

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

**MÁSTER UNIVERSITARIO EN INGENIERÍA DEL SOFTWARE –
EUROPEAN MASTER IN SOFTWARE ENGINEERING**



Application of User-centered Design in the Development of an Automated/Assisted Testing System for Self-service Devices

Master Thesis

Edwin Leonardo Téllez Sánchez

Madrid, July 2015

This thesis is submitted to the ETSI Informáticos at Universidad Politécnica de Madrid in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering.

Master Thesis

Master Universitario en Ingeniería del Software – European Master in Software Engineering

Thesis Title: Application of User-centered Design in the Development of an Automated/Assisted Testing System for Self-service Devices

Thesis no: EMSE-2015-05

July 2015

Author: Edwin Leonardo Téllez Sánchez

Ingeniero de Sistemas

Universidad Nacional de Colombia

Supervisor:

Xavier Ferré
Ph.D. in Computer Science
Universidad Politécnica de Madrid

Departamento de Lenguajes y Sistemas
Informáticos e Ingeniería de Software
Escuela Técnica Superior de Ingenieros
Informáticos
Universidad Politécnica de Madrid

Co-supervisor:

Jose González-Alonso
Ingeniero en Informática
Universidad de A Coruña, Spain
Master in Business Administration
University of Oxford, UK

Head of professional services for
Payments – Europe
NCR Corporation



ETSI Informáticos
Universidad Politécnica de Madrid
Campus de Montegancedo, s/n
28660 Boadilla del Monte (Madrid)
Spain

To God

To my family

To my friends in Spain and Colombia,

And to everyone that has helped me be a better person and professional

ACKNOWLEDGEMENTS

I thank to Xavier Ferré, my advisor, and Jose Gonzalez, my co-advisor at NCR, for their feedback, patience, and trust.

I thank also the people at NCR Iberia, who made possible this project, especially Susana Miranda and Eva Pindado. They were a supportive team during the nine months of work.

Finally, I thank the Graduated International School of Campus Moncloa (EIP Campus Moncloa), which supported my studies through a scholarship for the academic year of 2014-2015.

CONTENTS

ABSTRACT.....	9
1 INTRODUCTION	10
1.1 ATM software layers	10
1.2 ATM software layers	10
1.3 The AX certification process.....	11
1.4 Problem statement	12
1.5 Project goal.....	12
1.6 Solution Proposal	12
1.7 Expected benefits.....	12
1.8 Contents of the document	13
2 SELECTING USABILITY TECHNIQUES IN SOFTWARE PROJECTS	15
2.1 Usability techniques for agile projects.....	17
2.2 Usability Planner	20
2.3 Integrating UCD with agile methods	20
3 THE AX CERTIFICATION PROCESS	22
3.1 NCR AX Middleware	22
3.2 Business process model	23
3.3 Activities and performers.....	25
3.4 Process issues.....	26
3.5 Analysis of effort required by the current process	27
3.5.1 Time by activity (hours required by an expert)	27
4 SELECTION OF USABILITY METHODS IN THE PROJECT	29
4.1 Methods selection with Usability Planner	32
4.1.1 Activities.....	32
4.1.2 Method selection	33
4.1.3 Methods for design.....	34
4.1.4 Methods for evaluation	35
4.2 Usability plan.....	36
5 SPECIFICATION OF THE CONTEXT OF USE	38
5.1 User profiles	38

5.2	User tasks	39
5.2.1	Test execution tasks.....	40
5.2.2	Test reporting tasks	42
5.3	Physical environment.....	43
5.4	ATM technical restrictions	45
5.4.1	Hardware restrictions	45
5.4.2	Software restrictions.....	45
5.5	Preliminary conclusions	46
6	INTERACTION DESIGN PROCESS.....	47
6.1	Milestone 1: Inception and Proof of concept	47
6.1.1	Analysis of the AX certification process and training on ATMs platforms	48
6.1.2	Product concept.....	49
6.1.3	Definition of main product features	51
6.1.4	Bringing concepts from software testing discipline	51
6.1.1	First research on similar products (Competitor analysis).....	52
6.1.2	Functional and non-functional requirements.....	53
6.1.3	First UI design (wireframe)	54
6.1.4	First proof of concept	55
6.1.1	Selecting the product name.....	56
6.1.2	Evaluation	56
6.2	Milestone 2: First stable version	56
6.2.1	Ethnographic observations	57
6.2.1	Second research on similar products (Competitor analysis)	58
6.2.2	Construction of the first application version	60
6.2.3	First usability test.....	62
6.2.4	Design concept of the command runner	63
6.2.5	Second usability test	64
6.2.6	Improvements after the usability test	65
6.2.7	Navigation map: optimizing the navigation of the execution view.....	68
6.3	Milestone 3: Second stable version	70
6.3.1	Implementing a new execution view.....	70
6.3.2	Third usability test	73

6.3.3	Design of a new test case summary view	73
6.3.4	Giving full control to the user: Pause and Resume.....	76
6.3.5	Implementation of a new Command Runner	76
6.3.6	Making usable dynamically generated UI forms	78
6.3.7	Fourth usability test	81
6.4	Milestone 4: Evaluating the final application version.....	83
6.4.1	Fifth usability test	83
6.4.2	Lines of improvement	83
7	EVALUATION.....	85
7.1	Methods, metrics, participants and location	85
7.2	Laboratory usability test 1	88
7.2.1	Goal	88
7.2.2	Activities.....	88
7.2.3	Results.....	88
7.3	Laboratory usability test 2	89
7.3.1	Goal	89
7.3.2	Activities.....	89
7.3.3	Results.....	89
7.4	Laboratory usability test 3	92
7.4.1	Goal	92
7.4.2	Activities.....	92
7.4.3	Results.....	92
7.5	Laboratory usability test 4	94
7.5.1	Goal	94
7.5.2	Activities.....	94
7.5.3	Results.....	95
7.6	Thinking aloud protocol for evaluating Reports	96
7.6.1	Goal	96
7.6.2	Activities.....	96
7.6.3	Results.....	96
7.7	Heuristic evaluation	97
7.7.1	Goal	98

7.7.2	Activities.....	98
7.7.3	Results.....	99
7.7.4	Recommendations	102
7.8	Usability test 5: Effort required in a real certification by non-experts.....	103
7.8.1	Goal	103
7.8.2	Activities.....	103
7.8.3	Usability issues found	103
8	RESULTS.....	107
8.1	Application functional modules	107
8.2	Project benefits	111
8.2.1	Tracking the effort required by the new system	111
8.2.2	Expected benefits per year	112
9	DISCUSSION	114
9.1	Critical analysis of the integration of UCD and Agile Methods.....	114
9.2	Critical analysis of the use of Usability Planner	114
9.3	Critical analysis of the usability methods applied.....	115
10	CONCLUSIONS AND FUTURE LINES OF WORK	117
10.1	Achievement of project goals	117
10.2	Business benefits of automation.....	118
10.3	Lessons learned about our use of UDC	118
10.4	Benefits and costs of usability evaluation.....	119
10.5	Future work.....	120
11	REFERENCES	121
12	APPENDIX A. TIMELINE OF APPLIED UCD METHODS PER ITERATION	125
13	APPENDIX B. INTERACTION DESIGN ITERATIONS FOR OTHER MODULES	130
14	APPENDIX C. SUMMARY OF THE FIRST ETHNOGRAPHIC OBSERVATION	143
15	APPENDIX D. SOFTWARE ARCHITECTURE	146
16	APPENDIX E. SATISFACTION QUESTIONNAIRE.....	154

ABSTRACT

Automated Teller Machines (ATMs) are sensitive self-service systems that require important investments in security and testing. ATM certifications are testing processes for machines that integrate software components from different vendors and are performed before their deployment for public use.

This project was originated from the need of optimization of the certification process in an ATM manufacturing company. The process identifies compatibility problems between software components through testing. It is composed by a huge number of manual user tasks that makes the process very expensive and error-prone. Moreover, it is not possible to fully automate the process as it requires human intervention for manipulating ATM peripherals.

This project presented important challenges for the development team. First, this is a critical process, as all the ATM operations rely on the software under test. Second, the context of use of ATMs applications is vastly different from ordinary software. Third, ATMs' useful lifetime is beyond 15 years and both new and old models need to be supported. Fourth, the know-how for efficient testing depends on each specialist and it is not explicitly documented. Fifth, the huge number of tests and their importance implies the need for user efficiency and accuracy. All these factors led us conclude that besides the technical challenges, the usability of the intended software solution was critical for the project success.

This business context is the motivation of this Master Thesis project. Our proposal focused in the development process applied. By combining user-centered design (UCD) with agile development we ensured both the high priority of usability and the early mitigation of software development risks caused by all the technology constraints.

We performed 23 development iterations and finally we were able to provide a working solution on time according to users' expectations.

The evaluation of the project was carried out through usability tests, where 4 real users participated in different tests in the real context of use. The results were positive, according to different metrics: error rate, efficiency, effectiveness, and user satisfaction. We discuss the problems found, the benefits and the lessons learned in the process.

Finally, we measured the expected project benefits by comparing the effort required by the current and the new process (once the new software tool is adopted). The savings corresponded to 40% less effort (man-hours) per certification. Future work includes additional evaluation of product usability in a real scenario (with customers) and the measuring of benefits in terms of quality improvement.

1 INTRODUCTION

Software testing is a critical activity for self-service devices. Testing is required not only during development but also after maintenance activities, when a software module is updated or a hardware module is replaced.

Automated Teller Machines (ATMs) are sensitive systems that require important investments in functional testing. And their prolonged useful lifetime –which can reach the 20 years-, increases the need for updates and system testing.

The NCR Company is a leading manufacturer of ATMs which also offers software products for non-NCR ATMs. Specifically, the **AX Middleware** is a product that allows customers to build vendor-independent applications. So customers that own diverse ATM brands and models can have the same application running on all them. In other cases, the middleware also eases the transition from old to newer ATM models and software platforms.

The AX platform requires a complex testing process to verify that it works on each ATM independently of the manufacturer. The process requires extensive and repetitive human effort which produces high costs and is error prone.

The goal of this Master Thesis project is to automate as much as possible the AX certification process, in order to increase efficiency and the quality delivered to customers.

1.1 ATM SOFTWARE LAYERS

Regarding hardware, ATMs are modular systems composed by several self-service financial peripherals such as Card Reader, Receipt Printer, Cash Dispenser, Note Acceptor, etc.

On the software side, ATMs have an architecture composed by two layers, according to the CEN/XFS standard:

1.2 ATM SOFTWARE LAYERS

Regarding hardware, ATMs are modular systems composed by several self-service financial peripherals such as Card Reader, Receipt Printer, Cash Dispenser, Note Acceptor, etc.

On the software side, ATMs have an architecture composed by two layers, according to the CEN/XFS standard:

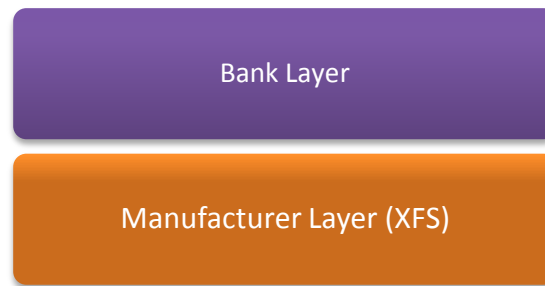


FIGURE 1. ATM software layers

- **Manufacturer Layer.** This layer is made up of the XFS Middleware (which is a standard for accessing financial devices independently of the manufacturer). It includes device drivers and the operating system.
- **Bank Layer.** The bank layer contains the bank application, which is the front-end that connects to banks transactional systems.

Theoretically, bank applications can operate on any ATM of any manufacturer thanks to the standardized manufacturer layer. But in practice the CEN/XFS standard is interpreted differently by manufacturers. Therefore, the AX Middleware is a layer added by NCR to even the differences of XFS interpretations.

The middleware only supports the ATMs that have been certified. Given the number of different ATM configurations in the market –such as brands, models, operating system versions, and peripherals- certifications are performed on demand for each individual ATM.

1.3 THE AX CERTIFICATION PROCESS

Usually, certifications are requested when a customer buys a new ATM model that is not supported by the AX, or after an important update in the ATM manufacturer's layer, such as the migration to a newer operating system. And certifications are critical, because all the ATM operations will rely on the AX middleware, once the ATM is certified and it is in use.

The AX Certification Process is a system testing process performed on a singular ATM that runs the AX Middleware. The process is performed by a team of specialists that travel to the ATM place, and performs an exhaustive set of tests on the system. That requires physical manipulation of ATM peripherals. For that reason, the tests cannot be done remotely or automatically.

The outcome of this process is the Certification Report, which summarizes the results of the tests, the issues found, and the responsible party (the software provider). The report size depends on the number of peripherals tested, but it usually contains more than 150 printed pages, which are edited manually.

Once the responsible party fixes the faulty software, the process of testing and reporting is repeated until no issues are detected. This can imply additional travels.

The full process can require more than 10 working days. Although, depending on the department workload and business priorities, it can be expanded to more than a month.

The company wants to automate the Certification Process because is too costly in terms of money and time, and the specialists are a small team.

1.4 PROBLEM STATEMENT

Currently, the Certification Process has the following problems that need to be solved:

- Many process tasks are carried out manually and are error-prone and costly.
- Tasks that require user interaction require high user efficiency and accuracy.
- The know-how for efficient testing depends on each specialist and it is not explicitly documented. This is also a business risk for the company.
- The huge diversity of ATM brands and models (that range from 20 years ago until today) need to be supported, as any of these systems can require certification.

In summary, it is required a software solution for automating specific tasks and for assisting users in others. Besides the technical challenges, the efficiency and accuracy of use are critical and the context of use is more constrained than ordinary software. Software usability is critical for the success of the project.

1.5 PROJECT GOAL

The goal of the project is to develop from scratch a software tool that effectively assists users through the AX Certification Process, in order to overcome the current problems in the challenging context of use.

1.6 SOLUTION PROPOSAL

Our proposal is the application of user-centered design (UCD), along with agile software development for building the desired software solution.

The former is for ensuring that product usability will have a high priority, and the latter will allow us to early identify and mitigate development risks to guarantee a working solution will be delivered on time and budget.

The agile development method will be based on several SCRUM practices such as: Short iterations (of 1 or 2 weeks), backlog management, sprint reviews, and sprint planning.

1.7 EXPECTED BENEFITS

With the incorporation of the new software we expect the following benefits:

Time and costs reduction

- Reduced specialists' effort and time during ATM testing
- Reduced travelling costs because of fewer visits to customer facilities
- Automatic generation of the Certification Report.

Improved perceived quality

- Easier issue detection and tracking (through better tests tracing)
- Reduction of typing errors during report generation

1.8 CONTENTS OF THE DOCUMENT

This document is structured in 12 chapters and 3 appendixes:

- **First Chapter: INTRODUCTION.** This introductory chapter presents the project, the goals and the solution proposal.
- **Second Chapter: SELECTING USABILITY TECHNIQUES IN SOFTWARE PROJECTS.** It is an overview of the state of the art of usability methods selection in user-centered design.
- **Third Chapter: THE AX CERTIFICATION PROCESS.** It presents a description of the business process where the software solution is required, and also it provides an analysis of effort required by the company in the current state.
- **Fourth Chapter: SELECTION OF USABILITY METHODS IN THE PROJECT.** It shows the criteria used for selecting the usability methods in the project and the usability plan.
- **Fifth Chapter: SPECIFICATION OF THE CONTEXT OF USE.** It is the specification of the context of use as the result of analysis.
- **Sixth Chapter: INTERACTION DESIGN.** It describes the interaction design process carried out along project iterations, from the conception of the product to its first release, according to the user-centered design process.
- **Seventh Chapter: EVALUATION.** It describes the evaluation methods applied in the project
- **Eighth Chapter: RESULTS.** This chapter describes the final product obtained along with the project benefits after its usage.
- **Ninth Chapter: DISCUSSION.** This conclusive chapter is a critical analysis of the usability methods applied, the use of Usability Planner and other relevant issues around UCD and agile methods.
- **Tenth Chapter: CONCLUSIONS AND FUTURE LINES OF WORK.** This chapter exposes the conclusions once the project was finished and also proposes future lines of work.
- **APPENDIX A. TIMELINE OF APPLIED UCD METHODS PER ITERATION.** This appendix is a detailed timeline of the usability methods applied along iterations and the products per iteration.
- **APPENDIX B. INTERACTION DESIGN ITERATIONS FOR OTHER MODULES.** This appendix is a catalog that illustrates the evolution of the product design for other features not described in the document.

- **APPENDIX C. SUMMARY OF THE FIRST ETHNOGRAPHIC OBSERVATION.** This is the original document produced after the first ethnographic observation.
- **APPENDIX D. SOFTWARE ARCHITECTURE.** This appendix describes the resulting software architecture design.
- **APPENDIX E. SATISFACTION QUESTIONNAIRE.** This section shows the form template used to evaluate user satisfaction.

2 SELECTING USABILITY TECHNIQUES IN SOFTWARE PROJECTS

The application of user-centered design (UCD) in a particular project requires several considerations. Bevan [1] proposes a practical approach which consists of two parts: 1) the identification of the user-centered design activities that are actually required, and 2) the selection of the most appropriate methods based on the design and organizational context.

Regarding the first step, we need to explore the UCD activities and take from each category (Analysis, Design or Evaluation) the ones that are relevant in each moment of the project. These activities are listed below for reference.

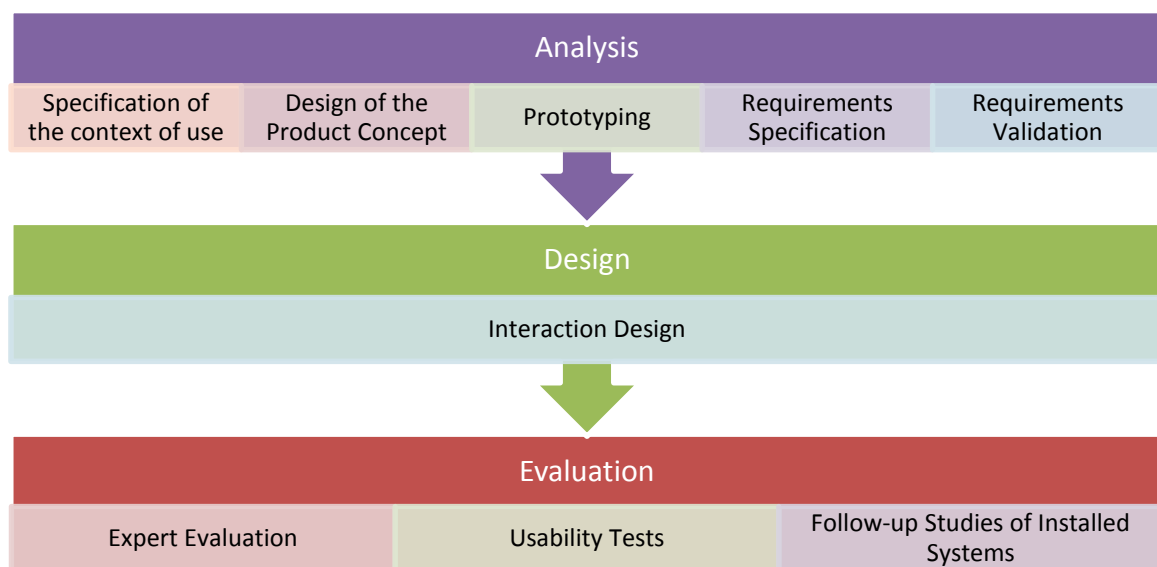


FIGURE 2. UCD categories and their activities

The decision depends on the actual needs. At the beginning of the project we will typically require more activities of analysis, while at advanced phases we will need to emphasize more on design and evaluation activities. This will constitute the usability plan.

Concerning the second part, the goal is to find adequate UCD methods for our project based on the context. For this, Bevan [1] suggests a straightforward approach, which is illustrated below.

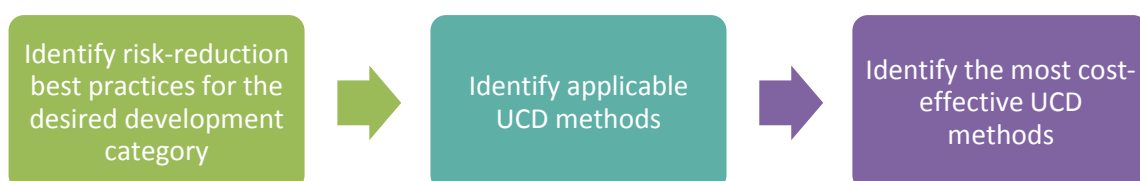


FIGURE 3. Steps for finding adequate UCD methods in a project according to [1].

:

First, we have to focus on a development activity category (i.e., Project Envisioning, Architecting Solutions, Evaluation, etc.), from which we take a sample of best practices aimed at reducing risk of poor product quality due to lack of usability. These best practices are defined in the ISO PAS 18152 standard.

Second, we must identify which UCD methods are applicable in the development category (i.e., Field studies and Observations can be applied during Evaluation). Moreover, we need to assess to what extent the methods achieve best practices.

Finally, we need to assess which UCD methods are the most cost-effective given the time and effort required and the project constraints. The last step, cost-effectiveness evaluation, must take into account at least the following factors [1]:

- a) **Constraints:** Time, cost, skills available, access to stakeholders and other users (refer to ISO-16982 standard).
- b) **The nature of the task:** Complexity, amount of training required, consequences of errors, time pressure.
- c) **The nature of the product:** Whether it is a new or an existing product, the product complexity, etc. (As defined in the ISO-16982 standard).
- d) **Context of use:** Range of contexts, how well understood is the product and the context.

A summary of the selection process is depicted in the following figure. The filtering of choices starts from the more general (i.e. select a development activity) to the concrete search (i.e. select the most cost-effective method).



FIGURE 4. Summary of the process for selecting UCD methods

For reference, we provide a summary of UCD method types, which will be referred along this document (taken also from [1]).

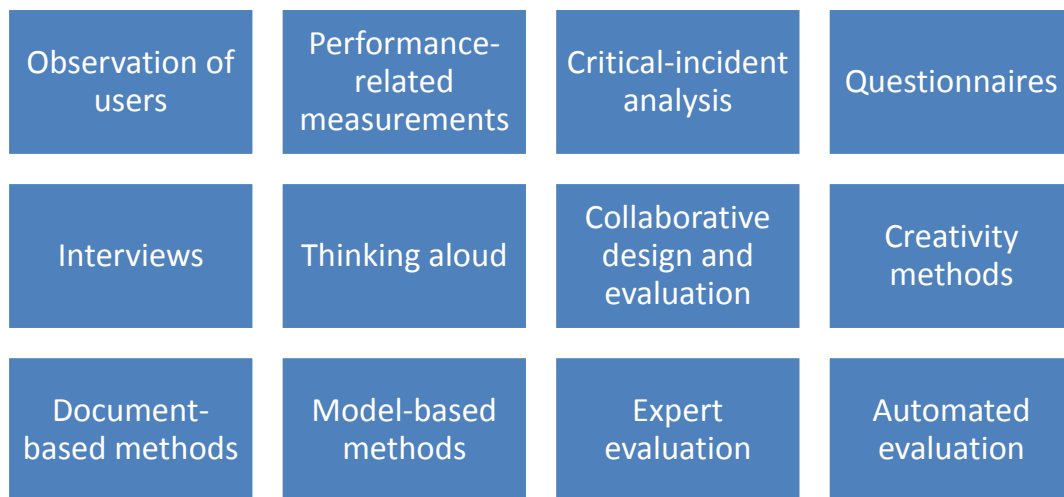


FIGURE 5. Examples of types of usability methods

In each method type, we can find several usability techniques. In the literature we find more than 40 usability techniques, which can be applied in a UCD project. Some examples of popular techniques are shown below:

- Participatory design
- Participatory workshop
- Performance measurement
- Personas
- User profiles
- Photo surveys
- Preliminary field visit
- Paper Prototypes
- Scenarios
- Storyboards
- Task analysis

2.1 USABILITY TECHNIQUES FOR AGILE PROJECTS

Silva [5] presents a systematic review of usability methods for agile projects. This study shows the most common recommendations of integrating usability into agile projects in literature from 2001 to mid of 2010.

A first quantitative analysis shows the usability artifacts used by frequency. This is shown in the figure below:

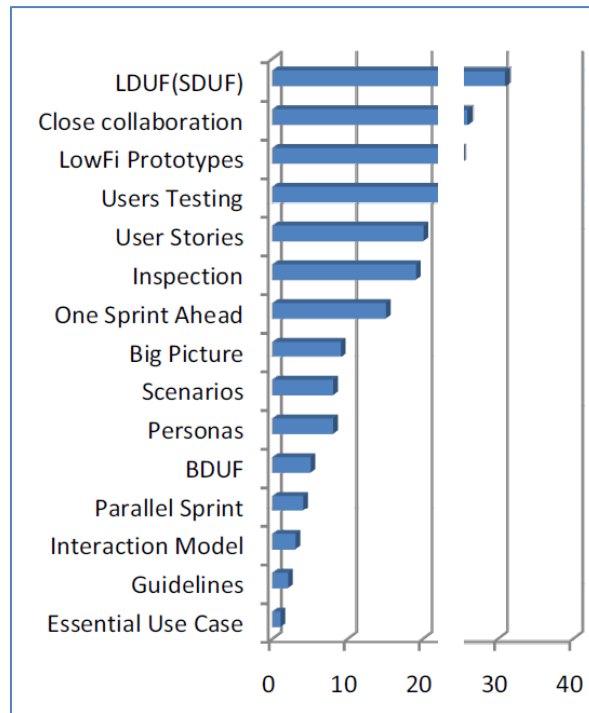


FIGURE 6. Common user-centered design artifacts applied in agile projects. Figure taken from [5].

This list mixes integration strategies, UCD methods, and positive results (e.g., Close collaboration), but it gives a good picture of what people is using and getting in practice.

For instance, the first is LDUF (SDUF) which corresponds to Little Design Up Front. It is mentioned by 30 studies that say that it is more suitable and compatible with agile methods than BDUF (Big Design Up Front). Other 15 studies say that projects should apply One Sprint Ahead integration.

On the other hand, the most recommended methods in decreasing order are: low-fidelity prototypes (25 studies), user testing (22 studies), and usability inspections (19 studies), and finally (with less than 10 studies each): Scenarios, Personas and Essential use cases.

Secondly, as a result of a qualitative analysis of the studies, the author suggests a strategy for integration user-centered design in agile projects. It is shown in the following picture.

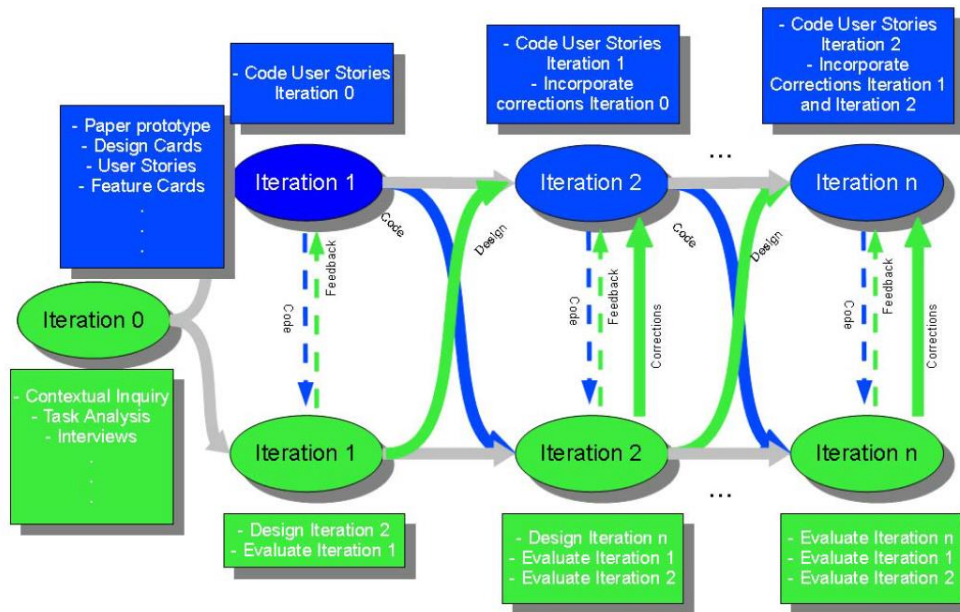


FIGURE 7. Suggested integration of user-centered design for agile projects. Figure taken from [5].

The suggestion corresponds to a generic usability plan and also involves the introduction of UCD roles and the use of a Sprint 0. The UCDS (user-centered design specialist) team shall comprise at least two roles: A *UCD researcher* and a *UCD prototyper*. The iterations are described as follows [5]:

Iteration 0: The activities involved are contextual inquiry, tasks analysis and interviews.

The methods are: Paper prototypes, Design Cards, Feature Cards, and User Stories with acceptance criteria with usability issues.

Iteration 1: Perform same analysis activities for the next iteration and also perform Inspection Evaluation on the code of the current iteration.

The methods are: Oral Storytelling (for getting feedback in the current sprint), Prototypes, Design Cards and User Stories for next iteration.

Iteration 2: Perform same analysis activities for the next iteration and also perform Inspection Evaluation on the code of the current iteration and User Testing on the Iteration 0's design that was coded in iteration 1.

The methods are: The same as before and Issue Cards to report problems of the code implemented in iteration 1 (designed in iteration 0).

Iteration N: Evaluate implementation with Inspection Evaluation on the code of the current iteration. And perform Inspection Evaluation and User Testing on the code of iteration N-1, which was designed on iteration N-2. And perform Inspection Evaluation and User Testing on the code of iteration N (designed on N-1).

The methods are: The same as before including Issue Cards to report problems.

In conclusion, the author provides a concrete set of recommendations suitable (or commonly applied) in agile projects, where big upfront design is not welcomed, and the preferred methods are the cheapest and easily-driven by developers (for instance, low-fi prototypes and inspections). However, richer methods that effectively involve users and their environments are left out. This of course will have consequences on the quality in use of the product.

2.2 USABILITY PLANNER

As a final thought, it is important to consider the cost of the usability methods selection in typical software development scenarios. Typically, there is no access to usability experts and the schedule is too tight to perform a research on usability method selection for the concrete project. The process does require a good understanding of the methods and their applicability on different contexts; otherwise we can end up with decisions that are out-of-context or beyond-budget.

For alleviating this, Bevan et al [2] developed an automated assistant tool which aims to reduce the overhead of manual searching and assessment during UCD method selection, by providing recommendations based on search criteria in terms of project constraints. We will describe the tool –called Usability Planner- when we present its usage in the project (see Chapter 4).

2.3 INTEGRATING UCD WITH AGILE METHODS

Integrating UCD with agile development has the potential to help developers involve customers and users, which is recognized as a difficult practice. And also it helps integrate HCI practices with software engineering [6].

Integrating UCD with Agile Development brings some challenges as both have similarities but also differences. The three main similarities according to [6] are:

- 1) They are ***iterative processes***
- 2) ***User involvement***: For instance, Scrum promotes user evaluation and user participation in sprint reviews, while in UCD this is one of the six founding principles.
- 3) ***Team coherence***: The full team should be united and have a common understanding of (and focus on) the project and the users.

On the other hand, the main differences are:

- 1) UCD advocates for building and maintaining design artefacts, while Agile methods encourages the reduction of design documentation overhead; and

- 2) UCD promotes a deep initial analysis and learning of users and their needs, while Agile methods discourages any big up-front research or design in favor of producing a working product sooner.

In practice, the integration of these two approaches requires the consideration of 5 relevant principles [6]:

- 1) **User involvement:** The user must be involved and assume specific roles in the team, such as having a proxy user in the team.
- 2) **Collaboration and culture:** Collaboration is a must among UCD designers, developers and users (which cannot be passive bystanders). Communication and joint work should be daily.
- 3) **Prototyping:** UCD designers should be willing to provide prototypes and user feedback to developers in a timely manner and in a cycle that is suitable for everyone (UCD prototypes are cheaper and quickly to build while code is more expensive and takes longer, so UCD designers should plan for the waiting time)
- 4) **Project lifecycle:** UCD designers should have ample time for the initial research of users and needs before the actual development start.
- 5) **Project management:** It is required a cohesive project management framework that avoids power conflicts between UCD and development roles and bureaucracy.

3 THE AX CERTIFICATION PROCESS

In this chapter we describe the business process that we want to assist, then we describe its issues and finally we analyze the effort required to users. First, before describing the actual process, we briefly explain why this process exists.

3.1 NCR AX MIDDLEWARE

In the past, ATM front-end applications were programmed by accessing directly device drivers. This meant that an application designed for a specific ATM was impossible to be deployed on an ATM of a different manufacturer.

To overcome this limitation, the CEN/XFS¹ standard was created, which allowed the access to financial devices through an intermediary layer providing a standard Application Programming Interface (API). Ideally, any application developed on top of the XFS standard will work on any ATM model. This is illustrated in the following figure.



FIGURE 8. a) With the XFS standard any app works in any ATM system. b) Reality of different XFS interpretations made by manufacturers²

However, in practice many incompatibility problems appeared, as there were subtle different interpretations of the standard by manufacturers, which is unacceptable in the ATM business sector.

As a response, NCR created a solution named the NCR ActiveX Middleware (AX Middleware) which provides a unified and simplified API on top of XFS to front-end applications. The main benefit of using this middleware is that the end-user application is fully independent of

¹ According to Wikipedia, CEN/XFS (eXtensions for Financial Services) is client-server architecture for financial applications on the Microsoft Windows platform, especially peripheral devices such as ATMs. It is an international standard promoted by the European Committee for Standardization (CEN). The standard is based on the WOSA Extensions for Financial Services or WOSA/XFS developed by Microsoft.

² Images taken from: www.atmmarketplace.com

the XFS layer, and therefore the differences among incompatible XFS interpretations don't affect it.



FIGURE 9. AX MIDDLEWARE

However, this solution has a cost: *Each single ATM must be certified to ensure AX is fully compatible.* This is the goal of the AX Certification process.

3.2 BUSINESS PROCESS MODEL

The AX Certification Process can be considered as a software compatibility testing process, where the AX platform is functionally tested against the underlying ATM XFS platform. The goal of the process is to guarantee that the AX platform works as expected on a specific ATM.

The process consists on the following core activities:



A detailed BPMN model of the AX Certification Process was elicited during analysis, which is shown in the figure below. Core activities are highlighted in green.

