it, it was considered important to provide all the description about the script, an evaluation on its performance, made by users, the success rates, and all the images that the script would search for.

Considering script description, users are presented with information about the script name, its author, description, which browsers are used (if used), and other applications which it will interact with. This is considered important because gives important descriptive information to people.

Success rate is presented on a scale from one to five. It is also presented the number of people that have already evaluated it. The success rate consists on the percentage of times which the script was successfully performed. Scripts might not be successfully performed if, for example, some image is not found, some click was not properly accomplished, there is some change to the interface.

The performance evaluation made by users, is important because it allows for a crowd communication and can be an indicative of the effectiveness of the script and its usability.

The purpose of presenting all the images that the script will look for is to give people the information of what steps are to be taken, and the elements on the interface which the tool will interact with.

Briefly, the player consists on views that provide all the information related with the scripts, allowing their choice, the capability to see their description, given rate and success percentage and check the images which the PDC will search. The main point is to provide users with the most complete information possible, being able to anticipate what will happen.

4.3. IMPLEMENTATION

The implementation was divided into three main steps. The first step was the recorder, capturing all user interaction and writing it into a file. The second, the script player, consisted on the integration of the Sikuli library on the tool and the interpretation of the previously created file. The last step (third) was the GUI development which allows users to easily use the developed tool.

4.3.1. Recording scripts

To record scripts the tool uses mostly the library JNativeHook[5] and the Java Robot[6].

JNativeHook is a java library that provides mouse and keyboard listeners. From this library the *nativeMouseClicked* and *nativeKeyPressed* methods are used. The first has the information of every click performed, its location (coordinated on the screen), and if it is a right or left one. The second method captures every key pressed. To record all this interaction codes were set and are used when writing the script file, for example typing letters corresponds to a code. These codes are posteriorly used by the script player to know if that line corresponds to a click or a typing action. When clicking, a screenshot is taken. This screenshot will be, at the end, cropped to match the clicking coordinates (process explained below).

To get screenshots the method *createScreenCapture* of the java Robot is used. Screenshots are saved on a default folder and are indexed sequentially. Apart from the

[5] https://github.com/kwhat/jnativehook
[6] https://docs.oracle.com/javase/8/docs/api/java/awt/Robot.html

screenshot a text file is created (see example on Figure 23), this file contains the relation between the screenshot number and the coordinates where the mouse has clicked.

At the end of the recording every image is cropped. When cropping the tool tries to make square images with a pre-defined height and width. It reads the coordinates file row by row. Each row has the id of the images and the coordinates of the click. Using the method *getSubImage* of the BufferedImage library the image is cropped equally around the coordinate point. For example, if the coordinate of the click was 400x400 the image will be cropped so the final image consists on the pixels from 300x300 to 500x500 considering that the desired images is of 200x200 pixels.

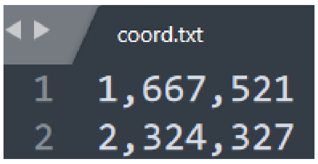
When creating a script, a json file is written with the information that the user has filled out regarding the title, description and applications used. This is done with the use of the *FileWriter* and the BufferedWriter. On Figure 24 an example of a script with typing and clicking actions is presented.

4.3.2. Playing scripts

To play already created scripts the use of the Java Scanner[7] and Sikuli libraries are necessary. The scanner is used to read the input file (automation script) line by line and adds it to an array of strings. This way the commands are read from an external file and can be iterated by the correct execution order. The second step consists on going through all strings on the array and checking if the code corresponds to a click on an image or text insertion.

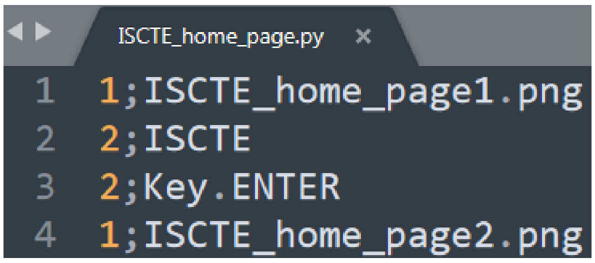
When the instruction is a click, Sikuli searches for the correspondent image for a maximum of ten seconds, this value was chosen by trial and error corresponding to medium wait times on applications where scripts were made for. If the image was recognized on the screen the element is clicked and if not, an error is thrown, and script execution is aborted, it fails. The wait time of ten seconds concerns GUI changes where screens may be loading, and the image may take some time to show up. Aiming at the improvement of image recognition, when searching for them two internal operation are performed:

[7] https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html



```
coord.txt
1,667,521
2,324,327
```

Figure 23 - Example of a coordinates file. It contains the association between the image number and its coordinates on the screen



```
ISCTE_home_page.py
1;ISCTE_home_page1.png
2;ISCTE
2;Key.ENTER
1;ISCTE_home_page2.png
```

Figure 24 - Example of a script. The first number gives the information about the action (click or type) and then the image to be clicked is named or the text/key to be pressed.

- Operations on the most common pixels-colors: Normally the most common pixel-color is the background and in cases where it is not those are not the pixel-colors which will mostly influence proper image recognition. From this premise, the tool discards the most common pixel-color focusing on the ones that are not that common. This way it is focused on the details of the widget (buttons shape or contour) and the complexity of finding a match of more pixels is removed (see Figure 25).
- Re-dimensioning: When recording the script, images are saved with a default size, this does not guarantee that only the widget (button, search bar, …) is on the image. To try to reverse the possible mal-effects, the developed tool tries to find the image (with the default size) and if not possible, it starts generating new images, smaller and with the central information of the original. This way a zoom-in is made and the most distant pixels are deleted (see Figure 26).

Text insertion is simpler since images do not have to be searched for. It is considered that when inserting text, the input field has been previously selected. The text is not normally inserted key by key but is pasted as a full string. This is possible because special keys are treated differently creating string breaks and being treated as special commands.

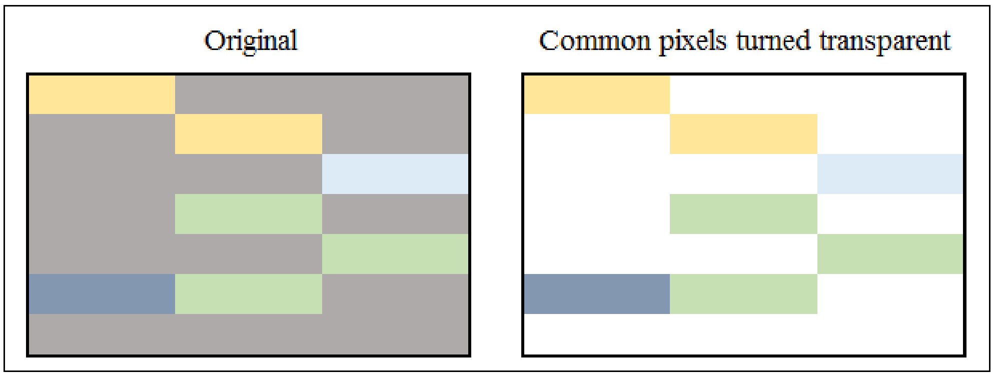


Figure 25 - Example of switching the most common pixels to transparent.

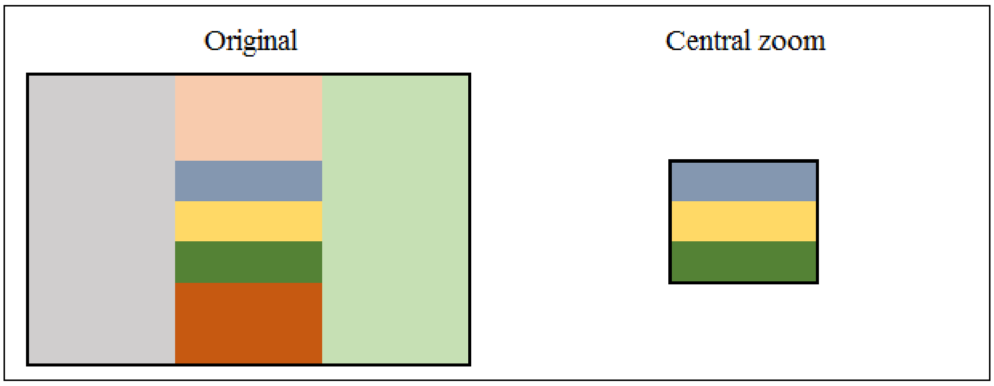


Figure 26 - Example of making a zoom to focus the centre of an image.

4.3.3. Interface

Interface elements were fully developed using the JavaFX platform. It consists on a set of graphics and media packages enabling the development of rich client applications, operating across diverse platforms [8]. Some alternatives to JavaFX were Java Swing[9], SWT: The Standard Widget Toolkit[10] or apachepivot[11]. JavaFX platform was chosen because of the ease of implementation provided by Scene Builder[12].

The tool was divided into three main screens and one secondary (see Figure 27):

- Main GUI;
- Recording GUI;
- Script details GUI;
  - Displaying script images.

[8] https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm
[9] https://www.javatpoint.com/java-swing
[10] http://www.eclipse.org/swt/
[11] http://pivot.apache.org
[12] https://gluonhq.com/products/scene-builder/

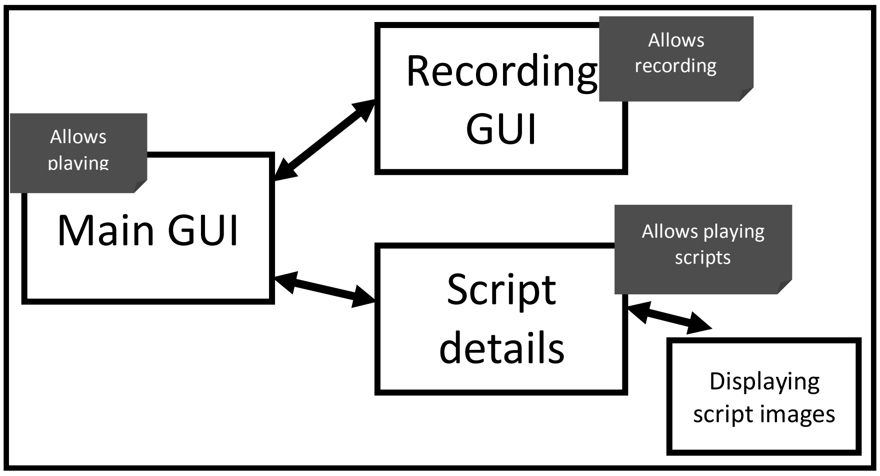


Figure 27 - Synthesis of available screens and navigation.

On the Main GUI (see Figure 28) scripts are presented shortly, with information to their name, description and current rate. On the view it is possible to see six scripts and there are two buttons on the bottom that allow page navigation. A search function is available, users must type keywords on the search bar and the scripts containing those keywords on the title or description will be presented. On this view it is also possible to record new scripts, view the detailed script information and run them.

The Recording GUI (see Figure 29) asks the user for information as script name, script description, which browsers will be used during the action recording and allows the insertion of other applications to be used. The information regarding the script name and description are mandatory. Only after filling the information the user can start recording his automation.

The Script GUI (see Figure 30) provides more information about the chosen script. This information is all stored on a *json* file created when the script was created. It is displayed the script name and description, the browsers which the script has been tested on, other application that are used, the performance rate (one to five) with the number of votes and the success rate. The success rate is automatically calculated each time a script is ran, if it succeeds it counts as a positive vote and if it fails it counts as a negative vote. On the *json* file the number of times that the script has been ran is also saved. On this screen people can access the performance of the script and it is instantly updated. When displaying script images (see Figure 31), it is a three by three grid which on each slot one image is displayed. Images are displayed by the order they will be searched when running the script and there are two buttons allowing the navigation between pages, for cases where a script uses more than nine images.

