Lappeenranta University of Technology
School of Industrial Engineering and Management
Degree Program in Computer Science

Manuel Delgado Sánchez

AUTOMATED UI TESTING. WBCT, A COMPUTER-AUTOMATED USER TEST ENVIRONMENT TOOL

Supervisors: Professor Ahmed Seffah
ABSTRACT

Lappeenranta University of Technology
School of Industrial Engineering and Management
Degree Program in Computer Science

Manuel Delgado Sánchez

Automated UI testing. WBCT, a computer-automated user test environment tool

Master’s Thesis

61 pages, 38 figures, 1 table, 1 appendix

Examiners: Professor Ahmed Seffah

Keywords: user interface, user experience, usability, software testing, testing methods

Today, the user experience and usability in software application are becoming a major design issue due to the adaptation of many processes using new technologies. Therefore, the study of the user experience and usability might be included in every software development project and, thus, they should be tested to get traceable results. As a result of different testing methods to evaluate the concepts, a non-expert on the topic might have doubts on which option he/she should opt for and how to interpret the outcomes of the process. This work aims to create a process to ease the whole testing methodology based on the process created by Seffah et al. and a supporting software tool to follow the procedure of these testing methods for the user experience and usability.
ACKNOWLEDGEMENTS

I would first like to thank to my supervisors Professor Ahmed Seffah of the School of Business and Management at Lappeenranta University of Technology.

I would also like to thank the rest of professors who were involved in teaching on the Master in Computer Science at LUT.

Finally, I am very grateful to my parents Manuel and Agueda, my girlfriend Anastasia Hru and the rest of my family and friends, Oscar, Felipe, Pili, Angela, Mico Chico, Enano, Yuri, Sofia, Lolo and Mansoureh for the support and continuous encouragement during my years of study.
# TABLE OF CONTENTS

1 **INTRODUCTION** ........................................................................................................... 6

1.1 **BACKGROUND** .......................................................................................................... 6
1.2 **RESEARCH PROBLEM, OBJECTIVES AND DELIMITATIONS** ............................ 7
1.3 **RESEARCH METHODOLOGY** ..................................................................................... 8
1.4 **STRUCTURE OF THE THESIS** .................................................................................. 10

2 **USABILITY AND USER RESEARCH: SOME DEFINITIONS, PERCEPTIONS AND QUALITY ATTRIBUTES** ........................................... 11

2.1 **SOFTWARE TESTING** .............................................................................................. 12
2.2 **USER INTERFACE** .................................................................................................... 14
2.3 **TESTING USABILITY AND USER INTERFACE** ........................................................ 17

3 **TAXONOMY OF USABILITY AUTOMATED TESTING METHODS** ............. 24

4 **COMPUTER-AUTOMATED USER TEST ENVIRONMENT TOOLS** ............ 34

4.1 **WHAT IS CAUTE?** .................................................................................................. 34
4.2 **WEB CAUTE-TOOL AND PROCESSES** ................................................................ 37

**SUMMARY** ..................................................................................................................... 57

**REFERENCES** .................................................................................................................. 58
**LIST OF SYMBOLS AND ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPMN</td>
<td>Business process model and notation</td>
</tr>
<tr>
<td>CASE</td>
<td>Computer-aided software engineering</td>
</tr>
<tr>
<td>CAUTE</td>
<td>Computer-automated user test environment</td>
</tr>
<tr>
<td>CIF</td>
<td>Common industry format</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical user interface</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of electrical and electronics engineers</td>
</tr>
<tr>
<td>ISO</td>
<td>International organization for standardization</td>
</tr>
<tr>
<td>LUCID</td>
<td>Logical User-Centered Interaction Design</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Matrix Laboratory</td>
</tr>
<tr>
<td>NUI</td>
<td>Natural user interface</td>
</tr>
<tr>
<td>SUS</td>
<td>System usability scale</td>
</tr>
<tr>
<td>TDD</td>
<td>Test-driven development</td>
</tr>
<tr>
<td>UAT</td>
<td>User acceptance test</td>
</tr>
<tr>
<td>UI</td>
<td>User interface</td>
</tr>
<tr>
<td>UX</td>
<td>User experience</td>
</tr>
<tr>
<td>WBCT</td>
<td>Web based CAUTE tool</td>
</tr>
<tr>
<td>WIMP</td>
<td>Windows, icons, mouse and pointer</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 Background

Today, the user experience and usability in software applications are becoming a trending due to the adaptation of many processes using new technologies. Amid the advantages of usable human interfaces, one may specify security, business viability and human efficiency, and execution (Abran et al. 2004). Therefore, the studying of the user experience and usability might be included in every software development project and consequently they should be tested in order to get traceable results.

Unfortunately, most software developers do no use or apply correctly any particular model for testing the user experience. This might be caused because there exist several non-unified methods and a lack of proper guidance on how to model and integrate the user-oriented tests in the software engineering lifecycle.

System Testing is well defined with documented automated methods and tools whereby a software developer can analyze a system (Fisher 1991), therefore, it might result useful to create and introduce a consolidated user-testing method and a tool that would help on these tasks.
1.2 Research problem, objectives and delimitations

Software applications are present in our everyday life and these applications are used by different people with different profiles, ages, cultures, skills or disabilities. It is a challenge for software companies to develop applications that fulfill all the user requirements related to usability and provide a good user experience while using the software.

This work is aimed to help non-expert users in selecting the appropriate technique to test and provide a tool to assist them during the whole testing process and build testing outcomes in an understandable manner.

As a result of different testing methods to evaluate the concepts, a non-expert on the topic might have doubts on which option he/she should opt for and how to interpret the outcomes of the process. This study aims to create a process to ease the selection of the best suitable method for this purpose. A supporting software tool will be used as a case study on how to follow the procedure of these testing methods for the user experience and usability.

This work aims to answer on the following research questions:

- What are usability and user interface (UI) and how are they tested?
- What are the benefits of using automated user interface testing?
- Which processes should be followed to test the usability and UI of one application?
1.3 Research methodology

This research is mainly based on exploratory research. The first chapter provides a conceptual background about usability, user interface and the different approach on how to test them. The second chapter aims to clarify the classification of the usability testing methods and explains the benefits of using automated testing methods. The last chapter introduces an innovative framework that helps to automate the testing process.

This research follows the five step guidelines for design science research proposed by Vaishnavi and Kuechler (2004):

1. **Awareness of the problem**: a lack of proper tools helping during usability and UI testing process was noticed. The research problem and the research questions stated on the previous section are thus proposed.
2. **Suggestion**: theoretical information was collected in order to create a basic understanding about the related relevant topics. The usage of an innovative framework assisting on the usability and user interface testing process is suggested.
3. **Development**: a new tool that uses the previously explained framework in order to clarify the procedure is proposed.
4. **Evaluation**: the software created on the previous step is evaluated.
5. **Conclusion**: exposition of the benefits of using the proposed framework and tools.

Figure 1 sketches the design science research process that has been followed to proceed with this study and the generated outcome in the conclusion of each step.
Figure 1. Design Science Research Process Model (Vaishnavi & Kuechler 2004)
1.4 Structure of the thesis

The first chapter contains some core concepts that need to be understood in the human-computer interaction field. Concepts as usability, user experience and user research, what software testing is, what a UI is and different approaches on how to test and measure the previous concepts are explained.

The second chapter of the thesis details the importance of using automated tools and processes in order to ease, improve and focus the efforts dedicated on testing a UI within a software project including taxonomy for categorizing the testing methods.