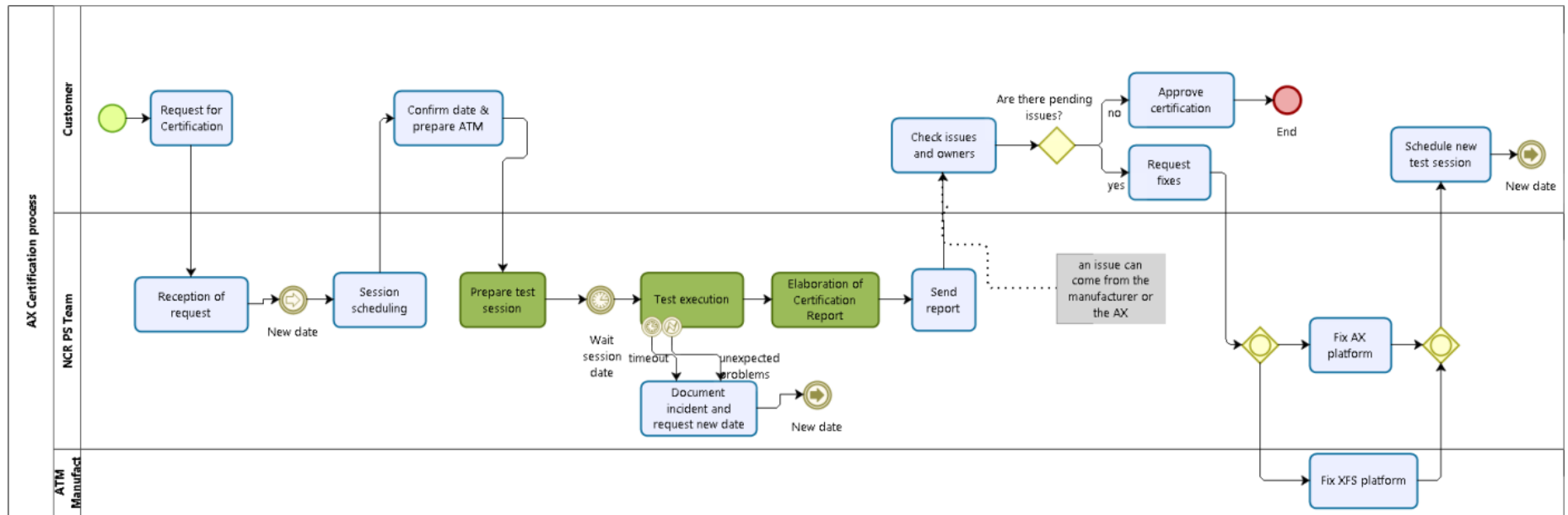


Figure 10 the AX Certification Process

3.3 ACTIVITIES AND PERFORMERS

The participants of the Certification Process are: Customer, Project Managers, Specialists, and third-party ATM manufacturers. The activities and the performers are described in the following table:

Activity	Description	Role
Certification request	The customer requests the certification of a non-NCR ATMs to NCR	Customer
Request reception	The project manager receives the request	Project Manager
Session scheduling	The project manager schedules the testing session with the customer	Project Manager
Test preparation	Identify hardware and software settings Get latest AX binaries Get latest documentation Prepare registry settings for ATM Prepare certification template for the specific ATM devices Prepare test resources (USB sticks, external input devices, laptop, tools)	Specialist
Test execution	Install tools and latest binaries Check ATM settings Execute all test cases (one by one) Document results & issues Collect runtime traces <i>(Occasionally : Develop fixes on the fly)</i>	Specialist
(In case of failure) Stop the session	In case of failure (of HW/SW) that impedes to start or continue the tests, the session must be stopped, and the partial results must be documented. A new date must be proposed.	Specialist
Elaboration of Certification Report	For each device: <ul style="list-style-type: none"> Review results and runtime traces Identify error causes and assign owners Fill the report template 	Specialist
Deliver report to customer	Deliver report to customer	Project Manager
Request fixes	The customer request the platform vendor to fix the problems found (written in the report). The vendor can be either NCR or a third party.	Customer
Fix platform errors	The owner of the platform fixes the error and delivers an update. This enables a new certification cycle.	NCR / Manufacturer
Close certification	Once the certification is approved by both the customer and NCR the process is finished.	Customer

Table 1. Main activities of the certification process

3.4 PROCESS ISSUES

The basic flow of the AX Certification Process seems quite clear and simple: test design, planning, execution, and reporting. However, as software testing process for financial devices it is required that activities be rigorous, systematic and standardized. But this is difficult to achieve with the current process as most of the activities are performed manually. That is, there are no software tools for supporting the process. This makes tasks be error prone, complex, and tedious for users. All these factors put in risk the quality delivered to customers.

Activity	Issues
Test Session Preparation	<ul style="list-style-type: none">• The procedures of test cases are not documented. Only the specialists know the exact procedures. This knowledge is not shared among certification team members• Specialists heavily depend on personal memory of previous sessions to perform the exact testing steps.• Besides the certification report template there are no standardized test scripts that can serve both as documentation and reusable artefacts for assisting to anyone who wants to make a certification.
Test Execution	<ul style="list-style-type: none">• Tests are performed manually, there are not specialized tools.• The test execution session takes place in the customer's premises (which usually requires travelling)• A single test session can span between 4 and 5 working days depending on the specialist and the number of devices• A certification can require a second or a third iteration of tests: And each iteration produces different data that need to be appended to the latest certification report
Report Elaboration	<ul style="list-style-type: none">• Report usually is done one or more weeks after the test execution. So results are difficult to remember, and finding error causes is cumbersome.• The report is about 140 pages, written by hand in a text document. If the test results were in a standard format, the report could be automatically generated.• The document is error prone as command results are manually written and inserted from several runtime traces.• The results obtained from different iterations need to be merged by hand.
Report Presentation	<ul style="list-style-type: none">• The document generated (PDF) does not provide navigation (nor bookmarks) which makes it hard to read.• Failed tests are hard to find and easy to miss by customers

Table 2. Certification process' issues

Moreover, specialists invest significant effort in manual activities that could be automated. This affects business efficiency and also employees' satisfaction. We identified the issues in each activity, which are discussed in the table above.

In summary, the AX Certification process requires important manual work that demands significant time and elevated costs for NCR and its customers. Moreover, most of them can be automated or assisted with software tools.

3.5 ANALYSIS OF EFFORT REQUIRED BY THE CURRENT PROCESS

The goal of this analysis is to get a realistic measure of the current effort required to perform a full certification as a baseline for measuring the benefits after introducing the new tool (which is discussed in Chapter 8 RESULTS).

Given that the main testing activities correspond to **test planning**, **test execution** and **test reporting**, we focus the performance measurement on these activities only. We analyze the effort currently required by the team by tracking the time required by each activity (in man-hours).

We take into account that a certification requires several testing sessions. Therefore we have a measure for the first, second, and third sessions (additional sessions are rare).

Moreover, the session effort depends on the complexity of the financial device being tested. So, we measured the effort required by three different devices with varying complexity:

- **A simple-to-test device:** Receipt printer
- **A medium-complexity device:** Cash dispenser
- **A complex device:** Bill acceptor

Finally, the effort required heavily depends on the user. Users are usually specialized in one or two financial devices but not in all. So the measures were asked to the specialist in each case. Therefore, we considered these measures as the “best case”.

3.5.1 TIME BY ACTIVITY (HOURS REQUIRED BY AN EXPERT)

The following table contains the information collected from users, about the effort required for certifying each device following the current procedures³.

³ These numbers do not take into account the time for fixing the errors found (except for very small fixes).

(*)The execution session can include partial generation of report and small AX fixes in-situ.

Activity	Device	Session 1	Session 2 (**)	Session 3	Average
Test preparation	Cash dispenser	3	0	0	2.5
	Bill acceptor	1.5	0	0	
	R. Printer	3	0	0	
Test execution (*)	Cash dispenser	8	4	2	10
	Bill acceptor	8	3	1	
	R. Printer	2	0.75	0.25 - 0.5	
Report elaboration	Cash dispenser	4	0.5	0.25	2.8
	Bill acceptor	1.5	0.5	0	
	R. Printer	1	0.25	0.25	
				Total	27

Table 3. Average effort required to certify one peripheral

In conclusion, the certification of a single ATM device made by an expert takes in average **15.3** man-hours.

Finally, having in mind that an ATM has typically between 7 and 9 financial devices (or even 10), a full certification can require between 107 and 153 man-hours. This is the average raw cost for certifying a single ATM of a single customer.

(**) Time of successive sessions is relative, since it depends on the number and kind of errors detected. It does not take the same time to verify an output state that to reproduce the faulty conditions.

4 SELECTION OF USABILITY METHODS IN THE PROJECT

This chapter describes the procedure we followed to select the methods for user-centered design. A quick characterization of the project is as follows:

- A bespoke software product is needed.
- The specification is clear as it depends on a mature and well-established process.
- The software is targeted to in-house users who have advanced technical knowledge.
- This is the first time user-centered design is introduced in the development environment.

According to one of his preliminary studies [7], Bevan establishes a guide for selecting usability methods, which he considers appropriate for commercial environments. There, he emphasizes that besides using general characteristics of projects for the selection process, it is required more guidance on the appropriateness of the methods in different contexts of use. For instance, he suggests that participatory design methods are much easier to apply for systems developed for in-house users. While some other methods are only required in specialized circumstances, such as focus groups, parallel design, wizard of Oz, remote evaluation, etc.

In our case, we opted for using a tool for getting a quick list of recommended usability methods for the project, based on its characteristics. And then we filtered out methods that were equivalent or similar to others, the methods that were not feasible, and finally before applying each method, we jointly discussed each method with the customer to approve its application. Despite all the steps done, this selection process was rather fluent and easy.

In this section we present the description of the project context which was the main input for the selection process. These factors correspond to constraints over the project, the users, the tasks, the product, the context and the team, as defined by the Usability Planner tool. Each one is analyzed in depth below.

Project Constraints

Budget: The budget was established before the beginning of the project and cannot be modified. The allocation of people to the project was also predefined. We consider the budget limited.

Time: The project duration is exactly 1 year, which is considered enough and allows the introduction of several UCD activities. If this time was not enough, the scope of the project could be reconsidered.

Usability importance: The system under development is a bespoke product which is targeted for assisting a core business process. This process requires technical expertise and high interaction with ATMs. The context of use is very specific and the results of tests cannot depend on the use conditions or the users. Inconsistencies in the use of the tool or in the information provided must be avoided.

Specification (un)certainty: The business process has a maturity of more than 15 years. So, the process is clearly defined and well-known by users. However, it does not imply that the process is formally specified in paper.

User Constraints

Access to users: Developers and users are collocated in the same working space. This is an invaluable advantage for having easy and frequent access to users. Moreover, users are willing to participate and get involved. They are about 8 people.

Impaired users: There are no functionally impaired users. And there are no explicit requirements for accessibility.

Habitual use: Habitual users will always be the PS team members. Currently, they are employees with at least 14 years of experience in the company. Therefore, the system is not required for first time users, but for experienced habitual users.

Task Constraints

Complexity of tasks: ATM testing requires a background in platforms, standards, vendors' compatibility, issue tracking, etc. User's knowledge will vastly impact his/her performance along the process activities and the use of the application. This can be considered as a complex process.

Quantity of tasks: The most complex activity (execution of tests) is typically composed by more than 100 test cases, which is full of details and need to be performed by the user one by one.

Safety or business critical tasks: The manipulation of the ATM through the new software does not produce risks either for users or the physical components. In business terms, it cannot produce losses or any other important harm derived from wrong usage. Therefore, the product is not considered safety or business critical.

Organizational changes: The Certification Process is not going to be altered by the introduction of the tool. The activities and the performers will continue being the same. The impact is in terms of efficiency and quality.

Product Constraints

Efficiency of use: The efficiency of the process is the ultimate business goal. The speed of the process determines its cost and the customer's perceived quality. Therefore, the efficiency of use is fundamental for the product's success.

Accuracy importance: During ATM testing the collection of data must be accurate and complete, as it will be the data for decision making, and it has a direct impact on the quality of the certification service delivered to customers.

Adaptation of an existing system: This is a fully new product and there are no previous systems with the same purpose. Although there are individual tools that are used separately and as such they are the baseline that the new system must supersede.

A well understood product: The process is mature and well defined. But the product has not been fully conceived and specified. However the use cases and the features are clearly understood. So this is not considered a constraint.

Customizable product: The product is for professional use only. It is not expected to be customizable in any sense. In the contrary is conceived as a standardized tool.

Context Constraints

General purpose: The tool purpose is very specific for ATM certifications. It is not going to be used for a variety purposes.

Team Constraints

Usability expertise availability: There is no access to usability experts. This is considered an important limitation. However, the development team has basic knowledge on interaction design is available.

In the following table we summarize the most relevant project characteristics and constraints.

Project Constraints
<i>Budget is restricted</i>
<i>Usability is important</i>
<i>Complex tasks</i>
<i>Many tasks</i>
<i>Efficiency and accuracy are important</i>
<i>A well understood product</i>
<i>No usability expertise available</i>

Table 1. Constraints of the ActiveX Certify project

4.1 METHODS SELECTION WITH USABILITY PLANNER

The selection of methods was performed with different criteria. One important step was the use of a tool for getting recommendations of methods for our particular project. And then, after an additional filtering, we proposed a sample of the methods to the client to decide its application and make a plan.

We used Usability Planner, which is a tool developed by Nigel Bevan together with a team from the Technique University of Madrid (UPM) [2, 3, 4]. The tool follows a practical 3-step process which is pretty straightforward. This is shown below.

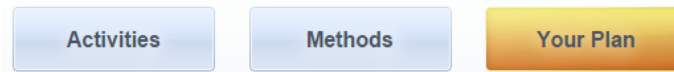


Figure 11 Method selection menu of Usability Planner

4.1.1 ACTIVITIES

This step consists in the selection of the user-centered design activities to be performed. In our case, we selected the following activities:

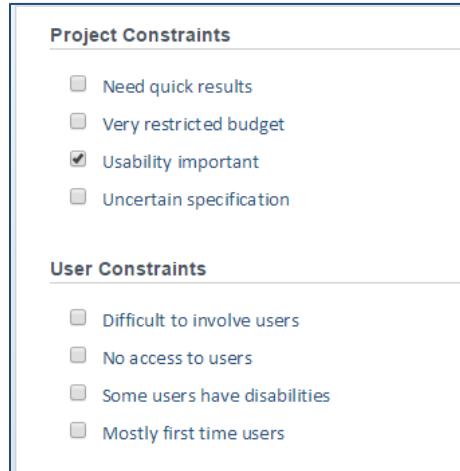
- Analysis:
 - Specification of the context of use
 - Design of the product concept
 - Prototyping
 - Requirements specification
- Design:
 - Interaction design
- Evaluation:
 - Usability tests
 - Expert evaluation

Basically we will not perform the following activities:

- Strict usability requirements specification and validation: Because the development will not be outsourced, compliance will not be verified by a third-party. Moreover, the team is very small (3 people) and resources are very limited.
- Follow-up studies of installed systems: Because there are no previous systems in use. And on the other hand, this project will finish with the system deployment.

4.1.2 METHOD SELECTION

This step consists in filling the search criteria to get instantaneous suggestions. A fragment of the search criteria offered by the tool is shown in the figure below.



Project Constraints

- ☐ Need quick results
- ☐ Very restricted budget
- ☒ Usability important
- ☐ Uncertain specification

User Constraints

- ☐ Difficult to involve users
- ☐ No access to users
- ☐ Some users have disabilities
- ☐ Mostly first time users

Figure 3. A fragment of the search criteria offered by Usability Planner

Applying the project constraints discussed previously, the tool produced a list of 23 usability methods. The methods were ranked according to the scale:

- Strongly recommended
- Recommended
- Slightly recommended
- Neutral
- Not recommended.

Due to the vast list of suggestions, we took only the strongly recommended methods. From these suggestions we filtered the methods that were redundant or equivalent to other methods. This selection is explained in the following tables.

Methods for Analysis

Sub activity	Method	Why
Specification of the context of use	Ethnographic observation	Selected because its effectiveness and viability (visits to real certifications are allowed). The information is collected from the source, and it is good for discovering tacit information.
	Competitor analysis	There are no competitors in the market; however, there are lots of similar products (for software

	Affinity diagramming / Participatory workshops	testing) that can be inspiring. (Not selected) because they require several people at the same time, which is not viable for the company.
	Contextual inquiry / Field study	By selecting Ethnographic observation we discarded these methods, which share the same objectives.
	Essential use cases	Not selected. Instead a definition of standard use cases is planned with users.
Design of the Product Concept	Scenarios	Selected because is good and practical for devising desired product features.
	Braindrawing	This method requires from 5 to 12 participants. So it was not considered.
	Parallel design	This was not selected because it also requires several people working at the same time.
Prototyping	Paper prototyping	Selected because it is familiar, cheap (in time) and effective.
	Wireframe	Selected to be applied, although with less frequency than prototyping.
Requirements Specification	Usability specifications	Selected as an integral part of requirements elicitation.
	User & task analysis [added]	Informal description and analysis of user profiles and tasks through the analysis of the business process (tasks vs. assignees, tasks and resources required, etc.)
	Common industry specification for usability requirements	Not considered because the development is not outsourced. And strict requirements compliance validation is not required
	Function allocation / Task allocation	Task analysis is performed through other methods (above).

4.1.3 METHODS FOR DESIGN

Sub activity	Method	Why
Interaction Design	Navigation map	Selected for the navigation analysis and design for selected functionalities.
	Participatory design	This method requires several participants in long sessions, which is difficult to justify and approve in a department with few people and tight schedules. So it was not considered.

4.1.4 METHODS FOR EVALUATION

Sub activity	Method	Why
Usability Tests	Laboratory usability testing	Selected as the primary evaluation method because of its popularity and because NCR has a dedicated laboratory for testing with easy access for employees. And the lab and the users are collocated in the same building.
	Thinking aloud	Selected as a secondary evaluation method, if the time allows its application.
	Performance measurement	Selected as a secondary evaluation method, if the time allows its application.
Expert Evaluation	Heuristic evaluation	Selected for being widely used and accepted, and because it's quick and practical. However, we acknowledge our lack of access to experts. The method is selected as a complementary method to laboratory testing (according to availability of time).

In conclusion, we selected 12 methods from the suggestions offered by the tool:

- Analysis: 7
- Interaction design: 1
- Evaluation: 3 (being two of them subject to time availability)

These methods are shown in the figure below.

Analysis	Interaction Design	Evaluation
<ul style="list-style-type: none"> • Specification of the context of use <ul style="list-style-type: none"> • Competitor analysis • Ethnographic observation • Design of the Product Concept <ul style="list-style-type: none"> • Scenarios • Prototyping <ul style="list-style-type: none"> • Paper prototyping • Wireframe • Requirements Specification <ul style="list-style-type: none"> • Usability specifications • User & task analysis 	<ul style="list-style-type: none"> • Navigation map 	<ul style="list-style-type: none"> • Expert Evaluation <ul style="list-style-type: none"> • Heuristic evaluation • Usability tests <ul style="list-style-type: none"> • Laboratory usability testing • Thinking aloud* • Performance measurement*

FIGURE 12 List of selected usability methods (*optional methods)

4.2 USABILITY PLAN

After method selection, we designed a usability plan. This plan is not the scheduling of activities as the project is developed iteratively (according to agile development). So iterations cannot be planned in advance. Therefore, our usability plan will consist on establishing when UCD activities can be performed according to expected project milestones.

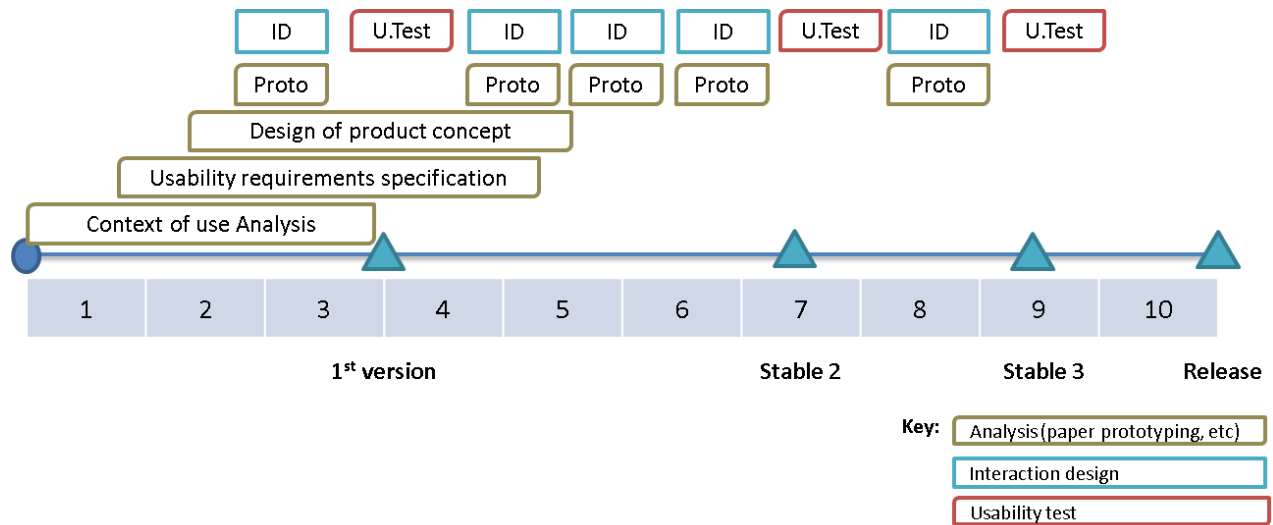


FIGURE 13. Project usability plan

Having in mind that the project has an established duration of 8 months and iterations will be short (two weeks in average), then we can assume that the project will have at least 10 iterations. Second, the company wants to have at least one release at the middle of the project. So, we can plan for two releases before finishing the project (milestones 1 and 2) and a final release (milestone 3).

So the usability plan basically consists on assigning UCD activities along the iterations (as shown in the figure above):

- **First Version:** The first milestone (a third of the project) will be the most intensive in UCD analysis, but they are not limited to that, as it is expected a working software for iteration 3.
- Then, a usability test will be performed.
- **Stable version 2:** The second milestone will consist mainly in development through prototyping, wireframes, interaction design, and coding along several iterations.
- A second usability test needs to be performed.
- **Stable version 3:** The third milestone will fix the usability problems identified and produce the last functional software features.
- A final usability test will be performed in order to find usability issues before producing a final release.
- **Release:** This is the final version, which can be used for measuring the project benefits in a complete certification process.

5 SPECIFICATION OF THE CONTEXT OF USE

In this chapter we describe the results of the analysis of the context of use. This is an important activity in the user-centered design as it is foundation for other analysis and design activities: some usability requirements are derived from the context of use, and usability evaluation must exhibit the fundamental conditions found in the real context of use [8].

The user's analysis, task analysis and the context of use specification in general were done mainly through observations of Certifications in the lab and in real Certifications at the company's customer facilities. These findings were then refined through several users meetings. This analysis was considered finished on iteration 12 (that is more than 50% of the project time). A detailed description of the UCD methods involved is shown in Chapter 6.

5.1 USER PROFILES

Several people participate in a certification process, such as the company's customers, project managers, specialists and ATM manufacturers' representatives. However, the users directly involved in the testing activities are the certification specialists from the company division for software support. Currently, there are about 5 specialists, although only three of them are in charge of certifications. The general user profile is the folioing:

- **Users population:** 5 concrete people
- **Location:** All users are allocated in the same building and office
- **Job title:** All users are consultants at the software support division
- **Profession:** All users are software professionals with at least 10 years of experience in software development and support activities.
- **Requirements:** The requirements for preparing, running, and reporting tests are applicable for all the users (there is no distinction of tasks between user profiles).

Moreover, the users have different specialties. There is a user that certifies several devices, but in general each user is expert in one or two devices. A sample mapping of users (specialists) and devices (specialties) is shown below (only 4 devices are shown):

	Cash dispenser	Note acceptor	Receipt printer	Pin Keypad
Specialist 1	X	X		
Specialist 2			X	X
Specialist 3				X

Table 4. Sample mapping of users (specialists) and devices (specialties)

Therefore, we defined two different user profiles: Junior and Senior. Below, we summarize the user profiles, which are very concrete and particular in our project:

Junior User Profile <ul style="list-style-type: none">• A user that is expert in the certification process• A user that certifies a specific device for the first time (or in very few occasions)
Senior User Profile <ul style="list-style-type: none">• A user that is expert in the certification process• A user that certifies the device of his or her specialty

Table 5. User profiles

5.2 USER TASKS

Task analysis consists on getting a task organization model that describes user tasks and practices in the business process. The methodology to obtain the users tasks according to [9] is:

- 1) Obtain information about the background of the work that is going to be automated.
- 2) Collect and analyze data through contextual observations of real users doing their job in their work environment.
- 3) Build a task organization model of the current users tasks

These steps can be done in several iterations, for instance, the first iteration can be to perform the contextual observations, the second iteration can be the documentation and analysis of the work environment, in the third iteration we can build tasks scenarios, and a final iteration can be the documentation of the task analysis.

Some recommendations for performing task analysis are [9]:

- The construction of a formal model for each observation can be cumbersome and unworthy, while textual descriptions are usually richer than a specific predefined template that could be hard to interpret.
- It is not recommended to perform lots of observations in a row, instead it is better to stop to digest and analyze the findings of each one.
- In small projects with only one user type (or profile), 3 to 6 contextual observations – distributed in 1 or 2 iterations- can be enough. While in bigger projects with many different actors, 15 or more observations are usual.

- Regarding the participants, select people who are well-known for their best practices. As they are experts, they work with efficiency and effectiveness. So they are the most suitable for describing what they do by using principles and abstractions.
- Finally, it is also important to also involve less-experienced people, to know their perspective of the tasks. But this is not recommended at the beginning.

In the case of the Certification Process, we have performed two ethnographical observations of real certifications (they are described in Chapter 6), and elaborated textual descriptions (one of them is presented in APPENDIX C. SUMMARY OF THE FIRST ETHNOGRAPHIC OBSERVATION). Finally, we build a schema of the activities with their tasks and the place where they are performed. This schema is presented in the figure below. We focused the analysis on the main activities of the process: **test planning**, **test execution** and **reporting**.

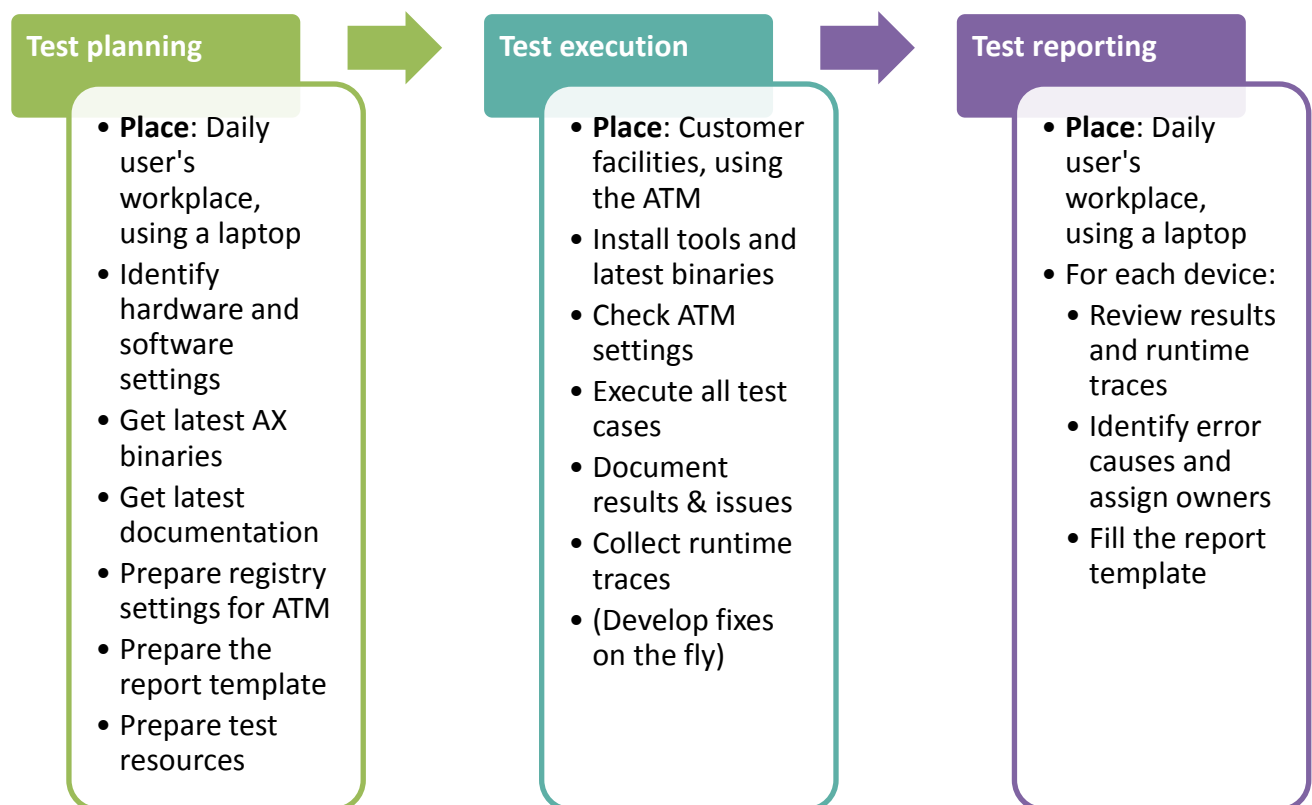


FIGURE 14. Activities and tasks of the Certification Process

The activities where automation is more relevant and critical are Test Execution and Reporting. So we describe them in more detail.

5.2.1 TEST EXECUTION TASKS

The test execution activity is the most complex and critical activity in the process. The tasks that compose a test session are illustrated in the task model below:

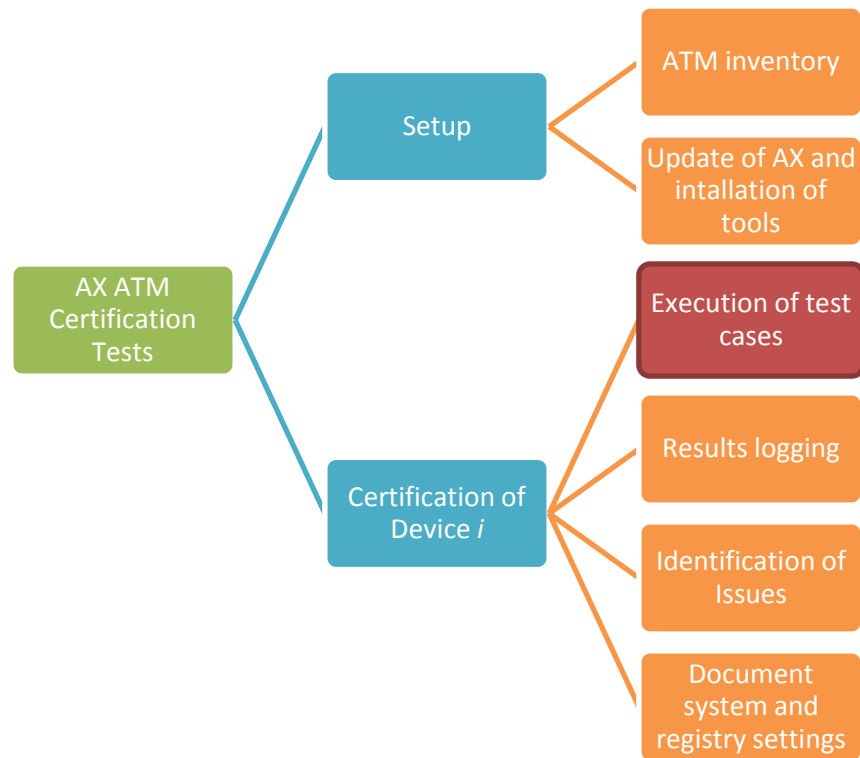


FIGURE 15. Task model of the Test Execution activity (the most important task in red)

The task model defines a setup step which is done once, and also a generic task that can be performed as many times as the number of financial devices installed in the ATM. Standard ATMs involves from 5 to 9 financial devices. Moreover, all tasks are performed manually though all them could be assisted or automated with software.

The most important task is Execution of Test Cases. Each device requires a test suite which tests all the commands and events of the device in its different states. A typical test suite can comprise up to 100 combinations of device functions and device states. And all of them need to be verified with test cases. Each test case is composed by four parts:

- A. Setting the initial device state: It requires the execution of several commands and user actions (depending on the device and the test case).
- B. Running the target command (the command under test)
- C. Make assertions: The comparison of results against the expected output (baseline)
- D. Documentation: Each test case result must be documented for the Certification Report

Finally, when the session is over, the user collects the runtime traces with the results (copying them out of the ATM).

It is worthy to mention that the test session can be repeated up to three times (in different dates): Each time a software fix is applied by its manufacturer (as it was described in Chapter 3).

Finally, a quick and dirty computation of the number of tasks performed along the test execution activity is:

$$\left(\begin{array}{c} 5-9 \\ \text{devices} \end{array} \right) \times \left(\begin{array}{c} 100 \\ \text{test cases} \end{array} \right) \times \left(\begin{array}{c} 1-3 \\ \text{sessions} \end{array} \right)$$

Thus, for the simplest case an ATM requires the execution of 500 test cases, while in the worst case it requires 2700 test cases. Moreover, the specialist usually needs to run the test case more than once when a test case fails, because the user must be sure that the procedure was done correctly before reporting any failure. These factors make the number of tests higher.

5.2.2 TEST REPORTING TASKS

The elaboration of the Certification report consists on the following tasks:

Collection of results. This information can be taken either directly from the screen or from the runtime traces written by the testing tool. Usually, the user takes notes directly from the screen if the result is simple and common. Otherwise, it will tag the test case with a timestamp which will allow him to find the particular result in the logging file in the future.

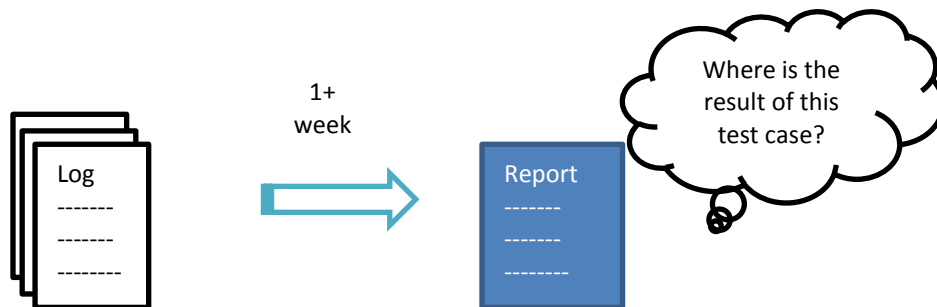


FIGURE 16. Report elaboration out of runtime traces and issues

Report elaboration. The user uses a report template to elaborate the report. The template contains the list of test cases describing: The test case description, the obtained result, the expected result, the status (either pass or fail) and comments. The user must fill all the fields with the information collected. The user normally starts filling the template while is executing the tests, but this is a task that need to be finished afterwards when he is back to office.

Moreover, the elaboration of the report could be delayed or postponed depending on the user's workload for one week or more.

Assigning issue owners and commenting results. Once the report is filled-in with results, the user checks the failed test cases to be sure that all of them have a comment explaining the errors and also identifies the owner of the issue. The owner of the issue can be either the company or the third-party ATM manufacturer.

Report delivery. The report is converted to PDF and delivered to the customer.

Report versioning and merging. In the cases where the certification has been repeated, a final activity is required to compile and merge the previous certification reports with the new partial results into a single final certification report.

5.3 PHYSICAL ENVIRONMENT

This section describes the physical context of use of the test execution activity, which is always performed in the customer's facilities, where the ATM is installed. The place usually corresponds to building basements and ATM maintenance laboratories. The people involved in this activity are the tests specialist, the customer staff, and the ATM manufacturer staff. And test resources include USB sticks (with software tools), external keyboard and mouse, a laptop and financial test material. The following figure depicts a typical testing scenario.



FIGURE 17. Context of use and typical tools used during the testing session

The ATM is usually prepared for testing with the housing uncovered for easy access to USB devices ports. A PC keyboard is always available but there is no room to place it. Typing needs

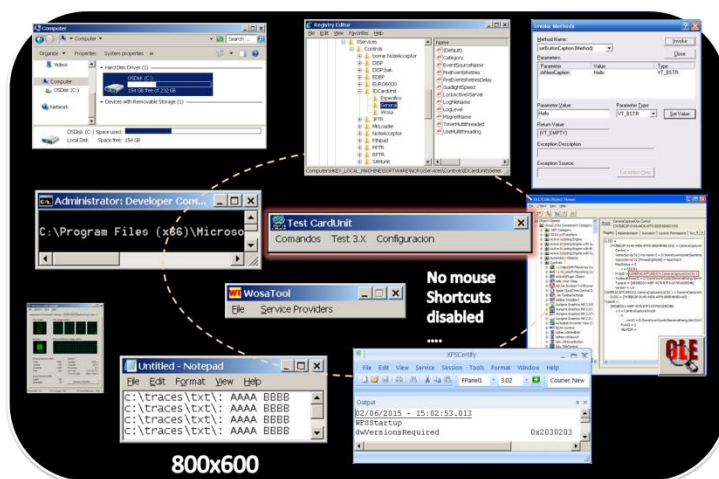
to be performed on top of the ATM. Moreover, the ATM casing doesn't allow chairs or tables in front of it, so the user typically works standing. Other resources like the user's laptop, cards, and the external mouse must be located on a separate table or desktop. So, switching between the laptop and the ATM screens is periodical.