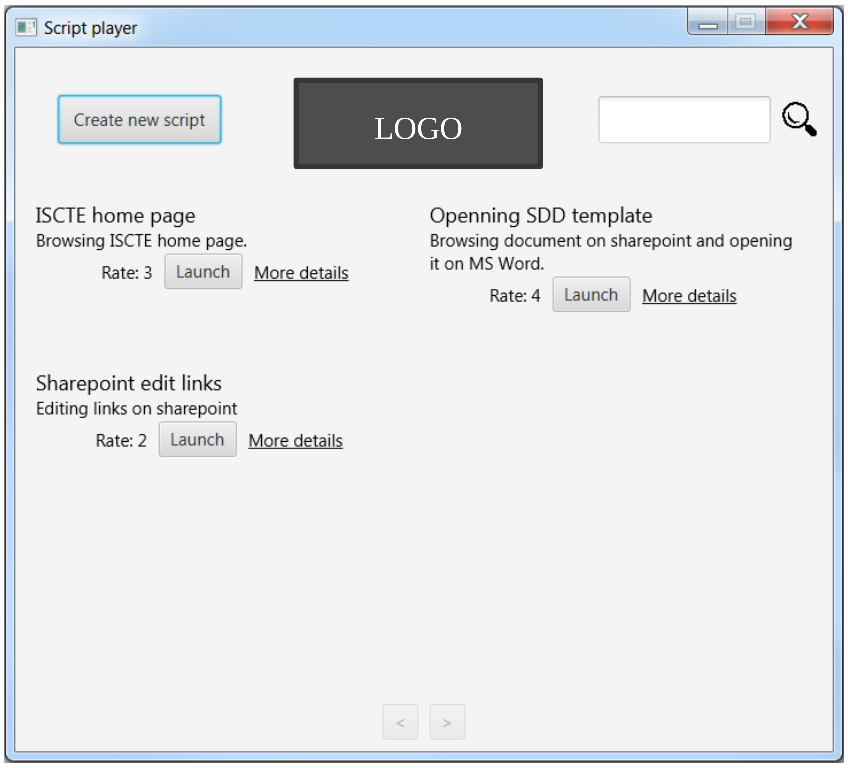


Figure 28 - Main GUI of the help tool

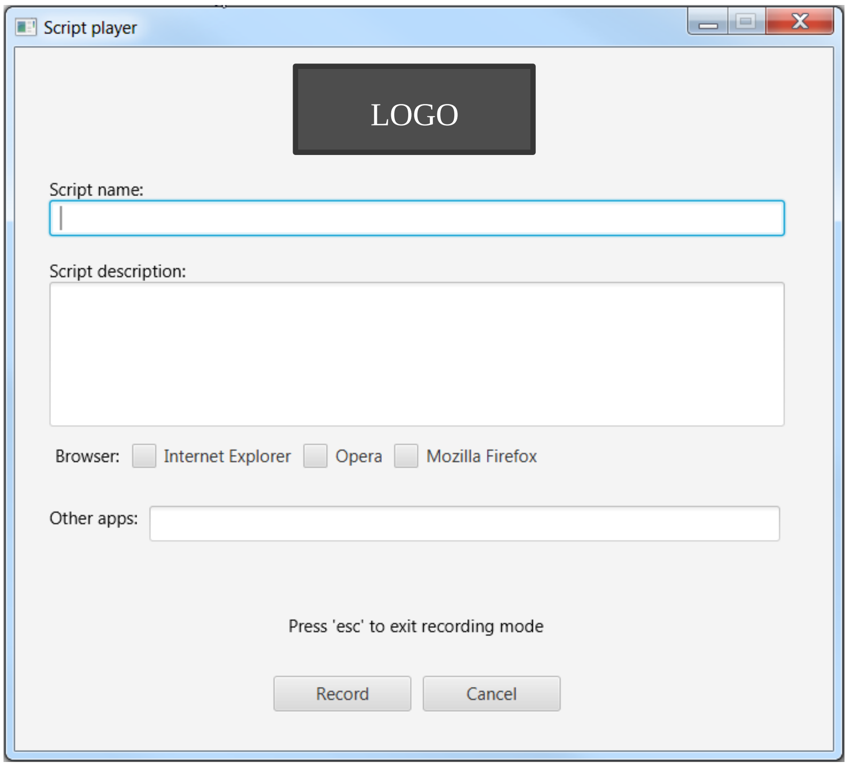
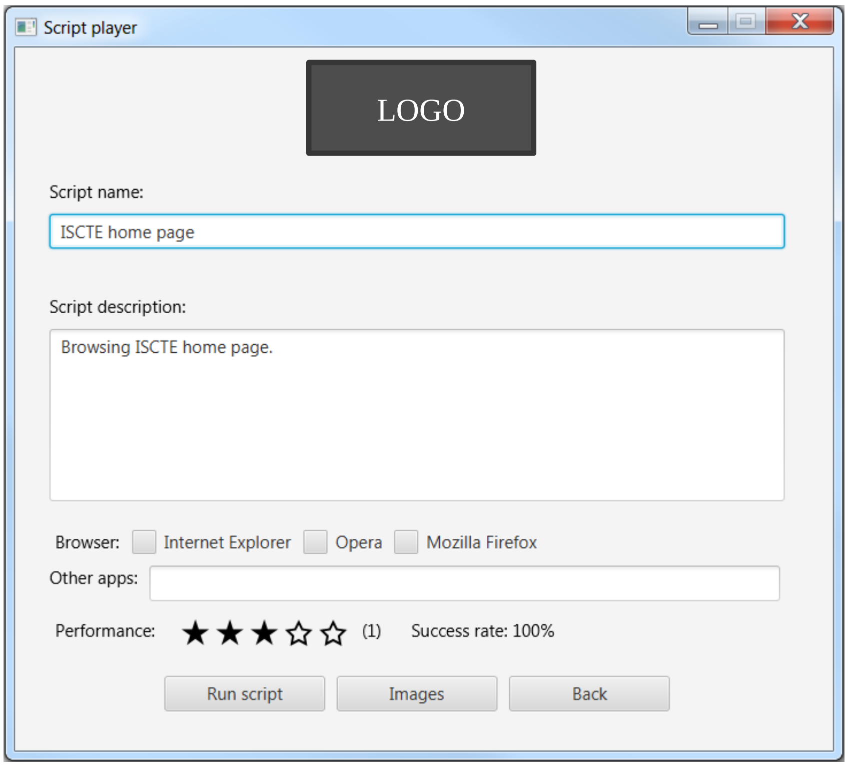


Figure 29 - Recording GUI of the help tool



Script player

LOGO

Script name: ISCTE home page

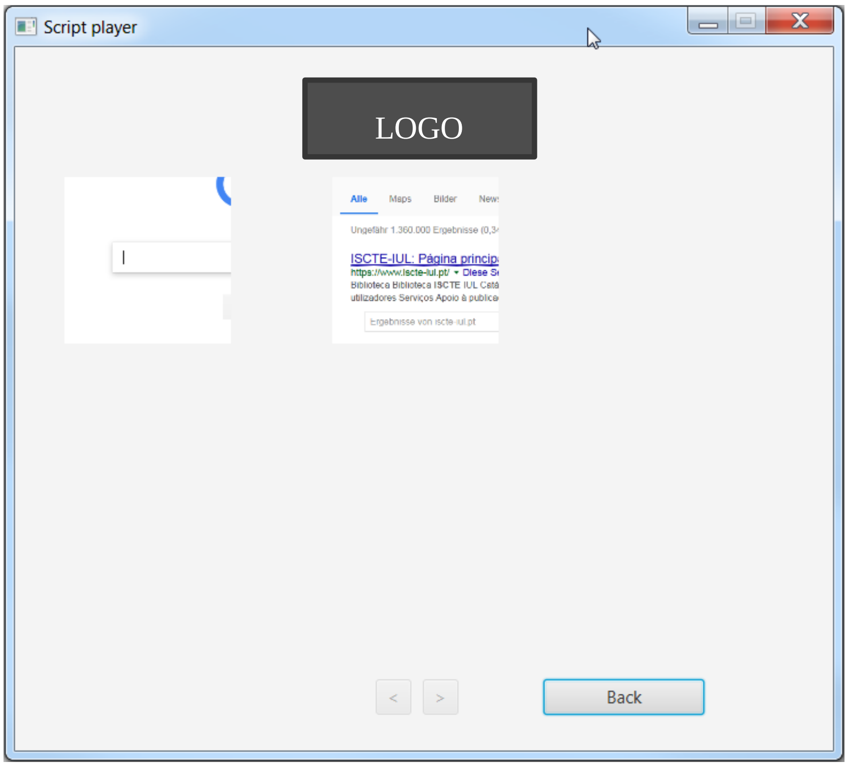
Script description: Browsing ISCTE home page.

Browser: ☐ Internet Explorer ☐ Opera ☐ Mozilla Firefox

Other apps:

Performance: ★★★☆☆ (1) Success rate: 100%

Figure 30 - Script information GUI of the help tool



Script player
LOGO
Alle
Maps
Bilder
News
Ungefähr 1.360.000 Ergebnisse (0,3
ISCTE-IUL: Página princip
https://www.iscte-iul.pt/
Biblioteca Biblioteca ISCTE IUL Catá
utilizadores Serviços Apoio à publica
Ergebnisse von iscte-iul.pt
<
>
Back

Figure 31 - Script images GUI of the help tool

4.4. CONCLUSION

This chapter focuses mainly on the development of the tool. It starts with the description of a proof of concept. This was done with a preliminary tool that interacted with external services (Sikuli) to be able to play pre-made help scripts. This proof of concept was done to verify the feasibility of the idea, making a help system based on both automation and PDC. Despite the reduced number of users, results tended to be positive. Additional feedback was collected and was considered on further developments.

Then, the two principal modules of the improved application are described. These are the Recorder which allows users to create and share new scripts and the Player that displays all the script metadata/information and allows the automation to be reproduced through PDC.

The last section described the technical implementation of all needed features. It describes the implementation of the recorder (how the user actions on the computer were monitored), the player (how the tool is able to re-play the automation) and the GUI (interface where everything is presented to the user and allows him to interact with the tool).

Chapter 5

EVALUATIONS

This chapter presents user studies. The first two evaluations were the basis for the development of the final tool. The first was presented on section 4.1.FEL TOOL – PROOF OF CONCEPT. The last evaluation was designed to perceive the benefits of the work developed as the users' acceptance and opinion.

On the first section of this chapter the second evaluation made with people is presented. It consisted on a questionnaire. The first evaluation was already presented on 4.1. FEL TOOL – PROOF OF CONCEPT.

The second section presents a usability study made with the developed help tool.

5.1. IDENTIFYING PLATFORMS AND TASKS

The second evaluation was planned to identify problematic platforms and tasks on Technological. It was considered that platforms and tasks which are harder to use should be the ones to try a different help-system, being possible to figure if the help would cause any perceived usability improvement.

This identification was done through a questionnaire on Technological (available on:Attachment A), company previously presented. On this questionnaire first, the demographic characterization of the people answering was made. Then, they were asked to identify platforms and tasks that might causes doubts to new comers, platforms and tasks that cause errors or doubts when performing periodic tasks and to identify platforms and tasks with insufficient help.

On the Table 3 the demographic data from the 59 participants is presented.