The last chapter introduces an innovative framework that explains how to perform a specific UI testing procedure, Computer-Automated User Test Environment (CAUTE). In addition, this chapter contains an example of a CAUTE-tool and how it may help a software development team in order to find problems and issues within the UI.
2 USABILITY AND USER RESEARCH: SOME DEFINITIONS, PERCEPTIONS AND QUALITY ATTRIBUTES

The goal of this thesis is to determine how to select the appropriate automated testing method for a UI among the several existing methods. This chapter provides an overview of the core concepts about usability and user research. Some general concepts about software testing, UIs and testing UIs are also included.

Usability is a prevailing topic which has typically been defined in distinct manners that may lead the developer to confusion. Some of the definitions for usability in different standards are listed below:

- "The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions" (ISO/IEC 9126, 1991).
- "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO 9241-11, 1998).
- "The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component" (IEEE Std.610.12-1990).

Regardless of the popularity of user experience (UX), the definition of the term has been deliberated in order to create an agreement to scope the UX. It is defined by the ISO FDIS 9241-210 (2010) as "A person's perceptions and responses that result from the use and/or anticipated use of a product, system or service.".

In order to handle with the previous terms, the developers and businesses might collect information of users' behaviors, motivations and needs. This is called user research. The user research is an expansive term that includes different techniques which create quantifiable outcomes; including questionnaires, surveys and usability testing through observations, feedback procedures or task analysis. It is defined by Rosenzweig (2010) as "The systematic study of the goals, needs, and capabilities of users so as to specify design, construction, or improvement of tools to benefit how users work and live".
The developing of applications that offer excellent usability and user experience may result somewhat challenging without the existence of standard methods that help to gather the data provided by the users and, subsequently, to analyze the data and propose possible solutions for the given problems. Before entering into further detail on how to select the methods, is necessary to underline some terms on which this work is based; the software testing and the UI.

2.1 Software testing

The purpose of software testing consists of finding, reporting and fixing existing bugs in software. A bug implies any type of issue with the system that might conduce to a failure that could influence on the user’s experience. The existence of bugs is unavoidable, even in simple functions that have already been evaluated using different tests (Dijkstra 1970). This implies that once the tests are performed proving the existence of bugs, the effort of the developers might be focused on fixing them. In others words, the software testing can be regarded as a manner with which the developers might assure that the system fulfills the specified requirements with bug-free instances.

A bug can be triggered when a set of actions are performed. At first, one error needs to be made due to human actions, such as typos in the code made by the developers or when the requirements have not been well defined by the clients. This leads to a defect, which consists of a change on the expected behavior of the system. If the defect is severe, the system provokes a fault, a state where it should not have been able to enter. This fault can bring a failure, a behavior that causes a divergence between the expected behavior and the current one or contravenes the user’s expectation.

One important aspect of software testing is that typically not all the bugs can be fixed and not all the bugs can be found. This idea leads to the dilemma of which bugs have to be repaired, so that the developers focus only on those ones that have an important and relevant influence on the user experience. This is important due to the time constraint that is usually present in most development process.
Two separate parts in a test could be distinguished; the test procedure and the test oracle (Barr et al. 2015). The test procedure contains detailed instructions on how to perform the test for a given case, including step-by-step guidance if necessary. On the other hand, the test oracle is a set of desired behaviors that the system should fulfill to pass it. The former can be executed in order to detect possible bugs but, however, the latter is indispensable to identify the bugs through the discrepancies between the current results of the test and the expected ones.

The automation of these processes may reduce the effort required for running the tests. The reason for this is that once the tests are written and set, they are frequently performed and rarely maintained; while, on the other hand, using manual testing would require an increased human effort on the exploratory testing, test design and testing execution while trying to find and force possible and potential system failures.

Besides finding bugs, automated testing may bring other benefits. The automated testing may clarify the specifications in a test-first environment (test as specification). For instance, in the user acceptance test, the developing team may consider a use case as complete when the tests pass. Other benefit resides on the fact that the automated tests may help to understand the performance of the software; this is the compliance of the requirements (test as documentation). Finally, it may find bugs in complex system by reproducing the origin of the bug (defect localization). It is noted that the usage of automated software testing helps to improve the quality of the software and eases the entire process of fulfilling the software requirements (Meszaros 2004).

Tests can be designed from the perspective of the developers or the one of the users. The developers create unitary and integration tests to verify that a piece of code, class or method works as expected in an isolated context, and that a group of classes and methods effectively work together to fulfill a specific target respectively. These types of tests may assure or increase the confidence about the exactitude of the code but they do not manifest the compliance of the user’s requirements and needs. For example, a tester may use a characteristic of the application using shortcuts or elements that are not available in the UI. To solve the problems related to the UI, testers may create user oriented tests that interact
straightforward with the software’s GUI and perform typical actions that a user might do in order to certify that it complies with the user’s requirement.

Other approach for software testing is the test-driven development (TDD). It consists of determining the tests before the target code is completed. In TDD, tests are contrived by developers and testers based on their assumptions before any code is finalized. Through TDD, the developers obtain a different point of view of the application and may identify possible issues that might appear during the development process and, therefore, evade them on its first stages (Astels 2003).

2.2 User interface

In modern software, UIs play an important role for users, they englobe the experience that the user is receiving and they are usually the only meeting point between the application and the systems.

Basically, a UI is the part of the application that is responsible for receiving the commands and data from the user and displays the results through any type of peripheral. If any of the features in not accessible from the UI, they might be invisible for the user and could never be used.

One type of UI is the Command-line interface (CLIs). It allows the user to type text commands through simple and successive lines of text and receive an output through a monitor (Mann 2002). Although, it might seem fairly simple, one writes some lines of text and the computer displays the result, the user has to learn the possibly limited and strict commands prior to using the application and entails a relatively high learning curve. Even CLIs bring certain difficulties they are broadly used in engineering applications such as Matrix Laboratory (MATLAB) or AutoCAD and play an important role in some desktop interfaces as GNOME or KDE.

Figure 2 exposes two of the common problems of the CLI; firstly, user must know the accepted commands before using the interface, even for performing simple actions as
moving a file or a folder, and secondly the help and results are poor (using the current standards) and might result difficult for the user.

Other type of UI is the graphical user interfaces (GUIs). GUIs offer improve esthetics, a simplified UI but often requiring greater computational resources and creating new vulnerabilities due to its complexity. GUIs are usually more user-friendly since the graphical environment could be designed to be intuitive for the user, even avoiding sometimes the need for previous knowledge (Mann 2002).
A GUI consists on a set of graphical elements with which a user can introduce and receive information to and from the system using different peripherals such as a keyboard, a mouse or a touch display. GUIs are typically grouped in windows, which eases the interaction.

Figure 3 exposes a common case of overlapping windows with several graphical elements grouped by functionality that improve and ease user experience.

![Overlapping GUIs in Windows 7.](image)

Figure 3. Overlapping GUIs in Windows 7.

The newest type of UI is the natural user interface (NUI) in which the user may interact with the software application with natural and human gestures (which will be processed for the application by specialized peripherals) instead of addressing specific commands and input directly to the application (Câmara 2011). This type of UI eases the process of learning since the system is in charge of understanding the user in contrast to the CLI and GUI where the user must know the predefined codes or explore the application. NUIs are used in several operating systems such as Android or iOS and devices as Xbox Kinect or Google Glass.
Figure 4 shows the NUI interface used by Xbox Kinect. One may notice that the users do not handle any kind of peripheral; instead, they use the gestures of their bodies to perform actions that the system will translate to inputs to the application and exhibit the outputs on the TV display.