The physical space tends to be uncomfortable for certain tasks, like taking detailed notes, exchanging files, reading large logging traces in a small screen, asking for help to remote coworkers, etc.

The resources a tester has are limited by host's facilities. Usually, phones are available but WI-FI spots, power outlets, tables and chairs are not (or they are far away the ATM).

The time frame allocated for testing is usually between 4 to 5 full days. This slot is enough in average, but unexpected conditions (e.g. a broken device) can lead to session cancellation and rescheduling.

Finally, users typically use between 4 and 9 different software tools or utilities. The typical appearance of the ATM desktop and the laptop's is shown in the figure below.



a) Typical ATM desktop (a single window actually fits the whole screen)



b) Typical laptop desktop (for editing the report template)

FIGURE 18. Typical desktop appearance during the testing session (screenshots of the applications used).

In summary, ergonomics is compromised by the physical conditions and the combination of the navigation by keyboard, the small screen size, and the low screen resolution produce a negative user experience which affects user's productivity during the long test sessions. All this suggests that an integrated software solution is definitely required.

5.4 ATM TECHNICAL RESTRICTIONS

The development of a tool for an ATM needs to meet additional requirements in comparison to standard desktop applications.

First, an application targeted for ATM testing should be multiplatform. Customers have a broad range of brands and models that can be 15 years old to versions released few months ago. This means that the application runtime requirements cannot be too high in terms of both hardware resources and operating system versions. About 5% of the ATM installed base works on Windows 7, while the remaining works on older versions. ATM installations older than 15 years and operating systems previous to Windows XP are considered out of the scope of the project.

5.4.1 HARDWARE RESTRICTIONS

Input devices: The main input device is the standard PC keyboard. Pointing devices are not always available (like mouse and touch screens).

Touch screens: Standard Windows applications are not usable with ATM touch screens, as the precision of these displays is too low for the standard (small) icons and controls. Moreover, privacy filters make the screen harder to use (deeper and less precise). Finally, display sizes are small (about 14").

Connectivity and data transfer: The only enabled input/output media are USB ports. Usually, they are so limited that it is required to disconnect some devices (e.g. mouse) to insert memory sticks. ATMs have network connectivity but are strictly reserved to customer's applications.

5.4.2 SOFTWARE RESTRICTIONS

Compatibility and software installation: Operating system settings cannot be modified and software installation is forbidden. Testing applications need to be multiplatform and portable (i.e., no need of installation). If the application relies on a framework or virtual machine, it must be fully compatible with the expected operating system versions (Windows version, Service Packs installed, settings, etc.).

Screen resolution: Typical screen resolution is low (800x600 pixels) and operating system's visual styles are disabled. Development settings shall take this in mind to avoid unexpected user interface behavior once the application is deployed on ATMs.

Network communications: Network access is restricted by the system administrator.

Error handling: ATMs have special error handling policies which provokes immediate system reboots whenever an application halts. This can produce data and time loss for users if a reboot occurs during a testing session. An auto-save mechanism can be considered.

5.5 PRELIMINARY CONCLUSIONS

Given that ATMs impose high restrictions on user interaction, the efficiency, effectiveness and satisfaction of use are compromised and very limited. In this environment, productivity can be improved by reducing the dependency of external resources (like documentation, report templates, and ultimately technical knowledge), by unifying the separate software tools, and by automating tasks that does not require user intervention (such as results logging, collection of system information, etc.).

An early strategy to overcome this quickly and cheap was proposed without success. Using the client-server architecture the user could control the ATM server application from a separate device (a laptop or tablet). This way, the user could enjoy a richer user experience provided by a mobile application without losing the direct physical interaction with the ATM. This option was discarded as network connectivity is disallowed for security reasons (as mentioned before).

So the solution necessarily needed to be a native GUI desktop application running on the ATM. On the other hand, an alternate solution could have been a command-line interface application, which is naturally optimized for keyboard interaction and does not demand complex runtime dependencies. However, the customer wanted to be able not only to execute test cases but also to allow edition and reporting in the same application. This suggested that a GUI application was the most suitable alternative for this project.

6 INTERACTION DESIGN PROCESS

In this chapter we describe how we applied user-centered design through the development process. Along this retrospective we will see how the most business-critical activity: **Certification Tests Execution**, required greater effort through more iterations of design and evaluation, as its usability requirements were higher.

The project required 23 iterations along 9 months. The most important project milestones were the following:

- **Milestone 1:** Implementation of a proof of concept for running Test Suites
- **Milestone 2:** First stable application version (edition and execution of test suites for 2 ATM devices)
- **Milestone 3:** Second stable application version (new execution model, reports, and support for 8 ATM devices)
- **Milestone 4:** Final application version (improved user experience and support for 10 devices)

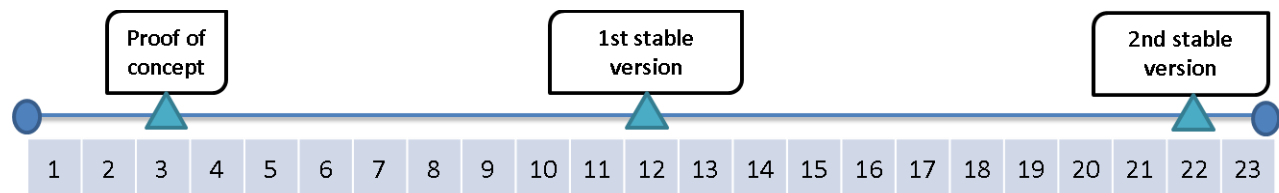


FIGURE 19. Project iterations and milestones

User-centered design was incorporated into the development process from the very beginning, and their activities were performed in parallel to software development tasks. The way of carrying out both UCD and development was through the **One Sprint Ahead** strategy. In which each new product feature is studied and designed in the current iteration, then implemented in the next one, and evaluated later on. However it was not a strict sequence as many UI designs were implemented several iterations later.

Given the high number of iterations, we describe the interaction design process grouping them in project milestones.

6.1 MILESTONE 1: INCEPTION AND PROOF OF CONCEPT

The project started from an idea for automating/assisting the AX Certification tasks. Before the project, there were tools used by users: A set of executable applications for invoking AX commands, and a set of standalone web pages (VBScript and ActiveX) for invoking AX

commands. However, they were used in isolation for specific tasks and just by few users. Thus, the project goal was defined as creating a solution from scratch that could assist all users in the whole process.

6.1.1 ANALYSIS OF THE AX CERTIFICATION PROCESS AND TRAINING ON ATMs PLATFORMS

This first iteration is considered the inception of the project which consisted on the analysis of the business process and the introduction to technical background of ATMs platforms. The business process was described in terms of: Activities, People, Information (entities and concepts), Resources (tools) and sequence of flow. The result of this analysis was presented in Chapter 3. The activities performed in this iteration are the following:

- **Meetings.** A specialist explained the concepts behind ATM software platforms and their architecture through several meetings.
- **Learning sessions in the lab.** A specialist explained the AX Certification in the ATM laboratory through examples of real certification tests on the ATM. Two sessions with different specialists were performed.
- **Reading of reference material.** We read different material for getting detailed information about the AX platform, the underlying XFS specification, and also real examples of Certification Reports, which is the outcome of the Certification Process.

The sessions in the laboratory were not contextual observations as the specialist was not performing the real work and he was also continuously interrupted during the process. However, the kind of information obtained was similar: user tasks, context of use, functional requirements, etc. So we elaborated a description of the session with these observations. The following figure shows a snapshot of the first report.

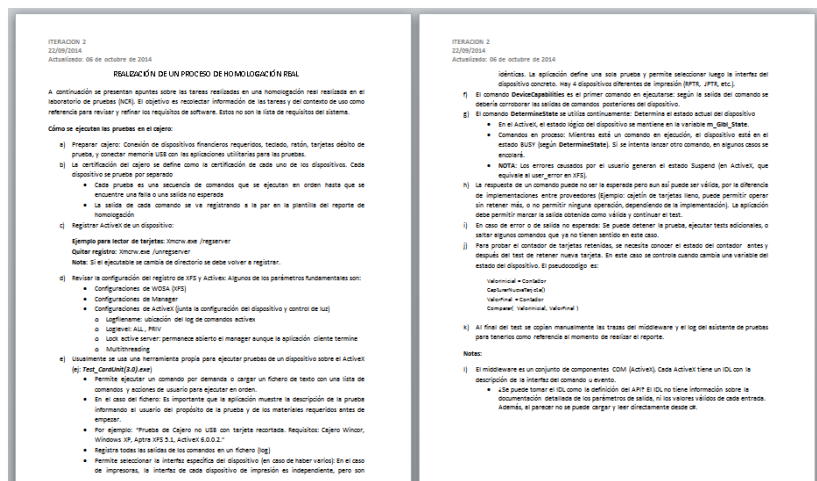


FIGURE 20. Picture of the observations in the laboratory learning sessions (a 2-page report)

6.1.2 PRODUCT CONCEPT

The first description of the product proposed by users in early meetings was: *“We need something similar to **XFS CERTIFY** which fulfills a similar goal for testing the XFS layer; it allows the execution of commands and allows testing several devices at the same time”*. A screenshot of the product is shown below:

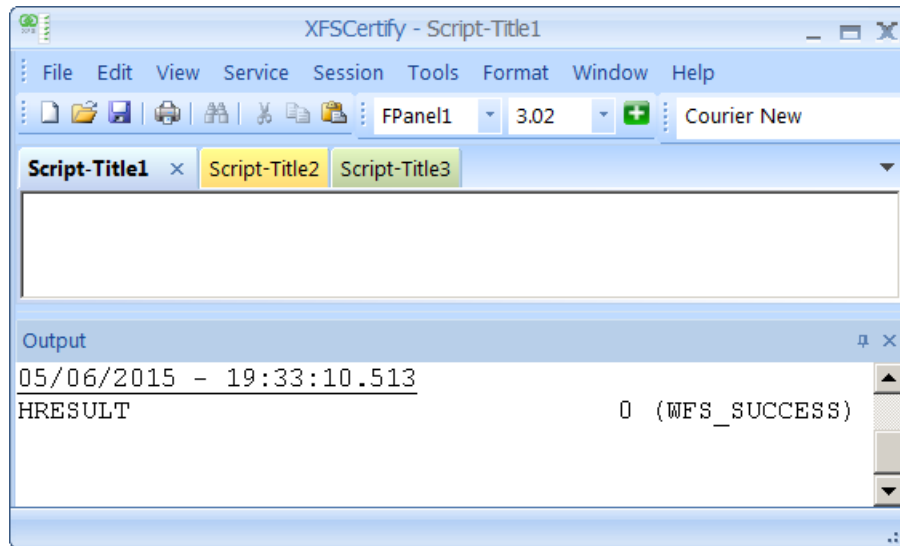


FIGURE 21. A product referenced by users as an example (rights reserved)

However, users soon presented a clear description of the product concept: *“The device certification is a sequence of test, where each test needs a situation (or device state), the execution of a command, and the verification of the result based on a baseline. Tests change the state of the device.”*

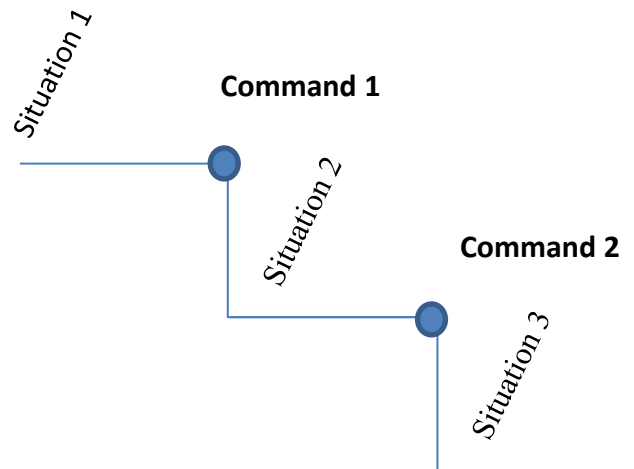


FIGURE 22. Concept of a device certification test

A further refinement to this idea was obtained through successive meetings with users: “*The certification requires the definition of a series of steps that should execute sequentially to achieve the desired device state and the command under test. These steps can be of three types: AX Commands, AX Events, and user actions*”. This idea is illustrated in the following *paper prototype*, which again, was proposed by users.

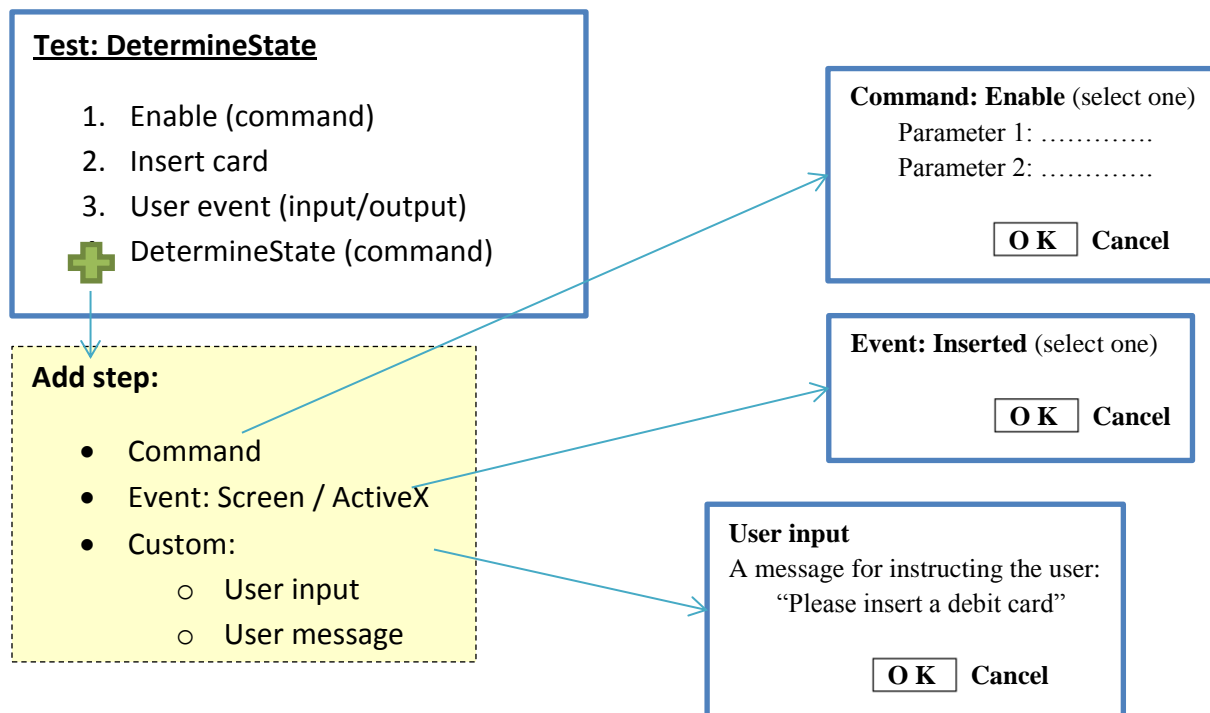


FIGURE 23. A Certification Test Script (paper prototype proposed by users)

6.1.3 DEFINITION OF MAIN PRODUCT FEATURES

From use cases and the initial product concept we depicted the general features the product should provide:

Tests design: The design of tests consist on describing all the steps that must be executed to certify a device or an entire ATM. Each step can execute an action or prompt the user to put the device in a situation. The general features of the editor are: create a test, save to a file and open a test from a file. The editor can be used in a PC (design time) or on an ATM.

Test execution: Execution of a test consists of the sequential execution of each step in the test set. Execution can be in two modes: All tests of the device or stepper. Results are checked according to expected values and execution creates execution traces (logs). Test execution is performed on the ATM only.

Results review: This consists on checking that all the steps were executed properly, adding comments to explain incidents found and identifying causes of failures. This can be done in the user's PC or even on the ATM.

Report generation: Once the test results are available, a user will be able to generate the certification report. This consists on the results of each test and the revision comments. The scenario in which the report is made is the user's PC.

6.1.4 BRINGING CONCEPTS FROM SOFTWARE TESTING DISCIPLINE

The overall definition of the product is encompassed in the discipline of Software Testing. So we considered appropriate to refer to some of its concepts: Test Suites and Test Cases, along with assertions and expected results.

In this way, we mapped the definition of a device certification as a test suite. While a test case was mapped as the unit test of a device function. The situation required by each test case was defined in terms of tests steps (actions) that make up a test case. Finally, we defined four types of actions: commands, events, and user actions (used for asking user intervention).

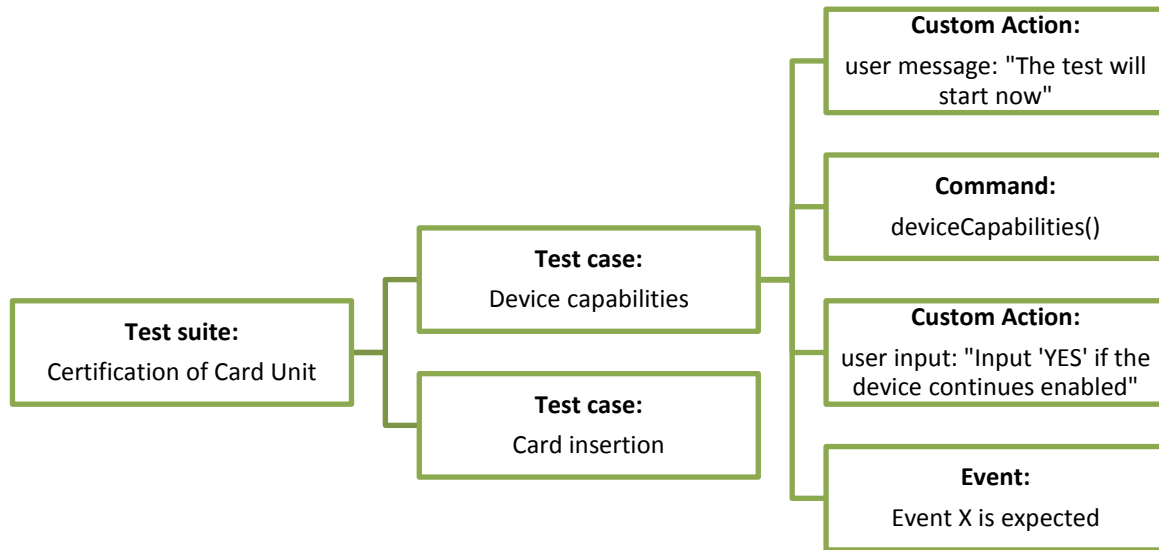


FIGURE 24. Application of software testing terms to define a certification test

Test suites and test cases are containers. Commands and events correspond to valid functions of the device API (ActiveX). Custom actions can be any of the following types:

- User Message: Displays a text message to the user to communicate something in a test point.
- User Input: Requests user's inputs such as comments.

At this point, a natural decision was to start searching for testing applications on the market.

6.1.1 FIRST RESEARCH ON SIMILAR PRODUCTS (COMPETITOR ANALYSIS)

First we reviewed *Selenium IDE for Firefox*, which is useful for web applications testing. A screenshot is shown below:

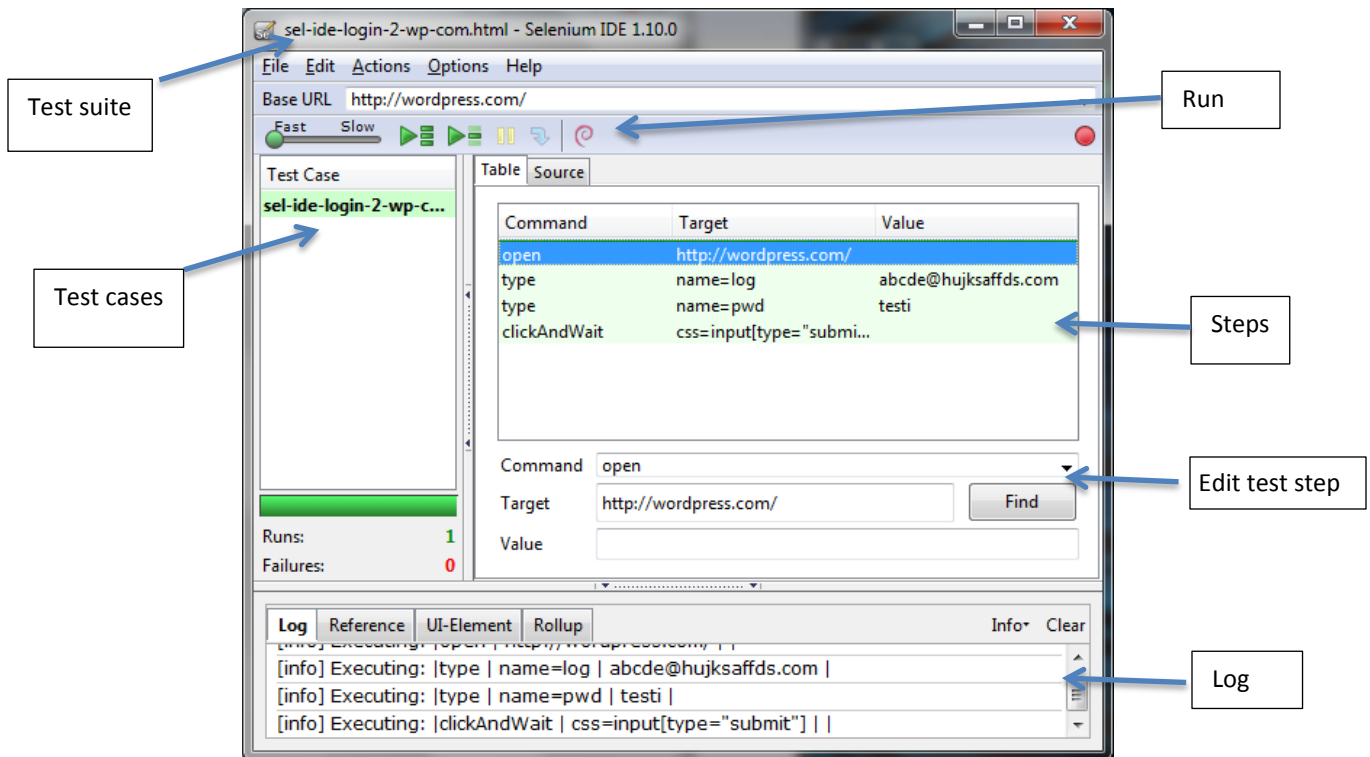


FIGURE 25. A reference application for product concept design (The Selenium IDE)

This application gave us a first idea on the basic layout of the user interface (UI) and the basic application features. They were basically:

- **Tests design:** Capabilities for edition of test suites, tests cases and test steps
- **Execution:** Execution of a test suite, a test case, or a test step.
- **Logging:** The recording of results and events.
- **Reference:** Display of API documentation for the referred actions.

6.1.2 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

The initial set of functional requirements was derived from the product features, the use cases, and the overall product concept. They first requirements specified were the following:

- Execution of individual commands:** The user shall be able to run individual AX commands at any time. The application shall ask for the arguments through a graphical user interface showing the names and descriptions of each parameter. The results shall be presented in a detailed view with the name, description and value of each output parameter.
- Modeling of test suites and test cases:** The application shall allow the user to define a test suite with its corresponding test cases, where the test case shall have a description and the containing test steps.

- C. Modeling of test steps:** The test steps in a test case can be any of the following types (the SRS document contains full details):
- Command step
 - Event step
 - User input
 - User message
- D. Logging:** all the actions (test steps) executed by the application shall automatically be logged in a text file.
- E. Keyboard accessibility:** The application UI shall provide access to functions through the keyboard.
- F. Error handling:** Any error produced in an action or in the application shall be properly managed in order to avoid data loss by a system reboot (according to the ATM settings).
- G. Security:** The application shall request a login name and password to allow its use. The application shall allow the configuration of users through a settings file.

6.1.3 FIRST UI DESIGN (WIREFRAME)

A following step after having defined the product concept was the design of the graphical user interface (GUI). Through this design (see the figure below) visibility is given to the following items:

- Current test suite (name)
- ATM device under test
- Execution toolbar
- Key functions: open and save test suite

The interface allows us to view on a single screen the exact status of the execution of the test (current Test Suite, selected Test Case, and current Test Step).

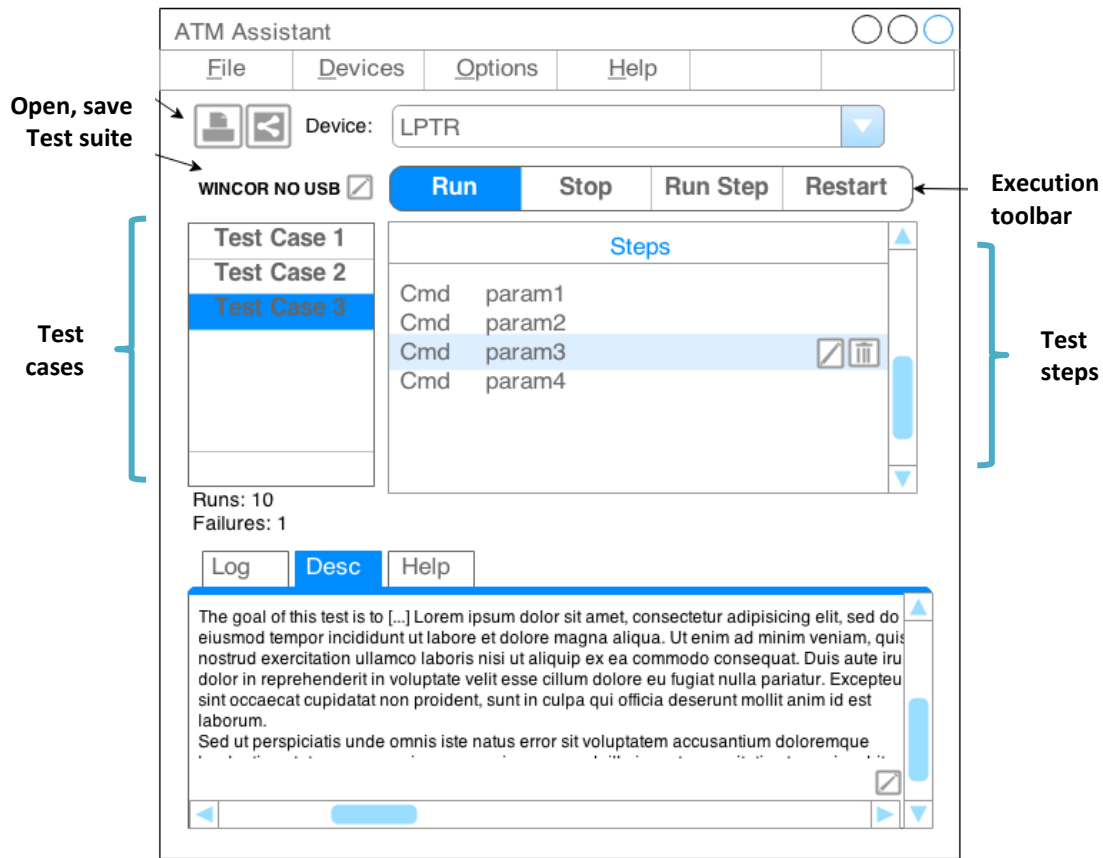


FIGURE 26 First wireframe

Lower tabbed panels allow us to view the execution log, test suite description (written by the user who designed the test), and the help panel (which could render content of the user manual). Finally, the main menu (File, Devices, Options, and Help) is entirely optional or could be relegated for user settings.

6.1.4 FIRST PROOF OF CONCEPT

During this milestone, the construction was focused on creating a proof of concept to touch the technology involved (*.NET, Windows Forms and ActiveX components*) and also to validate the main software architecture design, as the base for designing the whole product. The architecture is described in APPENDIX D. SOFTWARE ARCHITECTURE. Its goals were primarily to ease the modeling and reuse of Test Suites, and the design of an extensible catalog of ATM devices, which allows users to enhance the supported devices without software coding by using high-level models in JSON format.

The proof of concept was actually able to open, load and execute Test Suite models specified in a JSON document. A screenshot is shown below.

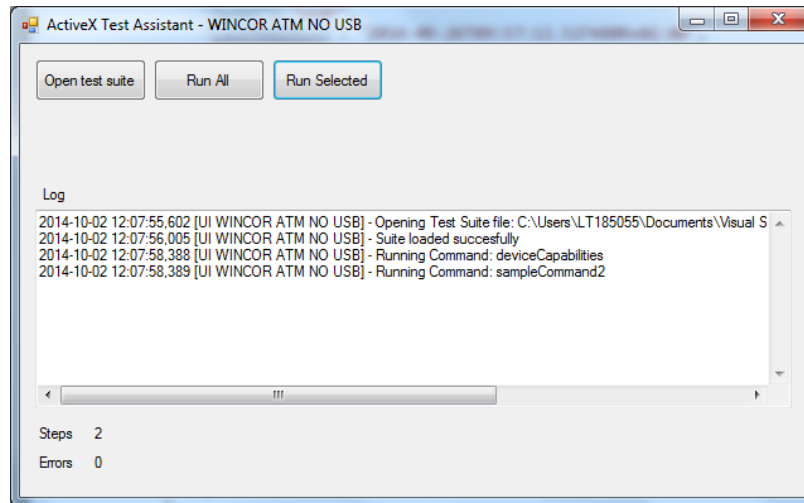


FIGURE 27. First proof of concept in .NET (Windows Forms)

6.1.1 SELECTING THE PRODUCT NAME

The first and final name of the product was defined by users based on the convention used in a tool referred in the conception of the product. The tool is called XFS Certify and allowed the testing of XFS commands. So it was natural for users to choose **ActiveX Certify** for our project, which clearly identifies the purpose of the product: To certify the ActiveX platform (AX) in a particular ATM.

6.1.2 EVALUATION

Along these iterations, we confide on the agile good practice of showing the products to users as soon as they were ready to get early feedback. So we presented the draft of the analysis document, the UI wireframe and the executable proof of concept. After the feedback we produced new versions of the analysis document; the wireframe was accepted as is; and the proof of concept worked as expected, which was enough to validate the product understanding and to foresee the goals for the next milestone.

6.2 MILESTONE 2: FIRST STABLE VERSION

The first milestone finished with an initial specification of functional requirements, usability requirements and the first validation of the architecture. The goal of the Milestone 2 was to implement the wireframe to produce a working version capable of running test cases on at least two ATM's devices. The scope of this milestone left out the generation of the Certification report to the Milestone 3.

The activities performed in Milestone 2 corresponded to an additional ethnographic observation, a new research in related software (competitor's analysis), the implementation of the first functional version, and two usability tests.

6.2.1 ETHNOGRAPHIC OBSERVATIONS

In order to get richer information of the business process, the context of use and to validate the current specification of requirements we needed to observe how a real certification was performed by specialists in the field. Fortunately, at the time there were two customers that required the certification service so it was possible to apply this method early in the implementation process, before the first stable version was finished.

The first observation performed was a 5-hour session whose goal was the certification of the PIN keypad for a well-known customer located in the same city. In this session, only one specialist and one observer (the developer) participated. The place of the session was the basement of the building, where a laboratory for ATMs maintenance is located. The activities carried out were:

- 1) Observation by taking notes (during the session)
- 2) Elaboration of report with the findings (after the session)
- 3) Presentation of the findings to users (to validate them and getting feedback)
- 4) Conclusion (the requirements and the analysis documents were updated accordingly)

The results of the observation were 8 new requirements, the information of the physical environment, and the description of user tasks (all these results are part of the previous chapter, see Chapter 5). The most important facts were:

- There are new kinds of parameters in inputs forms that need to be supported (multiple selection and default values). This was a new functional requirement.
- The session was fully performed with the keyboard (no mouse was available).
- The ATM user desktop was composed by at least 6 separate applications used at the same time for each test (see details in the Appendix).
- It is very frequent the reading of logs (which need to be searched in the file system and opened each time a change is made).
- The user did not follow the order of the certification report to run the tests. Instead he groups tests with the same pre-requisites, so it is not required to repeat the tests.
- A summary of each result is written by hand in the report document but the details are taken afterwards from the logs. A problem here is that the order of execution of tests is different to the order in the report, so the user must to search and find each specific result in the traces.

- Using Word (in a separate laptop) adds problems as the user needs to switch between machines. Moreover, the laptop performance adds important delays to the process.

Finally, the full report can be found in the APPENDIX C. SUMMARY OF THE FIRST ETHNOGRAPHIC OBSERVATION

The second ethnographic observation was scheduled several days later. The session was carried out for the same customer but for a different ATM and device. The goal was to certify the Card Reader. The participants were also the same user and the developer and the allocated time was 4 hours. We performed the same activities as in the previous observation: Taking notes during the session and analyzing the results afterwards. The results were as follows:

- The context of use was basically the same as specified previously
- In particular one of the test was repeated about 10 times in order to verify a boundary condition (number of maximum cards allowed). This simple test required a lot of time, it was suggested that a kind of “macro” or “template” or “reusable test case” could be useful as the input parameters were always the same and the number of repetitions is indeterminate (depends on the ATM model).

In conclusion, this session gave us important information though it was not as fruitful as the first one.

6.2.1 SECOND RESEARCH ON SIMILAR PRODUCTS (COMPETITOR ANALYSIS)

Regarding early visual and interaction design we started looking similar applications in the market. The product analyzed was Microsoft Test Manager (see image below) which is an online suite of tools for professional software testing (mainly but not limited to web applications). The picture below illustrates one of its UI screens for test suite edition.

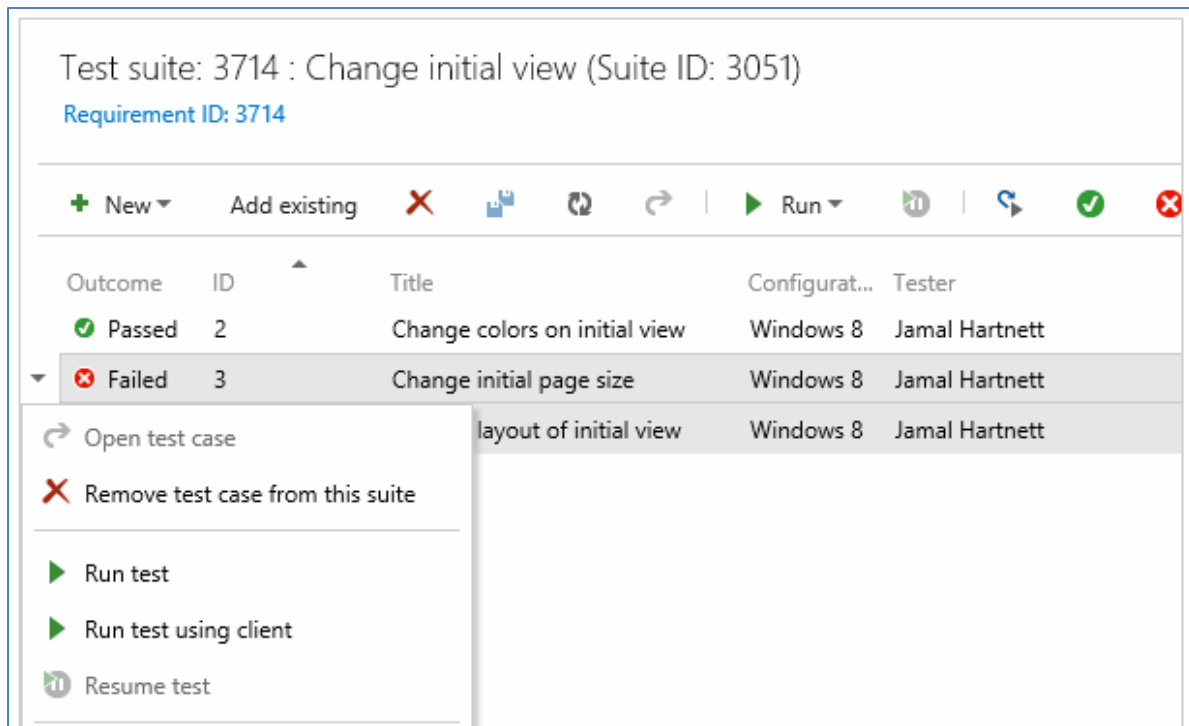


FIGURE 1. Microsoft Test Manager & Visual Studio Online (test suite edition).