Regarding platforms most people considered that there are platforms which might cause doubts to new users. The opinions regarding platforms with insufficient help system were splitted and most people considered that regarding periodic tasks there are not platforms causing doubts. When asking the identification of the platforms three came up. Those are related to human resources, time recording (with more incidence), and billing declaration. On Figure 32 it is shown the classification that people gave to the platform complexity on the three asked questions.

In what concerns tasks most people, again, do not recognize periodic tasks that cause doubts or errors. People’s opinion seems split when it concerns tasks that are hard to learn by new commers and when asked if automation and PDC would be beneficial on a help-tool most people seem to not know or not have an opinion. Regardless of these distributions filling work hours and booking travels were identified has complicated tasks being the hour allocation the one with more occurrence. On Figure 33 it is shown the classification given by people when evaluating the usability/complexity of tasks.

Table 3 - Demographic information of users participating on the questionnaire

		Count	Percentage
Number of participants		59	100%
Gender	Male	45	76%
	Female	14	24%
Age	Between 20 and 29	9	15%
	Between 30 and 39	34	58%
	Between 40 and 49	15	25%
	Between 50 and 59	1	2%
Years in the company	Less that 1	20	34%
	Between 1 and 2	12	20%
	Between 2 and 3	10	17%
	Between 3 and 4	11	19%
	Between 4 and 5	2	3%
	More that 5	3	5%
Company area/field	IT	57	97%
	IT/HR	2	3%

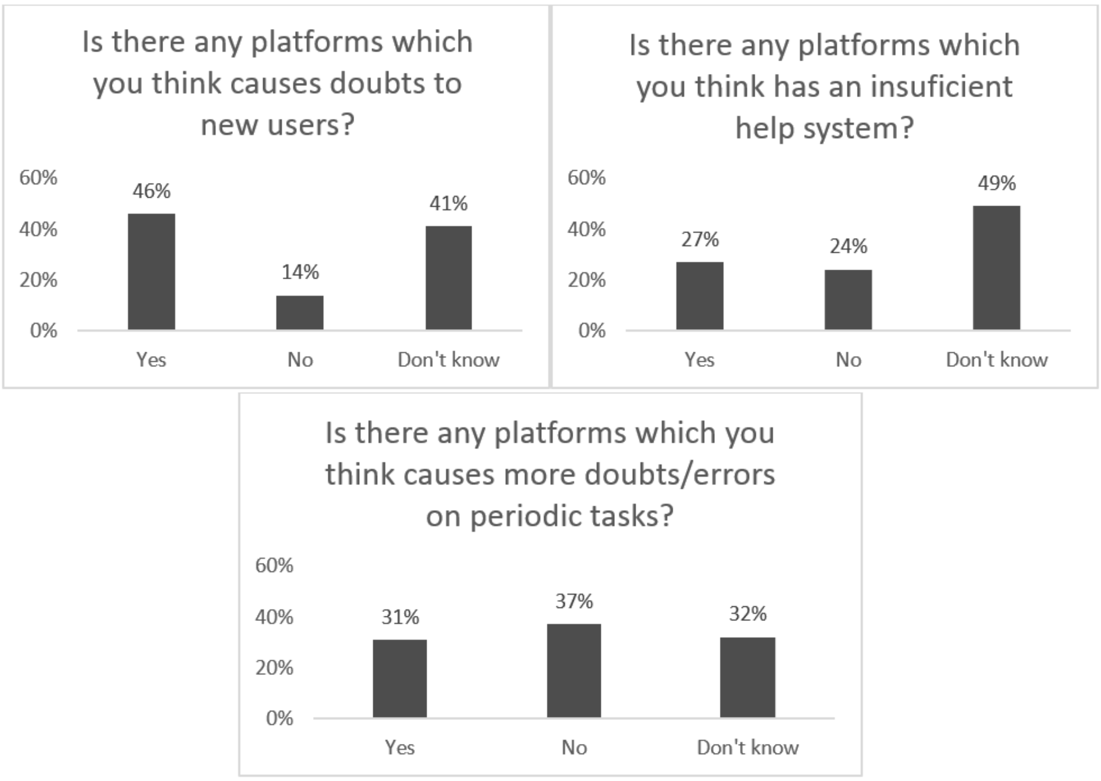


Figure 32 - People's opinion on platforms usability.

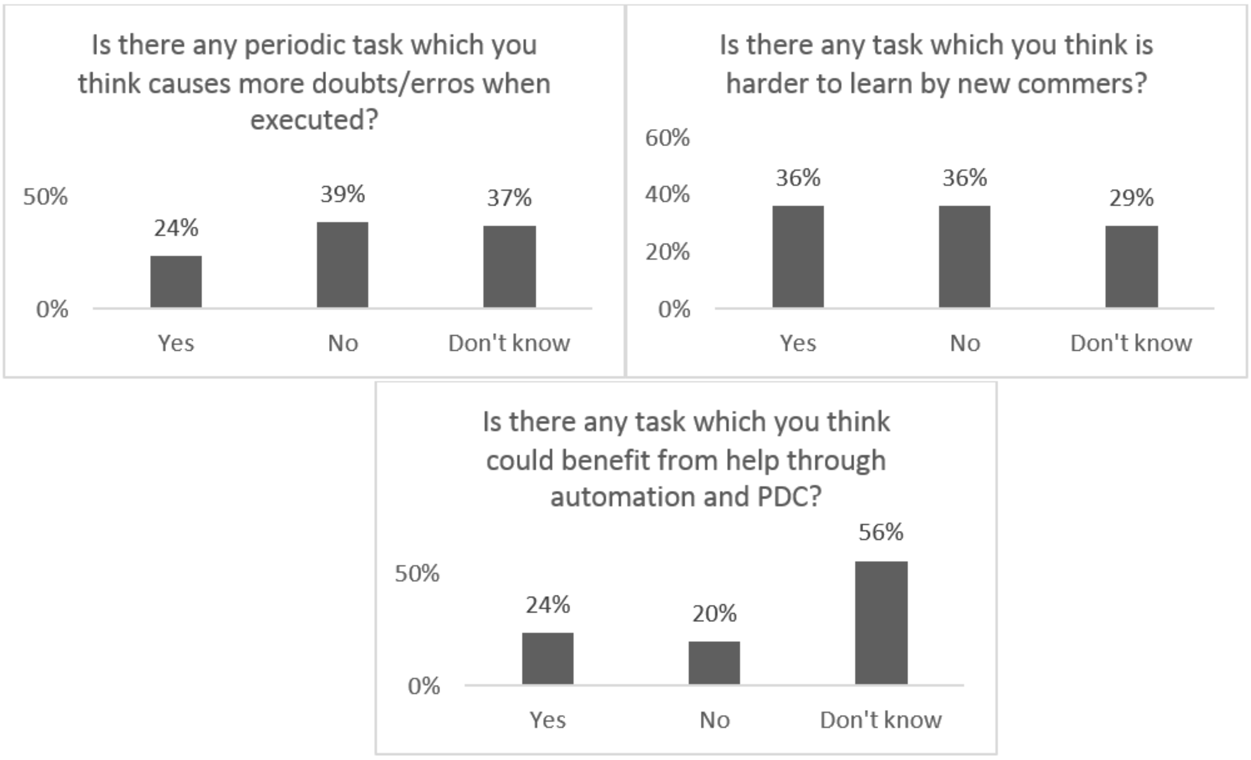


Figure 33 - People's opinion on tasks usability.

5.1.1. Conclusion

The number of responses obtained was satisfactory (59), and all questions were answered. Besides the amount of “Don’t know” answers when identifying problematic tasks and platforms a good variety of platforms/tasks was presented. For reasons of confidentiality to Technological these platforms and tasks can not be identified in further detail.

From the interception of faulty platforms and tasks that create doubts, the time reporting platform was identified as the one requiring improvements. People commented that the improvement could be at the design level or through new help mechanisms since now there are none focused on this task.

5.2. USABILITY STUDY

Considering the previous evaluations, the help-tool and some example scripts were developed (example scripts on Attachment B). With this new tool a user-evaluation was conducted.

This evaluation aimed to perceive the concrete benefits of an automation and PDC oriented tool. Besides the ability to perform the task and the opinion regarding the help tool, demographic data and perceived complexity of the tasks to be performed were gathered.

5.2.1. Participants

Participants chosen for this study were all employees from Technological, working under the IT department. A user study with 32 participants was made and approximately 80% of the sample was male.

Most (97%) had already used the platform were the tasks from this study had to be carried out.

5.2.2. Procedure

To evaluate our tool participants were split on two groups and only the group two used the developed help tool.

This evaluation was composed of two groups and five tasks. The choice of dividing people into two groups was due to the need to verify if the chosen tasks met some difficulties in their accomplishment without any kind of help mechanism, as well as to verify if the time spent was improved with the aid of automation. The use of only one group would be advised if we were measuring the "before" and the "after" having some apprenticeship in the middle. Users were presented with a description on the task to be performed. Group one had to perform the task without any kind of help system and group two used the developed help tool.

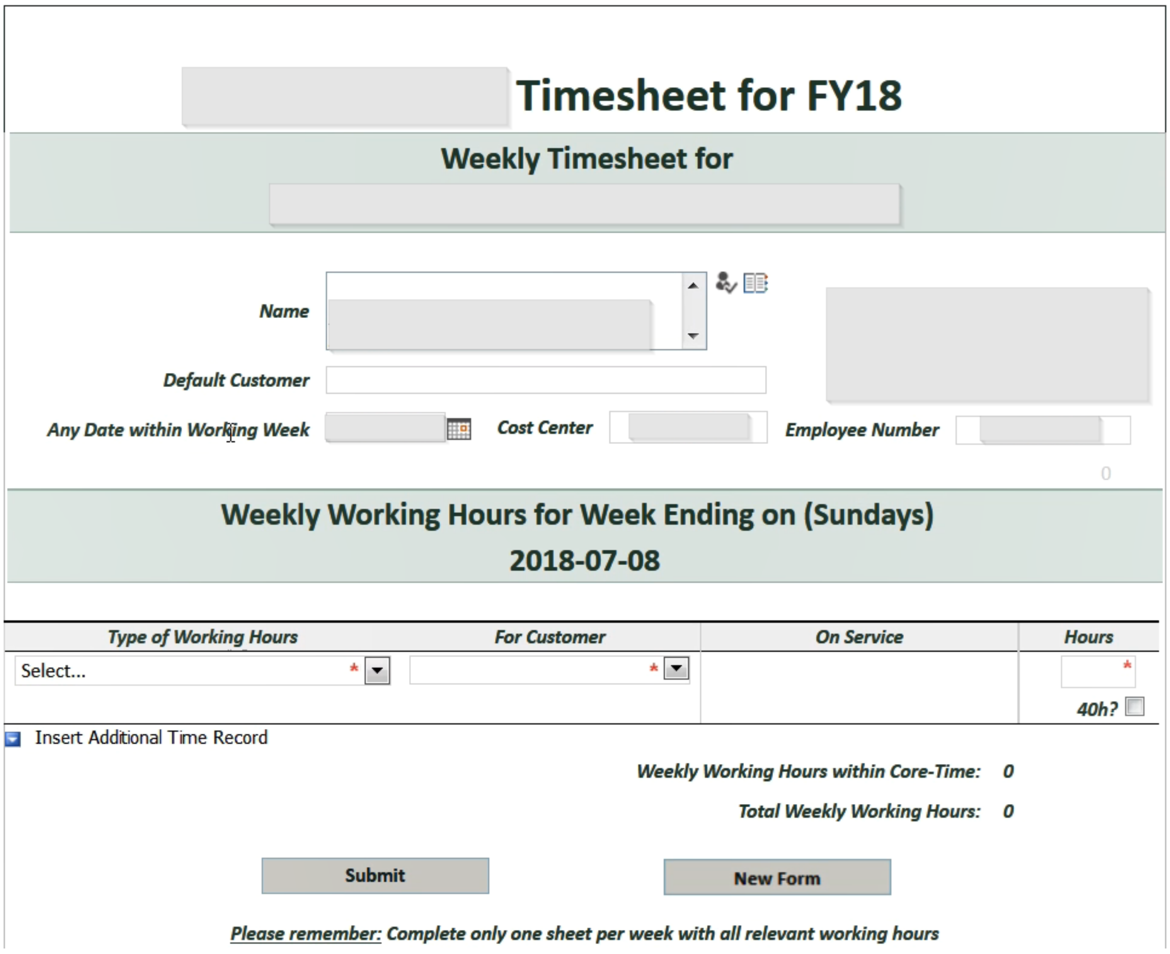
The five tasks chosen to be part of this evaluation were:

1. Please book 40 work hours on the Timesheet under client 'X' and service 'N' for this week;
2. Please book 16 absence hours and 24 work hours on the Timesheet under client 'X' and service 'N' for this week.
3. Please book 40 work hours on the Timesheet under client 'X' and service 'N' for the next week;
4. Please book 40 work hours on the Timesheet under client 'X' and service 'N' for this week. Also, fill in 20 work hours from the past billing period;
5. Please export the excel file with the hours booked and check how many hours are filled for User-ID 'Y' on the week ending at May 27.