![Figure 4. Users playing using NUI with Xbox Kinect through body gestures (Sergey Galionkon 2010).](image-url)

To summarize this section, there currently exist three types of UI (CLI, GUI and NUI), each one with its advantages and disadvantages. CLI offers a set of strict commands on a simple and secure UI that may result somewhat complex for the user. GUI offers a complex and exploratory UI, easing self-learning at the expense of having more complex code. Finally, the NUI offers an interface that is invisible to the user as a result of using body gestures, eye gestures or voice commands to interact with the system.

### 2.3 Testing usability and user interface

Although the UI is one of the most important points on the user experience; it is not the only matter that may affect it. Malfunctions of the logic of the application and other technical problems may alter the usability of the entire system. Some examples of these problems can be interface errors, incorrect data saved or retrieved from a database, and functionalities that were not added or do not correctly perform their tasks.
In order to avoid such issues on the production versions, software developers conduct tests to determine whether the customers are satisfied with the products or they face problems on using the software being the user acceptance tests the most used method among the software companies (Hambling & van Goethem 2013).

The selection of the appropriate method for testing one application may result a major headache, although all the processes essentially underlines several common activities that need to be performed during the testing process and that can be applied to the user testing process.

A generic case of usability testing process, such as the one sketched in Figure 5, starts with the planning of the whole evaluation. Afterwards the evaluators establish the scenarios and tasks that will be analyzed. The participants that will take part into the procedure are recruited. Thereupon, the evaluators conduct the usability test and analyze the generated data in order to create an evaluation report, including recommendations for further testing processes.

![Figure 5. Generic iterative usability testing process.]

The notation used to demonstrate the models and processes studied is the Business Processes Modeling Notation (BPMN), a standard graphical notation broadly accepted by a large industry consortium. The advantage is that BPMN notation brings consist of a simple manner for representing a complete flow of a process and respond to several questions such as the required information for the testing, the life cycle. Also, it is stated and when to
perform the key steps, how the activities need to be performed and the actor required in the process (BPMN, 2016).

User acceptance tests (UAT) focus on the final user feelings and thoughts of the performance of a system as a whole, or a large part of it, including functionality, quality requirements and usability issues. An acceptance test consists of a set of typical actions that a final user has to perform and the results obtained through all the process. UATs are conducted during the final stages of the development either with the final version or a beta. Considering one scenario, once all acceptance tests are approved, the scenario may be considered as complete and ready for production.

For better results, the usability acceptance test should be conducted in laboratories particularly designer for this purpose. This type of laboratories is named as “usability labs” (Albert et al. 2010). These studies can be performed observing the users in their usual work habitat or using Internet for collecting the data remotely (Tullis & Fleischman 2002).

Usability acceptance tests require a scale to identify and classify the user issues. Several researchers have defined a system to rate and classify the problems; Sauro (2011), Molich (2007), Wilson (2001), Dumas and Redish (1999), Rubin (2008) or Nielsen (1994).

One of the most broadly used methods to measure the usability and usability acceptance test is the System Usability Scale (SUS) designed by Brooke (1986) and redefined by Sauro (2011). It consists of ten questions that the user must reply considering five possible answers.

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

The response format for the ten questions usually follows the next structure and the user must select a single option of the five ones proposed by the method as shown in Figure 6:

<table>
<thead>
<tr>
<th>Strongly disagree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

Figure 6. Response format for System Usability Scale (SUS)

Figure 7 explains the process for a typical user acceptance test. It starts with the collection and analysis of the requirements specified by the client and the recruitment and training of the people who will have some role into the process. The second step consists of planning and designing the test: what needs to be tested, when needs to be tested and who will perform the test. Once the plan is fixed, the test is implemented and the results are analyzed and reported to the client that will expound the user’s point of view. Next step needs to be performed based on the client’s feedback; the client or the testers may notice some failures during the testing process so it might need to be repeated, some bugs on the code were found and they need to be corrected, the use case may require adjustments and the UAT process repeated or the client approve the use case and it is incorporated to the software.
Another method for testing UI consists of performing controlled experiments in human sciences and psychology, and applying scientific knowledge and studies in order to analyze and understand human behavior towards the software. Kerlinger and Lee (2000), Rubin (2008) and Kohavi (2009) indicate that controlled experiments are one of the most rigorous and extensively studies conducted to comprehend human comportment.

The main advantage of using controlled experiments over the usability testing resides in the possibility of assessing repeatable hypothesis in lieu of looking for single failures. Other advantage that supports the usage of controlled experiments is the possibility of predicting system behaviors and user performance based on psychological theories.
Kerlinger and Lee (2000) advocate that the controlled experiments follow the process MaxMinCon, which is the acronym for maximizing systematic variance between the treatment groups, minimizing the variability error within each treatment group, and Control extraneous systematic variances. This method implies that the studied groups must be different, the control and experiment groups ought not to be very distinct and the utilized measurements must be high-principled.

Figure 8 demonstrates a typical controlled experimental process valid for any science field customized for testing. It consists of five stages: definition of the problem, planning of the testing process, research conduct, analysis of the data, and interpretation of the results.

In conclusion, as part of the software, a UI needs to be tested but it may not result as simple as the unitary testing or the integration testing. In order to examine the UI, the simplest manner consists of manual work where a person, professional tester or user, performs some actions that have been previously defined in the documentation and controls if the output of the software meets the expectations. Manual testing brings some advantages over the automated processes when bugs appear, but humans are slower than
machines performing tests. If the tests need to be repeated numerous times, it might turn to an unviable process.

The next chapters use the concepts described above in order to create a better understanding of existing taxonomies and, in addition, show the benefits that automating user interface tools may bring to the development process and why it should always be considered as part of the software lifecycle.
3 TAXONOMY OF USABILITY AUTOMATED TESTING METHODS

This chapter describes the usability testing and the importance of using automatic tools and processes in order to ease, improve and focus the efforts dedicated for testing an UI within a software project. It also includes the taxonomy of methods, which has been already studied by several scientists.

There exist diverse approaches for assorting UI testing, the principal types that are performed during the development lifecycle of one application being:

- **Exploratory testing** during the early stages of the development lifecycle that may help establishing the validity of user requirements and high-level design before the development of functional prototypes (Abran et al. 2004; Itkonen & Rautiainen 2005).
- **Assessments test** are performed with tools that capture information on the early stages of the development in order to gauge the usability of lower-level operating and specific aspects of an application, focusing on the ergonomic properties of the UI (Rubin, J., & Chisnell 2008; Charfi et al. 2014).
- **Validation and verification tests** performed at the last stages of the development cycle used to certify the usability of the product by means of measures of effectiveness, efficiency and satisfaction (Tran et al. 2013; Nielsen 1994).
- **Comparative test** that can be collectively performed with the exploratory testing, assessments testing and validation testing. The comparative tests may be used to contrast two or more aspects of an application such as design element together with a functional element. This type of tests are conducted to establish the advantages of choosing a certain design over others evaluating the possible accessibility, acceptability and the satisfaction of the target users, and to ascertain the best design for easing the use of the software (Ivory 2000).

Usability testing is a long process that includes many tasks subject to the employed method. Most of these method include activities such gathering usability data, for instance errors, subjective ratings or activity completion time; analysis deciphering the data to recognize
usability issues in the interface; and critique, proposing solutions and enhancement to alleviate problems (Nielsen 2003).

The taxonomy of methods for usability and user testing has been already studied and certain subsets of the proposed techniques are prevalent used within the projects (Ivory & Hearst 2001).