We analyzed the terms used, the edition and execution functions and keyboard shortcuts. Here we have the results:

Edition view

- It uses the same terms: Test Suite, Test case, and Test step.
- The presentation is clear: test suite title, description, and list of test cases.
- Editing tools and contextual menu: supports edition and execution options.

Execution view

In this view, several elements are observed:

- Navigation of test cases (previous and next)
- Background information: Current test case and current step
- Presentation of all steps in the test case numbered (including non-executed)
- Presentation of the results of each step: pass and fail icons.
- The panel is reduced to make room for the application being tested.

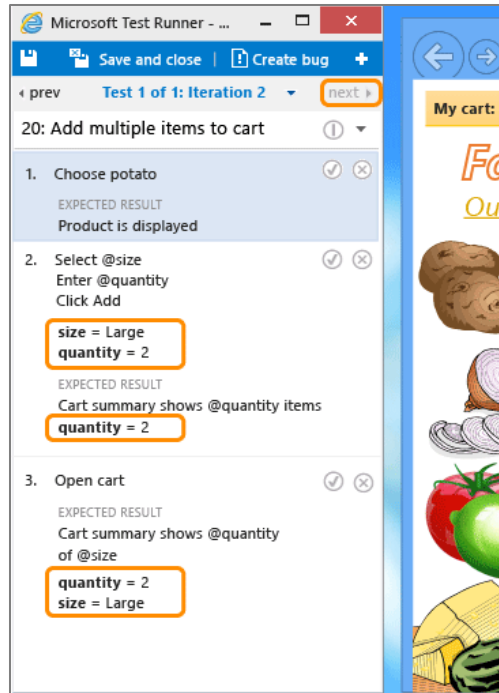


FIGURE 2. Microsoft Online Test. The execution view (left side) and a sample application being tested (right)

6.2.2 CONSTRUCTION OF THE FIRST APPLICATION VERSION

Following the main ideas of the wireframe we started the first implementation. However, we quickly devised two issues:

- 1) The prototype proposed a single view for edition, execution and the results. This made the UI simple to understand, but given that actions could be executed individually, it would not allow tracing the results in historical order. Moreover, providing both edition and execution functions in the same view seemed more complex to manage.
- 2) The wireframe did not include a way to display the results of actions, besides the logging view which shows unstructured text.

Therefore, we decided to divide the UI into two views: **Edition** and **Results**. This idea also matches with the GUI of the commercial product seen in the previous section.

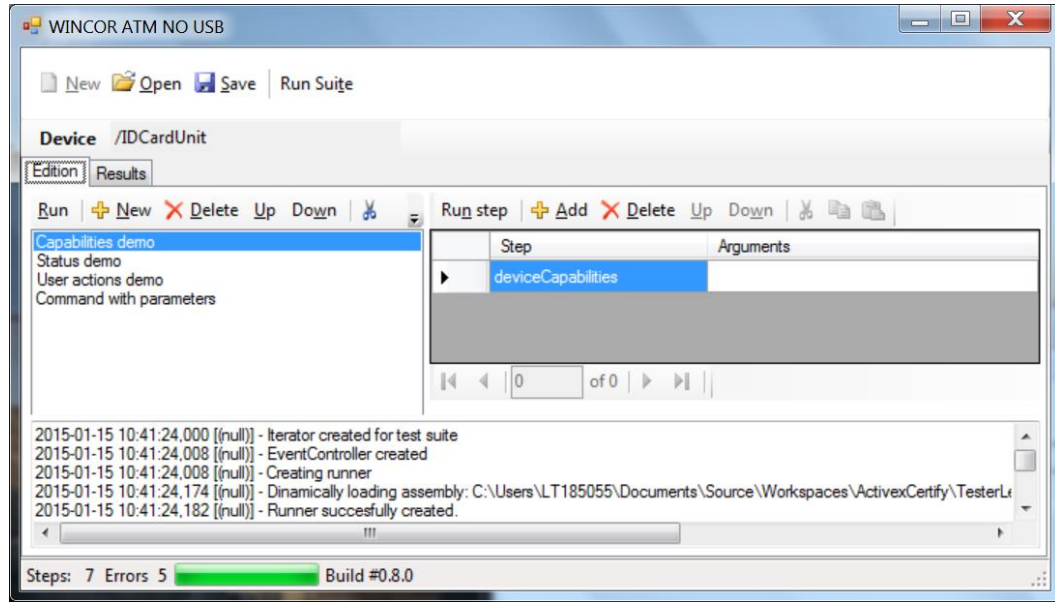
The implementation was done in C# using Windows Forms. The first implementations were pretty similar among them, so we describe only the resulting stable implementation of the design (corresponding to the result of iteration 7 and 8).

Test suite

Test cases

Main toolbar

Test steps



Results

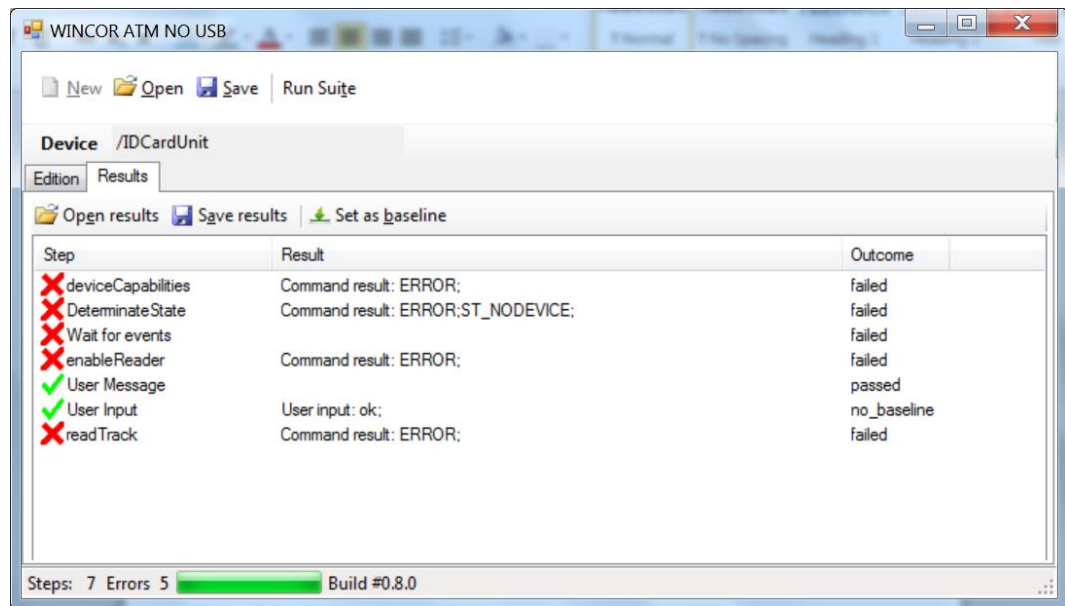


FIGURE 28 Edition and execution views (from iteration 4 to 8)

As the prototype did not include the Results view, the design was decided during the implementation. The presentation of results was defined as a tabular form showing the result of the action (command result) and the outcome of the test (either pass or fail). This window worked like a graphical logging window, i.e., it showed the results of actions in historical order (independently of test cases and where new results were always appended). The basic idea was simple and was liked by users.

The execution functions allowed were: Run Suite, Run Test Case, and Run Test Step. However, they were located only in the Edition View (except for the Run suite function).

6.2.3 FIRST USABILITY TEST

The first test was scheduled in the ATM laboratory with one of the users –a specialist with experience in the certification of several ATM’s devices (the test is detailed in Chapter 7). This test was also the first application deployment on a real ATM, because until now the development and tests were carried out on an ATM simulator.

The preparation of the session resulted in lot of unexpected results which affected the stability and the usability of the application, which produced big delays. So instead of cancelling the test session, we opted by performing a user exploration session, where the user took the control and started performing usual tasks while talking aloud to the evaluator. There were no previous training besides the verbal explanation of the general layout of the window and the main supported functions. The user started running commands (steps), looking at the results, trying again, etc. The idea was to get an initial feedback and perception of the current version.

At the end of the session, the user summarized the experience as positive but also suggested a list of concrete new requirements most of them being functional (we don’t put it here for the sake of brevity). The main issues observed by the evaluator were the following:

- The application shall inform the user when the AX service is not correctly installed and running (otherwise the application will fail silently).
- Background and font color contrast in some GUI controls were inadequate (either invisible or hard to read).
- Using the keyboard (controlling focus) was challenging (some toolbars have no shortcuts or keyboard access, and other functions had the same access key).
- Some GUI forms were not adapted correctly to screen size and resolution (forms too big).
- Stacked modal popups must to be avoided (some modal windows were locking the whole interface and were behind other windows difficult to reach with the keyboard).

The last suggestion was to avoid the use modal pop-ups, as this is one of the most usability common problems the users have experienced in the past while using windows applications in the ATM without a mouse pointer. Popups were used to inform the user about the arrival of AX events. Several implementation alternatives were reviewed: auto-hiding balloons, stacked messages in the results table, and status bars. The last option was selected because of its low cost and the time restrictions, and also because the standard UI controls available in Windows Forms did not match the kind of control required.

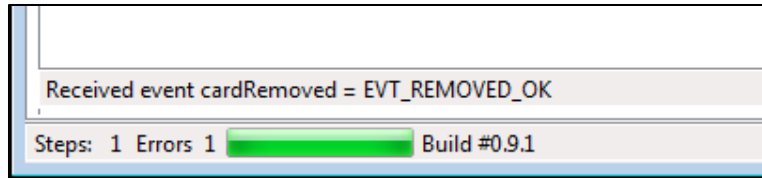


FIGURE 29. Iteration 8: Modal popups are replaced by a new execution status bar

The session findings were collected and discussed afterwards with the full team to decide on its implementation. The implementation of the fixes was done immediately, although many of them were almost unnoticeable in the external appearance of the application.

6.2.4 DESIGN CONCEPT OF THE COMMAND RUNNER

The findings of the usability test helped not only to improve the existing features, but also were taken into account for designing the new product features. The Command runner is a feature that allows the execution of commands at any time, as a shell console. It corresponds to the first requirement. In this milestone we designed the first concept of the feature using a wireframe (see the figure below).

The wireframe was presented to users and they provided the following feedback:

- The wireframe is attractive and simple which is good
- The idea of using templates for predefined sequence of actions seems useful but it is not currently required.
- It is desirable to be able to manage different devices at the same time (e.g., using tabs), as it is provided by the XFS Certify product.

Finally, the implementation of the feature was scheduled for the next milestone.

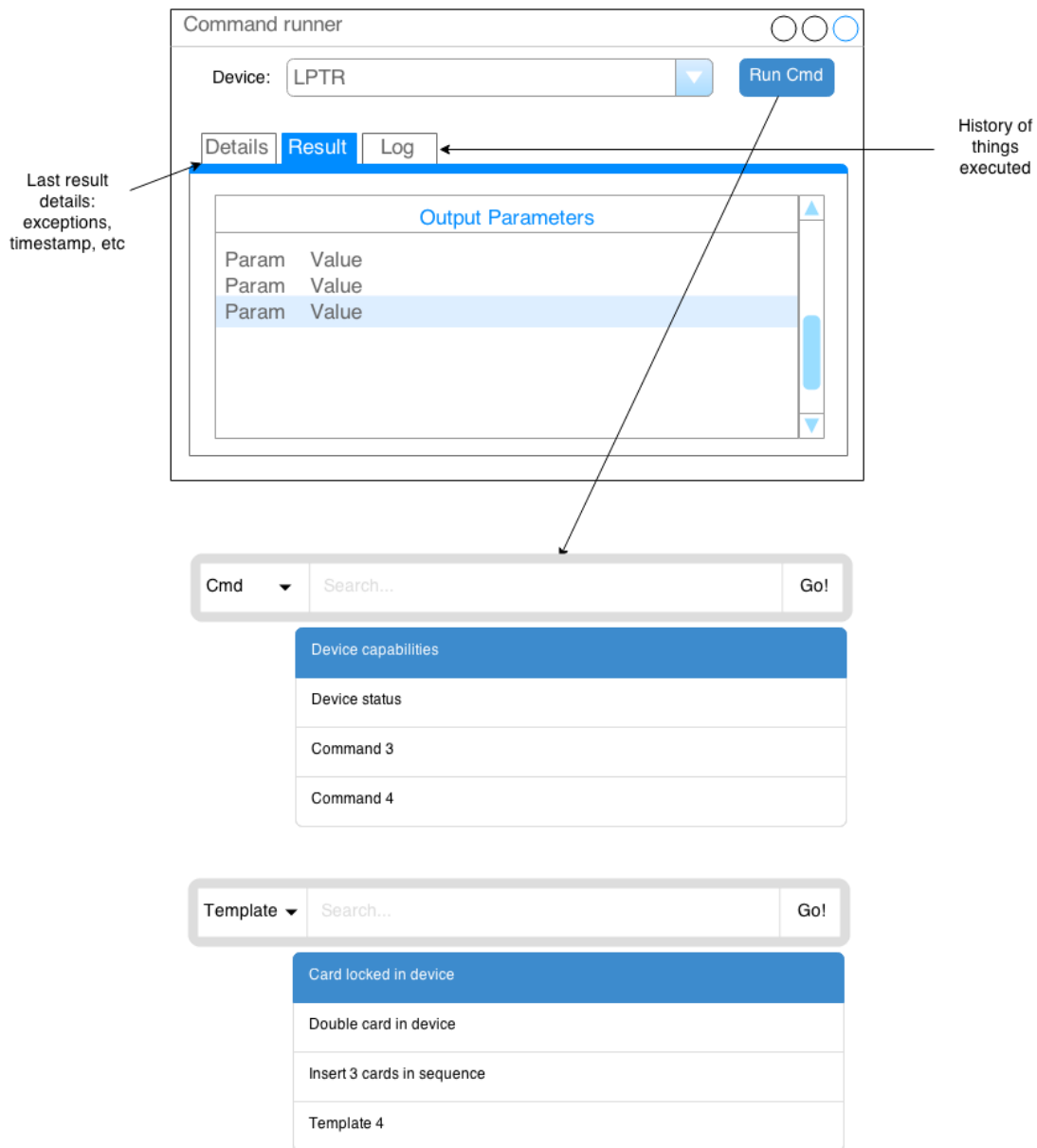


FIGURE 30. Wireframe of a new product feature: The Command Runner

6.2.5 SECOND USABILITY TEST

The second test was performed about two weeks after the first one. The participant invited was a different user with experience in the certification process and also part of the project team. However, this user had neither seen nor used the application before. The details of the test are also described in Chapter 7. The activities consisted on the execution of an existing test suite and the creation of a new test suite for the Card Reader unit in the ATM laboratory. The test was split in two sessions because of other user's commitments.

The result of questionnaire for user satisfaction was very high in all aspects although the number of user errors and suggested improvements was 14. The main findings were the following:

- The newly added start screen is too small and is unnecessary
- The name given to the test suite is not used for naming the file
- It is not clear which panel has the focus
- The result details view has no close button (though shortcuts exist to close the window)
- The comparison between expected and obtained result is not shown item by item (it is not practical to look at the text comparison and then go to the detailed table to find the position of the mismatch). This is shown in the next section.
- The “Run” button must be renamed to “Run test case” to avoid confusion with “Run suite” or “Run Step”.
- The logging text is not easy to read because there are no separators between test steps.
- The “Open Results” function was only found with help.

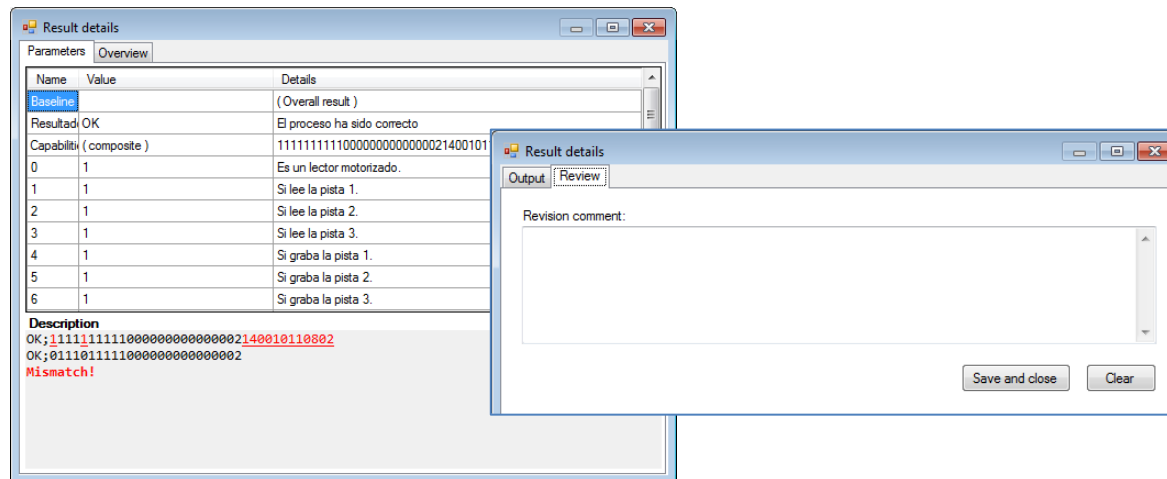
6.2.6 IMPROVEMENTS AFTER THE USABILITY TEST

The implementation of changes was done in several iterations afterwards. The first improvement was done in the Result Details window. The main changes were the following:

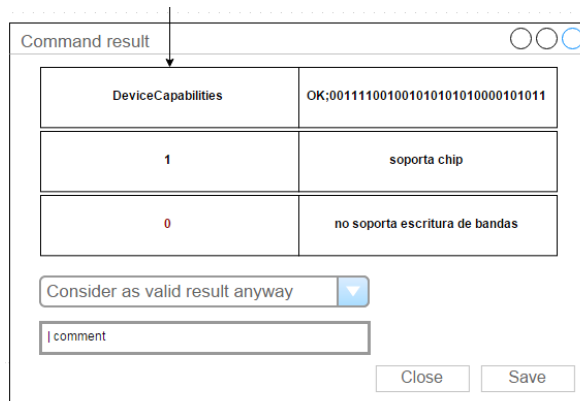
- Simplify the view by removing tabs: The revision options were unified into the main view.
- Explicit save and close buttons were added.
- Comparison of results item by item: colors and symbols were added to highlight the meaning of the comparison. The last image (Iteration 17) corresponds to Milestone 3 to get an overview of the evolution (moreover, the main improvement was in Milestone 2).

In the images below we show the comparison of the window before and after the usability test.

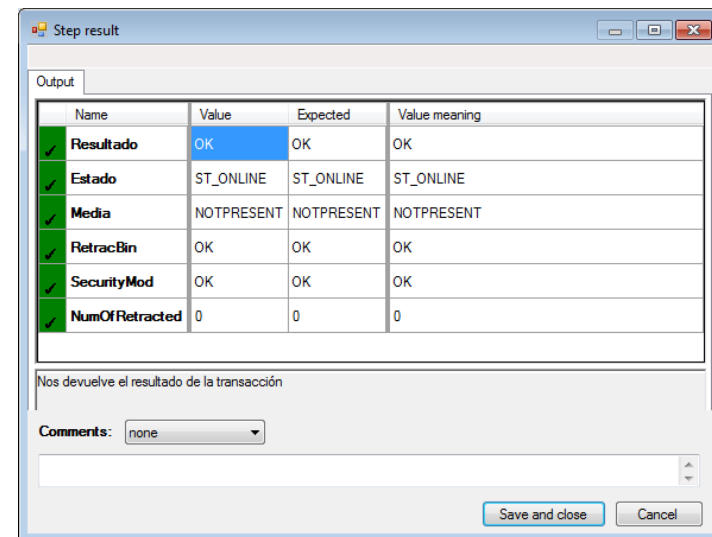
Another improvement was in the start screen. At the end of the Milestone 1 it was added a small start screen offering 3 basic options at the startup: Create Suite, Open Suite, and Open Device Manager. However, after the test we saw that this welcome screen was unneeded. The figure below shows how a new start screen was proposed, but again it was finally discarded by users, as they expected to start the application and get into the main window directly. So a welcome screen was not developed. The figures are shown below.



Iteration 10. Result details with 2 tabs: Expected and obtained result (left) and comments panel

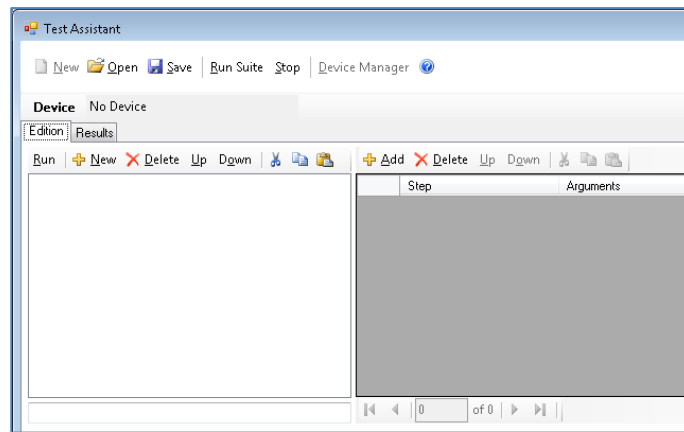


Iteration 11. Wireframe with comments incorporated in the same view

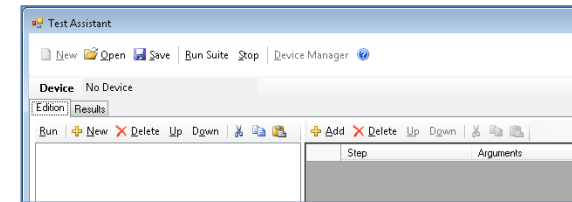
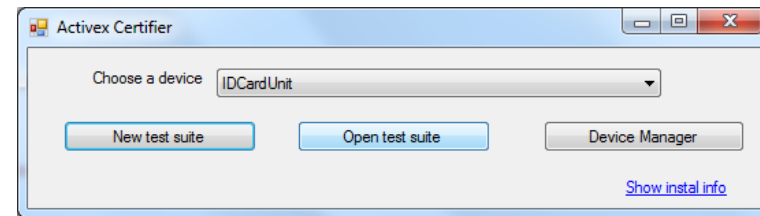


Iteration 17. Implementation of design with colors and symbols (This corresponds to Milestone 3)

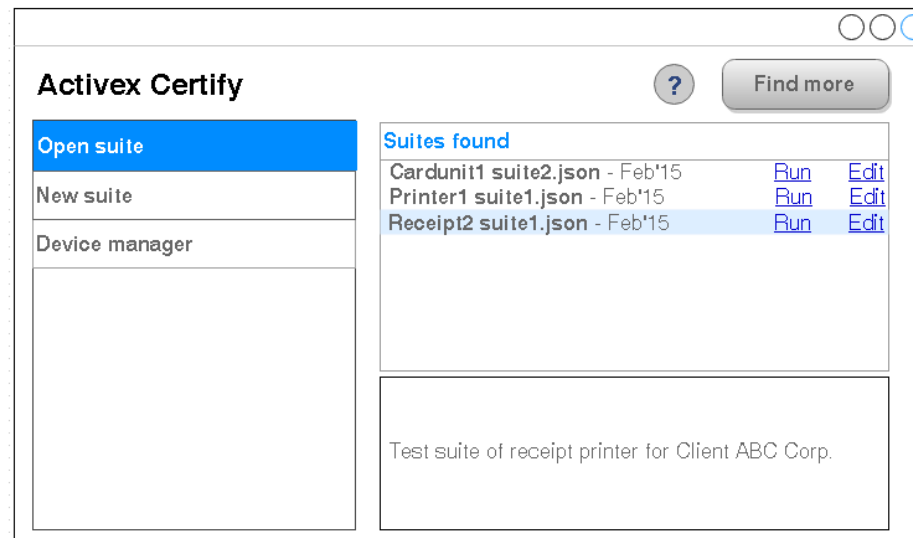
FIGURE 31. The design and implementation of the Result Detail window



Iteration 6



Iteration 7



Iteration 11. Wireframe proposal for starting screen. But it was not approved by users.

FIGURE 32. The Start Screen

Finally, a major enhancement was needed to improve the navigation between views during execution of Test Suites. For this, we applied the navigation map method presented in the next section.

6.2.7 NAVIGATION MAP: OPTIMIZING THE NAVIGATION OF THE EXECUTION VIEW

The usability test revealed how hard was the switching between views and jumping between controls while running a test case repeatedly (especially by using just the keyboard). The basic steps were:

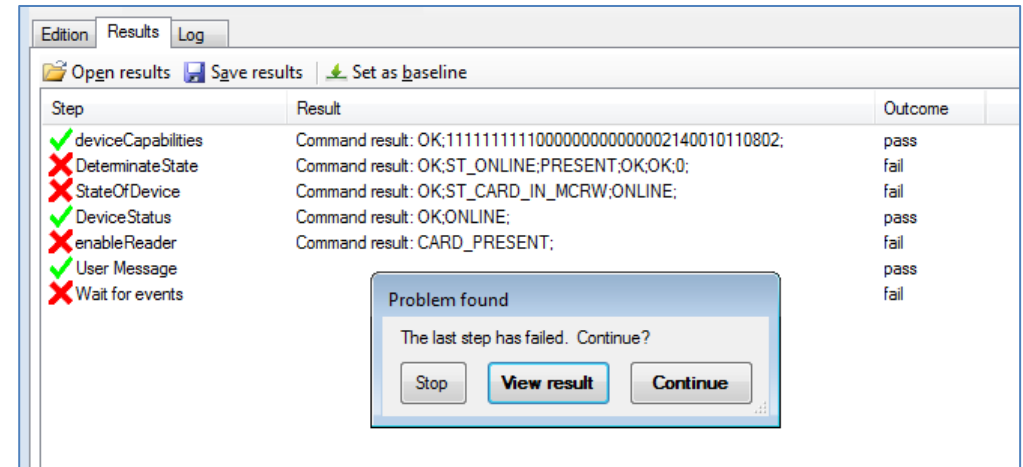
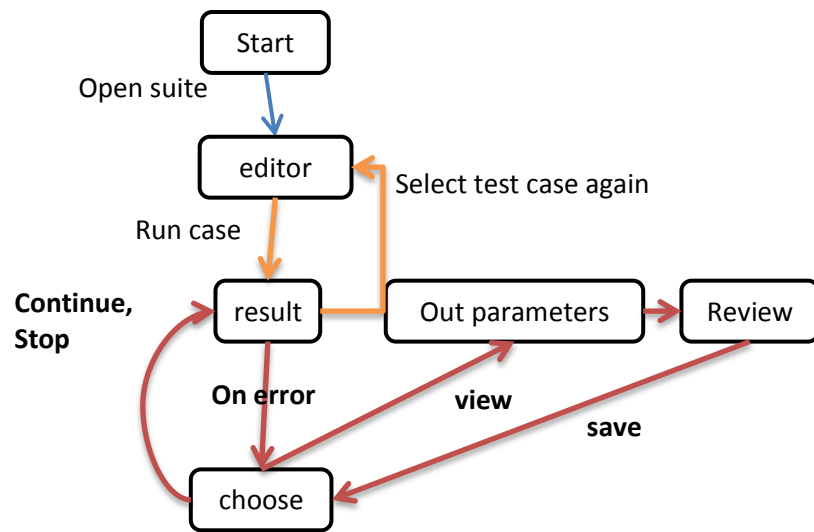
➔ In Edition View: Select a test case, click on Run, change to Results view, wait and then see the Result Details window, look at the details, add comments (changing view), return to the previous dialog to continue, etc.

The current navigation map along with the user interface is displayed in the next page. Each node is a view (e.g., a tab panel) or a dialog box. The orange cycle is the required path to run the same test case again; while the red cycle is the sequence of steps done in case of a test case failure, which requires the user to place comments and a revised result. The high number of steps is distractive for the user and is inefficient.

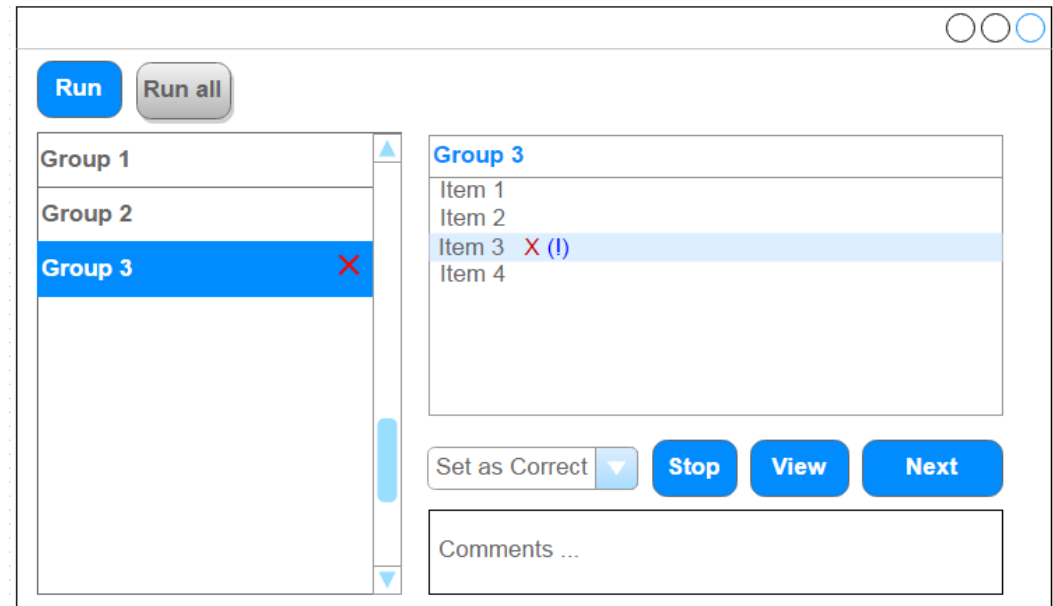
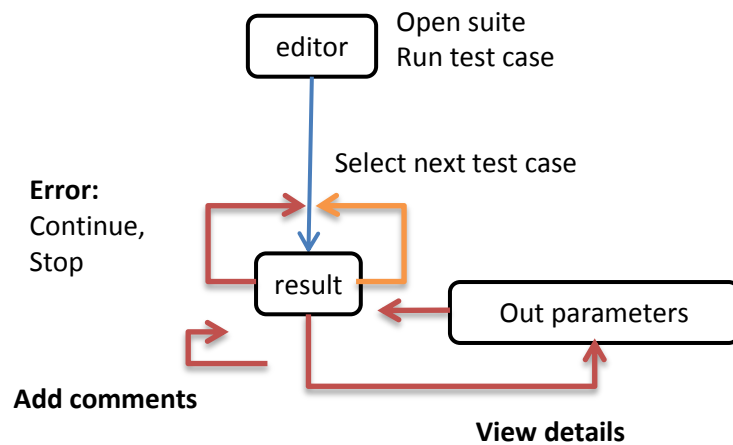
Afterwards, a new navigation map was proposed along with a wireframe that implemented its main ideas. The ideas of the new design were:

- To drop the modal dialog for failed test steps: provide the alternatives for continuing (Stop, Continue, and View Result) in the main Results view.
- To allow put comments and revisions in the general Results view.
- To place the execution functions (Run Test Case, Run Suite and Stop) into the general results view. The Run Step function had not sense anymore in a Test Suite.
- To preserve the hierarchy of test cases and steps in the Results view with a master-detail view.

The second part of the picture below shows the new navigation map with a new UI design that fulfills the navigation requirements. Finally, the implementation of the design was done in several iterations, as it required important changes in the application architecture and code.



Iteration 7. Navigation during test case execution (modal dialog is obtrusive)



Iteration 11. New navigation flow (all in one view)

6.3 MILESTONE 3: SECOND STABLE VERSION

The activities performed during Milestone 3 were less focused in analysis and more in software implementation, interaction design and evaluation. Analysis activities were focused on the elicitation of requirements and the validation of concepts for the latest module: Reports.

6.3.1 IMPLEMENTING A NEW EXECUTION VIEW

At the beginning of Milestone 3 (iteration 12) it was clear that the new model of execution proposed by the new navigation map required further refinement in order to plan its implementation and make it more viable. So, we proposed about 12 new refinements of the design until getting a viable proposal. The goal was to solve the following issues (refining the previous ideas):

- Results are not classified by test cases
- It is not clear the current step in execution until it has finished
- The AX events are not logged in the results view
- The execution of a test case requires dropping the result of other previous test cases
- Unnecessary switching between editor and results view for running a single test case

A selection of the proposed designs is shown in the next page. We started with a master-detail format for test cases and test steps. Then we simplified the view with to avoid repetition of test steps and added controls for navigating back and forward. Then we removed the master panel for simplifying the keyboard control. In the last proposal, users suggested that test cases should have a short name (identification) and a long description which would be useful for guiding the user, and also a master panel was introduced again but only for visualization of the current test case.

The implementation was split in two parts: First to implement the third design and then add the master panel (test cases) to get the fourth design. However, once the implementation started it was clear that this view was saturated and that a global view of results was missing. Therefore, we built several paper prototypes to show the problem to users. At the end of the meeting we agreed on the creation of a new view for having the results of test cases and ease the addition of comments directly. The first implementation of this new refinement is shown in the Figure 34, where there are now two views: **Execution view** and **Results view**. The implementation of iteration 17 (bottom left) was selected for a new usability test.

After the test, in iteration 18, a new layout was proposed placing the revision panel in the top of the view, but it was discarded as it was far from clear and introduced too much controls in a single region.

Certificación IDC 1 Selection All

Estado del dispositivo tras un atasco de tarjeta...

#1 enableReader

#2 "insert card please"

#3 retainCard

#4 wait events

Recepción del evento tras atascar el papel para que no pueda ser expulsado

DeviceCapabilities Fail
 Obtained: OK;11111010101010
 Expected: OK;11111010101011

retainCard Pass
 Obtained: OK;11111010101010
 Expected:

Events received: cardInserted, cardRemoved Pass

Its correct View result Continue

| Comment

Test 3 of 5

Locked card

#1 enableReader

#2 "insert card please"

#3 retainCard

#4 wait events

DeviceCapabilities Fail
 Obtained: OK;11111010101010
 Expected: OK;11111010101011

retainCard Fail (Revised)
 Obtained: OK;11111010101010
 Expected:

Events: Pass
 Received: cardInserted , ...
 Missing: cardRemoved , ...
 Unexpected: cardInsertedError , ...