On Figure 34 it is presented an image of the Timesheet.

The execution of the tasks was recorded and then analyzed. Notes were also taken during the sessions.

On the context of this user-test the most important aspects were the completeness of the task and the time to perform them. Since in this study people did not have to perform tasks for a second time, learning is not evaluated. The study was designed in this way because the main


Timesheet for FY18

Weekly Timesheet for

Name	
Default Customer	
Any Date within Working Week	
Cost Center	
Employee Number	

Weekly Working Hours for Week Ending on (Sundays) 2018-07-08

Type of Working Hours	For Customer	On Service	Hours
Select... *	*		* 40h? ☐

Insert Additional Time Record

Weekly Working Hours within Core-Time: 0

Total Weekly Working Hours: 0

Please remember: Complete only one sheet per week with all relevant working hours

Figure 34 – Timesheet

objective was to verify the effect of the aid tool in performing the tasks completely and because there was not another help mechanism which would allow the comparison in terms of apprenticeship.

At the end, users using the help-tool filled a questionnaire that addresses four aspects: Usefulness, Ease of Use, Ease of Learning and Satisfaction (as defined in the standard USE questionnaire[13]). Answers were on a 7-point Likert scale with values from 1 (very strongly disagree) to 7 (very strongly agree).

5.2.3. Measures

In this test will be measured the time of accomplishment of the tasks, its success, answers to a preliminary questionnaire regarding the platform and the tasks to be executed, and answers to a USE questionnaire.

In addition, a parametric test (2-sample t test). This was chosen because the amount of people sorted in each group is superior to 15 people, and the spread of the results is different

[13] Retrieved from: http://garyperlman.com/quest/quest.cgi?form=USE, on July 2018

from groups, due to users using the automation are restricted to the automation executions time.

5.2.4. Results

On Table 4 the demographic results are presented. Most users participating were male, under 20 to 29 years old and working on Technological for less than 1 year.

Table 4 - Demographic information of users participating on the usability study.

		Number of participants	Percentage
		32	100%
Gender	Male	26	81%
	Female	6	19%
Age	Between 20 – 29	17	53%
	Between 30 – 39	13	41%
	Between 40 – 49	2	6%
Years in company	Less than 1	21	66%
	Between 1 – 2	5	16%
	Between 2 – 3	3	9%
	Between 3 – 4	2	6%
	Between 4 – 5	1	3%

Most participants had already some experience with the tool for which the scripts were developed but not with the tasks to be performed. Almost everyone had checked the number of hours they had filled, some had already booked holidays, and most did not book extra-hours. On Figure 35 the answers to the questions are presented.

When asked about perceived complexity to accomplish the tasks (task 1 to task 5) the opinions were widely dispersed (see Figure 36). Taking it into account, mode values were analyzed (not considering N/A). The values were taken from a 7-point Likert scale being the value 1 a task with low complexity and the value 7 high complexity. The mode value for weekly hours booking was 2, for absence hours 3, for future hours 2 and 5, past billing hours 5 and export time records to excel 3 (see pointed that script description could be clearer and that task execution could be faster. One user also referred that there could be less popups.

Table 5).

On Figure 37 the success percentage when performing the five tasks for group 1 are presented. Task 1 to 3 were done by almost everyone, showing some difficulties on the last two tasks. Looking at the completeness of task 4 and 5 the results are very distinct, 81% of the participants could not perform task 4 and 88% could not perform task 5. People using the developed help system were all able to perform the five tasks.

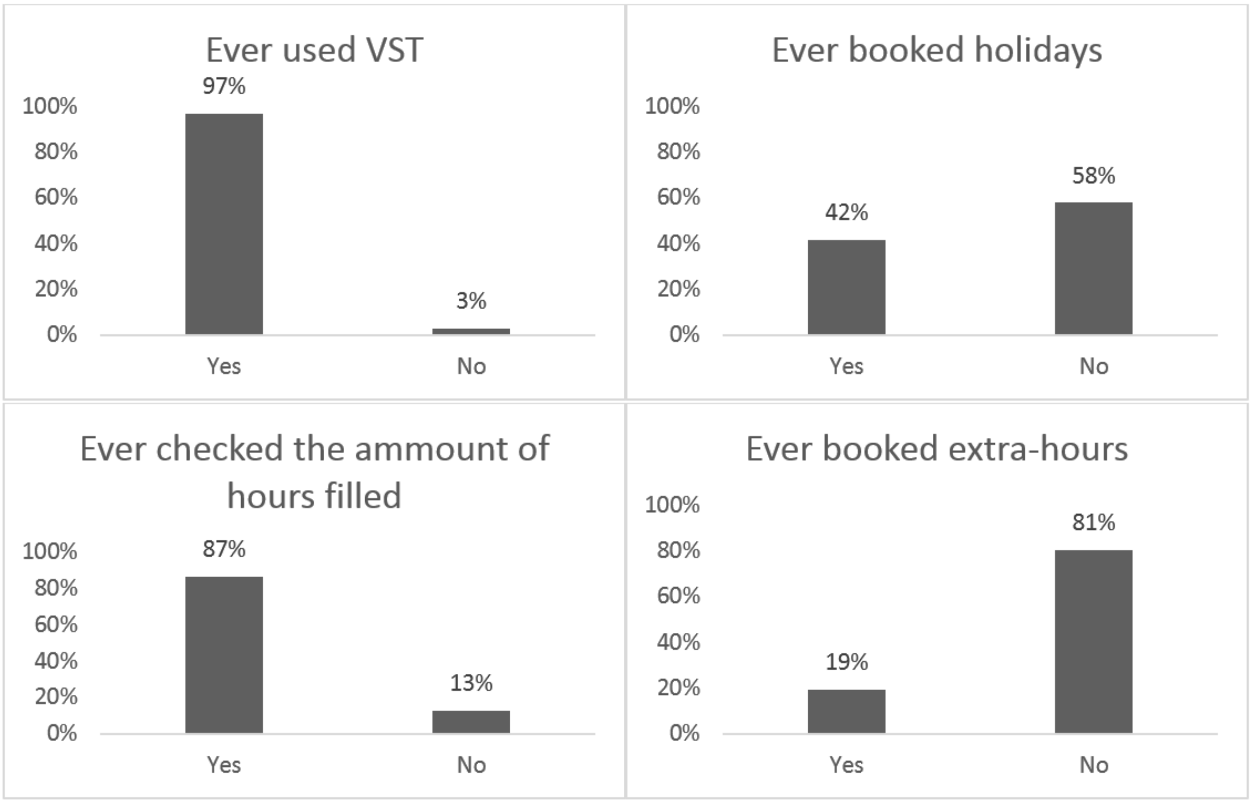


Figure 35 - People experience executing tasks on Timesheet (also known as VST).

Figure 38 presents the time spent by people from group 1 and from group 2 when performing the five tasks. Performance time for group two is equal to the time that the automation script takes and, so, the standard deviation is equal to zero. When looking at times that people from group 1 took we find high deviation values for times spent on task 4 and 5 and some deviation for task 2. Time spent by people from group 1 when booking the standard weekly hours (task 1) was similar.

When tracking the usability of the tool, through the USE questionnaire, positive values were gathered. The results were grouped by its categories Usefulness, Ease of Use, Ease of Learning and Satisfaction. Mode values for the previous categories are presented on the table below. On the figures bellow it is also presented the average values for each question under its category. On Attachment C, a table is presented with the full information of the USE Questionnaire, questions and results for each question.

When giving opinions on the help tool people referred that the tool was intuitive and easy to use and that it poses as an advantage for those that are not familiar with the targeted applications (application with tasks automated). That it was good to see the tasks step by step and that highlighting the components before interacting with them helps focusing attention. People also said that it is easier and faster to get things done without having to think much. Custom made scripts were also pointed as a good feature. On a less positive side, people

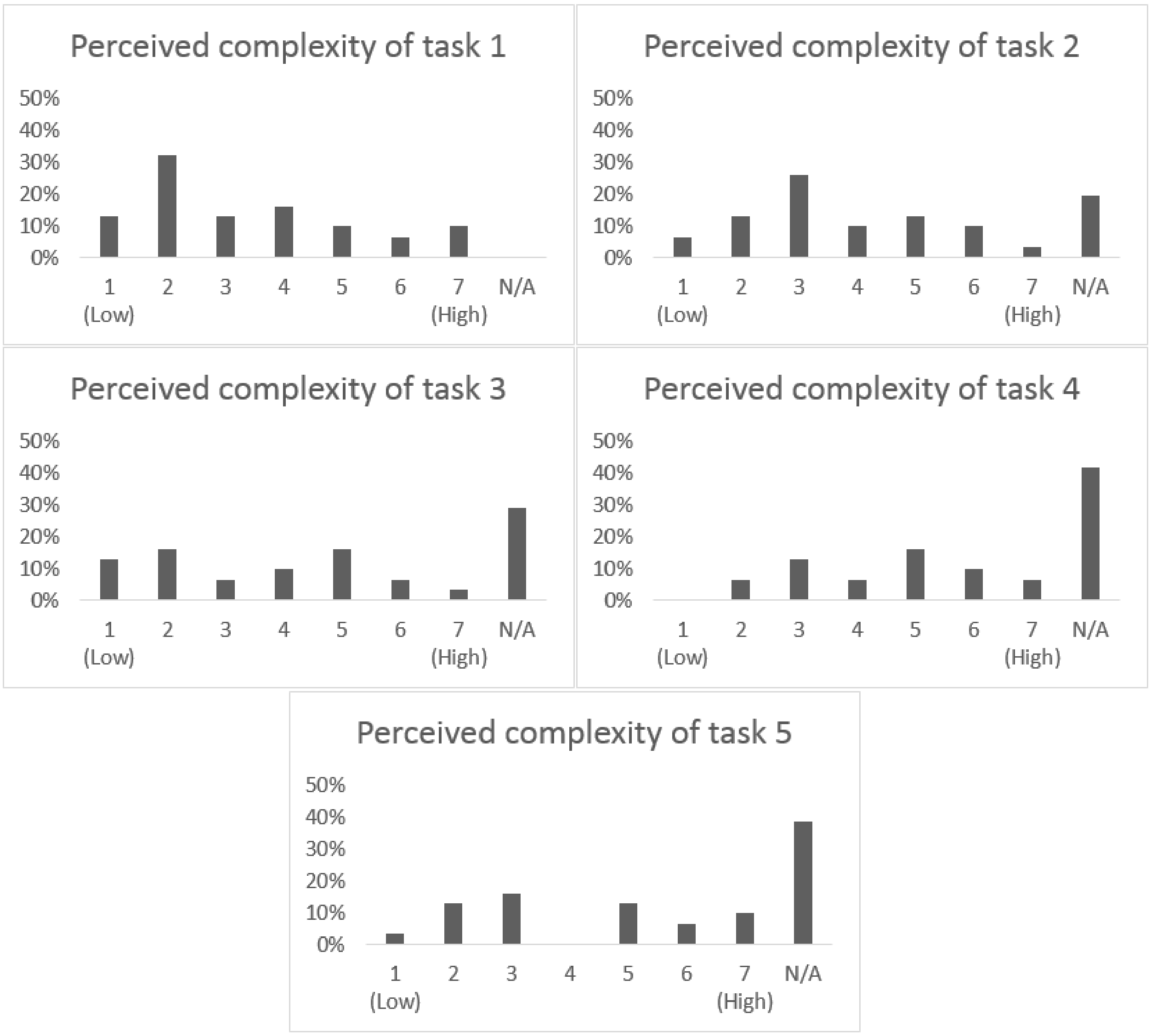


Figure 36 - Tasks' perceived complexity.

pointed that script description could be clearer and that task execution could be faster. One user also referred that there could be less popups.