Different analysts using the same usability method might collect distinct findings studying the same UI. This was stated by previous (Molich et al. 1999) where seven professional usability labs and one university student team carried out the usability testing of one website using the same technique. None of the 310 detected problems was reported by the eight teams. This outcome suggests an absence of consistency in the discoveries of usability evaluation. Besides, usability evaluation just covers a subset of actions that users might perform. Consequently, usability experts recommend using diverse evaluation techniques (Nielsen 2003).

How to reach a complete usability evaluation with methodical results and conclusions? One solution consists of increasing the amount of people involved on the project including professional testers and real users using non-automated usability testing methods. The other option includes the automation of some processes of the usability evaluation such as gathering information, analyzing the data and providing an appraisal of the activities.

Nonautomated usability testing typically involves testers that gather data while the users are performing previously defined tasks. Once a task is done, the tester evaluates how the interface fulfill the users' needs, user's task completion and other parameters such as time for completing the tasks, errors and difficulty of the process.

Automated testing within a project ordinarily comprises automated gathering of the data and its automatized evaluation in relation to some metrics or a model being not common the case of methods that includes both of them, gathering and evaluation of data (Ivory & Hearst 2001).
The automation of usability testing brings diverse advantages over the manual testing such as the following:

- Reducing the expenses of usability testing. Methods that automate the task of gathering data, analysis and provide an appraisal of the activities can decrease the time spent on the testing and therefore the overall expenses.
- Evaluation can be held during the design phase and not right after its completion. Rather than in non-automated processes, where the evaluation is performed once the interface is completed, using modeling and simulation tools brings the possibility of auditing the UI in early stages which provides the chance to detect errors and save costs.
- It is not always possible to evaluate each and every facet of an interface through nonautomated processes. Thus, while automated evaluation, time and error costs over a whole design may be predicted.
- Decreasing the necessity of experts and the quantity of necessary expertise among the participants in the project. Automating some processes might help the team and the individuals on the areas they are not experts.
- More areas of the interface can be evaluated. With nonautomated evaluation, it is not always possible to cover each and every single facet of an interface. Using software tools that performs traceable results may help the designer on the usability evaluation and may increase the facets that can be audited.
- Automated testing brings the possibility to evaluate different designs and increase the subset of audited features that otherwise could not be assessed due to lack of time, resources and cost.

In spite of human-computer interaction experts and empirical software engineers concurring on the relevance of automating user-oriented testing processes, a few methods and tools have been developed for this automation. Ivory and Hearst (2001) proposed a taxonomy to discuss the automation in usability evaluation using four characteristics; method class, method type, automation type and effort level.
**Method class** refers to the type of evaluation that is being performed. Five different method classes are considered: *testing methods*, where an expert monitors how the users interact with the interface and verify usability problems; *inspection methods* where an expert recognizes possible usability problems following a predefined criteria; *inquiry methods* where the data is collected via interviews, questionnaires or self-reporting logs; *analytical modeling methods* where an expert uses models to create possible forecast; and *simulation methods* where a reviewer creates a framework to imitate user interactions using an interface and describe the outcome of the study.

**Method type** includes an ample variety of procedures for testing, inspections, inquiries, analytical modeling and simulation classes. For the classification, the related methods are placed in categories which describe the typically performed mechanism and actions during its usage.

**Automation type** refers to the facets that can be automated; gathering data, analysis or critique. Balbo (1995) proposed a taxonomy with four approaches for the automation that considers what can be automated within the different activities: *no automation*, where no level of automation is supported so all the activities are performed by experts and testers such as in question-asking protocol or interviews; *capture automation*: where the software systematically collects the data referent to usability such as in remote testing; *analysis automation*: where the software automatically establish possible usability failures such as in log-files analysis; and *critique automation*: where the software analyzes and suggest possible improvements to avoid usability problems such as the guideline review.

At last, broadening Balbo’s automation taxonomy, the **effort level** comprises the necessary human interaction for performing a concrete method. This classification encompasses *minimal effort*, meaning that any interface or model is required to perform the test; *model development*, which necessitates a model developed by an evaluator; *informal use*, that requires a set of random tasks; and *formal use* which includes a sort of selected tasks designated by an evaluator or user.
Figure 9 stands the taxonomy developed by Ivory and Hearst remarking that every method belongs to a method class (testing, inspection, inquiry, analytical modeling or simulation), method type (such as interviews, surveys or teaching method), automation type (no automation, capture automation, analysis automation and critique automation) and to a effort level (minimal effort, model development, informal use or formal use).

![Usability Evaluation Method](image)

Figure 9. Summary of the taxonomy for usability testing methods proposed by Ivory and Hearst (2001).

As an example of the usage of this taxonomy, Figure 10 clarifies the classification for the VISVIP method (Cugini & Scholtz 1999) that can be positioned into the Testing method class, Log file Analysis method type, Analysis automation type and it could be used in informal and formal uses.
Ivory and Hearst surveyed 75 WIMP (windows, icons, mouse and pointer) methods and 57 methods than can be used for Web UI testing in order to determine whether they can be totally, partially or none automated. Table 1 demonstrates which of these methods offer any type of automation. In the first column, the model class is shown with grey shadow and the method types that belong to the concrete model class are displayed under each class. In columns 2-5, the level of automation of each method type is detailed. The letters refers to the effort level in constructing an interface or model that is required for performing the test being M for model development, I for informal use, F for formal use and blank space when minimal effort is required. The number between parentheses indicates the number of methods that were surveyed for being suitable for performing the tasks.
Table 1. Survey about automation support for WIMP and Web UE methods (Ivory & Hearst 2001)

<table>
<thead>
<tr>
<th>Model Class: Testing</th>
<th>Method Type</th>
<th>Automation Type</th>
<th>Capture</th>
<th>Analysis</th>
<th>Critique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thinking-Aloud protocol</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Question-Asking protocol</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shadowing method</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coaching method</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teaching method</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Co-discovery learning</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance measurement</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td>F (7)</td>
<td></td>
</tr>
<tr>
<td>Log file Analysis</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td>IFM (19)</td>
</tr>
<tr>
<td>Retrospective Testing</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote Testing</td>
<td>None</td>
<td>IF (3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model class: Inspection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guideline review</td>
<td>None</td>
<td>IF(6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cognitive walkthrough</td>
<td>None</td>
<td>IF(2)</td>
<td>F (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pluralistic walkthrough</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heuristic evaluation</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perspective-Based Inspection</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature inspection</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal usability Inspection</td>
<td>None</td>
<td>F (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consistency inspection</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standards inspection</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model class: Inquiry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contextual Inquiry</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field observation</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Focus groups</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interviews</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surveys</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Questionnaires</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td>IF(2)</td>
<td></td>
</tr>
<tr>
<td>Self-reporting logs</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Screen snapshots</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User feedback</td>
<td>None</td>
<td>IF(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model class: Analytical Modeling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOMS analysis</td>
<td>None</td>
<td>M (4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UIDE analysis</td>
<td>None</td>
<td>M (2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cognitive task analysis</td>
<td>None</td>
<td>M (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task-Environment analysis</td>
<td>None</td>
<td>M (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knowledge analysis</td>
<td>None</td>
<td>M (2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design analysis</td>
<td>None</td>
<td>M (2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmable user models</td>
<td>None</td>
<td>M (2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model class: Simulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information proc. modeling</td>
<td>None</td>
<td>M (9)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Petri net modeling</td>
<td>None</td>
<td>FM (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genetic algorithm modeling</td>
<td>None</td>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information scent modeling</td>
<td>None</td>
<td>M (1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automation type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>None</td>
<td>30</td>
<td></td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Percent</td>
<td>None</td>
<td>67%</td>
<td></td>
<td>13%</td>
<td>18%</td>
</tr>
</tbody>
</table>

Table 1 states that the 67% of the methods surveyed does not support any type of automation meanwhile the 33% support some type of automation, being: gathering
information methods 13%, analysis methods 18% and valuate methods 2%. One could assure that the automation has not been enough explored yet.