Its correct View details Continue

| Comment

Certification of IDC 1 Case All

Test with locked card

DeviceCapabilities Fail
 Obtained: OK;11111010101010
 Expected: OK;11111010101011

retainCard Fail (Revised)
 Obtained: OK;11111010101010
 Expected:

Events: Pass
 Received: cardInserted , ...
 Missing: cardRemoved , ...
 Unexpected: cardInsertedError , ...

Its correct View details Continue

| Comment

TC IDC 123 Case All

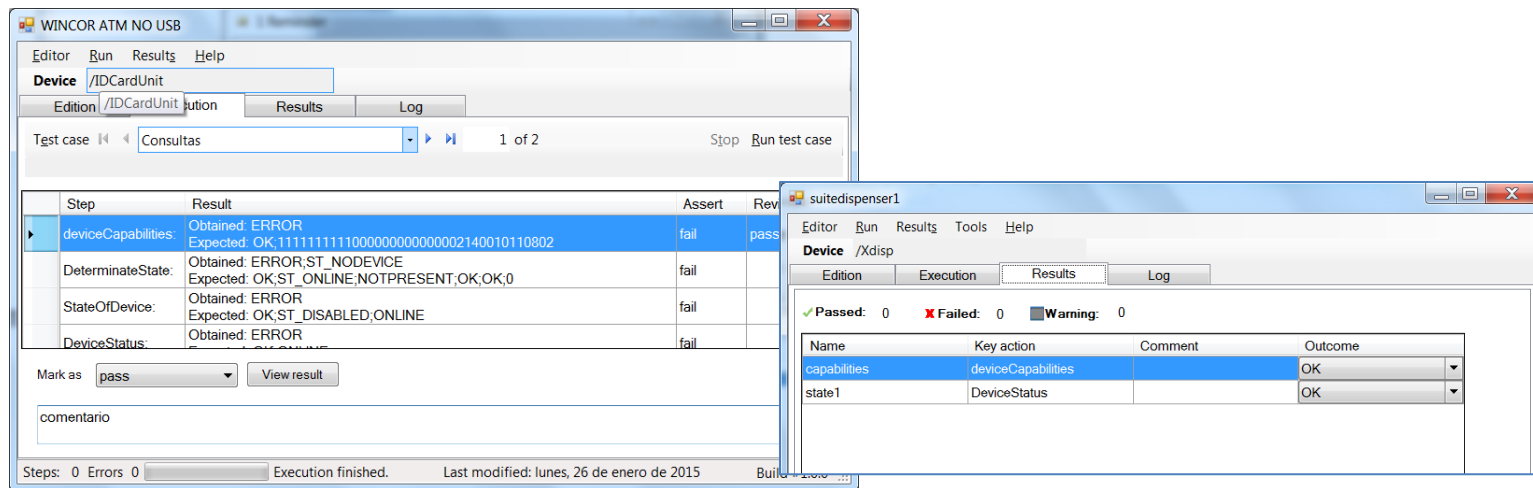
Test of Card Unit using a card that is locked in the device input

	Assert	Step	Result
Test suite 1			
Node1	Fail (R)	DeviceCapabilities	Obtained: OK;11111010101010 Expected: OK;11111010101011
Node2			
Node3			
Node4			
Node5	Ok	Wait events: cardInserted	Received: cardInserted Error Missing: cardInserted Unexpected: cardInsertedError
Node6			
Node7			
Node8			
Runnin g		Message: Insertar tarjeta	

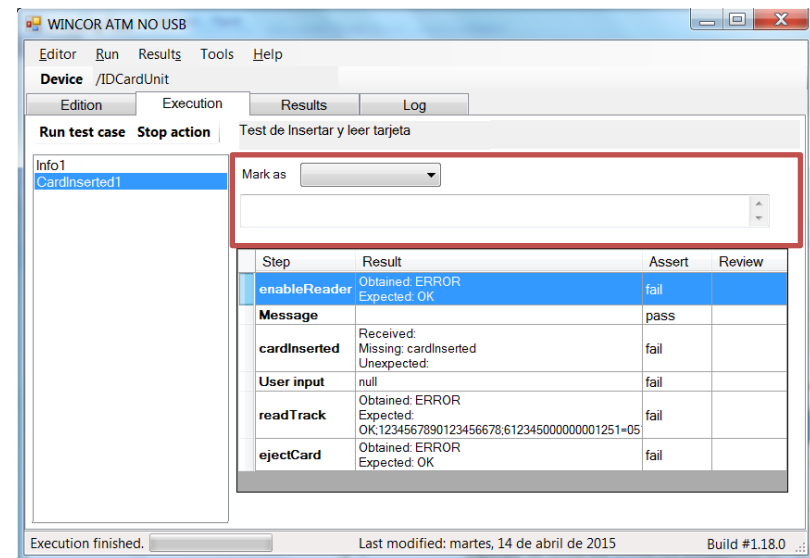
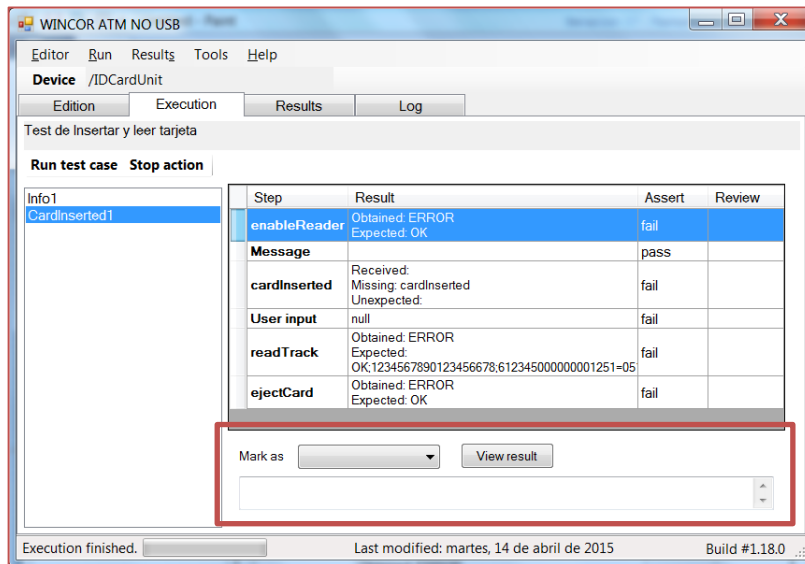
Its correct View details

| Comment

Figure 33. (Iteration 12) Wireframes for new execution view (attractiveness vs keyboard friendly vs coding cost)



Iteration 14. Implementation of new Execution and Results views (now two separate views)



Two alternatives for revision panel (Left: Design of Iteration 17. Left: Discarded proposal in iteration 18)

FIGURE 34. Implementation of the Execution View (Part 1: Top; Part 2: Bottom)

6.3.2 THIRD USABILITY TEST

Once the build of Iteration 17 was ready, it was scheduled a new usability test. The test was carried out in the ATM laboratory, with the same user who participated in the previous test (about 2 months before). This user had neither seen nor used this application version before. The goal of this test was the evaluation of the new Execution and Results view along with the implementation of the previously approved design of the Command Runner (there are additional details in Chapter 7). The activities of the test were:

- Activity 0: Execution of individual commands (for testing of the Command Runner)
- Activity 1: Run an existing test suite (for testing of the new Execution view)
- Activity 2: Create and execute full test suite (for testing the small enhancements of the Edition view)
- Activity 3: Partial execution of test suites

The results of this test were as follows:

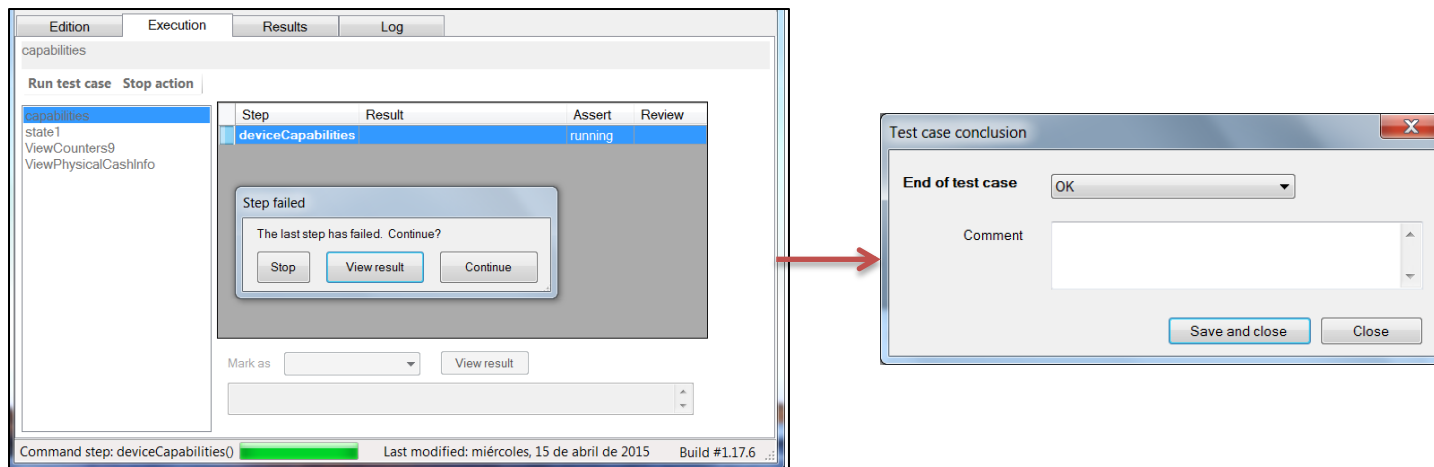
- Activity 0 was removed from the plan because the Command Runner was impossible to use without a mouse (this activity was postponed for a future usability test after a new version was ready).
- Activities 1 to 3 were successfully finished and within the predicted time (with a 4-minute delay).

Regarding error rate, the user made 5 errors caused by ambiguities in the GUI, an accidental error (the “Run test case” menu item is highlighted and very close to the “Edit test case” item, but the latter is more frequently used than the former). Other 11 errors corresponded to user doubts and other diverse issues. The most important issues found were:

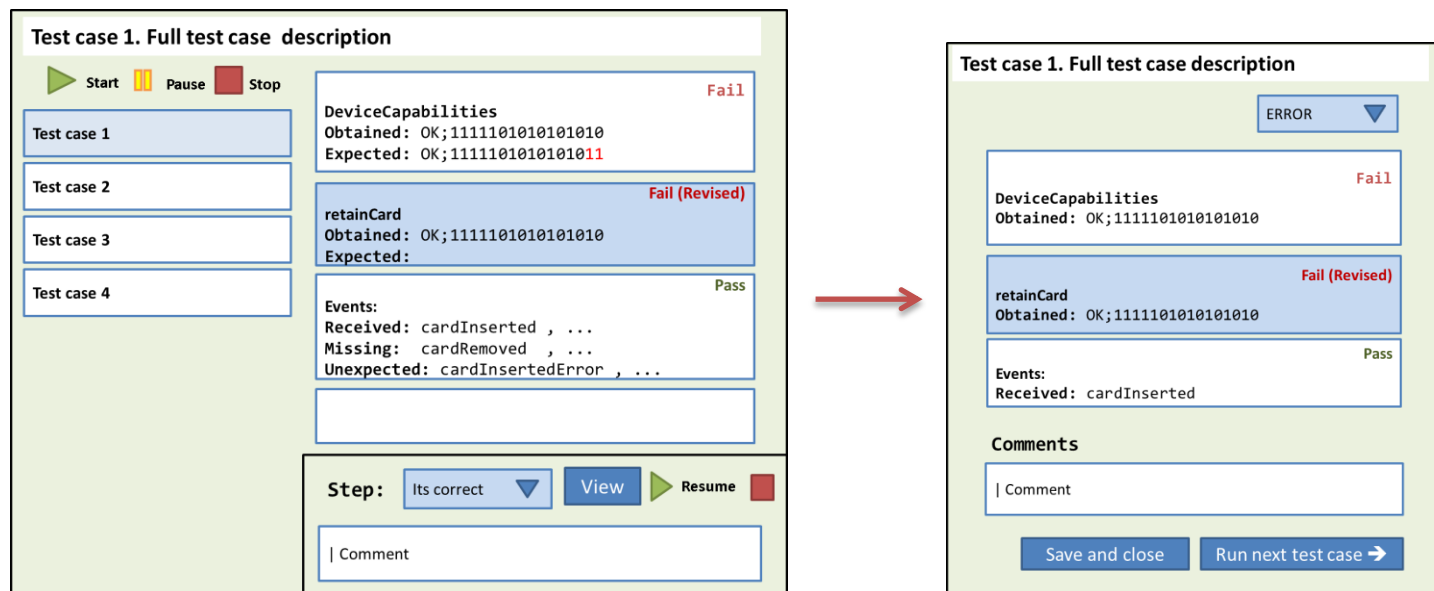
- The test case conclusion window is a modal dialog that asks for user comments. However, this dialog impedes to see the results, so a comment cannot be provided.
- The edition of test steps (of type Event) caused confusion. A new design was proposed to users to solve ambiguities.

On the other hand, the user satisfaction questionnaire was better compared to the previous test. Finally, out of the findings we proposed several changes, which are commented in the next sections.

6.3.3 DESIGN OF A NEW TEST CASE SUMMARY VIEW



Iteration 17. Execution view and test case conclusion



Iteration 18. Wireframe for execution view and test case summary (option 1)

FIGURE 35. The Test Case Summary View used in the Test 3 and the new design proposed (option 1)

Test case 1. Full test case description

Test case 1 Run

Test case 2

Test case 3

Test case 4

DeviceCapabilities
Obtained: OK;11111010101010
Expected: OK;11111010101011 Fail

retainCard Fail (Revised)
Obtained: OK;11111010101010
Expected:

Events:
Received: cardInserted , ...
Missing: cardRemoved , ...
Unexpected: cardInsertedError , ...

Its correct View details

Continue

| Comment

Test case 1. Full test case description

ERROR ▼

X DeviceCapabilities = OK;11111010101010

OK Message:

X Events: cardInserted=OK;

Comments

| Comment

Save and close

Run next test case →

Iteration 18b – Execution view and test case summary (option 2)

WINCOR ATM NO USB

Editor Run Results Tools Help

Device /IDCardUnit

Edition Execution Results Log

Start Pause Stop

Test case: Info1

Prueba los comandos de consulta del dispositivo

Step	Result	Assert	Review
deviceCapabilities	not run	none	none
DeterminateState	not run	none	none
StateOfDevice	not run	none	none
DeviceStatus	not run	none	none

Step revision: none step details

Suite loaded Last modified: martes, 14 de abril de 2015 Build #1.19.0

Review Test Case

Test case: Info1

Prueba los comandos de consulta del dispositivo

Step	Result
deviceCapabilities	OK:11111111110000000000002140010110802
DeterminateState	OK:ST_ONLINE;NOTPRESENT;OK;OK;0
StateOfDevice	OK:ST_DISABLED;ONLINE
DeviceStatus	OK;ONLINE

Outcome ok Set final outcome: ok 1

Comments

Funcionamiento correcto.

Save and close

Outcome none Reset outcome ok

Assigned NCR

Iteration 19. Implementation of new design of test case summary

Iteration 22

FIGURE 36. Design of Test Case Summary View (option 2) and its implementation

A solution for summarizing the end of a test case was needed. We proposed two alternative designs (using wireframes) with the aim to solve the following needs:

- Make visible the end of each test case
- Give enough context to decide test case outcome (show the results of the test steps) and provide comments (mark it as “pass” or “fail”)
- Make easier the navigation to the next test case

In Figure 35 we show the version used in the test and the first design proposed (wireframe), and Figure 36 shows the second wireframe. The latter was selected for the implementation. A feature was added in the implementation: The outcome of the test case is suggested automatically (out of the results of steps) and can be reset to the automatic value using a hyperlink.

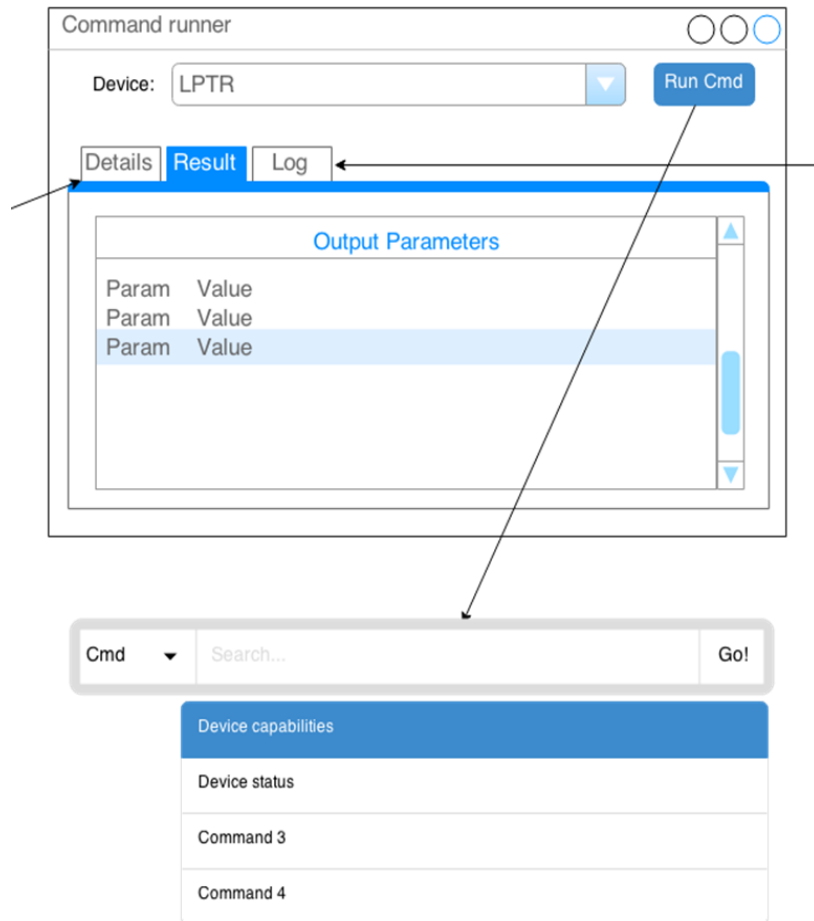
6.3.4 GIVING FULL CONTROL TO THE USER: PAUSE AND RESUME

Until the test 3, the user had no option to pause and resume a test case, which is something that gives flexibility to users during the execution of a long Test Suite or Test Case. Moreover, whenever a test step fails the execution should simply pause and let the user decide what to do. However, until now we used a modal dialog with three options: Continue, Stop, and View result (see Figure 35 in the top) that was too restrictive as it blocked the full UI. So users could not browse previous steps or see anything different to these options.

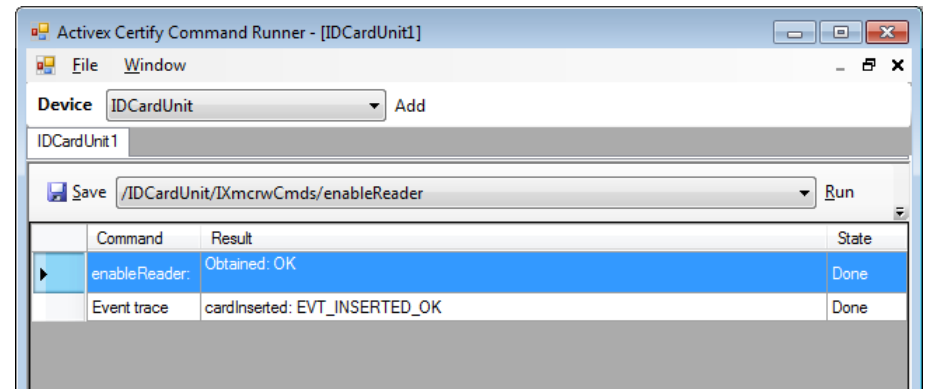
The visual design of this feature is trivial (just add the Pause button and allow to switch the Run button to the Continue function), however the implementation required a major development effort. Fortunately, the architecture was prepared for such a change as the execution was fully decoupled from the GUI and also used a separate thread from the beginning. The resulting implementation is also shown in Figure 36.

6.3.5 IMPLEMENTATION OF A NEW COMMAND RUNNER

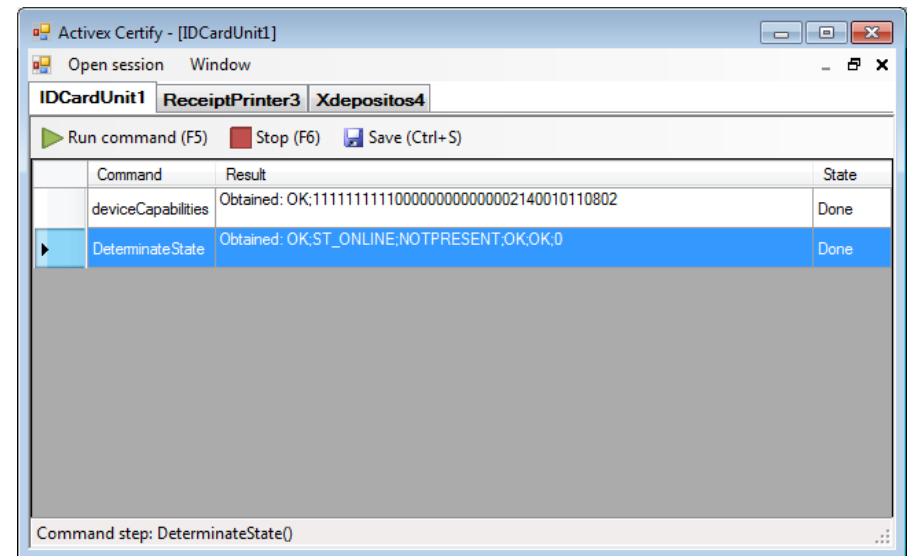
Going back to the results of the third usability test, we needed a solution for making the Command Runner more accessible and usable. On the first hand, the initial implementation did not fully followed the original wireframe design (where commands are selected in a popup). On the other hand, the wireframe does not allow multiple devices at the same time. So a new design in between was proposed and implemented. Moreover, the activity of the last usability test was cancelled because of the lack of keyboard accessibility. Therefore, a new implementation was created which fully addressed the mentioned concerns. In the figure below we see the images of the original prototype, the failing and the new implementations.



Iteration 8. First wireframe for the Command Runner. The wireframe does not allow multiple devices simultaneously.



Iteration 13



Iteration 19. Final implementation (popup is more complex than the wireframe)

FIGURE 37. Command Runner: The wireframe and the implementations (iteration 19's was used in the 4th usability test).

6.3.6 MAKING USABLE DYNAMICALLY GENERATED UI FORMS

The application uses dynamically generated input forms for the edition of test steps. These forms are generated at runtime by loading a device model (a JSON document with the definition of the device, of its commands and arguments)⁴.

The basic layout of the forms is rather simple, a heading panel (the command description), the content panel (a list of input controls), and the footer (the dialog buttons). Below, we show the input form corresponding to a specific device command.

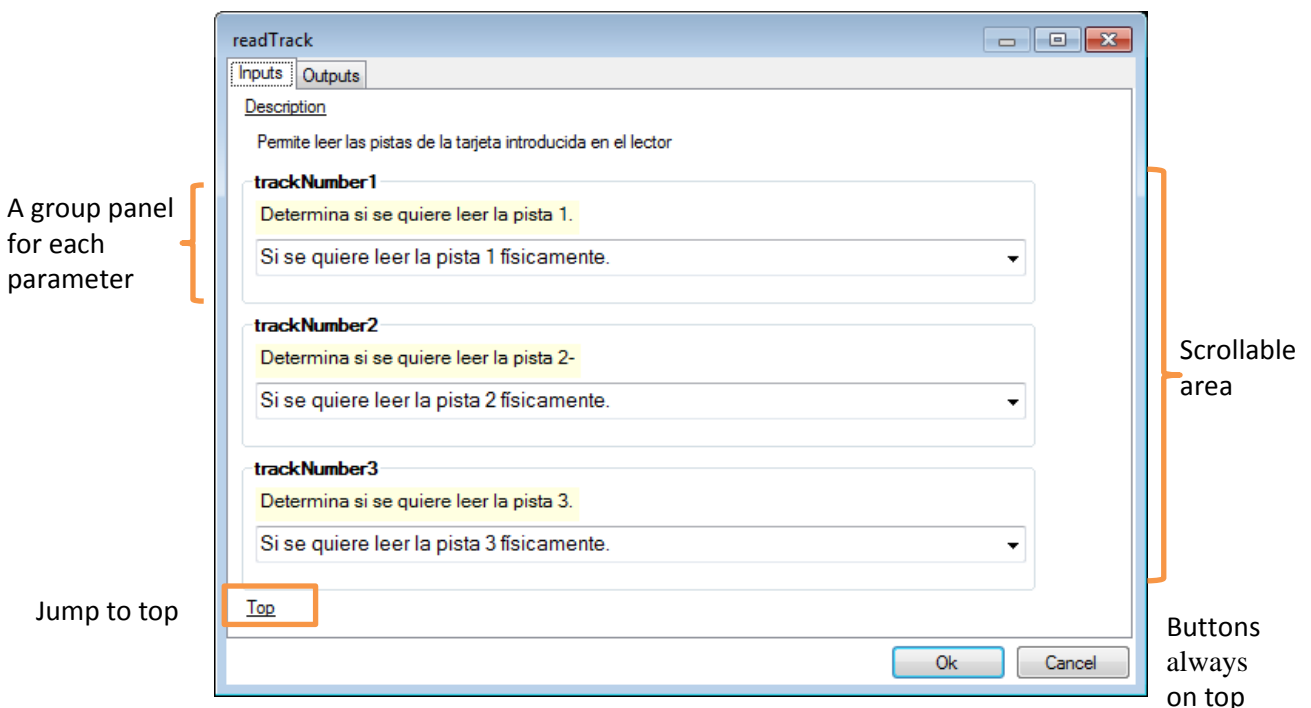


FIGURE 38. Dynamic form for editing test step arguments (Iteration 12)

The first design considerations were regarding the main layout. Its characteristics are:

- The name and description of each parameter are grouped for easy reading.
- Help texts are highlighted using a well-known color convention.
- The area of parameters is scrollable to allow any number of items while having a fixed window height, and always-visible dialog buttons at the bottom.

Second, the content of input forms was filled at runtime. So besides allowing any number of items, we found the issue of overlapping controls. Whenever the device model had descriptions that were too long or split in several lines of text the help text hid the input control.

⁴ This strategy is based on model-driven engineering (MDE)

A proposal for avoiding this was the addition of a new attribute in the device model: one for a short description and another for full descriptions. The latter could be displayed on demand in a separate view while the former could be always visible and located beside the input control (see the figure above). So the implementation consisted on the modification of Json models and the introduction of the popup note. However, the latter was postponed to a future iteration.

MtxNumber

Algoritmo o tabla de dispensación CEN XFS 3.x:

- "1": Mínimo número de billetes.
- "2": Vaciado equitativo de cajetines.
- "3": Máximo número de denominaciones.
- "x > 3": Tablas especificadas por aplicación.

Position

Indica la posición de salida de los billetes (CEN XFS 3.x)

Posición nula. El SP del fabricante determinará la posición.

Ok Cancel

Iteration 18

FIGURE 39. Layout issues of generated input form

Device.json

```
{
  "Name": "Currency",
  "Description": "Enter the currency using ISO format",
  "LongDescription": "The ISO format consists of [...]"
}
```

Command Input Form

Currency

Enter the currency using ISO

The ISO format consists of three letters such as:

- USD
- EUR
- JPY
- COP

FIGURE 40. Input form dynamically generated. In orange is shown the proposal for displaying long description.

Third, the requirements included the support for different kind of parameters, which were increasing in complexity:

- Free input fields (text boxes)
- Single selection fields (combo boxes)
- Multiple selection lists (list of check boxes)
- Composite parameters (a parameter with a linked edition form with parameters of any kind)

And the list continues. One of the most challenging types was the parameters with a command-line format. An example of this parameter is found in the command for controlling sensors. There are several sensors:

- Sensor 1
- Sensor 2
- Etc.

And the command allows operations on each of them:

- Turn on
- Turn off
- Unchanged

So the edition form for this parameter shall allow inputs like this: “sensor1=ON sensor2=OFF” or this “sensor1=ON”. So, the most intuitive and efficient editor that can be generated is the one at the bottom-right in the figure below: Through three-state check boxes we could set the operation for each sensor and the form would translate them into the target string format.

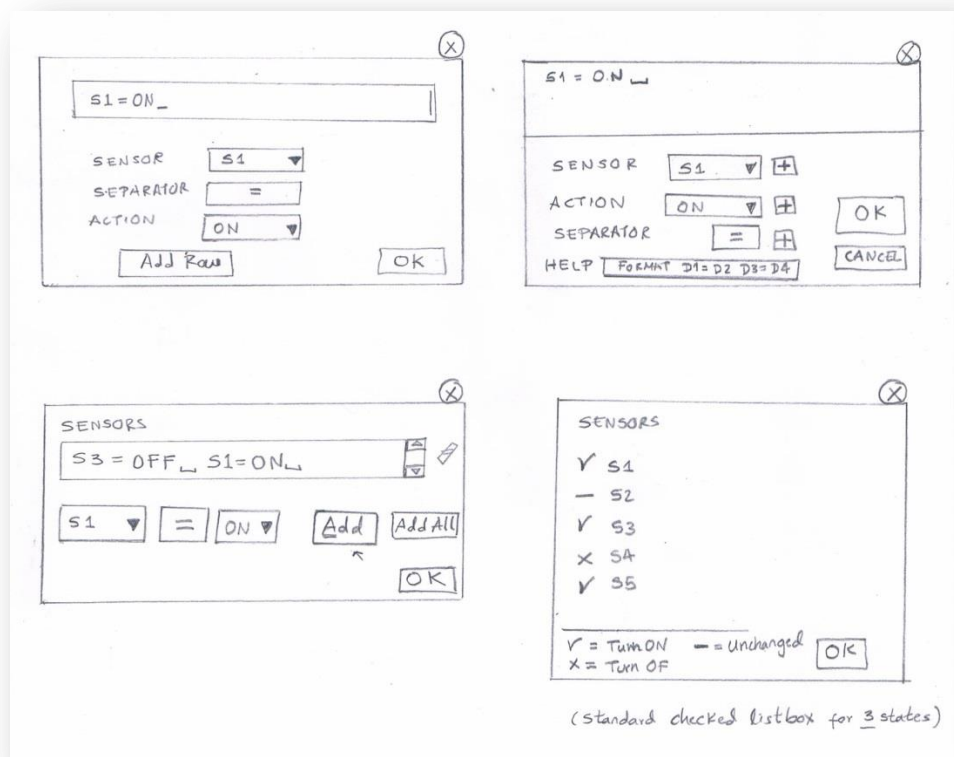


FIGURE 41. Four different design alternatives for the command-line format parameter editor

However, this option has several problems. First, the Json model is generic and allows any number of items and operations. So instead of three-state checkboxes we would require a generic list box per item.

A second issue is regarding the number of items: if there are too many items then finding the right item would be cumbersome. And a third problem is regarding the length of texts: text with very different text lengths would make the content hard to read: a bunch of list boxes and labels. A fourth issue is regarding the requirements: the tool shall allow both the input of valid and invalid format values, because the goal of the tool is testing.

Therefore, we looked for other alternatives (see the figure above). They are based on the idea of having an editor with free input text with a single toolbar with the items and operations available. So the user can freely edit the text or use the edition tools. We chose the bottom-left design, as it was a safe alternative with an acceptable usability and a viable development cost. The implementation of the design is shown below:

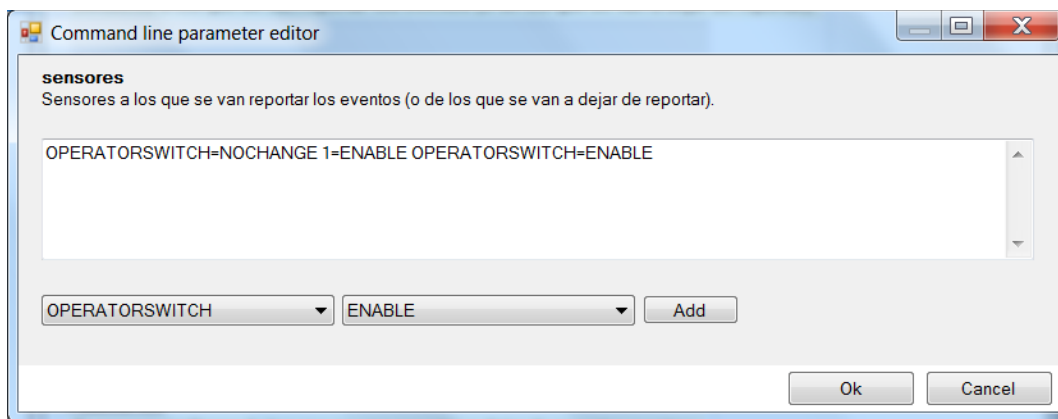


FIGURE 42. Implementation of the command-line parameter editor

This generic editor meets the requirements, as it allows values with valid and invalid format, and also has the following characteristics:

- Valid values can be built using the controls, while invalid inputs can be formed through direct typing.
- It is a generic editor: it accommodates to list of keys and values of any size without affecting easy reading and picking of items.

Regarding usability, effectiveness and performance could be furtherly evaluated with users, although its impact is not relevant, as it is a very specific parameter type that is found in very few devices.

6.3.7 FOURTH USABILITY TEST

The fourth usability test was the first test that covered the three main features (and use cases) of the application: Edition, Execution and Command Runner. The test was performed in the same ATM laboratory, but using a newer ATM model (NCR SelfServ 22). The goal of the test was to evaluate the usability through the certification of a new device: The cash dispenser. The participant was a user who has not participated till now, who is expert in the device and had not used the application version before (see Chapter 7 for more details). The activities of the test were:

- Running individual commands
- Run an existing test suite
- Create and execute a test suite

The user successfully finished all the test activities within the expected time (with just 2 additional minutes). The error rate obtained was 1 user error and 6 varied issues:

- The Command Runner reuses dialogs from the Execution View, but the options for setting a baseline and writing comments has no sense in its context (even if its disabled is confusing, it is preferred to be hidden).
- (Dialog) The label “Edit Test Case” is used for creating and editing test cases, which is confusing (this caused the user error).
- “Create test case” is a function that cannot be reached with the keyboard (using Tab). The shortcuts and contextual menu were not discovered by the user.
- List boxes in test step input forms show selection options that are difficult to recognize: The options should have a short name and long description, but it is used only the long description.
- Test Case summary view: The user tries several times to enter and see more details (but the option does not exist).

This test was followed by a Thinking Aloud Protocol test aimed to evaluate the usability of reports. This last test was the first usability evaluation for the Reports module. The test was carried out by the same user but using her laptop, as it is the usual context of use for report generation. The activities were the following:

- Generate a certification report out of two test suite result files (Device Card Reader and Cash Dispenser). The report was required to comply with a list of requirements.

The result of the test was positive. The user successfully finished the test without requiring external assistance in a third of the estimated time (10 instead of 30 minutes). The error rate was 2 user errors, 5 varied issues and 1 (minor) bug:

- The report wizard provides an “Open Folder” button: The user thought it was to open a source results file folder instead of the target report folder.
- The bug was a date field that should be automatically filled but was empty.
- The user wanted to open a report definition to edit it, but this function does not exist. However, creating a new report from scratch required almost the same effort.

Following the “talk” of the user, the flow of actions was coherent with its thinking most of the time.

Finally, before the end of the milestone, a new version was released which included most of the suggested improvements. This version was the last version produced which is presented in Chapter 8.