Table 5 - Mode value when evaluating task complexity.

	Mode (1 – Low to 7 – High complexity)
Task 1	2
Task 2	3

Task 3	2 and 5
Task 4	5
Task 5	3

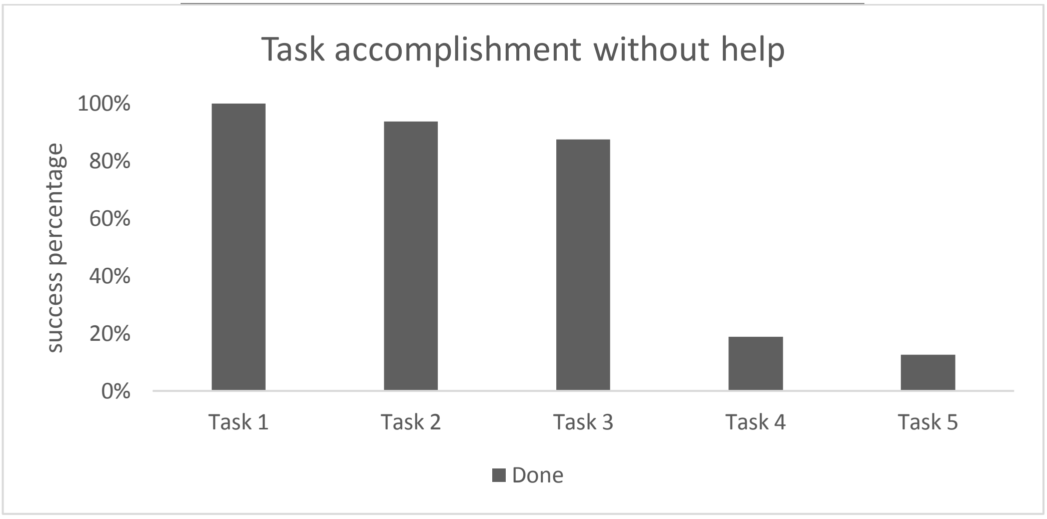


Figure 37 - Success rate while performing tasks without a help system

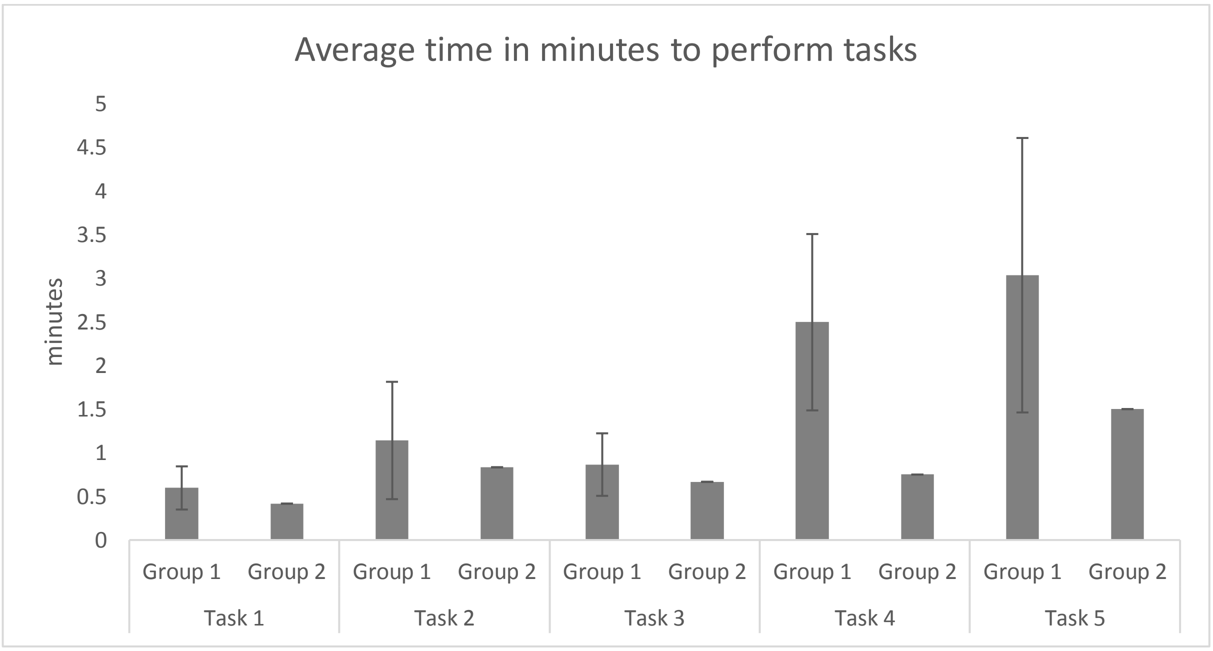


Figure 38 - Time spent executing tasks.

Table 6 - Mode value when evaluating the usability of the developed tool.

	Mode (Very Strongly Disagree to Very Strongly Agree)
Usefulness	Very Strongly Agree
Ease of Use	Very Strongly Agree
Ease of Learning	Very Strongly Agree
Satisfaction	Strongly Agree

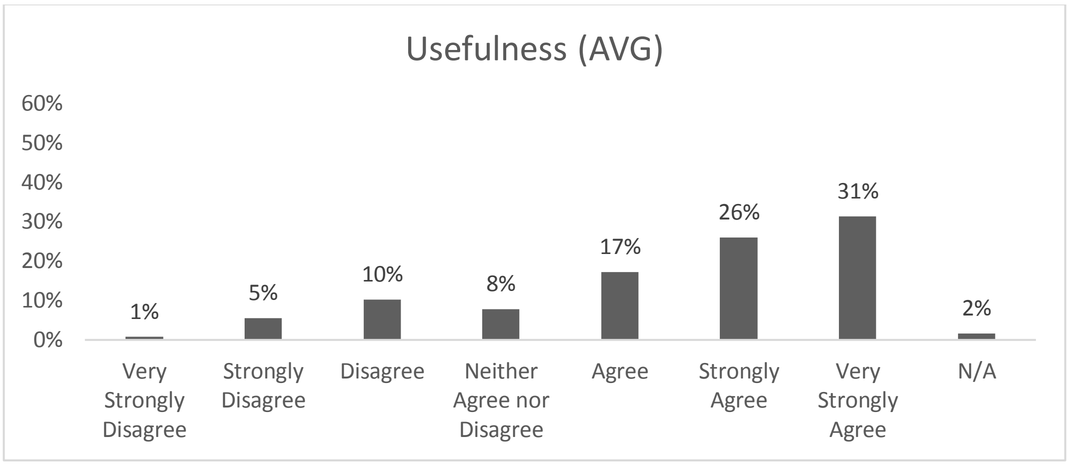


Figure 39 - Average result of the usefulness of the developed tool.

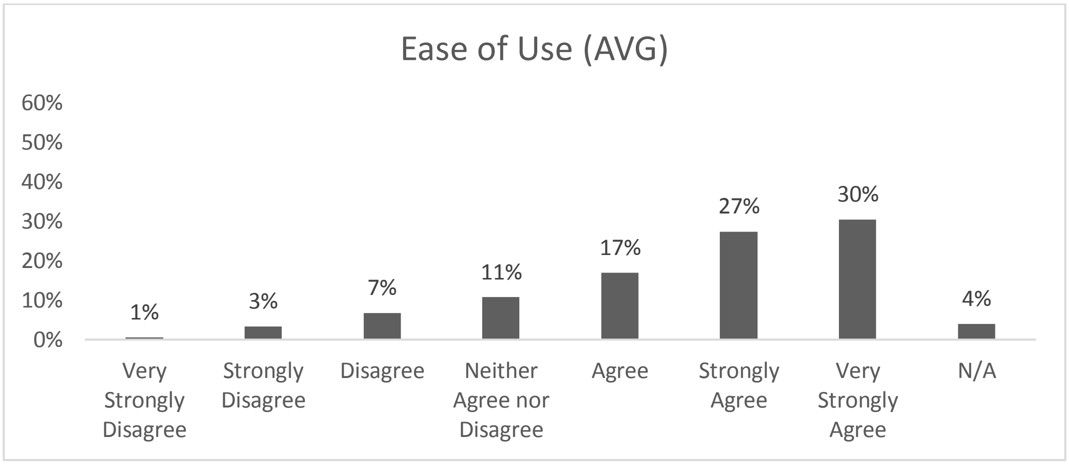


Figure 40 - Average result of the ease of use of the developed tool.

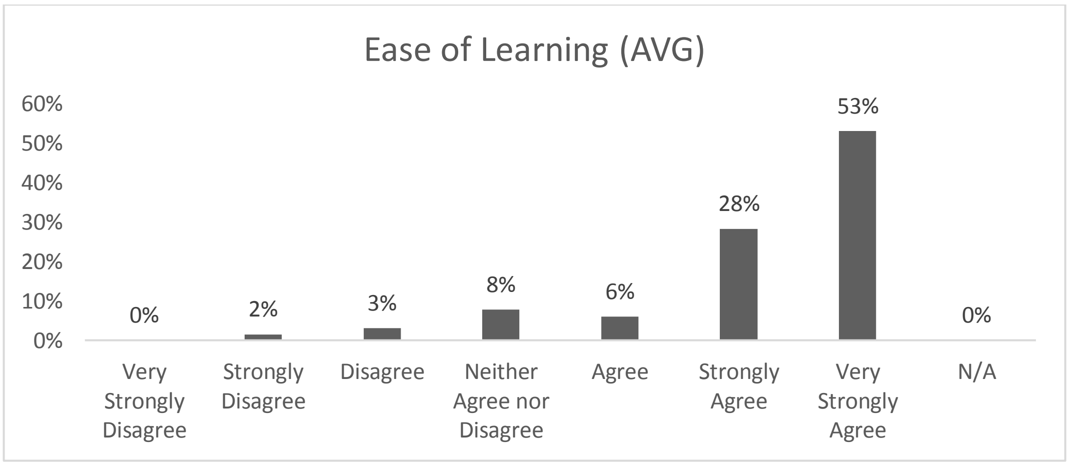


Figure 41 - Average result of the ease of learning of the developed tool.