By analyzing the data provided by Ivory and Hearst, one may notice that there exists a lack of automation within the testing methods. This lack of automation may lead to less cost-efficiency projects and delays. Nevertheless, the favorable advantages suggest that during the development, one of the processes of the project should be automated for user interface testing.

Several groups of researchers have tried to establish a model for the user testing process including different sets of processes and methodologies based on day-to-day best practices. Even though they result useful on the development process; these models are based on individual expertise mainly, thus they are not settled as a common agreement. Hornbæk (2011) stated that there exist few guidelines in order to design, run and report experiments. Helms et al. described four methodologies (2006):

- Logical User-Centered Interaction Design (LUCID), an iterative user-centered approach to user interaction design. It comprises six phases: (1) Envision, (2) User and task analysis, (3) Design and prototype, (4) Evaluate and refine, (5) Complete detailed design and production, and (6) Deployment and monitoring. LUCID is based on the feedback provided by the users due to they are a key component of the cycle.

- The star life cycle is an iterative and evaluation-centered usability engineering method that aims to create an environment that supports a regular and iterative analysis (Hartson & Hix 1989). One of the advantages of the star life cycle consists of the possibility of starting with any development activity among the highly interconnected activities. The activities must be analyzed before starting the next one.

- The waterfall model approach is a broadly used and rigid method that follows a linear evolution of tasks (Bell & Thayer 1976). It involves many stakeholders with diverse expertise to execute a different set of tasks that includes (1) System feasibility, (2) Operations and maintenance, (3) Implementation, (4) Integration, (5)
Code, (6) Detailed design, (7) Product design, and (8) Software plans and requirements.

- The spiral model of software development, an iterative model more flexible than the waterfall model being the iterations larger and slower (Boehm 1988).

Helms (2006) stated that the previous methodologies experience frailty towards one or several of the following aspects:

- The absence of tools to support and integrate the methods with the software development tools.
- The methods do not supply enough assistance for some of the main activities and milestones required amid the testing process such as data benchmarking and analysis, expert and user hiring, planning or reporting.
- During the early design stage, some methods do not provide clear relations with the overall software engineering process.
- The lack of integration with another methods and techniques due to the enforcement to use testing tools included in the process description.
- The lack of flexibility by cause of processes that are organized as a set of predefined and fixed activities.
- The dearth of customization using one of the methods in case of the project has special requirements such as mobile applications or critical services.

Accordingly to the list shown above, the existing methods for testing the UI lack either of versatility, wholeness and scope of coverage and/or integration with the software development tools and within the software development lifecycle. Therefore, it seems necessary to find a well-defined user testing process that gathers and evaluate user’s opinions, expectations, experiences, behaviors and actions. The collected data could be used in different manners such as:

- Design patterns (Borchers 2001) and establishing guidelines (Johnson 2010) for further developments and UI design,
• Identification of the user test benchmark that involve an ample range of user-oriented testing projects (Fidopiastis et al. 2010),
• The computation of the quantitative qualities of the user performance and productivity (Rubin, & Chisnell 2008),
• The comparison of different versions of one product or similar products from the end-user perspective (Mayhew 1999),
• The evaluation of the prototype from the user perspective including usability and user experience (Tullis & Fleischman 2002).

To summarize this chapter, we could say that there exist several methods for testing a UI, but still different groups testing the same UI may get different results since each of those methods is inherently qualitative or illustrative to different testing aspects.

Today, the trending consists of creating modern web applications in order to provide new and improved services that were previously offered through installed applications on the local machines (Andrews et al. 2005). These web applications usually comprise several back-end components and modules that may be written in different languages and that are integrated with the aim of offering complex systems and solutions that cover all the requirements of the clients. Such complex applications require specific testing methods that would cover all the possible issues.

In the next chapter, an innovative method that could be used for testing widgets and user interfaces of different applications is introduced: the CAUTE process. In addition, it presents (as a case study) a web application on how a CAUTE tool might be used by one test or lab manager.
4 COMPUTER-AUTOMATED USER TEST ENVIRONMENT TOOLS

In this chapter, the CAUTE process is introduced including the advantage of using it for testing a UI, as well the phases that must be followed to complete the procedure. In addition, a web-based CAUTE tool is presented. It aims to explain how a lab manager or a test manager could use the process through a simple web site.

For testing purposes, developers and testers have been using computer-aided software engineering (CASE) tools in order to audit and control the functionalities within the software lifecycle (IEEE 2009). The usage of CASE tools may ease the software development and maintenance (Orlikowski 1993). CASE tools supports the software lifecycle on several aspects including business modeling, the design of the phases of the life-cycle, the validation and correctness of the code specifications, the maintenance of the file repository, the analysis of the complexity and modularity of the code and the project management and scheduling (Case A.F. 1985).

The favorable advantages stated in the previous chapter suggest that during the development, one of the processes of the project should be the automated user interface testing. Considering that CASE tools ease the software development and maintenance, a development team may need a tool for monitoring and selecting the appropriate testing methods for the UI. However, it is not the main purpose of this thesis to develop a new framework. Therefore use the framework proposed by Seffah et Al will be used here.

4.1 What is CAUTE?

The major purpose of the framework consists of a sort of tasks that ease the process of systematization and personalization of the process model for dissimilar tests and projects. The Computer-Automated User Test Environment (CAUTE) aims to fulfill the user-oriented testing professional requirements as CASE tools are intended for software engineers and it may be used either in some stage of the user-oriented testing or during all the functions performed by the testers.
The inclusion of a detailed framework on the user-oriented testing processes, such as CAUTE, might solve or mitigate the incidence of several problems related to the integration of automated methods to the testing processes. Some of the related issues that can be found are:

- The incompatible of data types and the lack of communication and coordination among tools. This leads to additional manual work done by testers.
- The absence of pattern for designing and controlling the testing process during its execution.
- The lack of tools for processing documentation and manage the process.
- The difficulty to incorporate non-expert and software development teams to the testing process.
- The laboriousness to introduce the user-oriented testing in the software development lifecycle.
- The process do not provide a standardized method for collecting data regardless of the type of user since all of them enact an important role and impacts on the quality of the test.

Thus, the usage of CAUTE tools may accelerate the testing process by automatizing repetitive tasks, integrating other tools to the processes and allowing testers to concentrate on the creative facets of the user-oriented testing.

Seffah et al. claims that CAUTE is a user-oriented process formed by a combination of other processes already described including usability testing as defined in the HCI community (Nielsen 1994). This includes controlled experiments used in psychology and broadly used in life science and engineering fields such as HCI; empirical software engineering and software business economics (Sjoberg et al. 2005; Kerlinger & Lee 2000); and system and software user acceptance testing that consists of detecting differences between the behavior of a software piece and the expected one through the usage of it. Finally, it also includes unit testing and integration testing by developers, system testing by testers and the user acceptance testing by the users (Pressman 2009).
The process contains 11 stages as follows:

1. **Plan**: during the testing plan phase, one should create a draft with *what* will be tested (prototype, models, software system), *how* (user testing methods and tools), when (stage of the development lifecycle), *where* (in a lab, remotely via Internet, user’s workspace), *who* will participate (subjects, stakeholders, evaluators), *why* the test is performed (objectives) and *which aspects* might be considered during the test.