6.4 MILESTONE 4: EVALUATING THE FINAL APPLICATION VERSION

In this last milestone, we wanted to measure the business benefits of the project. Therefore we designed a new usability test (#5) whose goal was not only to evaluate the user experience but also to measure the total effort required in a full certification process by using the new tool.

6.4.1 FIFTH USABILITY TEST

The fifth usability test was designed to be more realistic and demanding. We searched for a participant with the *junior profile* (according to the user profiles defined in Chapter 5). And effectively we selected a user who knows the certification process but has never done a certification of any device before. The place of the test was the ATM laboratory, the ATM used was the NCR 5886 (an old model), and the device tested was the Card Reader. In Chapter 7 and Chapter 8 we describe the results of this activity in detail.

Besides the numbers obtained, a general conclusion was that a *junior user* will be able to perform successfully a full certification only if the Test Suite is modeled with very detailed instructions and the user also knows the usual practices for physically manipulating the device. That is, the tool does not provide any support for teaching the user how to interact with the ATM in the context of the certification process.

On the other hand, we observed that the paramount effort required by the user was not comparable with the effort required by an expert. So this test also led to a broader discussion about how experts really perform the testing steps and how the tool can be adapted to allow a closer assistance to any kind of users.

6.4.2 LINES OF IMPROVEMENT

We arranged a meeting with an expert user (*senior profile*) with the aim of getting his opinion regarding the last product version. The goal was to know how he exactly performs a test session in the most optimal way. To make more objective the session, we asked the user to try to model a Test Suite, while thinking aloud, with the exactly same configuration he usually uses in the execution of tests.

The session took about 1 hour and at the end the user modeled the skeleton of a test suite for the Card Reader. Instead creating 100 test cases with all the combinations of state-function, he defined a suite with the following structure:

- 1 big test case for testing the standard correct operation of the device in a single flow of actions.
- Separate test cases for error conditions and boundary cases

However, in a second iteration he produced the following structure:

- The flow of actions for the correct case was modeled as a Test Suite divided by several test cases
- The error condition tests were modeled as a separate Test Suite

In the final iteration, the user explained that the certification report was structured by defining a set of device states where several Device commands are tested. Therefore, testing 5 commands in a given device state corresponds to “1 test involving these 5 commands in a row” instead of running “5 tests in the same state”. So he modeled a Test Suite composed by test cases, each one corresponding to a device state (about 10 items). And each test case required the assertion of several commands under test. This optimal model required the modification of the application, as currently several assertions per test case are allowed but the report will produce only one record per test case, not a record per assertion made.

The last proposal was an important step for making possible the complete introduction of the tool in the business process. The following chapter (Chapter 7) describes in detail the evaluation of the product along the development process.

7 EVALUATION

In this chapter we describe the methods applied for usability evaluation (the selection of evaluation methods is discussed in Chapter 4).

In general, we will see that once a stable product version was available a test was performed before continuing developing new features. So, latter tests were more comprehensive than formers –more functional features were available.

The following picture shows the timeline of the project. The usability evaluation methods we applied are shown with the “UX” label. As an aside note, we highlight the first test because it was not successfully applied though its results were worthy to mention.

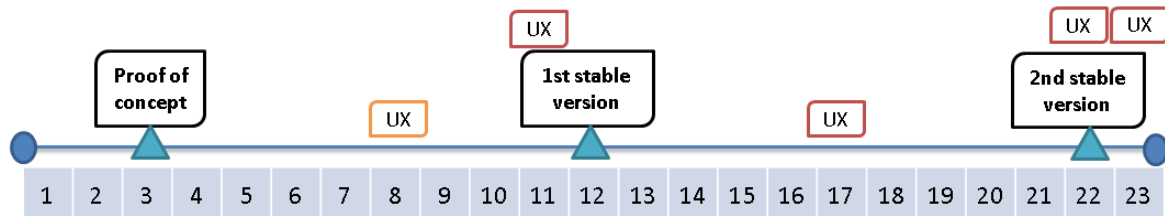


FIGURE 43. Usability evaluations performed along the project

7.1 METHODS, METRICS, PARTICIPANTS AND LOCATION

Methods and metrics

The usability metrics to be measured were selected according to project goals and actual resources (as defined in [20, 21, 23]):

- **Efficiency** (time on task)
- **Effectiveness** (success rate), and
- **Satisfaction** (like/dislike questionnaire)

Each method was applied according to the kind of task, its context of use, and the functional feature evaluated. They are the following:

- **Laboratory usability testing:** For measuring all three metrics interacting with the ATM
- **Thinking aloud protocol:** For measuring effectiveness and satisfaction using user's PC
- **Heuristic evaluation:** For measuring usability strengths and drawbacks according to Nielsen's principles.

Below, the figure shows how evaluations covered functional modules along the participants and the location:

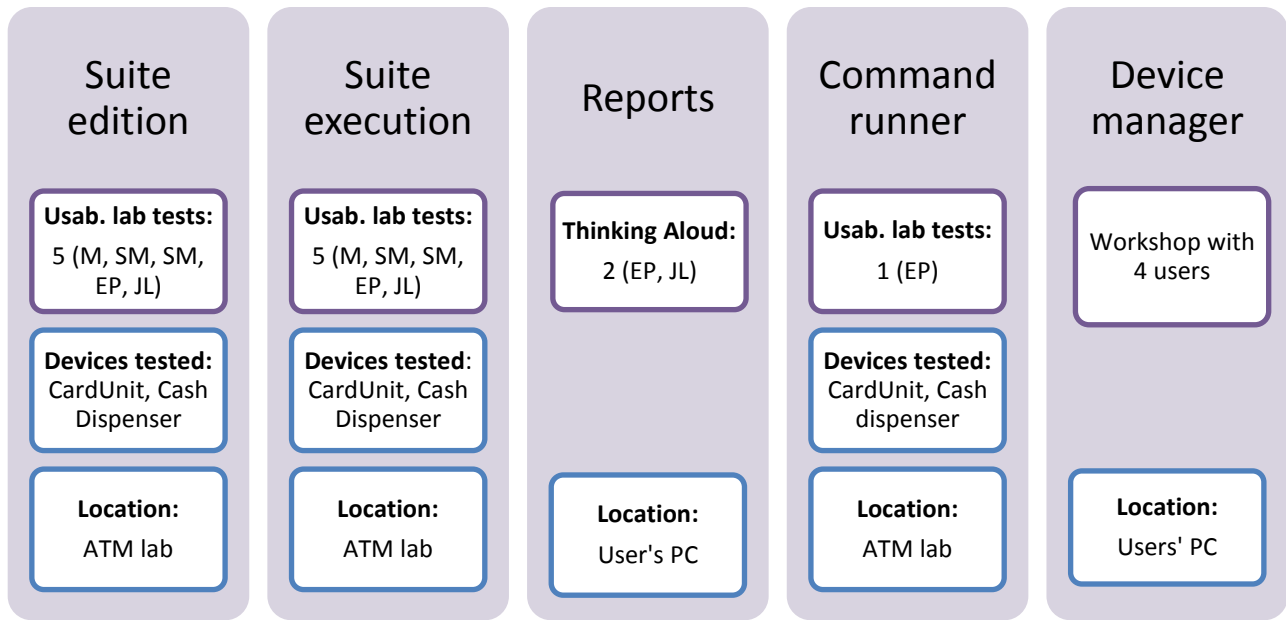


FIGURE 44. Usability evaluation methods applied per functional module

Participants

On the other hand, the number of participants in each case was always 1 user, who was always a different person. But now we will see why this is not a drawback for the validity of the evaluation.

First, the number of participants is not required to be always high. According to Nielsen [18], 5 participants can provide the most of the information of usability issues during a usability test, and increasing the number will provide us very little additional feedback. Second, in the context of the company, there are 3 people currently in charge of the Certification Process and potentially 2 more people can be involved depending on the company's needs. So, we have a total population of 5 users. This means we could have planned the 5 tests made at least in two different ways:

- Involving all 5 users in all tests
- Involving 1 different user in each test

We chose the second alternative because it was more viable (involving the whole team at the same time or along the same week for a 1-hour session is difficult to ask in a company); and also it gave us the advantage of having always a different “novice user” who had not used the application before.

Location and resources

The usability tests were performed in the company's laboratory where two different NCR ATMs were fully available for project evaluations. The physical settings of the laboratory are pretty similar to real context of use (keyboard and mouse are distant; the user is standing; test materials are on the table, etc.). In absence of a small table, the keyboard must be operated from the ATM's top.



FIGURE 45. ATM laboratory where evaluations were performed (The NCR SelfServ 22 ATM was one of the machines used)

In the usability laboratory tests the observer must be pretty close and behind the user, otherwise screen's privacy filter will impede the external observation. Note-taking during the test is made with notebook and pencil. The picture below shows one of the two ATMs used in the usability tests.

7.2 LABORATORY USABILITY TEST 1

Test date: 2015-01-15

Test duration: 120 minutes

Place: Laboratory using a NCR 5886 ATM

Version tested: Iteration 8 build

Devices tested: Card reader and receipt printer

Participants: 1 User (M)

7.2.1 GOAL

Perform the first application deployment on an ATM and evaluate its functionality and usability.

7.2.2 ACTIVITIES

- Activity 1: Create a test suite for the described list of test cases (commands)
- Activity 2: Enhance the test suite modeling the described list of test cases (events)

The user was asked to create a test suite with 17 test cases from text descriptions for the card reader and then run them in order. However, due to problems setting up the application on the ATM (it was the first time) the session was delayed and we needed to change it. So instead the plan, the user made an exploration of the tool trying to perform habitual tasks while talking aloud and providing comments, suggestions and feedback.

7.2.3 RESULTS

This test cannot be considered a valid laboratory usability test because the test plan was not followed. However, the session produced important feedback and new requirements that were quickly implemented in the tool thereafter. The most important issues found were:

- Background and font color contrast in some GUI controls were inadequate (either invisible or hard to read).
- Using the keyboard (controlling focus) was challenging (some toolbars have no shortcuts or keyboard access, and other functions had the same access key).
- Stacked modal popups must to be avoided (some modal windows were locking the whole interface and were behind other windows difficult to reach with the keyboard).
- Some GUI forms were not adapted correctly to screen size and resolution (forms too big).

Finally, this experience became the base for the design of the next usability test.

7.3 LABORATORY USABILITY TEST 2

Test dates: 2015-02-02; 2015-02-06 (two sessions)

Test duration: 45 minutes; 50 minutes

Place: Laboratory using a NCR 5886 ATM

Version tested: Iteration 11 build

Device tested: Card reader

Participants: 1 User (SM)

7.3.1 GOAL

Evaluate the usability of the application for creating and executing test suites

7.3.2 ACTIVITIES

- Activity 1: Run a full test suite (the test suite is given)
- Activity 2: Create and execute a full test suite according to 6 instructions for test cases
- Activity 3: Partial execution of test suites

Note: During the session, the Activity 2 was simplified to reduce the session time. Original estimated time was 60 min, which was adjusted to 40 min.

7.3.3 RESULTS

Despite having only one participant, the activity was important in determining a number of software enhancements. It also set a benchmark for comparison against future usability assessments.

7.3.3.1 EFFICIENCY OF USE

The indicator measures the time spent by the user (minutes) versus an estimate for novice users.

Date	User	Activity 1	Activity 2	Activity 3	Total	Comments
Session 1	SM	45 min	<i>Postponed</i>	<i>Postponed</i>	45 min	The session was interrupted because of other user's commitments.
Session 2	SM	30 min	15 min	5 min	50 min	The user finished 30 minutes before the estimated time.
Estimated time		30 min	40 min	10 min	80 min	

7.3.3.2 EFFECTIVENESS OF USE

The indicator measures the level of completion achieved by the user in each activity.

Date	User	Activity 1	Activity 2	Activity 3	Total
Session 1	SM	80%	-	-	1 Partially done
Session 2	SM	100%	100%	100%	100%

In the first session it was not possible to complete the Activity 1. However, on the second date, the activity was finished successfully by the same user.

7.3.3.3 ERROR RATE

Errors made by the user while using the application. We consider four different error types.

- User error: An error caused by a usability issue.
- Accidental input errors: Not considered a usability issue, except if it's too recurrent or generalized
- Issues and questions: Any other issue found like help requests and questions.
- Bugs: Application errors

	Activity 1	Activity 2	Activity 3	Total
User errors	8	5	1	14
(Accidental) Input errors	0	0	0	0
Issues / questions	0	0	0	0
Bugs	0	0	0	0

In the first session, the user made 4 errors while in the second she made 10 (there were more activities).

7.3.3.4 USER SATISFACTION

At the end of the session, we asked the user to fill the UEQ questionnaire⁵, and then we used the data analysis tool to summarize the results. This questionnaire measures six user satisfaction categories through 26 questions.

The results per dimension are shown below. The figure highlights the ranges that are considered good (greater than 1.5), and bad (less than 1).

⁵ <http://www.ueq-online.org/>

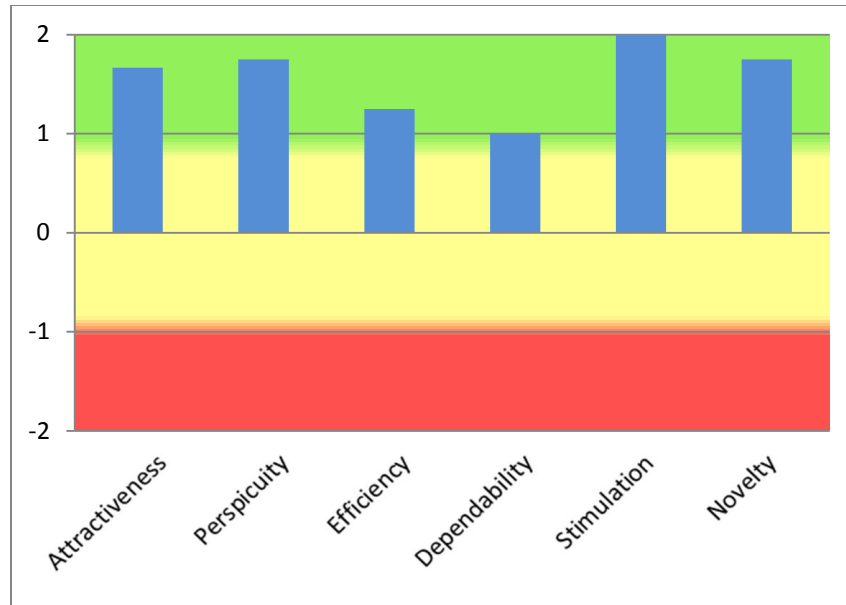


FIGURE 46. Results of the UEQ questionnaire

Attractiveness: Overall impression of the product. Do users like or dislike is?
Very high
Perspicuity: Is it easy to get familiar with the product?
Very high
Efficiency: Can users solve their tasks with the product without unnecessary effort?
Good
Dependability: Does the user feel in control of the interaction?
Good (somehow neutral)
Stimulation: Is it exciting and motivating to use the product?
Excellent
Novelty: Is the product innovative and creative?
Very high

So **Efficiency and Controllability** were the least scored, which can require broader attention for future iterations. This can be explained by the difficulty of efficient control of the UI using only the keyboard, and to quickly find the desired information (according to the user's requests for help).

In conclusion, all activities were developed entirely in the right time, without having knowledge or training on the tool before. However, assistance was required several times and several suggestions for improvement were recorded.

7.4 LABORATORY USABILITY TEST 3

Test date: 2015-04-15

Test duration: 55 minutes

Place: Laboratory using a NCR 5886 ATM

Device tested: Card reader

Version tested: Iteration 17 build

Participants: 1 User (SM), 2 observers

7.4.1 GOAL

Evaluate the usability of the modules: Test suite edition, Execution, and Command runner.

7.4.2 ACTIVITIES

- Activity 0: Running individual commands: A list of commands is given for execution
- Activity 1: Run a full test suite: The test suite is given
- Activity 2: Create and execute a full test suite: Instructions are given for creating and running a desired test suite
- Activity 3: Partial execution of test suites

Note: Activity 0 was removed from the original plan because it was impossible to use the application feature without a mouse. Therefore, this activity was not taken into account in any of the indicators presented below (this activity was postponed for a future usability test).

7.4.3 RESULTS

7.4.3.1 EFFICIENCY OF USE

The indicator measures the time spent by the user (minutes) versus an estimate for novice users.

	Activity 1	Activity 2	Activity 3	Total	Comments
Estimated time	5'	25'	5'	35'	<i>Time estimated for a user who has never used this application version</i>
Time spent	8' (10.15)	22' (10.23)	9' (10.45)	39' (10.54)	<i>The time spent was slightly greater by 4 mins. (this is not considered an issue)</i>

7.4.3.2 EFFECTIVENESS OF USE

The indicator measures the level of completion achieved by the user in each activity.

	Activity 1	Activity 2	Activity 3	Total	Comments
% completion	100%	100%	100%	100%	<i>Everything was done</i>

7.4.3.3 ERROR RATE

Errors made by the user while using the application. We consider four different error types.

- User error: An error caused by a usability issue.
- Accidental input errors: Not considered a usability issue, except if it's too recurrent or generalized
- Issues and questions: Any other issue found like help requests and questions.
- Bugs: Application errors

	Activity 1	Activity 2	Activity 3	Total
User errors	2	3	0	5
(Accidental) Input errors	0	1	0	1
Issues / questions	5	6	0	11
Bugs	0	1	0	1

In conclusion, besides the bug found, the user made 5 errors caused by ambiguities in the GUI, an accidental error (which seems to produce frequent errors in the future: “Run test case” is very close to “Edit test case” and is highlighted despite editing is more common than running). Other 11 errors corresponded to doubts and diverse issues. These results provided important feedback that became suggestions for change.

7.4.3.4 USER SATISFACTION

At the end of the session, we asked the user to fill the UEQ questionnaire⁶, and then we used the data analysis tool to summarize the results. This questionnaire measures six user satisfaction categories through 26 questions. Below we show the results along with the meaning of the categories assessed (according to [11]).

⁶ <http://www.ueq-online.org/>

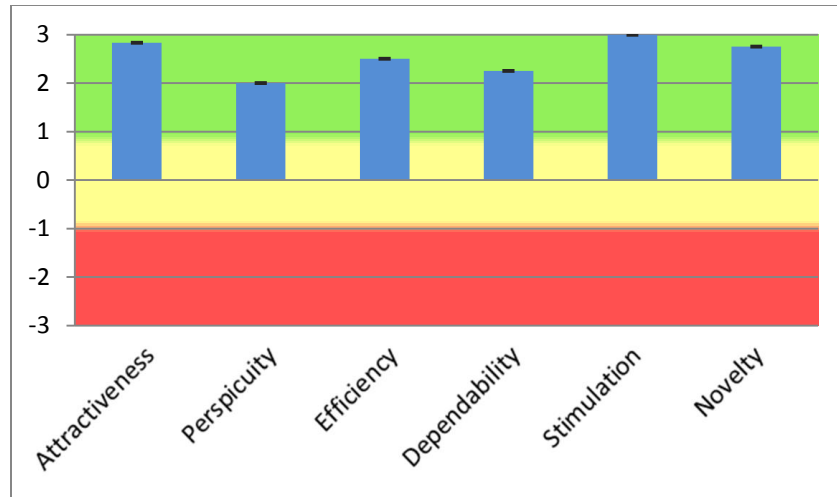


FIGURE 47. UEQ satisfaction questionnaire results

According to this scale, values above 1.5 are considered as “very good” while values below -1.5 are considered as “really bad”.

In our case, we see that the overall user perception is very positive. In particular, Perspicuity (or clarity) and Controllability had the lowest scores. The lack of transparency (which produced confusion and required assistance) is corroborated by the corresponding error rate metric seen before (issues / questions).

7.5 LABORATORY USABILITY TEST 4

Test duration: 22 minutes

Test date: 2015-05-25

Place: Laboratory using a NCR SelfServ 22 ATM

Device tested: Cash dispenser

Version tested: Iteration 20 build

Participants: 1 User (EP)

- The user has performed certifications of this device previously and is familiar with it
- The user has never used the application before

7.5.1 GOAL

Evaluate the usability of the modules: Test suite edition, Execution, and Command runner.

7.5.2 ACTIVITIES

- Running individual commands: A list of commands is given for execution
- Run a full test suite: The test suite is given

- Create and execute a full test suite: Instructions are given for creating and running a desired test suite

7.5.3 RESULTS

7.5.3.1 EFFICIENCY OF USE

The indicator measures the time spent by the user (minutes) versus an estimate for novice users.

	Activity 1	Activity 2	Activity 3	Total	Comments
Estimated time	5'	5'	10'	20'	<i>Time estimated for a user who has never used the application version</i>
Time spent	4' (10.59)	3' (11.03)	15' (11.06)	22' (11.21)	<i>We consider that the user finished in the estimated period of time.</i>

7.5.3.2 EFFECTIVENESS OF USE

The indicator measures the level of completion achieved by the user in each activity.

	Activity 1	Activity 2	Activity 3	Total	Comments
% completion	100%	100%	100%	100%	<i>Everything was completed</i>

7.5.3.3 ERROR RATE

Errors made by the user while using the application. We consider four different error types.

- User error: An error caused by a usability issue.
- Accidental input errors: Not considered a usability issue, except if it's too recurrent or generalized
- Issues and questions: Any other issue found like help requests and questions.
- Bugs: Application errors

	Activity 1	Activity 2	Activity 3	Total
User errors	0	0	1	1
(Accidental) Input errors	0	0	0	0
Issues / questions	3	1	2	6
Bugs	0	0	0	0

Seven drawbacks were found in total, from which, the most prominent are:

- The label "Edit Test Case" is not appropriate for the case of creating and editing
- Difficulty to activate a button with keyboard navigation
- There are difficulty to find functions without using mouse

7.6 THINKING ALOUD PROTOCOL FOR EVALUATING REPORTS

Test date: 2015-05-25

Session duration: 10 minutes

Version tested: Iteration 23 build (latest version)

Scope: Reports module

Place: ATM lab using the user's laptop

Participants: 1 user (EP)

- She had not used the Reports before

The thinking aloud protocol consists on asking users to verbalize their thoughts, feelings, and opinions while interacting with the system. This test is useful for capturing a wide range of cognitive activities. Its main benefit is a better comprehension of the user's mental model and the terms he or she uses to express an idea or function [14].

The method is not designed to measure efficiency or quantitative data. So the test was designed to have only one activity and we only measured the overall time.

7.6.1 GOAL

The goal of this test was to evaluate the usability of the Reports module (the wizard and navigation).

7.6.2 ACTIVITIES

The user was asked to generate a certification report with two test suite results (Device Card Reader and Cash Dispenser). The activity also had a list of features that the report should meet.

7.6.3 RESULTS

The user successfully finished the test without requiring external assistance. A minor bug and two errors were found. The latter were functions missed by the user but finally she found a way for finishing the activity and acknowledged these functions were not really important (opening a report definition vs. creating a new one from scratch).

- **Activity duration:** 10' (minutes)

The activity was finished in a much shorter time than expected. The estimated time was between 20' and 30'.

- **Effectiveness of use:** 100% successfully completed
- **Error rate:**

	Total
User errors	2
(accidental) input errors	0
Issues / questions	5
Bugs	1

7.7 HEURISTIC EVALUATION

Test date: 2015-05-26 and 2015-06-8 (revision)

Version tested: Iteration 23 build (latest version)

Scope: Edition and Execution features only

Evaluators: 1 (developer)

Heuristic evaluation is an inspection method performed by usability experts. It is part of the "Discount Usability" methods [12], which were proposed as a viable alternative for projects with restricted budgets⁷.

In heuristic evaluation, the benefit of the method depends on the number of evaluators. A single evaluator is not strongly recommended as he or she will not discover most of the usability problems. Instead, Nielsen [23] recommends at least 3 evaluators (and better with 5). A model he proposed which relates the number of evaluators and the problems found, is shown in the figure below.

In our case, we applied the method with a single evaluator (the developer) whose knowledge is based on academic training in usability. This means we will discover about one third of the usability problems. However, we find the method worthy whenever the results were complementary with results from previous tests.

⁷ Discount usability consists on: a) Simplified user testing: The idea that testing 5 users was "good enough", b) Narrowed-down prototypes: Using fast to design paper prototypes instead something that embodies the full user experience, and c) Heuristic evaluation: which consists on inspecting user interface designs relative to established usability guidelines [2].

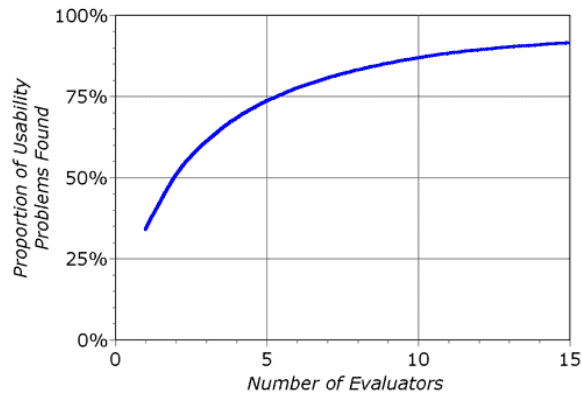


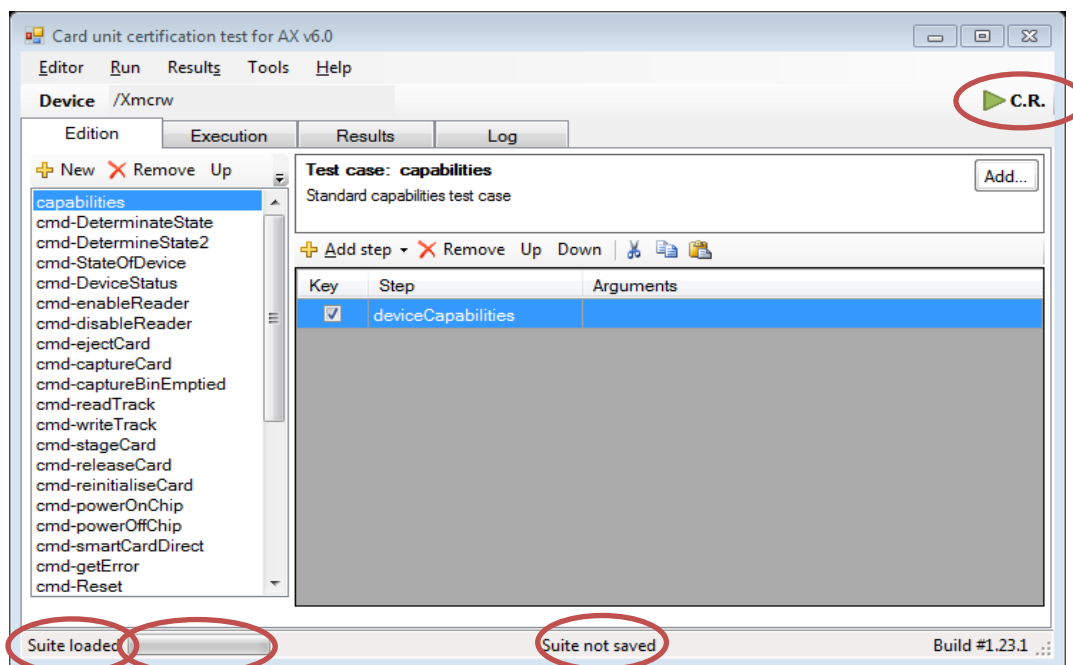
FIGURE 48. Proportion of usability problems found by heuristic evaluation using various numbers of evaluators.
Taken from [23]

7.7.1 GOAL

The goal of this test was to evaluate the usability of the product according to the heuristics recommended by Jacob Nielsen [13] in order to find usability flaws in the product, and obtain recommendations for improvement.

7.7.2 ACTIVITIES

The developer checks the compliance of the 10 heuristics and then makes a list of recommendations to solve the issues. The product evaluated is the latest version (produced in iteration 23). Screenshots of the product are provided below.



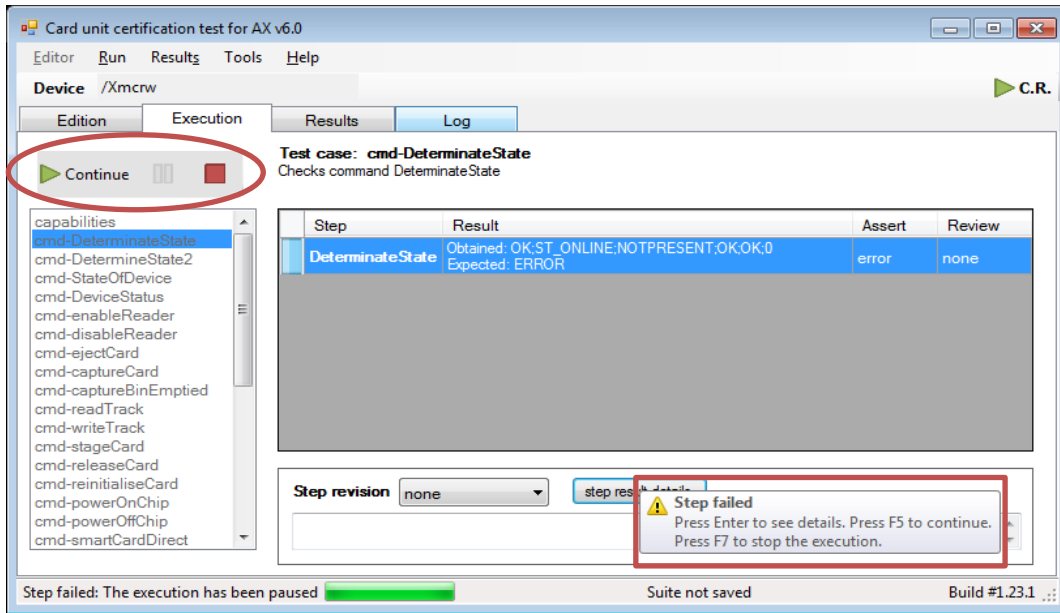


FIGURE 49. Screenshot of the latest product version (edition and execution views)

7.7.3 RESULTS

Below we provide the description of each heuristic, according to [13], and the results of the evaluation for each module. Results are given in terms of strengths and flaws found.

- 1) **Visibility of system status:** The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Module	Strengths found	Flaws found
Edition	<ul style="list-style-type: none"> ✓ UI components are disabled according to App state. ✓ User actions are verbalized in the status bar (during and after the action) ✓ Last saved time label 	<ul style="list-style-type: none"> ▪ No visual hint when an item has been cut, moved or pasted. ▪ Log panel shows details on last actions but is not always enabled. ▪ Current filename is not visible to users
Execution	<ul style="list-style-type: none"> ✓ Progress bar is updated during execution 	<ul style="list-style-type: none"> ▪ Users can't know which test cases have been done and which are not (in the left panel) ▪ Current results file name is not visible to users. ▪ Users can't know if results have not been saved

- 2) **Match between system and the real world:** The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

Module	Strengths found	Flaws found
--------	-----------------	-------------

Edition	✓ Terms: Device, command, event, parameter, etc., are always used along the user interface.	▪ None
Execution	✓ Terms are very generic but common and used by users: Run, Stop, Pause, Continue, Error, Warning, Owner, Comment, etc.	▪ “Assert” is not clear for users outside the project. Better options can be: Assessment/check /comparison/Test Outcome

- 3) **User control and freedom:** Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Module	Strengths found	Flaws found
Edition	✓ None	▪ Undo and redo are not supported
Execution	✓ Pause and stop is allowed during execution	▪ None

- 4) **Consistency and standards:** Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Module	Strengths found	Flaws found
Edition	✓ Shortcut keys assignments follow standard conventions: Open, save, close, copy, cut, paste, etc.	▪ None
Execution	✓ Screen layout for edition and execution are very similar (master-detail forms for test cases). ✓ All views support contextual menus.	▪ Shortcuts are not shown in contextual menus (as they are in others)

- 5) **Error prevention:** Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Module	Strengths found	Flaws found
Edition	✓ The user is informed before exiting, overwriting results, changing the context (opening another suite) ✓ GUI controls are enabled/disabled according to current state	▪ Not all wizards (only report wizard)*use error providers for validating errors while typing editing form fields.
Execution		

- 6) **Recognition rather than recall:** Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Module	Strengths found	Flaws found
--------	-----------------	-------------

Edition	<ul style="list-style-type: none"> ✓ Main functions are explicitly visible in toolbars in each view. 	<ul style="list-style-type: none"> ▪ None
Execution		<ul style="list-style-type: none"> ▪ The user can't know which the previously executed test case was (if he changes the selected item). ▪ The review panel for test cases is somehow hidden or not visible enough despite being one of the most used functions. So users need to learn how to show it up to use it.

- 7) **Flexibility and efficiency of use:** Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Module	Strengths found	Flaws found
Edition	<ul style="list-style-type: none"> ✓ Shortcut keys are enabled for (almost) all functions ✓ Key accelerators (textual menus) 	<ul style="list-style-type: none"> ▪ A way to copy several items (test steps, test cases) is not available. The same for moving, deleting and other editing functions (in batch) ▪ Creating a single test case with a single action, setting the key flag, and setting a baseline requires about 10 clicks, and possibly more than 20 keyboard actions (when there is no a mouse). Of course modeling 100 test cases is error-prone, tedious and really costly. A generator of multiple test cases is required based on the selection of commands for instance.
Execution	<ul style="list-style-type: none"> ✓ Quick button for commands console 	<ul style="list-style-type: none"> ▪ None

- 8) **Aesthetic and minimalist design:** Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Module	Strengths found	Flaws found
Edition	<ul style="list-style-type: none"> ✓ None 	<ul style="list-style-type: none"> ▪ None
Execution	<ul style="list-style-type: none"> ✓ Popups were replaced by status bar messages, small floating balloons and a summary form 	<ul style="list-style-type: none"> ▪ The table with steps results is not minimalist at all. It tries to show everything at once (in text) which is somehow cumbersome for quick reading. And there are important things that are not shown. ▪ Result Details form has similar problems

Screenshot of the issue found: obtained and expected are not clearly distinguished for 1 record, and for many is event worst.

Step	Result	Assert	Review
DeterminateState	Obtained: OK;ST_ONLINE;NOTPRESENT;OK;OK;0 Expected: ERROR	error	none

- 9) **Help users recognize, diagnose, and recover from errors:** Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Module	Strengths found	Flaws found
Edition	✓ None	<ul style="list-style-type: none"> None found. However, an in-depth analysis through other methods can identify risks and kinds of failures that could be produced.
Execution	<ul style="list-style-type: none"> ✓ Typical suite execution errors are explained with a possible solution ✓ Log window provide details in case of runtime failure ✓ Device validator find and inform the user about several kinds of modeling errors 	<ul style="list-style-type: none"> No Auto save during suite execution (Non-implemented requirement, despite ATM reboots are not rare)

- 10) **Help and documentation:** Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Module	Strengths found	Flaws found
Edition	✓ Selected forms provide examples and instructions (event step edition form)	<ul style="list-style-type: none"> (No documentation) Documentation available is limited to configuring new devices
Execution	✓	

7.7.4 RECOMMENDATIONS

From the diagnostics obtained, we now propose the following suggestions for development in priority order (to be reviewed by users):

- Auto-save results and show last save time (as it is used for the test suite in Edition)
- Support undo and redo in the test suite editor
- The execution view should differentiate the test cases that are done from the pending ones (in the global view).
- Support actions in batch: copy, cut, move, paste
- Create accelerators for creating a test case quickly, with a key action and baseline (for instance in 3 clicks)

- f. Simplify the presentation of test steps results (less text more icons)

7.8 USABILITY TEST 5: EFFORT REQUIRED IN A REAL CERTIFICATION BY NON-EXPERTS

Test date: 2016-06-01

Session duration: 1h 48'

Version tested: Iteration 23 build (latest version)

Device tested: Card reader

Participants: 1 User

- The app was fully unknown for him
- He has never made a certification before
- He knows the process at a high level

7.8.1 GOAL

The goal of the test was to measure the time it takes to perform the certification of a single financial device, for a user who never has made a certification before.