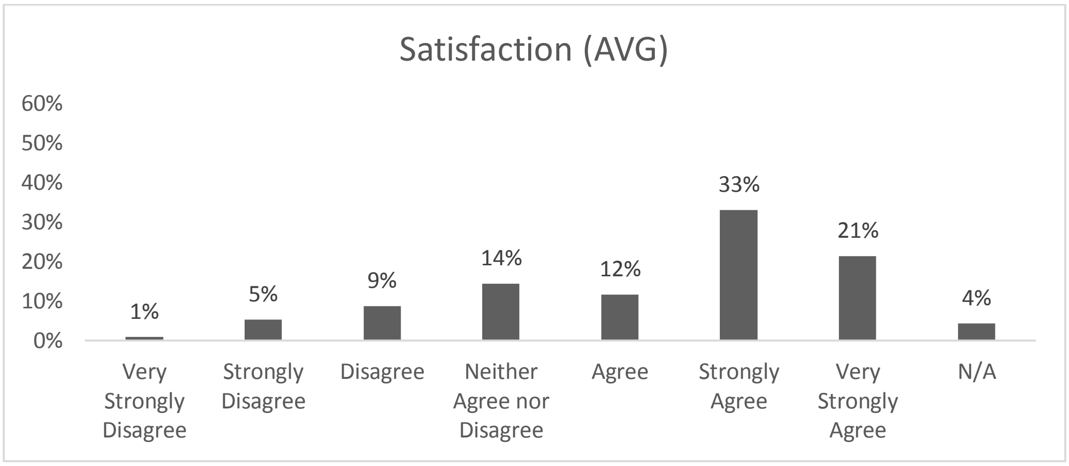


Figure 42 - Average result of the satisfaction of the developed tool.

On Figure 43 and Figure 44, a line representing the average values for the platform complexity and the perceived complexity for task 4 and 5 are shown. On the pictures it is also represented (only on the second) if the task was ever done and if the task was performed successfully. Success and if the task was ever done, if positive were given the value 6 of the Y-axe, and if negative the value 2, considering that platform and task complexity were rated on a 7-point Linkert-Scale (7 represents high perceived complexity). Values 2 and 6 were chosen to pass the visual perception of failure and success, respectively. These were the extreme values that allowed the standard deviation to not reach negative values

For this study the H0 was defined as “The use of the developed demonstration-based help tool has no influence in the execution of the tasks”. The t-student test was run with the information of the times that people of each group took to complete the tasks. The result of the 2-sample t test was approximately 0,000002. This result allows to reject H0 with a confidence of 99% and to conclude that the use of the developed demonstration-based help tool has influence on the execution of tasks.

5.2.5. Conclusions

Reaching the end of this evaluation it is considered that the platform and the tasks chosen were appropriate because difficulties and doubts were recognized throughout the process, but users were able to maintain the iteration

It is identified as a weak point of the evaluation all people executing the tasks on the same order. It is thought that a random question order would be better since, when reviewing through video, is seems like people tend to give up more easily on task 5 if they have failed to perform task 4.

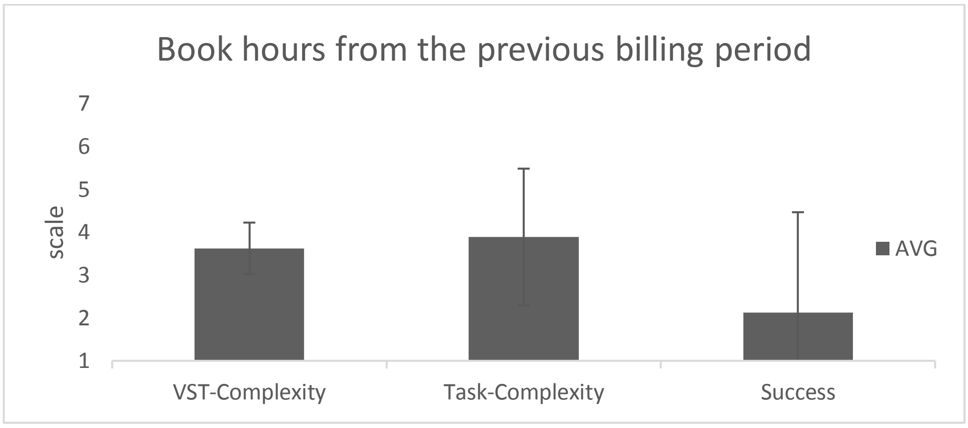


Figure 43 - Representation of the perceived platform and task complexity with the success rate for booking hours from the previous billing period.

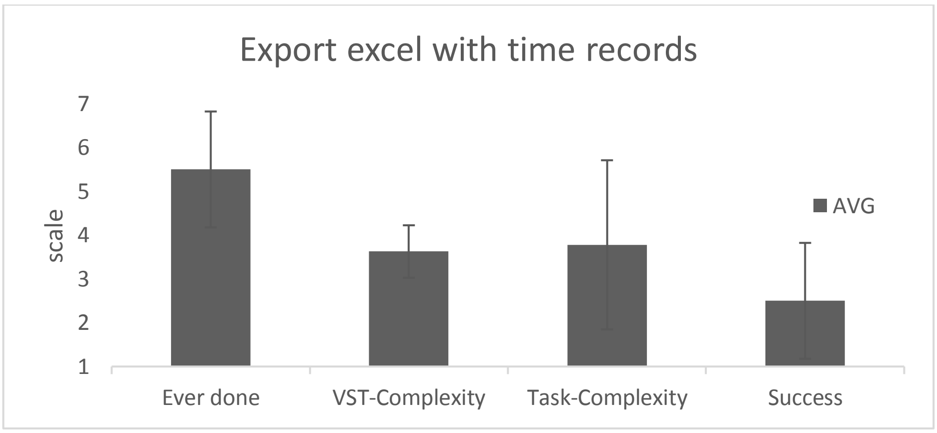


Figure 44 - Representation of the perceived platform and task complexity with the success rate for exporting an excel file with time records.

Considering the interference of the help-tool, the fact that everyone was able to execute all proposed tasks is an excellent indicative. Besides that, when comparing average times when performing tasks, the average time was smaller for every task when using the help tool. The average time-performance gain was on a factor of 40%. It is believed that, at the level that the user learns to perform the tasks, this factor decreases, allowing the user to perform the task faster than the automation.

This page was intentionally left in blank

Chapter 6

CONCLUSIONS AND DISCUSSION

This chapter presents the work conclusions and discussion. A reflection on the work developed is done.

Chapter 6 – Conclusions and discussion

This chapter will present on its first section the summary of contributions made. After, the research question, does a help tool based on demonstration, automation and PDC improve the success of task completion, time spent, and decreases the cost of learning, is answered. The last two sections present the discussion and future work.

6.1. SUMMARY OF CONTRIBUTIONS

The final tool contemplates the desired characteristics. It is a help application and the aid is given on a demonstration-based way. Every interaction of the automation script is executed on the target application and users learn the task while watching it getting done. Interaction across multiple systems in just one help script is also possible. This way, on multiple application tasks the user does not have to search multiple help systems.

The tool has been deployed on a shared system, allowing scripts to be synchronized, and people who have access to the tool to also have access to all the scripts that are developed and available. Like it has been said before, this is also a tool that does not require computer or programmer wise users, it is a tool that requires task wise users that want to share their knowledge and help others.

The proof of concept allowed the confirmation of user acceptability and shown indicative results of good usability and demonstration-based apprenticeship. With the questionnaire it was possible to improve the usability of an application used by most collaborators of Technological by its identification and the development of scripts representing the tasks that can be performed there.

The last study, the usability study, presented the final tool and accessed users' opinion regarding it and evaluated task performance when using the tool side-by-side with the application. With this one it was possible to answer the research question and to have results on the usability, ease-of-use, ease-of-learning and satisfaction of the developed work.

6.2. ANSWER TO RESEARCH QUESTION

The work developed allows the answer to the research question defined. The results of the tests performed with the developed demonstration-based help tool indicate an improvement on task performance.

Regarding the usability study (5.2. USABILITY STUDY) results support to claim that the developed demonstration-based help tool has positive influence on the execution of the tasks. Results show an improvement on the time spent on every tested task. With the tool tasks which users could not fully execute also became executable with an 100% success rate.

When comparing the learning effect with the proof of concept (chapter 4.1.), despite the reduced number of user-tests, the results seems to indicate that the demonstration-based tool enables users to perform tasks easier and faster without compromising too much the learning. The study shown much faster execution times while executing tasks with the developed tool in comparison with traditional help mechanisms (written tutorials).

Focusing on these two interactions with people and analyzing the results the opinion is that the demonstration-based help tool can improve time spent on task execution, allow tasks to be completely executed, without compromising learning.

6.3. DISCUSSION

By researching the existing tasks and platforms on Technological, it was possible to recognize that most applications do not yet have help mechanisms, or that existing ones are written tutorials available on an external site.

While running the usability test the learning effect compared to old help systems was not studied. This was because the chosen platform had no help system and people could not perform the tasks. At the time, people could only learn task execution by asking a colleague.

The number of errors committed was also not analyzed in the usability test. This choice was made since the automation accomplishes most of the steps of the task leaving a small margin for the occurrence and measurement of errors. If there were several scripts in the tool this could be a confounding factor leading to error, however, this was not the case. Another case would be the scripts made available not working properly forcing people to perform the task manually.

With the usability study the confirmation that there are still interfaces hard to use and that the inability to execute tasks completely, by people on group 1 (no help mechanism), causes frustration was also observed. People tended to complaint on the platform and to give up easier on following tasks.

Given the time to perform tasks comparisons, it is possible that as users are more familiar with the task, the manual execution time of it is close to the automation time. However, for this to happen people have to want to learn how to accomplish the task. It is possible that there are users who want the task to be performed only through automation, using this time to do other tasks.

The creation of new scripts was not evaluated. This choice was taken because it is considered that the recording of a good script is dependent on the knowledge that the user has on the task. The developed demonstration-based help tool does not influence the user in any form when recording a task. It poses as a transparent layer capturing all inputs (mouse and keyboard) and is only turned off when the user presses “Esc”.

The long-term use and the creation and sharing of scripts was also not studied. This was due to the fact that this evaluation would require more time that the one available for the development of the thesis. The final tool would have to be released and only after some time it would be possible to access the increase (or not) of the number of available scripts. Nevertheless, when accessing the tool on the USE questionnaire people gave classifications that indicate interest in using it and satisfaction with its use.

6.4. FUTURE WORK

This dissertation indicates that there is still work to be done in the area of systems usability. It verifies that there are still applications hard to use and that help mechanisms are not a priority, existing day-to-day application with no kind of help. It also shows good results on the development of demonstration-based help tools.

As further work, continuous learning capabilities would be a major improvement. On pixel and image recognition this capability could help the tool to better search and identify the screen elements and improve the success rate. Also, an assessment of people long-term use of the tool would be desired proving the willingness to create, share and help others.

REFERENCES

Akiki, P. A. (2018). CHAIN: Developing model-driven contextual help for adaptive user interfaces. *Journal of Systems and Software*, *135*, 165–190. https://doi.org/10.1016/j.jss.2017.10.017

Akiki, P. A., Bandara, A. K., & Yu, Y. (2013). Cedar studio: an IDE supporting adaptive model-driven user interfaces for enterprise applications (p. 139). ACM Press. https://doi.org/10.1145/2494603.2480332

Allen, J., Chambers, N., Ferguson, G., Galescu, L., Jung, H., Swift, M., & Taysom, W. (2007). PLOW: A Collaborative Task Learning Agent. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence* (pp. 1514–1519). Menlo Park, Calif: AAAI Press.