2. **Design**: on the design phase one needs to select the appropriate research method, and prepare the required resources to perform the test, including the preparation of the documentation that will be followed to interact with the participants. In addition, development of questionnaires, surveys and definition of the profile of the partakers and their groups, and how the information will be collected and analyzed considering the source of the information should be made.

3. **Acquire**: on the acquire phase, the participators of the test should be contacted, and a list with their data and schedules for the tests should be created. During this process one may hire the participants or use internal resources.

4. **Setup**: on the setup phase, the software and hardware that will be used in order to perform the test process should be installed, configured and tested. In case of using a HCI lab, or similar, special equipment might be necessary and these should be purchased, installed and tested during this process. At last, if the tests are performed remotely, a tool that will monitor the user actions on their working place is required.

5. **Preview**: on the preview phase, several tests should be performer to increase the trust on the deployed software, hardware and environment. At this stage, the lab manager should assure that all the options previously selected suits the needs of the functionalities of the interface that will be tested.

6. **Conduct**: on the conduct phase, the data should be collected through the different tests that were programed. On this phase, the selected methods on the previous steps should be used to collect qualitative and quantitative data through log files of human actions, video observations, feedback and screen captures.
7. **Debrief**: on the debrief phase, the participants should be audited and questioned about their actions, feelings and reactions during the tests.

8. **Compile**: on the compile phase, all the data is collected and stored in a secure and accessible environment in order to assure that it can be accessed by the people who will analyze it.

9. **Analyze**: on the analysis phase, the pertinent data analysis approach and mining methods should be selected in order to convert the results into conclusions and possible improvements.

10. **Report**: this phase usually consists of three types of report that are written during the latest phases of the framework. The first report can be drafted after the debrief phase, or compile phase, is completed. The second report details the results of the test and how the test was performed and its conclusions. The last report focuses on the mistakes that were found during the whole process and possible improvements for future plans.

11. **Capitalize**: on this phase, the users’ and evaluators’ feedback should be evaluated in order to find strengths and vulnerabilities of the process.

### 4.2 Web CAUTE-tool and processes

As mentioned before, Molich et al. (1999) carried out research indicating that even though usability groups may find several usability issues while testing an application, these will broadly defer from the results of a different usability testing group. This conclusion does not mean that a particular group had a lack of skills or knowledge, but instead that it does not exist a standard framework that a usability group could follow to reach some clear statements.

For this purpose the Common Industry Format (CIF) (ISO 25064 2013) was developed to establish a framework that might help for creating consistent reports while executing usability tests. CIF emphasizes on the incorporation of the usability as a part of the decision-making process for computer software.

In this work, one application that may assist on the creation of CIF for usability and user interface test reports is proposed: the CAUTE framework introduced first by Seffah et al.
The application Web Based CAUTE Tool (WBCT) aims to help managers and testers during the whole testing process; from the planning until the final report offering a relatively simple and intuitive user interface accessible from a web navigator.

Figure 11 shows the main screen of WBCT. A simple web interface has been used in order to ease the testing procedure for the companies; thus, a high level of previous testing knowledge is not required. The main screen gives access to the different sections of the application; this is the list of projects that some user of the application is currently testing and the list of possible audience for the testing. It also contains graphics about the performance of the testing and notifications with useful information and possible problems.

The first stage for using WBCT consists of creating an initial population for the database by means of persona (Wolpert et al. 2011) representing the people who might participate during the testing process: final users, stakeholders, managers and professionals. Figure 12 displays a populated data base that includes some of the profiles defined by Seffah et al. for the CAUTE process.
On the second step, one of the projects should be selected or the user should create a new one. As shown on the figure below, as one clicks on “Projects”, the system lists all the existing projects.
To ease the process, we have divided the eleven steps proposed by Seffah et al. in three stages: “Prepare test”, “Gather data” and “Analyze data”. Each one of these stages is decomposed in a set of smaller phases. “Prepare test” stage includes “Plan”, “Design”, “Acquire” and “Setup” phases. “Gather data” stage includes “Preview”, “Conduct” and “Debrief and compile” phases. “Analyze data” stage includes “Analyze”, “Report” and “Capitalize”. In addition, each of the phases is divided on several sub-steps that request different information to succeed on the testing process. Figure 14 discloses these different stages that should be performed while using the application.

During the preparation of the test, the general information about the project should be introduced, the testing method, the audience of the testing process, materials (hardware and software) that will be used and the environment where the testing will be executed. At “Gather data” stage, the WBCT suggest to conduct a pretest and a formal inspection, to prepare the test and to give some recommendations on how to perform the testing method that the user selected on the previous stage. At last stage “Analyze data”, the application invites to search for possible failures and patterns, creating a categorized list by priority, severity and frequency of occurrence. Next, the possible improvements based on the
detected failures should be developed and prioritized. At last, one can download a CIF report with all the detailed information from the project screen.

Once the process starts, the first stage “Prepare test” and on the first phase “Plan”, basic information of the project should be introduced: name, version and a short description of the application, the purpose and goal of the test and the frame for the whole process as stated on figure 15.

![Figure 15. WBCT: Plan phase 1. The user introduces general data about the project.](image)

Next, the testing method that will be applied during the process should be selected as shown in the image below. Once the methods is selected, the application gives a short description of the testing method, the stages of the development where it can be applied, the typical required personnel, what can be measured, the type of data that can be collected, how to perform the test, the possibility for remote testing, and the weakness of the method.
At last, the widgets and scenarios that will be tested should be set introducing a title for the action and an adequate description. Since these scenarios might change before starting the process, the information can be stored and modified if necessary. After introducing the data, one should press “Next” to continue with the second step, “Design”.

On the “Design” phase, the application suggest some requirements categorized in three sections, indispensable elements for the selected method, useful elements that would help to perform a better test and elements that are recommendable to use as good practice. Figure 18 displays the requirements of the thinking-aloud method related to the location and the environment where the test will be performed. On this phase, the available requirements should be identified by marking the checkboxes placed close to the options.
Figure 18. WBCT: Design phase 1. The user fills in a form about the requirements related to the location.

Figure 19 shows the requirements connected with the necessary equipment while testing by using the thinking-aloud method. As in the previous step, the resources should be selected for performing the test.

Figure 19. WBCT: Design phase 2. The user fills in a form about the requirements related to the equipment.

The “Design” step continues with the selection of the audience. The application offers the different roles defined by Seffah et al. on the CAUTE process, briefly explaining the
functions of each role. The necessary roles for the testing process should be selected using the assistance provided on the previous step while selecting the testing method.

Figure 20. WBCT: Design phase 3. The user selects the roles that will participate during the test process.

On the last step, the amount of people for each of the previously selected role should be set as shown in the image below. Once this action is performed the “Design” phase is completed and is possible to proceed with the next phase, “Acquire”.

Figure 21. WBCT: Design phase 4. The user selects the amount of participants of each role.

On the “Acquire” phase, the individuals who will participate on the process should be selected, as shown on the image below. The system displays the persona of the people who belong to any of the previously selected roles. In this view, people can be included or excluded from the process and the dates when they will perform the tests or meetings
should be set. In addition, one should include the role of the user while using the software, the amount of years that the user has been performing the role, the user’s experience using the platform or operating system, where the application will be deployed, the experience of the user in application related to the same domain and the user’s experience with previous versions of the product or similar products. After the individuals who will participate on the process are selected, the “Acquire” step can be considered as complete.