7.8.2 ACTIVITIES

The usability test consisted on performing the following activities for certifying the Card Reader unit on a NCR-5886 ATM located in the lab:

- Creating a Test Suite (user 1)
- Running the Test Suite (user 2)
- Generating the report (user 2)

They corresponded to real tests on the ATM device, but were delimited to a subset of the certification test cases because a full certification would be too long (many hours or days for novices). From the 17-page long test suite we selected a 2-page fragment (i.e., 11% of the real case) for designing the activities. The first activity (creation of the Test Suite) was performed with a different user (the developer) to reduce the amount of work and session time for the second user.

7.8.3 USABILITY ISSUES FOUND

7.8.3.1 EFFICIENCY OF USE

The indicator measures the time spent by the user (minutes) versus an estimate for novice users.

	Activity 1	Activity 2	Total	Comments
Estimated time	50'	10'	60'	<i>Estimated time for a user who is novice in certifications.</i>
Time spent	90' (11:10)	18' (12:44)	108' (1:02)	<i>The test delays correspond mainly to lack of information for performing ATM manipulations as required for test cases. And also for lack of knowledge on recovering the device to normal state after an error.</i>

7.8.3.2 EFFECTIVENESS OF USE

The indicator measures the level of completion achieved by the user in each activity.

	Activity 1	Activity 2	Total	Comments
% completion	75%	100%	87,5%	<i>The activity was stopped due to the time limit was exceeded.</i>

7.8.3.3 ERROR RATE

Errors made by the user while using the application. We consider four different error types.

- User error: If the user consciously makes a wrong decision, follows a different procedure, receive a different response, or is blocked and cannot finish the task. This is considered a usability issue.
- Accidental input errors: These correspond to accidental keystrokes and other input actions. This is not considered a usability issue, except if it's too recurrent or generalized
- Issues and questions: Any other issue found is counted here, along with help requested by the user, questions made, and things he tried but did not worked (e.g., inexistent shortcuts).
- Bugs: Application errors are counted here.

	Activity 1	Activity 2	Total
User errors	6	4	10
(Accidental) Input errors	0	0	0
Issues / questions	2	4	6
Bugs	0	0	0

The details of the errors found are not described here for the sake of brevity. However, they are really good points of improvement for the tool.

7.8.3.4 USER SATISFACTION

At the end of the session, we asked the user to fill the UEQ questionnaire⁸, and then we used the data analysis tool to summarize the results. This questionnaire measures six user satisfaction categories through 26 questions. Below we show the results along with the meaning of the categories assessed (according to [11]).

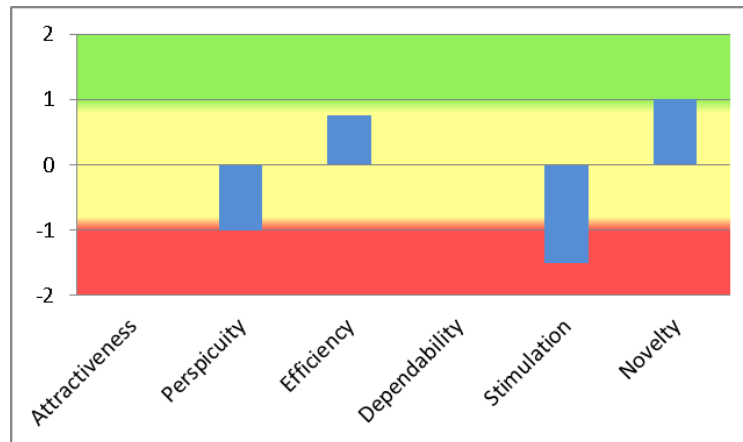


FIGURE 50. UEQ Satisfaction questionnaire results

Attractiveness: Overall impression of the product. Do users like or dislike it?
Neutral.
Perspicuity: Is it easy to get familiar with the product?
Negative.
Efficiency: Can users solve their tasks with the product without unnecessary effort?
Good.
Dependability: Does the user feel in control of the interaction?
Neutral.
Stimulation: Is it exciting and motivating to use the product?
The user strongly disagrees. He feels discouraged.
Novelty: Is the product innovative and creative?
Very positive.

The negative perception of **stimulation** –and neutrality in other categories- reflect the lack of support perceived by the user each time he had not enough information for continuing the tasks. Moreover, keyboard navigation was painful and extremely slow, as he did not use shortcuts and the number of tasks was very high.

Talking with the user after the questionnaire, he emphasized that it was the lack of information and knowledge for doing the tasks what produced so many difficulties when performing the

⁸ <http://www.ueq-online.org/>

activities, however, he recognizes the software was not implied directly in the issues, except for the costly navigation through the keyboard.

8 RESULTS

The resulting application is composed by the following functional modules:

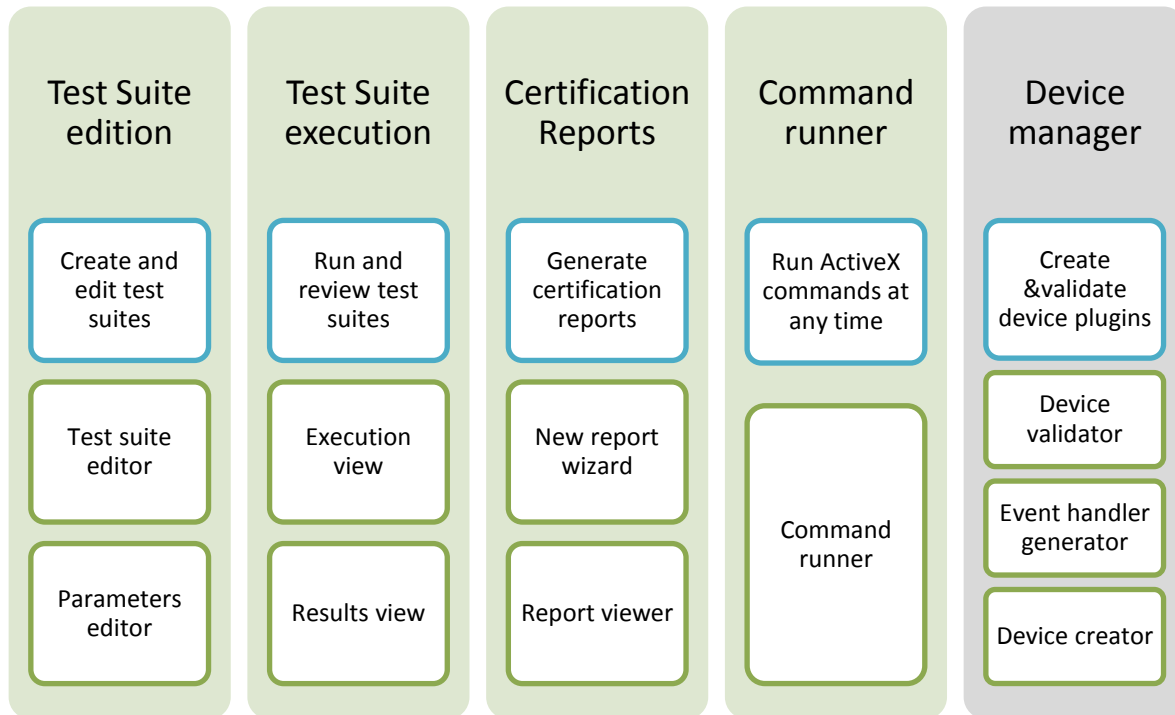


FIGURE 51. FUNCTIONAL MODULES OF THE APPLICATION

The view shows each module with its corresponding responsibility and sub-modules. These modules coincide with the main use cases: Edition of test suites, execution of test suites and commands, and elaboration of reports.

Regarding project goals, the product was ready on time (and budget) and provided a suitable usability and user experience which was measured through several usability tests made by real users in the expected context of use of each task (this is discussed in Chapter 7).

8.1 APPLICATION FUNCTIONAL MODULES

In this section we provide an overview of the main features implemented in the final product.

Edition of Test Suites. Differently to the previous process which required up to 9 or 10 software utilities to perform a single action, the new application provides a unified user interface designed to ease both modeling and execution of test cases. Before, a single test could require several steps which needed to be repeated once an again. However,

with the new tool, test cases can be written once and reused and ran as many times as desired.

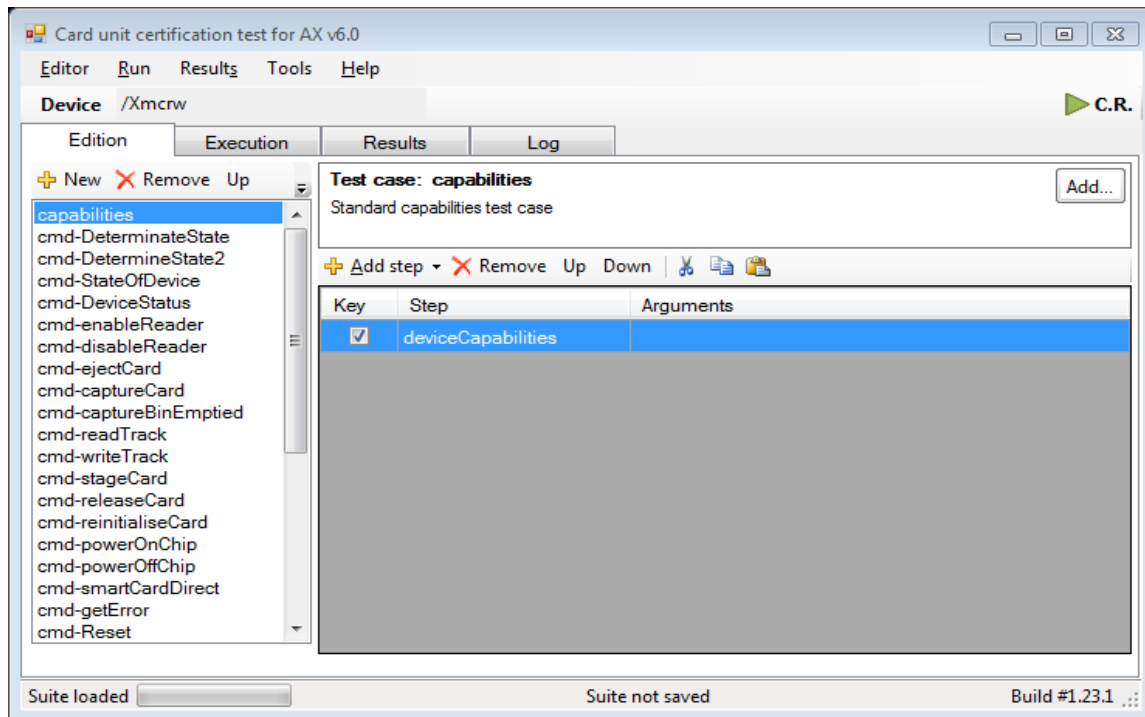


FIGURE 52. TEST SUITE EDITOR

Execution of Test Suites and Commands. It is no longer required to take notes of commands results or events, as everything is recorded by the tool during the execution. System information, registry dumps and other contextual information (test date, test suite version, etc.) is automatically collected by the tool, even when these settings are modified along the session.

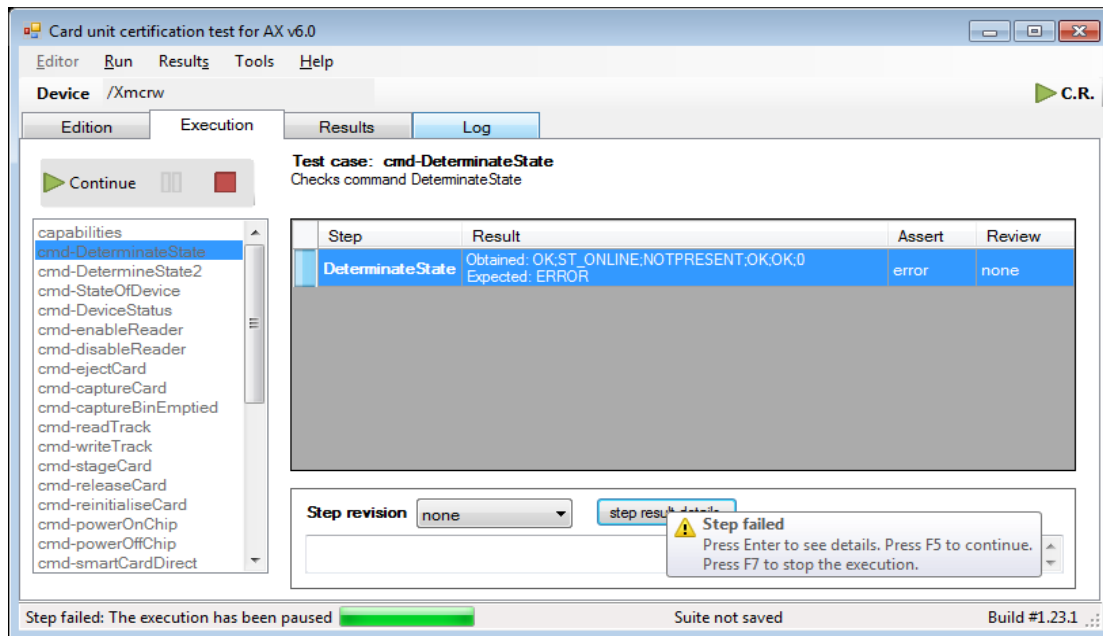


FIGURE 53. TEST SUITE EXECUTION

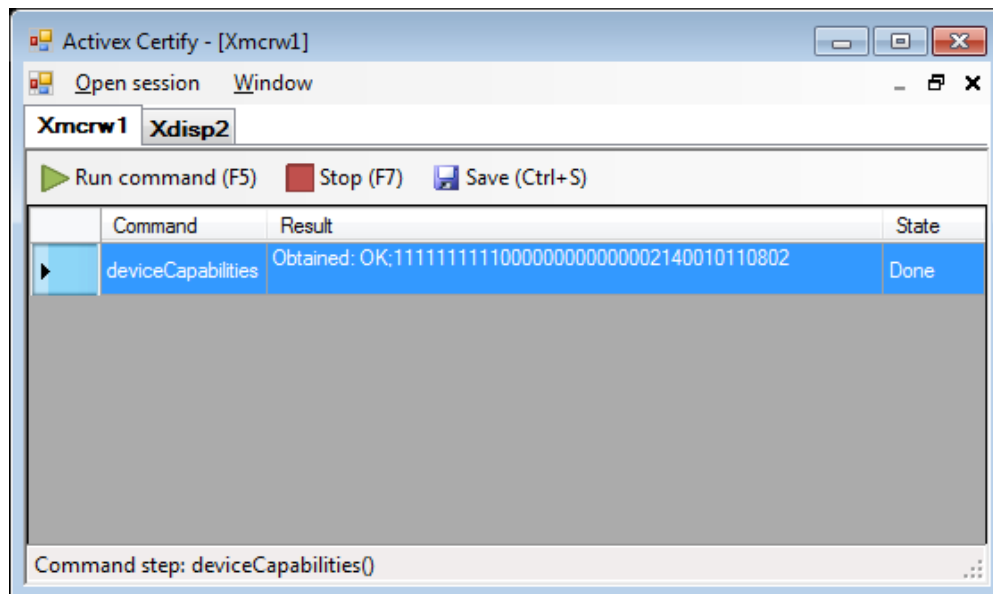


FIGURE 54. COMMANDS CONSOLE (COMMAND RUNNER)

Certification Reports. The reports module was designed to provide the following benefits (which solve the problems mentioned in the introduction):

- The report is automatically generated (as a local web page)
- It is an interactive report with navigation by devices and commands
- It provides a separate view for the failed tests
- Results of different sessions are controlled and easily managed by the application

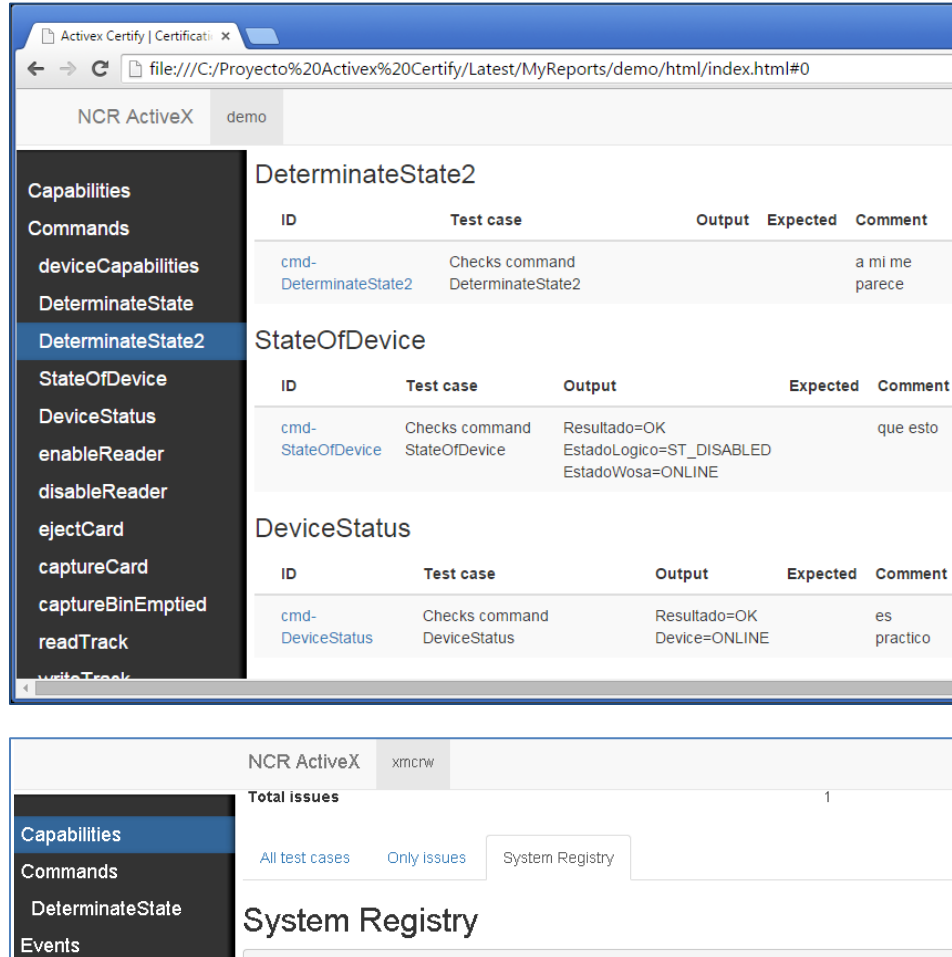


FIGURE 55. CERTIFICATION REPORT

Device manager. This is a product feature that allows the extension of the tool to support future devices without the need of programming or re-compilation of code. Thanks to the architecture used, which follows the principles of Model-Driven Engineering, the ATM's financial devices are modeled as platform independent models using the JSON data format. This extensibility mechanism was an important requirement for easing product maintenance and evolution along time with a low cost. The feature provides functions to create a device, to check the model syntax, and to make a connection test with the AX platform.

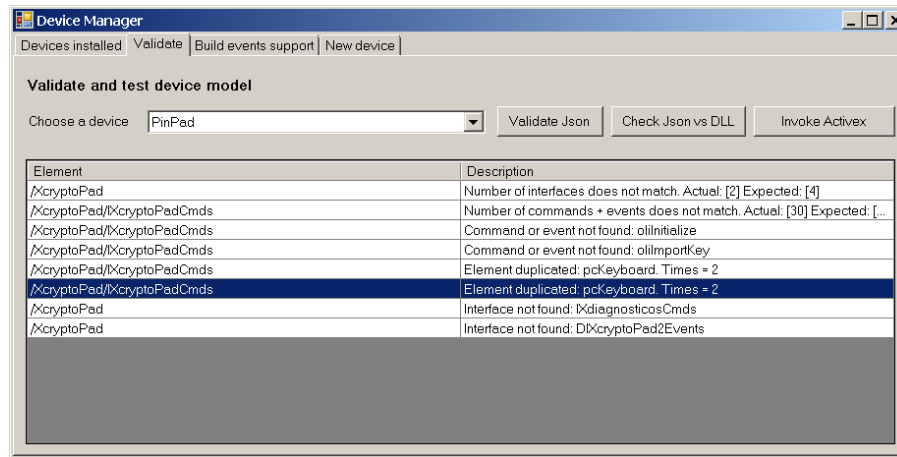


FIGURE 56. DEVICE MANAGER

Finally, the product supports 10 ATM devices, which makes it ready to be used in real certifications.

8.2 PROJECT BENEFITS

8.2.1 TRACKING THE EFFORT REQUIRED BY THE NEW SYSTEM

The primary goal of this usability test was to gather realistic metrics for the effort required by the Certification Process in a realistic scenario. In this section we summarize the times required by each activity and then we extrapolate this numbers for getting the effort of a full certification for a single ATM device.

Activity	Duration (minutes)	Idle time	Effective time	% Finished	% Full certification	Extrapolation to full certification
test suite creation	105'	10'	95'	100%	11%	863,6'
test suite execution	90'	46'	44'	75%	11%	533,3'
Report generation	18'	0'	18'	100%	100%	18'
Total						1414.9'

TABLE 6. Effort required by a full certification with the new tool

The results show that an inexperienced user will spend **23.5** hours in creating a test suite, executing it and getting the certification report for a single device.

However, if the user does not start from scratch but reuses an existing test suite, then he will spend **9.19** hours for getting the same results. Notice that a novice user always will require a previously created test suite; otherwise he or she will not be able to perform the process.

Therefore, the average efforts for certifying one ATM's device are respectively:

Before the project: Effort = 15.3 man-hours (for expert users)

With the new tool: Effort = 9.19 man-hours (for novice users)

Therefore, the tool produces a saving of **39.9%** in time, which could be increased when the process is performed by an expert (which will be always the case, as certifications are only performed by experts).

8.2.2 EXPECTED BENEFITS PER YEAR

After introducing the tool in the Certification Process we can compute the expected benefits during a year based on the statistics of last year (2014-2015), as shown in the following table. The total efforts were computed according to the average time required for certifying a single device (which is 15.3 man-hours).

Customer	Devices/ATM	Hours
Client 1	7	107.1
Client 2	6	91.8
Client 3	8	122.4
Client 4	8	122.4
Client 5	5	76.5
Total effort (2014)		520.2

TABLE 7. Certifications made by the company during last year

We can see that the company performed 5 certifications during last year, which summed 520.2 man-hours (specialist's effort). Now, we can compute expected benefits in terms of time as follows:

$\text{Time savings} = 520.2 \text{ hours} * 40\% = 208 \text{ hours/year}$

Similarly, using a base salary per hour⁹ we can compute expected cost savings:

Cost savings = 8323 EUR/year (*)

Finally, other business benefits are not computed here but are also expected, such as **higher quality**, in terms of:

- Reduced risk of incomplete, inconsistent or missing tests during test execution.
- Elimination of inconsistencies between runtime traces and the final report.
- Reduced manual work (and rework) in both version control and merging of results obtained along several test sessions.

⁹ A base salary rate per hour is tricky to compute (cost for the company is different to the cost for the customer), moreover this is sensible information. So an estimate of 40 EUR/hour was used based on public information on the internet.

9 DISCUSSION

9.1 CRITICAL ANALYSIS OF THE INTEGRATION OF UCD AND AGILE METHODS

Some questions arise regarding the real advantages of using user-centered design (UCD) in a project instead of using agile software methods best practices alone, such as user focus, user involvement, continuous iteration, incremental evolution, frequent testing, etc. If Agile Methods (AM) promotes principles similar to UCD [24], what difference did we find on using UCD instead AM alone?

First, UCD and AM are more and more jointly adopted in industry [24], so it is reasonable to think that their principles and practices will evolve and converge into a set of best practices that cannot be considered as fully independent from each other. In this sense, the benefits reported by using AM can also be reported as benefits produced by applying UCD.

On the other hand, Agile Methods usually say what to do, but not how to do it. For instance, Scrum promotes regular meetings for getting feedback from users, but it does not propose methods or tools to guide the meeting and achieve a goal. UCD has a large variety of concrete usability methods that effectively indicate the steps to perform analysis, design and evaluation. Moreover, UCD offers methods for different kinds of projects and contexts. In this sense, UCD and AM is a lot richer than AM alone.

Finally, two additional perceived benefits of using UCD along agile methods were increased user engagement and higher added-value. The former is because users feel owners of the product – as they were involved during analysis, design and evaluation. This promotes a positive attitude towards the product, eases users' feedback and favors improvement. The second benefit – greater added value- can be explained by the continuous usability evaluation. This is far better than a single final user acceptance test. As stated in the IEEE's definition of software quality, quality is not abstract, but needs to be measured against real users' needs and expectations along with the specified requirements.

9.2 CRITICAL ANALYSIS OF THE USE OF USABILITY PLANNER

The Usability Planner tool (which was explained in Chapters 2 and 3) offered us several benefits which we discuss here along with its possible drawbacks. The benefits we perceived were:

- At the beginning of the project, as it is expected in many projects, we did not know how to start developing the product, and less how to start the user-centered design activities, under so much uncertainty. Here, from just a basic understanding of the project (the goal and actual resources), Usability Planner gave us (almost) immediately

concrete suggestions of UCD activities that could be proposed to the customer (or users) and which could be supported with justified reasons: Make observations, research on similar products, and paper prototypes. And these were effectively what we did.

- As a non-expert usability engineer proposing UCD in a work environment where UCD had not been introduced, this tool really eased the adoption of UCD in practice. It made the first step of UCD for free: The method selection.

The drawbacks and possible enhancements we find in the tool are:

- As a drawback, both Budget and Time constraints are tricky to interpret correctly (there is ambiguity), because almost any project has a limited budget and time. So we can interpret them as: “A project with limited or too limited budget” and the same for the time. But we also can understand them as “A project with or without a limit of budget” (and the same for the time). The fact is that we find examples that exist in all cases (for instance, a company that provides a product for the massive market launching continuous versions through time will not have a “limit of money” as it will if it were a specific project. In the case of our project, we considered the first interpretation: Highly restricted vs loosely restricted.
- As an enhancement, a new method can be added, which certainly is outside UCD but is a complementary practice: “Make a proof of concept of technology”. The method can be recommended whenever the search criteria includes “the developer is not expert in the intended UI technology or the latter is too constrained.

9.3 CRITICAL ANALYSIS OF THE USABILITY METHODS APPLIED

The usability methods we applied during the project were selected on the fly throughout the project development. And now, once the development has finished we want to discuss the contributions of the most relevant:

Ethnographic observations: Seen as a type of field study the first perception is that the method is costly: it requires big upfront design, resources, a place, sample content and several people. However, we find it one of the cheapest and more worthy methods for analysis. Differently from other methods, this actually did not require previous planning or design. After its application, we summarized the findings and presented them to users to verify the coherence and validity of observations.

Paper prototypes: At first we used paper prototype as a quick and dirty design method for internal (developer) purposes: to print several ideas on paper choose the most viable and then present it to users in the form of a wireframe. However, it was a misconception of the method. As the method is for analysis and requires the interaction

of users. Therefore, it is a method that does require planning, elaboration and evaluation. So it is not that cheap to be used for validating 10 different UI designs.

Usability laboratory tests: Usability laboratory tests were the primary method of evaluation, and the feedback obtained was abundant. At first we saw it as a simple-to-apply method once you have the place and resources (i.e., the laboratory) and the people's time (users). However, our first experience was a "failed" test, and the latest was challenging and a bit discouraging. Now we have a broader view of the method costs and benefits. The preparation of test's activities was the most critical and costly step. Before designing the test you need to understand in depth how users work in practice (*know-how*), and the more you know it the better the test script will be. This suggests that usability tests require intensive work making observations beforehand (or using any equivalent method for task analysis).

On the other hand, especial care must be taken regarding usability metrics. Not all evaluation methods are appropriate for all the metrics. Care is also required for designing short and meaningful activities, to save users time but also to find the most critical usability problems. Finally, the selection of users and their distribution for future tests requires careful planning, especially if it is difficult to access users or they are a very small group. Finally, we need to plan the sessions in advance, as users usually have a tight schedule.

10 CONCLUSIONS AND FUTURE LINES OF WORK

The project consisted in the development of an ATM application for assisting users in the AX Certification Process, a software support service that NCR provides to its customers.

Once the project has finished, we ask ourselves: Is the product going to produce real business benefits? How can we measure them? How can we validate the product?

These questions were covered in part in Chapter 7. On the one hand, it is clear that there is no definite evidence on the business benefit yet, as the software has not been used with real customers. On the other hand, we certainly know from the results of usability tests, that the product fulfills the user requirements and users are satisfied with the usability of the product.

Now we analyze more in detail the goals achieved, the benefits of applying UCD during the project and the future lines of work.

10.1 ACHIEVEMENT OF PROJECT GOALS

In the Introduction, we stated the goal of developing a software product for assisting users in the Certification Process by solving the process issues. The solution that the final product offers to each issue is presented in the following table.

Current process problems	Solution implemented
Many process tasks are carried out manually and are error-prone and costly.	Several tasks were automated: <ul style="list-style-type: none">• Execution of test cases• Automatic collection of runtime traces• Automatic generation of the certification report
Tasks that require user interaction require high user efficiency and accuracy	The context of use was studied and the solution took into account the most important constraints: <ul style="list-style-type: none">• Tasks are grouped by activity: Edition, Execution, and Reporting (each activity is performed in a different context)• The tool is fully accessible through keyboard.• The tools is self-contained and portable (no installation requirements)• And efficiency and error rates were measured and continuously improved through usability tests.
Users can certify only the devices they know. The know-how for efficient testing depends on each specialist and it is not	The test suites for certifications are modeled by experts, but can be executed by any user. Test cases guide the users with instructions and provide the information they need to perform each activity. The know-how is now explicitly written in the Test Suites.

explicitly documented.	
Very old and new ATM models can require certification and need to be supported	<p>The software is extensible: It can support old and new ATM devices through device models without requiring programming.</p> <p>Moreover, the software works on .NET 4.0, which is supported by all the operating systems in the project scope</p>

TABLE 8. Achievement of project goals

10.2 BUSINESS BENEFITS OF AUTOMATION

The expected project benefits were the efficiency and quality improvement of the Certification process. That is, an improvement of a real process in a real company. This makes the project worthy and interesting. As it impacts users (employees), customers and the business as a whole.

Companies like General Electric (GE) have proven the power of 1% of efficiency improvement in other industries. They say that an industry can report impressive savings even with 1% of efficiency improvement, because savings are proportional to the economical size of operations along several years.

In our case, the efficiency improvement was 40%, and although the size of operations is not comparable to sectors like avionics or mining, this paves the way for bigger process improvements in the future.

10.3 LESSONS LEARNED ABOUT OUR USE OF UDC

From our experience using UCD along the project we conclude several aspects. First, UCD is not technology-agnostic (in the sense of “fully independent”). The programming language and underlying technology can provide advantages or impose important restrictions for achieving good usability. Moreover, without proven skills in software architecture and user-interface implementation the designs cannot become a reality. Therefore, it is important to continually build functional proof of concepts using the implementation technology before investing too much effort on interaction design. Otherwise, implementation will be too expensive or even out of the developers’ scope.

Second, UCD methods can be easy to apply but it is also easy to apply them wrongly. Some methods seem equivalent but their purposes and the procedures are totally different. For instance:

- Paper prototyping Vs. wireframes
- Observations Vs. training sessions

- Usability tests Vs tutorials

In particular, we lately discovered that our use of paper prototyping was incomplete, because we use it as a wireframe to illustrate a design and present it to users before any coding. However, users need to interact and use the paper prototypes, and only after this, design decisions can be made. The developer must be an observer and facilitator.

Third, we learned that is better to focus the *main UCD effort* on critical user activities first. In our case, the execution of certification tests was the critical activity; it required a lot more iterations until getting a feasible and usable implementation (more than 12 different designs). So usability is important but it has a cost, so focus helped us to use our resources and time better.

Fourth, as a developer is often difficult to approach users to make them part of the development team. There is always a fear on showing a product that objectively can be minuscule and unworthy (as unfinished software). And this is an obstacle for getting early and frequent feedback. In our case, UCD helped to involve users more naturally along product design and evaluation. And users also found UCD worthy, interesting and motivating.

10.4 BENEFITS AND COSTS OF USABILITY EVALUATION

Our main evaluation method was the laboratory usability test. We discovered important advantages of applying it, like the strengthening of user commitment and collaboration. Moreover, it proved to be a very objective feedback method. On the other hand we also discovered the real cost behind usability tests, especially in the design of the activities (as mentioned in Chapter 9).

Doing a retrospective of the last usability test which was the most exhaustive test, and produced somehow “disappointing” results, let us learn several lessons.

We've discovered that the way the expert user makes the real activity is different from how the usability test was designed, which was based on the certification template. We assumed that a certification test suite could be designed with just that template. And the result was that the participant needed to perform too much tasks for getting a small number of tests done, while the expert usually groups similar tests in a single test case so he or she does not need to repeat the same flow of actions for each one. Of course, the efficiency achieved is very different.

So a first lesson was that usability tests are more realistic if they are prepared with expert users.

Then, after discussing this fact with other users, they said that each one has a different approach for doing (and modelling) the certification tests. So it is not required to model the tests exactly as described by one user. What is important is that the tool allows everyone to model Test Suites the way each one wants. So the second lesson was that users must be involved, not only individually but also as a group, before making decisions that can be biased.

In conclusion, we acknowledge the important feedback usability evaluation gave us during development. But also we discovered that the design of these tests requires real insight in users' practices which cannot be deducted just by asking users or by looking at report samples. In our case, additional observations could have anticipated this discovery and help design more adequate usability test activities.

10.5 FUTURE WORK

The AX Certify software can be improved and enriched a lot. Effectively, it requires extensive testing in several platforms, to enhance the support to all current ATM devices and ultimately it can be improved after its use in real AX Certifications.

Moreover, the biggest drawback of the software is that it requires all ATM tests be perfectly modeled in test suites before each testing session. The number of test cases that needs to be written is huge and this intimidates users to not use it in practice. A solution for this is to generate the certification test suites from previous testing traces (e.g., logs). These logs usually have all the information required to reconstruct a model of a test suite.

Another focus of work is on standardizing the certification test suites through guidelines. These can involve conventions for test case naming, minimum and maximum size of textual descriptions, a general approach for modeling the tests according to best practices, etc. The final goal of this is making explicit the underlying knowledge of each specialist in a way that can be used by non-experts and new employees.

A final feature that lacks the software is testing metrics. Good practices in software testing involve measuring key factors such as test coverage, performance profiling, etc. Thus, a certification report that shows 100% passed tests with 90% of function coverage is better than another that shows 100% passed tests with 50% of function coverage.

Additional improvement ideas can be generated by involving users through analysis sessions and continued use of the software.