Amershi, S., Mahmud, J., Nichols, J., Lau, T., & Ruiz, G. A. (2013). LiveAction: Automating Web Task Model Generation. *ACM Transactions on Interactive Intelligent Systems*, *3*(3), 1–23. https://doi.org/10.1145/2533670.2533672

Blue Prism - Robotic Process Automation. (n.d.). Retrieved 30 December 2017, from www.blueprism.com

Bolognesi, T., & Brinksma, E. (1987). Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, *14*(1), 25–59. https://doi.org/10.1016/0169-7552(87)90085-7

Boy, G. A. (1998). Cognitive Function Analysis for Human-centered Automation of Safety-critical Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 265–272). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. https://doi.org/10.1145/274644.274682

Card, S. K., Moran, T. P., & Newell, A. (2008). *The psychology of human-computer interaction =: XA-GB* (Repr). Boca Raton, Fla.: CRC Press.

Ceaparu, I., Lazar, J., Bessiere, K., Robinson, J., & Shneiderman, B. (2004). Determining Causes and Severity of End-User Frustration. *International Journal of Human-Computer Interaction*, *17*(3), 333–356. https://doi.org/10.1207/s15327590ijhc1703_3

Chilana, P. K., Ko, A. J., & Wobbrock, J. O. (2012). LemonAid: selection-based crowdsourced contextual help for web applications (p. 1549). ACM Press. https://doi.org/10.1145/2207676.2208620

Chilana, P. K., Ko, A. J., Wobbrock, J. O., & Grossman, T. (2013). A multi-site field study of crowdsourced contextual help: usage and perspectives of end users and software teams (p. 217). ACM Press. https://doi.org/10.1145/2470654.2470685

Dixon, M., Fogarty, J., & Wobbrock, J. (2012). A general-purpose target-aware pointing enhancement using pixel-level analysis of graphical interfaces (p. 3167). ACM Press. https://doi.org/10.1145/2207676.2208734

Dixon, M., Nied, A., & Fogarty, J. (2014). Prefab layers and prefab annotations: extensible pixel-based interpretation of graphical interfaces (pp. 221–230). ACM Press. https://doi.org/10.1145/2642918.2647412

Dragunov, A. N., Dietterich, T. G., Johnsrude, K., McLaughlin, M., Li, L., & Herlocker, J. L. (2005). TaskTracer: a desktop environment to support multi-tasking knowledge workers (p. 75). ACM Press. https://doi.org/10.1145/1040830.1040855

Dworman, G. (2007). Arbitration of a help system. *Interactions*, *14*(1), 39. https://doi.org/10.1145/1189976.1189999

Dworman, G., & Rosenbaum, S. (2004). Helping Users to Use Help: Improving Interaction with Help Systems. In *CHI '04 Extended Abstracts on Human Factors in Computing*

Systems (pp. 1717–1718). New York, NY, USA: ACM. https://doi.org/10.1145/985921.986198

Grossman, T., & Fitzmaurice, G. (2010). ToolClips: an investigation of contextual video assistance for functionality understanding (p. 1515). ACM Press. https://doi.org/10.1145/1753326.1753552

Hevner, A., & Chatterjee, S. (2010). Design Science Research in Information Systems. In A. Hevner & S. Chatterjee, *Design Research in Information Systems* (Vol. 22, pp. 9–22). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4419-5653-8_2

Hornbæk, K. (2006). Current practice in measuring usability: Challenges to usability studies and research. *International Journal of Human-Computer Studies*, *64*(2), 79–102. https://doi.org/10.1016/j.ijhcs.2005.06.002

iCS. (2010, 2015). HAMSTERS. Retrieved 9 March 2018, from https://www.irit.fr/recherches/ICS/softwares/hamsters/index.html

Instructional Systems Development (ISD). (n.d.). Retrieved 8 March 2018, from http://www.au.af.mil/au/awc/awcgate/doe/isd/paper.htm

Intharah, T., Firman, M., & Brostow, G. J. (2018). RecurBot: Intharah, Thanapong, Michael Firman, and Gabriel J. Brostow. "RecurBot: Learn to Auto-complete GUI Tasks From Human Demonstrations. Presented at the 2018 CHI Conference on Human Factors in Computing Systems. ACM.

Intharah, T., Turmukhambetov, D., & Brostow, G. J. (2017). Help, It Looks Confusing: GUI Task Automation Through Demonstration and Follow-up Questions (pp. 233–243). ACM Press. https://doi.org/10.1145/3025171.3025176

Järvinen, P. (2007). Action Research is Similar to Design Science. *Quality & Quantity*, *41*(1), 37–54. https://doi.org/10.1007/s11135-005-5427-1

Kourousias, G., & Bonfiglio, S. (2010). Picture-driven Computing in Assistive Technology and Accessibility Design. Presented at the Access for All in the desktop, web and mobile field: an end-user and developer perspective, Spain.

Lau, T., Bergman, L., Castelli, V., & Oblinger, D. (2004). Sheepdog: learning procedures for technical support (p. 109). ACM Press. https://doi.org/10.1145/964442.964464

Lau, T., Cerruti, J., Manzato, G., Bengualid, M., Bigham, J. P., & Nichols, J. (2010). A conversational interface to web automation (p. 229). ACM Press. https://doi.org/10.1145/1866029.1866067

Lazar, J., Jones, A., & Shneiderman, B. (2006). Workplace user frustration with computers: an exploratory investigation of the causes and severity. *Behaviour & Information Technology*, *25*(3), 239–251. https://doi.org/10.1080/01449290500196963

Leshed, G., Haber, E. M., Matthews, T., & Lau, T. (2008). CoScripter: automating & sharing how-to knowledge in the enterprise (p. 1719). ACM Press. https://doi.org/10.1145/1357054.1357323

Li, I., Nichols, J., Lau, T., Drews, C., & Cypher, A. (2010). Here's what i did: sharing and reusing web activity with ActionShot (p. 723). ACM Press. https://doi.org/10.1145/1753326.1753432

Machado, V., Lopes, N., Silva, J. C., & Silva, J. L. (2017). Picture-Based Task Definition and Parameterization Support System. In *Recent Advances in Information Systems and Technologies* (pp. 592–601). Springer, Cham. https://doi.org/10.1007/978-3-319-56538-5_60

Martinie, C., Palanque, P., Barboni, E., & Ragosta, M. (2011). Task-model based assessment of automation levels: Application to space ground segments. In *2011 IEEE*

International Conference on Systems, Man, and Cybernetics (pp. 3267–3273). https://doi.org/10.1109/ICSMC.2011.6084173

Martinie, Célia, Palanque, P., Navarre, D., Winckler, M., & Poupart, E. (2011). Model-based Training: An Approach Supporting Operability of Critical Interactive Systems. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (pp. 53–62). New York, NY, USA: ACM. https://doi.org/10.1145/1996461.1996495

Matejka, J., Grossman, T., & Fitzmaurice, G. (2011). Ambient help (p. 2751). ACM Press. https://doi.org/10.1145/1978942.1979349

Navarre, D., Palanque, P., Ladry, J.-F., & Barboni, E. (2009). ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. *ACM Transactions on Computer-Human Interaction*, *16*(4), 1–56. https://doi.org/10.1145/1614390.1614393

Ornelas, J. D., Silva, J. C., & Silva, J. L. (2016). Demonstration-based Help for Interactive Systems. In *Proceedings of the 2Nd International Conference in HCI and UX Indonesia 2016* (pp. 125–128). New York, NY, USA: ACM. https://doi.org/10.1145/2898459.2898478

Paternò, F. (2004). ConcurTaskTrees: an engineered notation for task models. *The Handbook of Task Analysis for Human-Computer Interaction*, 483–503.

Paterno, F., Mancini, C., & Meniconi, S. (1997). ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In S. Howard, J. Hammond, & G. Lindgaard (Eds.), *Human-Computer Interaction INTERACT '97* (pp. 362–369). Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-35175-9_58

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Pongnumkul, S., Dontcheva, M., Li, W., Wang, J., Bourdev, L., Avidan, S., & Cohen, M. F. (2011). Pause-and-play: automatically linking screencast video tutorials with applications (p. 135). ACM Press. https://doi.org/10.1145/2047196.2047213

Rodrigues, P., Silva, J., & Pereira, R. (2018). DEMONSTRATION-BASED HELP: A CASE STUDY. In *10th annual International Conference on Education and New Learning Technologies* (p. 10). Palma de Mallocar. https://doi.org/10.21125/edulearn.2018.1096

Rosenbaum, S., & Ramey, J. A. (2014). Current issues in assessing and improving information usability (pp. 1119–1122). ACM Press. https://doi.org/10.1145/2559206.2559211

Schatsky, D., Muraskin, C., & Iyengar, K. (2016). Robotic process automation: A path to the cognitive enterprise. Retrieved 23 January 2018, from https://www2.deloitte.com/insights/us/en/focus/signals-for-strategists/cognitive-enterprise-robotic-process-automation.html

Shi, J. J., Lee, D.-E., & Kuruku, E. (2008). Task-based modeling method for construction business process modeling and automation. *Automation in Construction*, *17*(5), 633–640. https://doi.org/10.1016/j.autcon.2007.10.010

Shorkey, C., & Crocker, S. (1981). Frustration theory: a source of unifying concepts for generalist practice. *Social Work*. https://doi.org/10.1093/sw/26.5.374

Sikuli Script - Home. (n.d.). Retrieved 31 December 2017, from www.sikuli.org

Silva, J. L., Ornelas, J. D., & Silva, J. C. (2016). Make It ISI: Interactive Systems Integration Tool. In *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive*

Computing Systems (pp. 245–250). New York, NY, USA: ACM. https://doi.org/10.1145/2933242.2935872

Stohr, E., & Zhao, J. L. (2001). Workflow automation: Overview and research issues. In *Information Systems Frontiers* (Vol. 3(3), pp. 281–296). https://doi.org/10.1023/A:1011457324641

Sun, Y., Chen, D., Xin, C., & Jiao, W. (2015). Automating Repetitive Tasks on Web-Based IDEs via an Editable and Reusable Capture-Replay Technique (pp. 666–675). IEEE. https://doi.org/10.1109/COMPSAC.2015.12

Sun, Y., Qiao, Z., Chen, D., Xin, C., & Jiao, W. (2016). An Approach to Using Existing Online Education Tools to Support Practical Education on MOOCs (pp. 696–705). IEEE. https://doi.org/10.1109/COMPSAC.2016.49

UI Automation Overview. (2017, March 30). Retrieved 21 October 2018, from https://docs.microsoft.com/en-us/dotnet/framework/ui-automation/ui-automation-overview

UiPath Robotic Process Automation. (2005, 2018). Retrieved 30 December 2017, from www.uipath.com

Vogel-Heuser, B., Diedrich, C., Fay, A., Jeschke, S., Kowalewski, S., Wollschlaeger, M., & Göhner, P. (2014). Challenges for Software Engineering in Automation. *Journal of Software Engineering and Applications*, *07*(05), 440–451. https://doi.org/10.4236/jsea.2014.75041

Vouligny, L., & Robert, J.-M. (2005). Online help system design based on the situated action theory (pp. 64–75). ACM Press. https://doi.org/10.1145/1111360.1111367

Winter, J., Rönkkö, K., & Rissanen, M. (2014). Identifying organizational barriers—A case study of usability work when developing software in the automation industry. *Journal of Systems and Software, 88*, 54–73. https://doi.org/10.1016/j.jss.2013.09.019

Yeh, T., Chang, T.-H., & Miller, R. C. (2009). Sikuli: Using GUI Screenshots for Search and Automation. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology* (pp. 183–192). New York, NY, USA: ACM. https://doi.org/10.1145/1622176.1622213

Yeh, T., Chang, T.-H., Xie, B., Walsh, G., Watkins, I., Wongsuphasawat, K., … Bederson, B. B. (2011). Creating Contextual Help for GUIs Using Screenshots. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (pp. 145–154). New York, NY, USA: ACM. https://doi.org/10.1145/2047196.2047214

ATTACHMENTS

This page was intentionally left in blank

ATTACHMENT A – QUESTIONNAIRE FOR IDENTIFYING PLATFORMS AND TASKS

This questionnaire was constructed in Portuguese since targeted people were all Portuguese native. They were all from Technological company.