The last phase on the “Prepare test” stage is “Setup”. On this phase, the system analyzes the introduced data and displays failures, in red, and recommendations, in blue, considering the testing method selected on the “Plan” phase.
At last, the system requires the supervision of all the options included on the review step in order to guarantee that all the requirements have been meet before proceeding with the test itself. As shown on the image below, the hardware and software should be verified, install and configure the product that will be tested, prepare the physical environment completing the requirements settled on the “Design” phase, review and assemble the test materials and prepare the legal documents that the audience must sign to proceed with the test.

After the “Prepare test” stage is completed, the web browser is directed to the product dashboard, as shown on the image below, where one can download a partial CIF report with a summary of the current information and continue with the next stage “Gather data”.
The “Gather data” stage starts with the “Preview” phase. In this phase, the WBCT suggests to conduct a pretest and a formal inspection in order to find possible failures on the designed plan for the testing process. The application also suggests how to perform these activities recommending a set of steps that the test manager may follow. Figure 26 stands how the pretest and the formal inspection should be executed. The test manager should mark the checkboxes after the activity has been finished.
Figure 26. WBCT: Preview stage 1. System recommendations for conducting a pretest and formal inspection.

Figure 27 shows the last step of the “Preview” phase. WBCT suggest which type of skills are required and on which field the usability testers and observers should be trained to reap the maximum possible benefits of the procedure.

The next phase within the “Analyze data” stage is “Conduct”. In this phase, the system suggests to prepare the test and the observers and observers’ positions as stated on the
The hardware and software should be verified, the test manager should check that they operate as predicted and that the observers are in their positions.

![Figure 28. WBCT: Conduct stage 1. System suggestions for test and observers preparation.](image)

On the “Conduct” phase and before starting the test with the audience, the application offers some recommendations about the selected testing method and how to proceed with some required tasks as shown in the image below. After this, WBCT proposes a list of actions that might help on conducting the test since the audience arrives until they finish with the assigned tasks.
Figure 29. WBCT: Conduct stage 2. System suggestions on how to proceed using thinking-aloud method and the recommend steps for performing the test.

Figure 30 shows the “Debrief and compile” phase. WBCT suggests to collect the last information from the users with different methods (open-mind opinions, interview the users, post-test questionnaires) that are specific for the diverse testing methods that the application offers. As well, one should perform the same actions with professionals in order to avoid a possible loss of some information prior to the “Analyze” process.

Figure 30. WBCT: Debrief stage 1. System recommendations for debriefing audience after performing the test.
On the last step of the “Gather data” stage, the application advocates to organize and clean the raw results and annotate where the data is going to be stored as can be seen in figure 31. In this case, since the thinking-aloud method had been selected, the WBCT proposes that the path for the video, audio and screen capture files should be included so that they are easily accessible to anyone who is using the application or the CIF generated by the WBCT.

![Figure 31. WBCT: Debrief stage 2. The system suggests how to organize the results.](image)

After the “Gather data” stage, WBCT displays the information related to the stages that has been completed as shown in the image below. To complete the testing process, “Analyze data” stage should be finalized.
The last stage of the testing process starts with the “Analyze” phase where one should identify and prioritize preliminary user problems using different methods such as data mining, discovery or prediction. With such information, WBCT creates a list of potential problems as shown in the image below.
The second step on the Analyze phase, the required values related to usability metrics and satisfaction should be included for every task and user so the test manager can establish some comparisons using the data.

![Usability metrics and satisfaction measurements](image)

**Figure 34. WBCT: Analyze stage 2: Introduce usability metrics and satisfaction measurements**

After this, the application suggests to prepare some comparisons in order to find new potential problems that were not detected by the users and the professional testers, as shown on the image below.

![Compare data and introduce patterns and findings](image)

**Figure 35. WBCT: Analyze phase 2: Compare data and introduce patterns and findings**

On the “Report” phase, the application gives some recommendations on how to study patterns and recommendations and suggests creating a list of recommendations that might
be included in further versions of the product as can be seen in the image below. WBCT also gives some guidelines for the internal report and the process that one should follow after the testing process.

![Image of WBCT tool]

Figure 36: WBCT: Report phase. Introduce recommendations and create internal report

The last phase on the testing process is named “Capitalize”. WBCT encourages the test manager to create, organize and send some questionnaires for the audience in order to collect possible feedback of the testing process that might be used in further testing procedures. Figure 37 shows how the capitalize phase could be presented in a CAUTE tool.
At the end of the “Analyze data” stage, WBCT shows all the information of the testing process and facilitates a complete CIF in a printable and understandable format that can be stored or used as final report of the whole procedure as shown in the image below.

As commented before, at the end of each stage WBCT offers a partial CIF related to the testing process. It is possible to consult the output results, in CIF format, for the case study
testing process that have been shown on this section, in the appendix attached at the end of this thesis.
SUMMARY

As stated during the document, there no exist common agreements on how to perform a whole test process for a UI. Several scientists have tried to categorize the level of automation of the different user interface testing method and they have found that the methods lack of automated process either on gather, analyze or critique data. Other scientist demonstrated that different testing groups examining the same user interface could obtain diverse results.

The usage of a new process called CAUTE has been suggested. This might ease the testing procedure along with one web application that may guide test managers on planning, gather, analyze and creating conclusions of a test process.
REFERENCES


APPENDIX 1. CIF results from the testing process

Common Industry Format

This report aims to incorporate usability as part of the decision-making process such as purchasing, upgrading or automating procedures of a software product. It provides information about the methods and results of usability test and user interface testing.

Product: Car Expenses Tool

1. Product description

Car Expenses Tool aims to help users in controlling the gas expenses of a given car. User must introduce the amount of liters that he/she refuels, the price total price of the refueling, the date and the mileage.

The product has been designed for a population that pretends to control the gas consumption in order to reduce it. The application will be mainly used in mobile devices or tablets and in order to use it the user must download it from one of the stores where the application will be available; App Store, Google Play or Windows Store. It can be also used as a web service with a computer.

The product version is the 0.8a, a Beta version of the final product that will be available by the end of 2016.

(continues)
APPENDIX 1. (continues)

2. Test objectives

The goal of the process is to test the log in user interface. User must feel that the UI is simple and nice.

This process should measure the effectiveness, efficiency and errors of the web service application in a computer. The users must register using a social network and log in after that. No mobile devices are required.

The testing process will start on 22/07/2016 and ends by 04/08/2016.

3. Method

The selected testing method was Thinking-aloud protocol. It is a technique where the participant is asked to vocalize his or her thoughts, feelings and opinions while interacting with the product. User must understand that the subject of testing is the application, not the user, and everything he/she is doing is right.

3.1. Participants

The selected participants on WBCT are listed on the table below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Education</th>
<th>Occupation</th>
<th>Professional Experience</th>
<th>Computer Experience</th>
<th>Product Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jose Nunez</td>
<td>Male</td>
<td>47</td>
<td>Undergraduate</td>
<td>Basic user</td>
<td>5-10 years</td>
<td>1-5 years</td>
<td>+10 year</td>
</tr>
<tr>
<td>Francisco</td>
<td>Male</td>
<td>52</td>
<td>Basic school</td>
<td>Professional user</td>
<td>+10 years</td>
<td>+5 years</td>
<td>2-5 years</td>
</tr>
<tr>
<td>Navarro</td>
<td></td>
<td></td>
<td></td>
<td>Test manager</td>
<td>+10 years</td>
<td>+10 years</td>
<td>5-10 years</td>
</tr>
<tr>
<td>Manuel</td>
<td>Male</td>
<td>52</td>
<td>Doctoral studies</td>
<td>Basic user</td>
<td>0-1 year</td>
<td>0-1 years</td>
<td>0-1 years</td>
</tr>
<tr>
<td>Lopera</td>
<td></td>
<td></td>
<td></td>
<td>Basic user</td>
<td>0-1 year</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rogelio</td>
<td>Male</td>
<td>51</td>
<td>Basic school</td>
<td>Basic user</td>
<td>0-1 year</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sosa</td>
<td></td>
<td></td>
<td></td>
<td>Basic user</td>
<td>0-1 year</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2. Context of product use in the test

The reason for testing these functionalities is that are the most frequent task. The functionalities that will be tested are (defined by the test manager):

(continues)
APPENDIX 1. (continues)

3.2.1. Test access window

User must be able to find what to introduce in the fields and where they are located. The fields that should be introduced are “Email”, “Password” and press enter or click on “Log in”.

3.2.2. Register using a social network

User must be able to find how to register using different networks. The included networks are Twitter, Facebook, LinkedIn and Instagram.

4. Physical environment

Here are the characteristics of the physical environment introduced in WBCT:

- Quiet and interruption free environment.
- Controlled and multiple light sources.
- A work surface.
- Sound proof room.
- Multiple electrical socks.
- Accessibility to the room.

5. Computing environment

Here are the characteristics of the computing environment introduced in WBCT:

- Computer with monitor, mouse and keyboard
- Audio capture.
- Webcam pointing to the user.
- Screen-capture software.
- Computer survey.
- Ambient microphones.
- Eye-tracking equipment
- Video editing software.

(continues)
APPENDIX 1. (continues)

6. Experimental design

The experimental design was controlled by the user in WBCT. The testing session followed the next process:

6.1. The application will be tested by 1 test manager and 3 users.
6.2. Prepare the environment for the users. Keep silence into the room and verify the lights.
6.3. Initialize the hardware and open the application.
6.4. Initialize observers’ application and UI capture software if any.
6.5. Prepare the environment for the observers.
6.6. Prepare devices and applications that the observers will use.
6.7. Welcome the user or users.
6.8. Ask the users to sigh the legal documentation.
6.9. Explain the content and the application that will be tested.
6.10. Ask the users to fill in the pre-test questionnaire made on the previous step.
6.11. Explain the task scenarios that the user will perform.
6.12. Monitor the test sessions and record the data.
6.13. Ask users to fill in the post-test questionnaire.
6.15. Review the post-test questionnaires with the users.
6.16. Thanks the users.
6.17. Review the post-test questionnaires with observers and usability professionals.
6.18. Add observers’ and usability professionals’ comments.
6.19. Store the data:
   6.19.1. Video: Z:/CET/08a/video
   6.19.2. Audio: Z:/CET/08a/audio
   6.19.3. Screen-captures: Z:/CET/08a/screencap

(continues)
APPENDIX 1. (continues)

7. Participant general instructions

Below stand the instructions given to the users for performing the actions:

7.1. You should create an account using different social networks.

7.1.1. Task 1: Find the button to register yourself using your Twitter account and complete the process.

7.1.2. Task 2: Find the button to register yourself using your Facebook account and complete the process.

7.1.3. Task 3: Find the button to register yourself using your LinkedIn account and complete the process.

7.1.4. Task 4: Find the button to register yourself using your Instagram account and complete the process.

7.2. Task 5: Log in into the application using your email and password.

7.3. Other details

7.3.1. There will not be interaction with other people.

8. Usability metrics

The tables below represent the usability criteria of the users performing the tasks:

<table>
<thead>
<tr>
<th>Jose Nunez</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task time (min)</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Completion rate (0-100%)</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Frequency of errors</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frequency of assistance</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User satisfaction (1-5)</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Francisco Navarro</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task time (min)</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Completion rate (0-100%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Frequency of errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frequency of assistance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User satisfaction (1-5)</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Manuel Lopera</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task time (min)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Completion rate (0-100%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Frequency of errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frequency of assistance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User satisfaction (1-5)</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

(continues)
## APPENDIX 1. (continues)

<table>
<thead>
<tr>
<th>User</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task time (min)</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Completion rate (0-100%)</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Frequency of errors</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frequency of assistance</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>User satisfaction (1-5)</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

### 9. Results

Here are listed the final results related to Car Expenses Tool listed by task.

#### 9.1. Task 1

<table>
<thead>
<tr>
<th>User</th>
<th>Unassisted Task Effectiveness (% Complete)</th>
<th>Assisted Task Effectiveness (% Complete)</th>
<th>Task Time (min)</th>
<th>Satisfaction</th>
<th>Errors</th>
<th>Assists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jose Nunez</td>
<td>-</td>
<td>100</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Francisco Navarro</td>
<td>100</td>
<td>-</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Manuel Lopera</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rogelio Sosa</td>
<td>-</td>
<td>100</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Mean</td>
<td>100</td>
<td>100</td>
<td>5</td>
<td>3,75</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Min</td>
<td>100</td>
<td>100</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>100</td>
<td>100</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

#### 9.2. Task 2

<table>
<thead>
<tr>
<th>User</th>
<th>Unassisted Task Effectiveness (% Complete)</th>
<th>Assisted Task Effectiveness (% Complete)</th>
<th>Task Time (min)</th>
<th>Satisfaction</th>
<th>Errors</th>
<th>Assists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jose Nunez</td>
<td>100</td>
<td>-</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Francisco Navarro</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Manuel Lopera</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rogelio Sosa</td>
<td>-</td>
<td>100</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>100</td>
<td>100</td>
<td>4</td>
<td>3,75</td>
<td>0,25</td>
<td>0,25</td>
</tr>
<tr>
<td>Min</td>
<td>100</td>
<td>100</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>100</td>
<td>100</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

#### 9.3. Task 3

<table>
<thead>
<tr>
<th>User</th>
<th>Unassisted Task Effectiveness (% Complete)</th>
<th>Assisted Task Effectiveness (% Complete)</th>
<th>Task Time (min)</th>
<th>Satisfaction</th>
<th>Errors</th>
<th>Assists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jose Nunez</td>
<td>100</td>
<td>-</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Francisco Navarro</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Manuel Lopera</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rogelio Sosa</td>
<td>100</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>100</td>
<td>-</td>
<td>3</td>
<td>4,5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Min</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>100</td>
<td>-</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
APPENDIX 1. (continues)

9.4. Task 4

<table>
<thead>
<tr>
<th>User</th>
<th>Unassisted Task Effectiveness (% Complete)</th>
<th>Assisted Task Effectiveness (% Complete)</th>
<th>Task Time (min)</th>
<th>Satisfaction</th>
<th>Errors</th>
<th>Assists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jose Nunez</td>
<td>100</td>
<td>-</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Francisco Navarro</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Manuel Lopera</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rogelio Sosa</td>
<td>100</td>
<td>-</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>100</td>
<td>-</td>
<td>2,75</td>
<td>4,5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Min</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>100</td>
<td>-</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

9.5. Task 5

<table>
<thead>
<tr>
<th>User</th>
<th>Unassisted Task Effectiveness (% Complete)</th>
<th>Assisted Task Effectiveness (% Complete)</th>
<th>Task Time (min)</th>
<th>Satisfaction</th>
<th>Errors</th>
<th>Assists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jose Nunez</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Francisco Navarro</td>
<td>100</td>
<td>-</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Manuel Lopera</td>
<td>100</td>
<td>-</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rogelio Sosa</td>
<td>100</td>
<td>-</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>100</td>
<td>-</td>
<td>2</td>
<td>4,75</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Min</td>
<td>100</td>
<td>-</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>100</td>
<td>-</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>