1. Género?
2. Idade?
3. Há quantos anos trabalha na empresa?
4. Em que área da empresa trabalha/já trabalhou?
5. Existe alguma plataforma que considere gerar mais dúvidas na sua utilização, por novos utilizadores?
   a. Se sim, qual/quais?
6. Existe alguma tarefa que considere que novos colaboradores tenham mais dificuldade em aprender?
   a. Se sim, qual/quais?
7. Imaginando uma aplicação género “welcome-kit”, que plataformas considera que deveriam estar contempladas?
8. Imaginando uma aplicação género “welcome-kit”, que tarefas considera que deveriam estar contempladas?
9. Existe alguma plataforma que gere mais dúvidas/erros aquando da realização de tarefas periódicas?
   a. Se sim, qual/quais?
10. Existe alguma tarefa particular que gere mais dúvidas/error aquando da sua execução?
    a. Se sim, qual/quais?
11. Na sua opinião, existe alguma plataforma em particular cujo sistema de ajuda seja insuficiente?
    a. Se sim, qual/quais?
12. Na sua opinião, existe alguma tarefa em particular que pudesse beneficiar de uma aplicação de ajuda através de automação e computação por image?
    a. Se sim, qual/quais?

This page was intentionally left in blank

ATTACHMENT B – DEVELOPED HELP SCRIPTS

Task 1 – Standard hour registry:

```
popup("The help-script will start now. Please do not use the mouse and keyboard except when asked.")
m= find("1527855123896.png")
a=Region(m).nearby(10)
a.highlight(1)
click("1527089674439.png")
wait("2018-05-23 17_03_16-.png",30)
type(Key.DOWN)
type(Key.ENTER)
click("1527091941931.png")
paste("40")
m= find("1527092629998.png")
a=Region(m).nearby(20)
m= find("1527092613074.png")
b=Region(m).nearby(20)
a.highlight()
b.highlight()
popup("Please search and select your customer and service. At the end press submit")
a.highlight(1)
b.highlight(1)
```

Task 2 – Absence hours registry:

```
popup("The help-script will start now. Please do not use the mouse and keyboard except when asked.")
find("1527089663327.png")
m= find("1527855123896.png")
a=Region(m).nearby(10)
a.highlight(1)
click("1527089674439.png")
wait("2018-05-23 17_03_16-.png",30)
type(Key.DOWN)
type(Key.ENTER)
click("1528053880953.png")
type("GS IT TRA PT")
type(Key.ENTER)
wait(1)
click("1528053972963.png")
type("A")
type(Key.ENTER)
wait(1)
m= find("1528054358833.png")
a=Region(m).nearby(10)
m= find("1528054501945.png")
b=Region(m).nearby(10)
a.highlight()
b.highlight()
```

```
popup("Under 'Hours' type the amount of absence hours you want to register.\nIf you want to register more hours select the option highlighted.")
a.highlight(1)
b.highlight(1)
```

Task 3 – Filling future hours:

```
popup("The help-script will start now. Please do not use the mouse and keyboard except when asked.")
wait("1528055762888.png",30)
click("1528055781384.png")
wait(1)
popup("Select any day of the week you want to fill hours in.")
wait("1528055830347.png",120)
wait(2)
popup("The script will continue, please do not use the mouse and keyboard excel if asked.")
click("1527089674439.png")
wait("2018-05-23 17_03_16-.png",30)
type(Key.DOWN)
type(Key.ENTER)
click("1527091941931.png")
paste("40")
m= find("1527092629998.png")
a=Region(m).nearby(20)
m= find("1527092613074.png")
b=Region(m).nearby(20)
```

```
a.highlight()
b.highlight()
popup("Please search and select your customer and service. At the end press submit.")
a.highlight()
b.highlight()
```

Task 4 – Filling previous hours:

```
popup("The help-script will start now. Please do not use the mouse and keyboard except when asked.")
#click("1528819329142.png")
find("1527089663327.png")
m= find("1527855123896.png")
a=Region(m).nearby(10)
a.highlight(1)
click("1527089674439.png")
wait("2018-05-23 17_03_16-.png",30)
type(Key.DOWN)
type(Key.ENTER)
click("1528403602711.png")
click("1528403569495.png")
wait("1528459752471.png",60)
find("1528459752471.png")
click("1528459752471.png")
wait("2018-05-23 17_03_16-.png",30)
type(Key.DOWN)
type(Key.ENTER)
m= find("1528403786656.png")
a=Region(m).nearby(1)
a.highlight()
popup("Insert the amount of hours from the last period on the highlighted box.\nThen, select your normal customer and service. At the end press and 'SUBMIT' and then 'SUBMIT EITHER WAY'")
a.highlight()
```

Task 5 – Checking hours filled:

```
popup("The help-script will start now. Please do not use the mouse and keyboard except when asked.")
#click("1528819329142.png")
wait("1527859953126.png",20)
m= find("1527859953126.png")
a=Region(m).nearby(10)
a.highlight(1)
click("1527859953126.png")
wait("1527860066678.png",60)
m= find("1527860369686.png")
a=Region(m).nearby(10)
a.highlight(1)
click("1527860369686.png")
wait("1527860432763.png",20)
click(("1527860432763.png")
wait("1527860135030.png",30)
click("1527860135030.png")
click("1527861074382.png")
wait("1527860981601.png")
click("1527860996705.png")
wait("1527861156427.png",30)
click("1527861156427.png")
wait("1527861652580.png",90)
#click("1528055025157.png")
wait(1)
#click("1527861669127.png")
type("1527861669127.png", "K1")
type(Key.ENTER)
wait(1)
click("1527860109418.png")
m = find("1527862143024.png")
a=Region(m).nearby(10)
m = find("1527861982952.png")
b=Region(m).nearby(10)
a.highlight()
b.highlight()
popup("Column H and column K give you information of the hours reported. The script
will end now.")
a.highlight()
b.highlight()
```

ATTACHMENT C – RESULTS FROM THE USABILITY STUDY (USE QUESTIONNAIRE)

Usefulness questions:

1. It helps me be more effective.
2. It helps me be more productive.
3. It is useful.
4. It gives me more control over the activities in my life.
5. It makes the things I want to accomplish easier to get done.
6. It saves me time when I use it.
7. It meets my needs.
8. It does everything I would expect it to do.

Table 7 - Classification given to the developed tool regarding usefulness.

Question	**#**	**Very Strongly Disagree**	**Strongly Disagree**	**Disagree**	**Neither Agree nor Disagree**	**Agree**	**Strongly Agree**	**Very Strongly Agree**	**N/A**	**Total**
1	16	0%	0%	0%	6%	31%	31%	31%	0%	**100%**
2	16	0%	0%	6%	19%	25%	25%	25%	0%	**100%**
3	16	0%	0%	6%	6%	6%	31%	50%	0%	**100%**
4	16	0%	19%	12%	6%	19%	25%	12%	6%	**100%**
5	16	0%	6%	6%	6%	12%	38%	31%	0%	**100%**
6	16	6%	0%	19%	6%	19%	19%	31%	0%	**100%**
7	16	0%	6%	12%	12%	12%	12%	44%	0%	**100%**

8	16	0%	12%	19%	0%	12%	25%	25%	6%	**100%**
Usefulness (AVG)	16	1%	5%	10%	8%	17%	26%	31%	2%	**100%**

Ease of use questions:

9. It is easy to use.
10. It is simple to use.
11. It is user friendly.
12. It requires the fewest steps possible to accomplish what I want to do with it.
13. It is flexible.
14. Using it is effortless.
15. I can use it without written instructions.
16. I don't notice any inconsistencies as I use it.
17. Both occasional and regular users would like it.
18. I can recover from mistakes quickly and easily.
19. I can use it successfully every time.

Table 8 - Classification given to the developed tool regarding ease of use.

Question	**#**	**Very Strongly Disagree**	**Strongly Disagree**	**Disagree**	**Neither Agree nor Disagree**	**Agree**	**Strongly Agree**	**Very Strongly Agree**	**N/A**	**Total**
9	16	0%	0%	0%	12%	25%	25%	38%	0%	**100%**
10	16	0%	0%	0%	6%	25%	25%	44%	0%	**100%**
11	16	0%	6%	0%	19%	31%	25%	19%	0%	**100%**
12	16	0%	6%	12%	0%	6%	31%	44%	0%	**100%**
13	16	0%	0%	25%	0%	25%	25%	19%	6%	**100%**
14	16	0%	0%	6%	12%	31%	12%	38%	0%	**100%**
15	16	0%	6%	0%	31%	12%	25%	25%	0%	**100%**

16	16	0%	6%	12%	6%	12%	19%	44%	0%	**100%**
17	16	0%	6%	6%	19%	12%	31%	19%	6%	**100%**
18	16	6%	0%	6%	0%	0%	44%	19%	25%	**100%**
19	16	0%	6%	6%	12%	6%	38%	25%	6%	**100%**
Ease of Use	16	1%	3%	7%	11%	17%	27%	30%	4%	**100%**

Ease of learning questions:

20. I learned to use it quickly.
21. I easily remember how to use it.
22. It is easy to learn to use it.
23. I quickly became skillful with it.

Table 9 - Classification given to the developed tool regarding ease of learning.

Question	#	Very Strongly Disagree	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree	Very Strongly Agree	N/A	Total
20	16	0%	6%	12%	0%	6%	31%	44%	0%	**99%**
21	16	0%	0%	0%	6%	6%	25%	62%	0%	**99%**
22	16	0%	0%	0%	6%	12%	38%	44%	0%	**100%**
23	16	0%	0%	0%	19%	0%	19%	62%	0%	**100%**
Ease Of Learning	16	0%	2%	3%	8%	6%	28%	53%	0%	**100%**

Satisfaction questions:

24. I am satisfied with it.
25. I would recommend it to a friend.
26. It is fun to use.
27. It works the way I want it to work.
28. It is wonderful.
29. I feel I need to have it.
30. It is pleasant to use.

Table 10 - Classification given to the developed tool regarding satisfaction.

Question	**#**	**Very Strongly Disagree**	**Strongly Disagree**	**Disagree**	**Neither Agree nor Disagree**	**Agree**	**Strongly Agree**	**Very Strongly Agree**	**N/A**	**Total**
24	16	0%	0%	12%	12%	19%	31%	25%	0%	**100%**
25	16	0%	0%	12%	6%	19%	31%	31%	0%	**100%**
26	16	0%	0%	12%	25%	0%	31%	25%	6%	**100%**
27	16	6%	0%	6%	19%	12%	25%	31%	0%	**100%**
28	16	0%	12%	6%	19%	19%	25%	6%	12%	**100%**
29	16	0%	19%	12%	0%	12%	38%	12%	6%	**100%**
30	16	0%	6%	0%	19%	0%	50%	19%	6%	**100%**
Satisfaction	16	1%	5%	9%	14%	12%	33%	21%	4%	**100%**