11 REFERENCES

- [1] Bevan, N. (2009). Criteria for selecting methods in user-centred design. I-USED'09 Workshop, INTERACT 2009, Uppsala, Sweden.
- [2] Ferré, X.; Bevan, N. & Antón Escobar, T. (2010) UCD Method Selection with Usability Planner. Proceedings of NordiCHI 2010, Reykjavik, Iceland.
- [3] Ferré, X.; Bevan, N. (2011) Usability Planner: Development of a Tool to Support the Process of Selecting Usability Methods. Proceedings of INTERACT 2011.
- [4] Usability Planner web application
<http://raptor.ls.fi.upm.es/usabilityplanner/activities/>
- [5] Silva da Silva, T.; Martin, A.; Maurer, F.; Silveira, M., "User-Centered Design and Agile Methods: A Systematic Review," *Agile Conference (AGILE)*, 2011, vol., no., pp.77, 86, 7-13 Aug. 2011 (doi: 10.1109/AGILE.2011.24)
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6005488&isnumber=6005485>
- [6] Chamberlain; Sharp; Maiden. Towards a Framework for integrating agile development and user-centered design. Springer-Verlag. p143-153. 2006
- [7] Nigel Bevan. *UsabilityNet Methods for User Centred Design*. Volume 1 of the Proceedings of HCI International 2003, 434-438. Lawrence Erlbaum. 2003
http://www.nigelbevan.com/papers/Usability_methods.pdf
- [8] Usability Body of Knowledge. *Context of use Analysis*. (online)
<http://www.usabilitybok.org/context-of-use-analysis> [Accessed on December of 2014]
- [9] Deborah J. Mayhew. *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design*. 1999.
- [10] Tellez, Leonardo; Ferré, Xavier. NCR ActiveX Certifier Software Project. May, 2015.
- [11] UEQ User experience questionnaire.
<http://www.ueq-online.org> [retrieved on June 2015]
- [12] Nielsen, Jacob. Discount Usability: 20 Years. 2009 (online)
<http://www.nngroup.com/articles/discount-usability-20-years> [retrieved on June 2015]

- [13] Nielsen, Jacob. 10 Usability Heuristics for User Interface Design. 1995 (online)
<http://www.nngroup.com/articles/ten-usability-heuristics> [retrieved on June 2015]
- [14] Thinking aloud protocol. Usability Home (online)
<http://www.usabilityhome.com/> [retrieved on June 2015]
- [15] User testing tips (online)
<http://wiki.fluidproject.org/display/fluid/User+Testing+Tips> [retrieved on June 2015]
- [16] User testing protocol (online)
<http://wiki.fluidproject.org/display/fluid/User+Testing+Protocol>
[Retrieved on June 2015]
- [17] System Usability Scale (SUS) (online)
<http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>
[Retrieved on June 2015]
- [18] Nielsen Jacob. Why You Only Need to Test with 5 Users. 2000
<http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>
[Retrieved on June 2015]
- [19] Planning a Usability Test (metrics). Usability.gov (online)
<http://www.usability.gov/how-to-and-tools/methods/planning-usability-testing.html>
[Retrieved on June 2015]
- [20] Nielsen, Jacob. Success rate: the simplest usability metric. 2001 (online)
<http://www.nngroup.com/articles/success-rate-the-simplest-usability-metric>
[Retrieved on February 2015]
- [21] Usability metrics and examples (online)
<http://www.userfocus.co.uk/articles/dashboard.html>
http://www.userfocus.co.uk/pdf/benchmark_report.pdf
<http://www.userfocus.co.uk/articles/making-usability-metrics-count.html>
<http://www.userfocus.co.uk/articles/satisfaction.html>
[Retrieved on February 2015]
- [22] Measuring Errors in the User Experience. 2012 (online)
<http://www.measuringu.com/blog/errors-ux.php>
[Retrieved on February 2015]
- [23] Nielsen Jacob. How to conduct a Heuristic Evaluation. 1999 (online)
<http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation>

[24] Stephanie Chamberlain, Helen Sharp, Neil Maiden. *Towards a Framework for Integrating Agile Development and User-Centred Design*. Extreme Programming and Agile Processes in Software Engineering. Lecture Notes in Computer Science. Springer Berlin Heidelberg. 2006

http://dx.doi.org/10.1007/11774129_15 [retrieved on June 2015]

APPENDIXES

12 APPENDIX A. TIMELINE OF APPLIED UCD METHODS PER ITERATION

In the following sections, we provide a summary of goals, methods and results along the time by iterations. We grouped this overview by milestones.

Conventions

Analysis Methods:

- TRAIN: Training in the ATM lab
- MEET: Meetings with users
- EXAMP: Reading of written material
- OBS: Ethnographic observation
- COMPET: (Competitors analysis) Research on similar tools

Design Methods:

- PROTO: Paper prototypes
- MAP: Navigation map

Construction methods:

- WIREFRAME: Wireframes
- IMPL: Software coding

Evaluation methods

- MEET: Present products to users and get verbal feedback
- UTEST: Usability test
- DIARY: Diary test

12.1 MILESTONE 1: INCEPTION AND PROOF OF CONCEPT

Iter.	ANALYSIS	DESIGN	CONSTRUCTION	EVALUATION	WORKING PRODUCT	PRODUCT DESCRIPTION
1	ATM platforms (TRAIN)					- Technical concepts and business entities
	Business process analysis (MEET)					- Process description: activities, users, outcome, tools. - Project goal
2	Use cases and product features (MEET & EXAMP)	Product Concept (PROTO)			1 st Analysis document	- Use cases and product features - Design of product concept
	Specification of context of use & Requirements elicitation (OBS) x2					- Context of use focused on technical restrictions - Initial functional requirements set
		General architecture design and selection		Review of analysis document	1 st Design documentation	-Architectural views: functional modules and

Iter.	ANALYSIS	DESIGN	CONSTRUCTION	EVALUATION	WORKING PRODUCT	PRODUCT DESCRIPTION
3		of programming language		(MEET)		deployment -Decision of programming language
		Design of entities and design of execution of test suites	Implementation of proof of concept in C# (IMPL)	Review of design document (MEET)	Windows App in C# (Proof of concept)	- UML class and sequence diagrams - Application for test suite loading and execution from a Json file
	GUI layout research (COMPET)	Screen layout (PROTO) Screen layout (WIREFRAME)			GUI screen layout design	

12.2 MILESTONE 2: FIRST STABLE VERSION

The activities performed during the iterations included in Milestone 2 corresponded to additional observations, competitor's analysis, intensive software development, and the first usability tests.

Iter.	ANALYSIS	DESIGN	CONSTRUCTION	EVALUATION	WORKING PRODUCT	PRODUCT DESCRIPTION
4	Analysis of requirement for <i>Events</i> (MEET)	Design of OO classes & presentation of <i>Events</i> (PROTO) Events view design (WIREFRAME)		Review of latest documents (MEET)	GUI screen layout design & business logic design	
5	Research on apps for testing (COMPET)		Setting of development environment (IMPL)		1 st version of the application	- Build first implementation version (loading and running generic test suites)
6		GUI design for results visualization (PROTO)	New GUI with results visualization (IMPL)	Validation of analysis and requirements (FIELD)	New GUI with results visualization	- App with results panel (tests ok and failed)

Iter.	ANALYSIS	DESIGN	CONSTRUCTION	EVALUATION	WORKING PRODUCT	PRODUCT DESCRIPTION
7	Study of 1 st ATM device requirements (EXAMP)	Design of dynamically generated input forms (PROTO)	Input forms coding & new device (IMPL)	Iteration review (MEET)	Application with edition forms and 1 fully-supported device	<ul style="list-style-type: none"> - App with edition of test suites - New device supported: card reader. - Input forms allow composite parameters. Forms are generated from Json models.
8		Command runner window (WIREFRAME)	<ul style="list-style-type: none"> - New kinds of input forms parameters - New device - Device validator (IMPL) 	Usability test # 0 (card reader) (UTEST)	<ul style="list-style-type: none"> - Revised requirements doc - Application with 2 supported devices 	<ul style="list-style-type: none"> - New device supported: receipt printer. - The test was “a failure”, however produced: <ul style="list-style-type: none"> - 11 requirements for improvement - 2 suggestions
9			Coding of usability test findings (IMPL)		New application version	
10			New results view (diff with baseline). Big code refactoring (IMPL)		New application version	
11		“Suite execution” map (MAP) Full new UI design (WIREFRAME) Business-oriented test case modeling proposal (PROTO)		Usability test #1 (card reader) (UTEST)	1st Stable application version	Usability Test found 14 issues User perception was very positive.

12.3 MILESTONE 3: SECOND STABLE VERSION

Similarly, below we describe the methods and results obtained during the iterations encompassed in this milestone.

Iter.	ANALYSIS	DESIGN	CONSTRUCTION	EVALUATION	WORKING PRODUCT	PRODUCT DESCRIPTION
-------	----------	--------	--------------	------------	-----------------	---------------------

Iter.	ANALYSIS	DESIGN	CONSTRUCTION	EVALUATION	WORKING PRODUCT	PRODUCT DESCRIPTION
12		12 design alternatives for execution view (WIREFRAME) Review of "execution view" UIs (COMPET) Study of JUnit architecture	New execution view UI (IMPL) State machine for runner (Proof of concept)	Review of execution view (MEET)	Application with 2 new views: Execution and Results	Fully renewed execution view
13			Change to execution results: Event traces (IMPL) Command runner (IMPL)		Application with Command runner	
14	Report requirements (MEET/EXAMP)		Proof of concept for Reports HTML (IMPL) Json (Device) generator macro (IMPL)		Guide for generating new devices (doc) Macro for creating device models (VBA) Report requirements (doc)	Preparation of tools for giving users ways for creating new devices
15	Users presentation for creating devices (MEET)		Devices DLL generator (IMPL)			Automation of steps of adding devices to the application
16		Design of reports Design of report generator (WIREFRAME)	Report generator Report viewer (jQuery) New device (IMPL)		1 st working version of reports Application now supports 3 devices	- New device supported: cash dispenser
17		Design of report versioning	Integrate user-made new device (IMPL)	Usability Test #2 (cash dispenser)	Application now supports 4 devices	- New device supported: note acceptor
18		4 designs for test case execution and results (WIREFRAME)	Coding of usability tests findings. Integrate 2 user-made devices.	Analysis of previous usability test	Application now supports 6 devices	- Usability test findings: 18 issues (command runner was not usable). - New devices supported: SIU & PINPAD.

Iter.	ANALYSIS	DESIGN	CONSTRUCTION	EVALUATION	WORKING PRODUCT	PRODUCT DESCRIPTION
19		New suite wizard (test case generator) (PROTO)	New report sections. New command runner UI. Device validator from IDL (proof of concept). (IMPL)			
20			Change of report section (capabilities). Coding of user test findings. Device creator from AX DLL. (IMPL)	Test of suite edition (DIARY) 2 developer usability tests (UTEST)		- User test was performed in a single day without direct observation: 2 issues found. - A full device can now be generated with minimal manual effort.
21	Requirements for new internal report (MEET)		New report section New device validation rules (IMPL)			-All existing devices were reviewed against the new validator
22			New input form parameter types All device models are fixed (IMPL)	2 developer usability tests.	2ND Stable application version	2 devices are now “fully supported”
23			Coding of user test findings. Integrate 2 user-made devices. (IMPL)	Usability test #3 (cash dispenser). “Thinking aloud” test (reports).		Usability tests results: - developer test: 3 issues - user tests: 7 issues
					Final application version	Devices supported: 10 Report types: 1

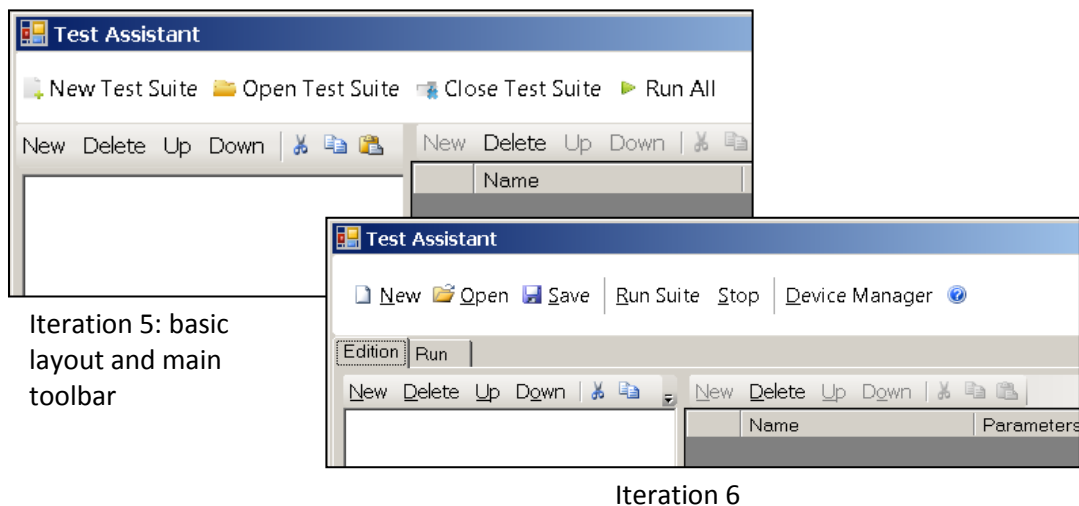
This milestone produced the latest application version that ultimately was used in the evaluation test for measuring the project benefits. The resulting application is presented in Chapter 8.

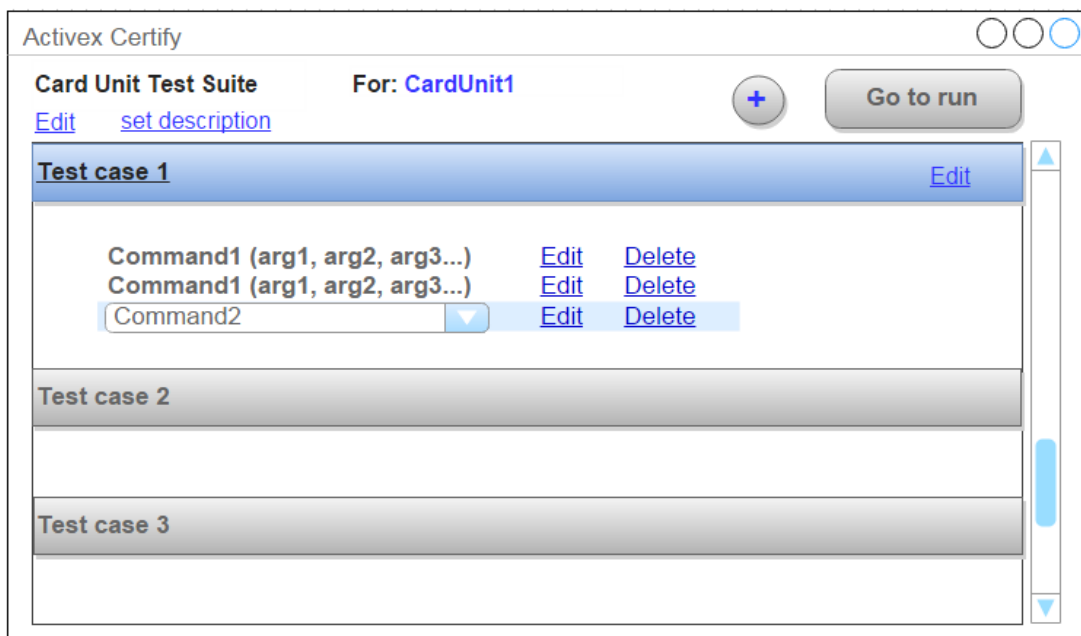
13 APPENDIX B. INTERACTION DESIGN ITERATIONS FOR OTHER MODULES

In this appendix we show the process of interaction design of additional features of the product not discussed in the document: Edition view, Reports, Device Manager, and others. We describe this evolution by grouping the results in product features. The description is rather visual through images, because of the large number of small changes. We describe and highlight the most important user interface variations in the captions of figures.

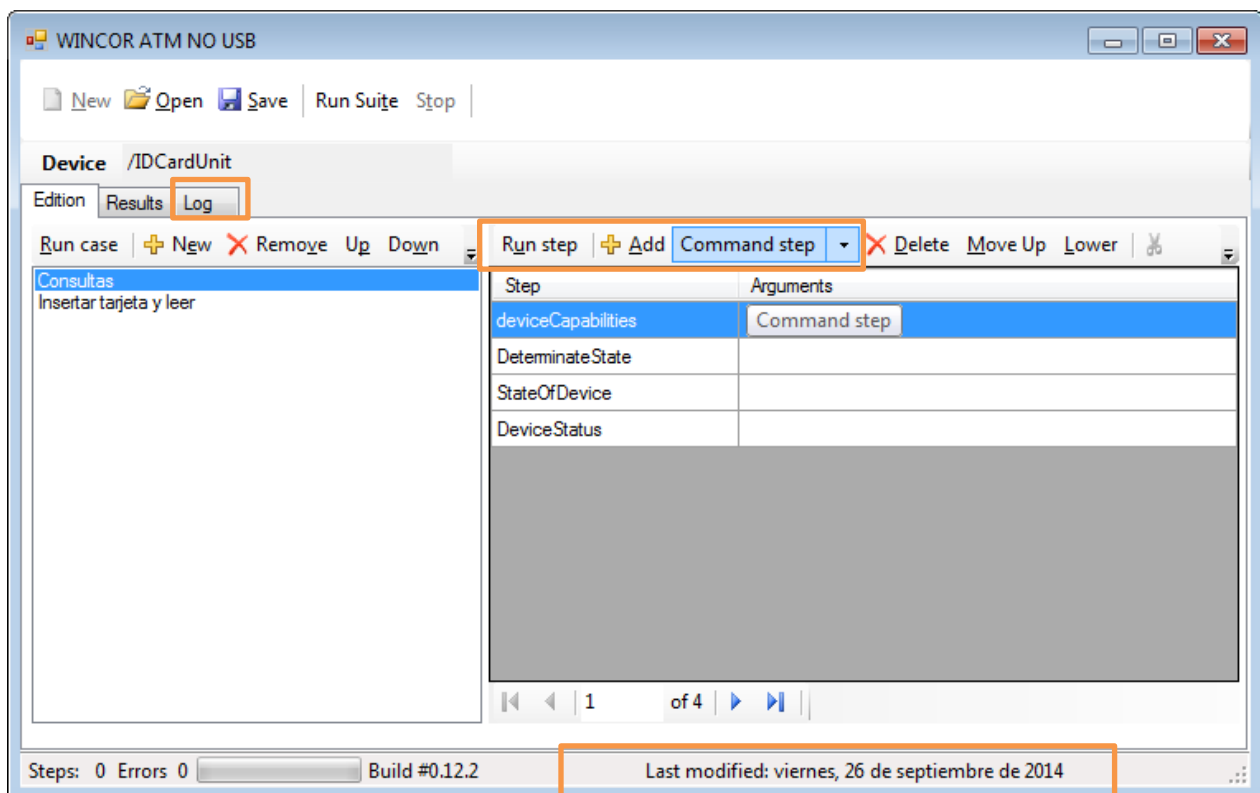
13.1 THE EDITION VIEW

This section describes the evolution of the edition view. This view consists on the presentation of test cases and the edition options available

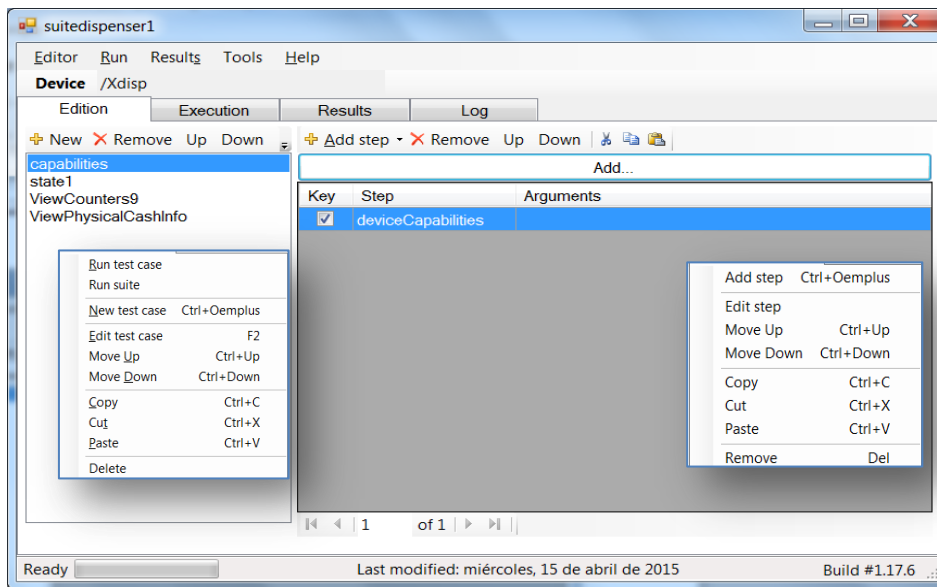




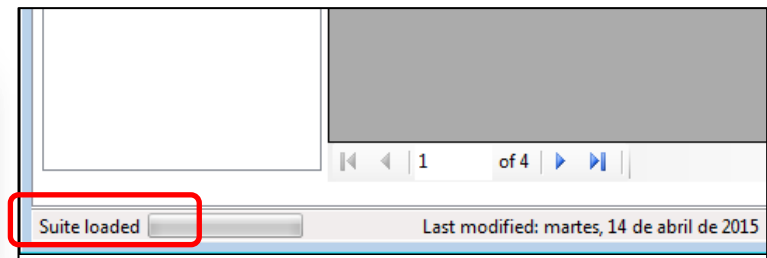
Iteration 11. Wireframe (mono-panel)



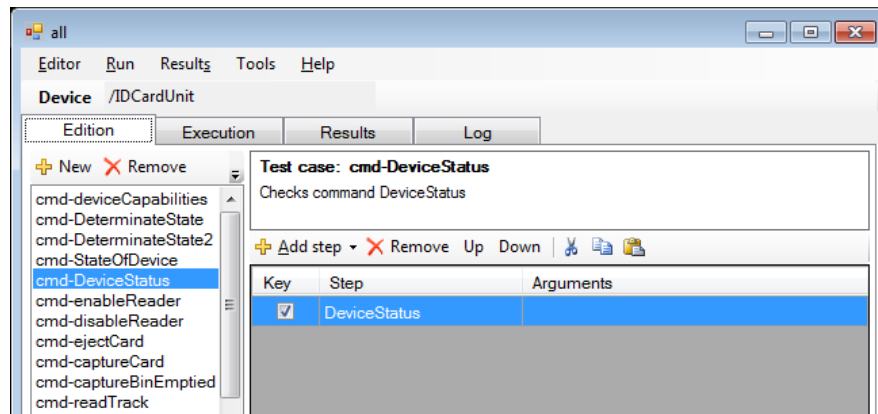
Iteration 12. Important changes are highlighted: Run and stop functions. Log panel now is a separate tab, last save label (most of them added in Iteration 10)



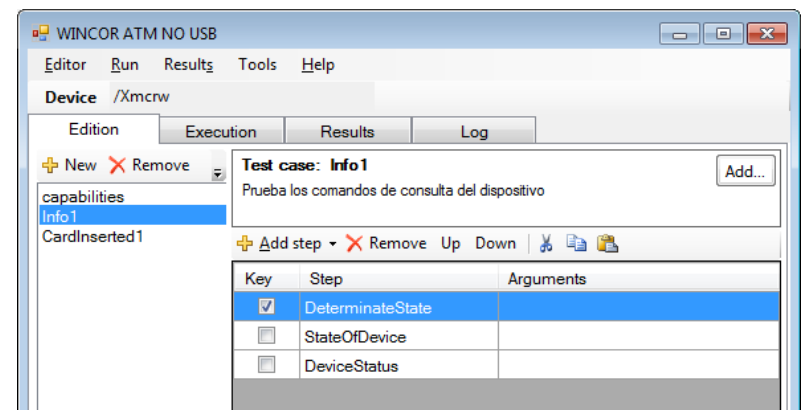
Iteration 17 (Stable version: Contextual menus and add button)



Iteration 19 (current state visible)



Iteration 20 (test case header)



Iteration 22 (new add button)

13.2 COMMAND STEP EDITION FORM (DYNAMIC UI)

readTrack

Permite leer las pistas de la tarjeta introducida en el lector

trackNumber1
Determina si se quiere leer la pista 1.
Si se quiere leer la pista 1 físicamente.

trackNumber2
Determina si se quiere leer la pista 2-
Si se quiere leer la pista 2 físicamente.

trackNumber3
Determina si se quiere leer la pista 3.
Si se quiere leer la pista 3 físicamente.

Ok Cancel

Iteration 7

smartCardDirect

Invoca la ejecución de un comando del sistema operativo del Chip. Para aclarar los datos de entrada se debe tener en cuenta que cualquier comando que se ejecute en el Chip tiene el siguiente formato CLA INS P1 P2 Length Datos.

PROTOCOL
Protocolo de comunicaciones del S.O. chip
Significa protocolo T0.

DATA
Datos que se envían al chip. Formato: cadena de caracteres Hexadecimales.
a0 a4 03 f0 2 data
[Edit](#)

Ok Cancel

Iteration 8 (parameters with linked edition forms)

CashDispense

Saca los billetes solicitados de los cajetines y los presenta al cliente.

Cantidad
Cantidad solicitada por el cliente
20

Currency
El tipo de divisa en la que el cliente ha solicitado la Cantidad. Son tres caracteres con la denominación según no
EUR

MixNumber
Algoritmo o tabla de dispensación CEN XFS 3.x:
- "1": Mínimo número de billetes.
- "2": Vacío equitativo de cajetines.
- "3": Máximo número de denominaciones.
- "x > 3": Tablas especificadas por aplicación.

Position
Indica la posición de salida de los billetes (CEN XFS 3.x)
Posición nula. El SP del fabricante determinará la posición.

Ok Cancel

Iteration 18 – Layout issues

13.3 EVENT STEP EDITION FORM

Add event action

Select the key event: (used by the test case)

/Xdisp/XdispCmds/billsTaken

Select additional events you expect: (Use space to check the boxes)

/Xdisp/XdispCmds/billsTaken

Set a timeout for the waiting:

1 (default 1 s)

Ok Cancel

Iteration 19 (First of two screens for setting the test step)

Add event action

Wait events:

1) Expected event

cardInserted

2) Expected event data (let it blank if you dont want to check event data)

EVT_INSERTED_OK

3) Additional events you expect in any order (Use the space key to check the boxes)

cardInserted
cardRemoved
cardInsertedError
cardRemovedError

4) Set a timeout for the waiting 4 (default 1 second)

Ok Cancel

Iteration 22. Simplified single edition window

13.4 REPORT VIEWER

Test case	Group by Key action	Key action result	Expected value	Comparison
	ReadTrack			
	Descripción	Resultado	Esperado	Estado
	Lectura de una tarjeta insertada previamente	OK; Valores de las pistas	OK; Valores de las pistas	OK
	Lectura de una tarjeta insertada previamente después de ser grabada	OK; Valores correctos de las nuevas pistas	OK; Valores correctos de las nuevas pistas	OK
	Lectura sin tarjeta dentro	OK; Datos tarjeta* *Cuando esta es insertada	OK; Datos tarjeta* *Cuando esta es insertada	OK
	Lectura con la tarjeta en la zona chip	OK; Datos tarjeta	OK; Datos tarjeta	OK
	Ejecución tras un comando STAGECARD	OK; Datos tarjeta	OK; Datos tarjeta	OK

Iteration 14. Concept of report (taken from real report sample)

Load test suite results

Choose File

myresult.results

Load

WINCOR ATM NO USB, /IDCardUnit

Contents

- [TC1 - Comandos de consultas](#)
- [TC2 - Insertar tarjeta y leer](#)

Test cases

- TC1 - Comandos de consultas**

Outcome: fail

Comment: El dispositivo no responde correctamente

Expected	Output	Outcome	Revision
OK:1111111111000000000000002140010110802	ERROR	fail	
OK:ST_ONLINE;NOTPRESENT;OK:OK:0	ERROR:ST_NODEVICE	fail	
OK:ST_DISABLED;ONLINE	ERROR	fail	
OK:ONLINE	ERROR	fail	

WINCOR ATM NO USB /IDCardUnit

Overview

Errors from NCR

Errors from Wincor

Warnings from NCR

Warnings from Wincor

Commands

Events

Load report

Choose File

customer_sample.json

Open a report from a ".results" file

Load

WINCOR ATM NO USB /IDCardUnit

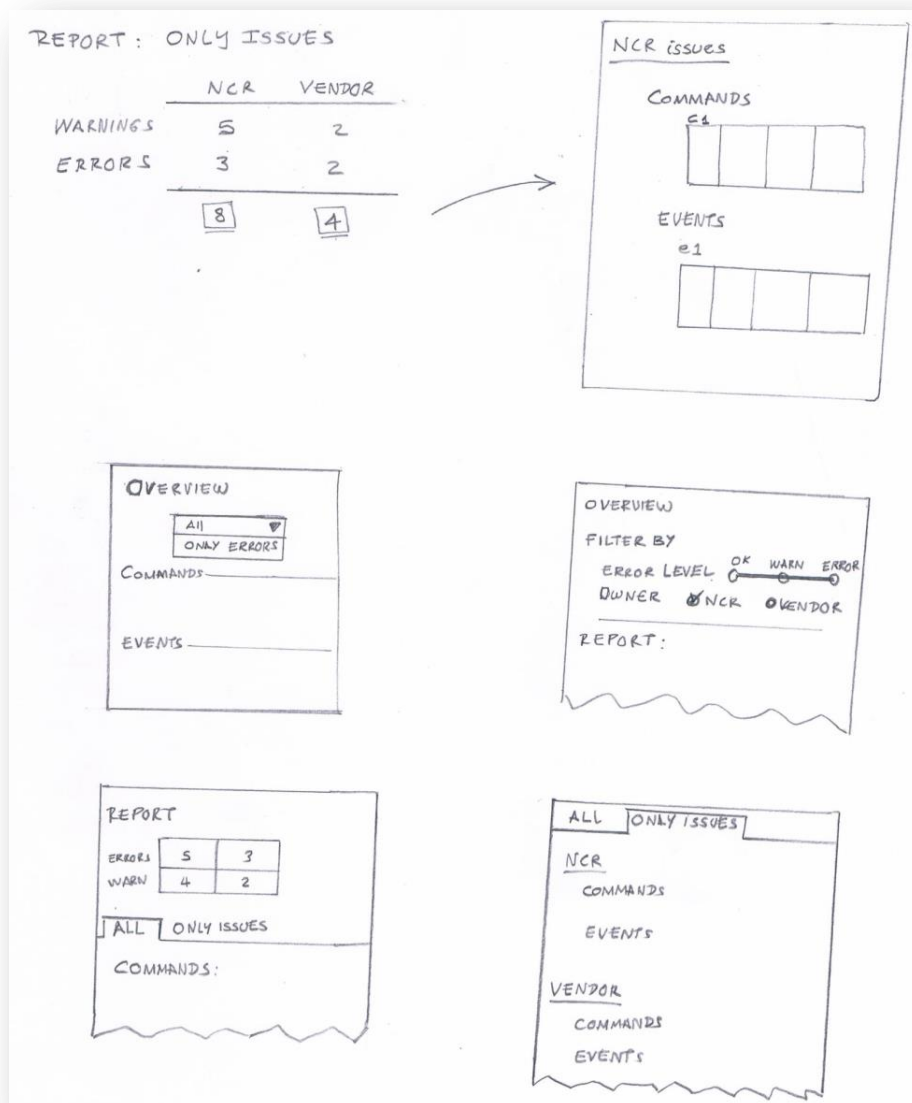
La homologacion se hace sobre el ActiveX version 6.0.0.0, que es la version actual ut

Overview

	NCR
Errors	10
Warnings	10
Total issues	20

Iteration 16. New implementation using the Bootstrap framework

Iteration 14. First proof of concept integrated to the app



Iteration 16. Paper prototypes of Report

Design alternatives for the customer report

For displaying results by owner and error level we have the following key concerns:

- Build a concise content unit which can be fluently read or event be printed
- Promote easy navigation for reading the full report and also for going directly to issues
- Avoid content repetition: a test case can appear in the list of test cases and the list of failed test cases (this is the problem of having a single page)

Device1

Device2

Device3

>>

Lector de tarjetas (IdCardUnit)

+ Overview

	NCR	Manufacturer
Errors		
Warnings		

+ Capabilities

+ Commands

+ Events

Iteration 16 Wireframe of full certification report (multi-device)

NCR ActiveX

Report 1

Report 2

Card unit /IDCardUnit

Overview

Capabilities

Commands

Events

Card unit /IDCardUnit

Description: La homologacion se hace sobre el ActiveX version 6.0.0.0, que es la version

Creation date: 2015-03-16T09:25:56.7950878+01:00

Device: /IDCardUnit

ActiveX version: 6.0.0.2

Overview

	NCR
Errors	10
Warnings	10
Total issues	20

Iteration 16. Implementation of multi-report

13.5 REPORT GENERATOR

Report

Date 1/1/2016

Title Homologacion Wincor - ActiveX 6 para XYZ Corp

Customer XYZ Corp

ATM Vendor Wincor **Model** Wincor XE **XFS** XFS 3.0

Operating System Windows XP **Update** SP3

Report version 1

Results

Add **Remove**

Title	Report
Pinpad	./pin.results
Card Unit	./card.results

Save **Generate report**

Iteration 16. Wireframe of New report wizard

Report Wizard

Please give a name for the report (which should be also a valid folder name)

Report name CustomerReportV1

Location C:\Proyecto Activex Certify\Latest\MyReports\CustomerReportV1

Add new **Delete** **File** **Select**

C:\Proyecto Activex Certify\Latest\data\3.results **Browse**

Report Wizard

Report name CustomerReportV1

Version 1

Introduction text Certificacion realizada sobre el modelo de NCR 5886

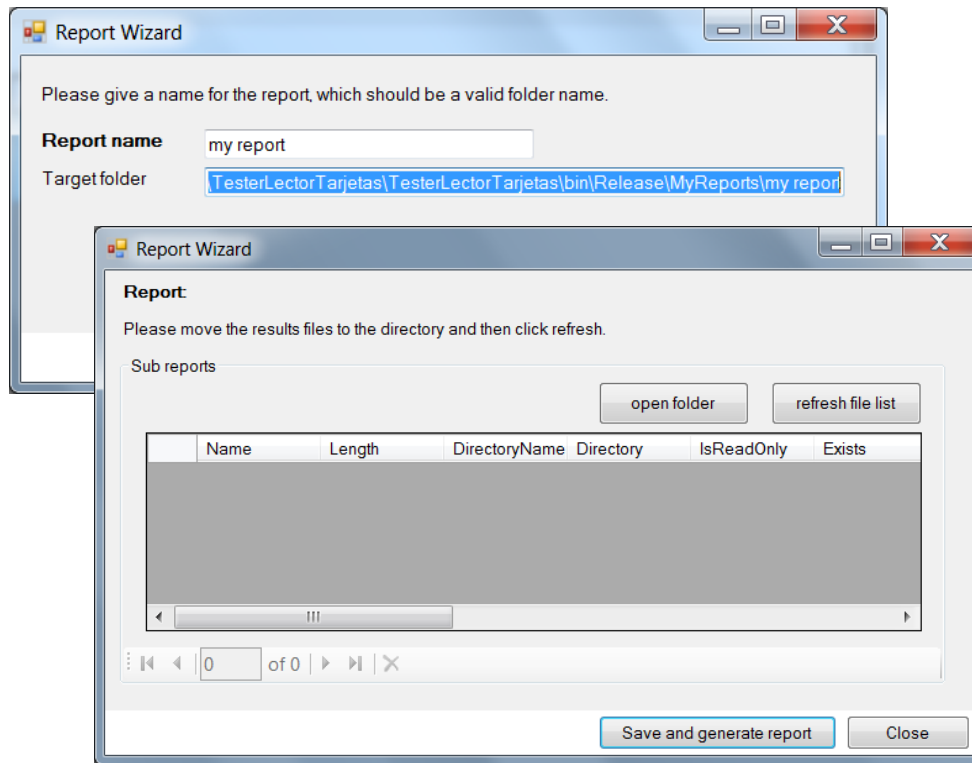
Add suite result **Remove** **File** **Select**

C:\Proyecto Activex Certify\Latest\MyReports\CustomerReportV1\3.r... **Browse**

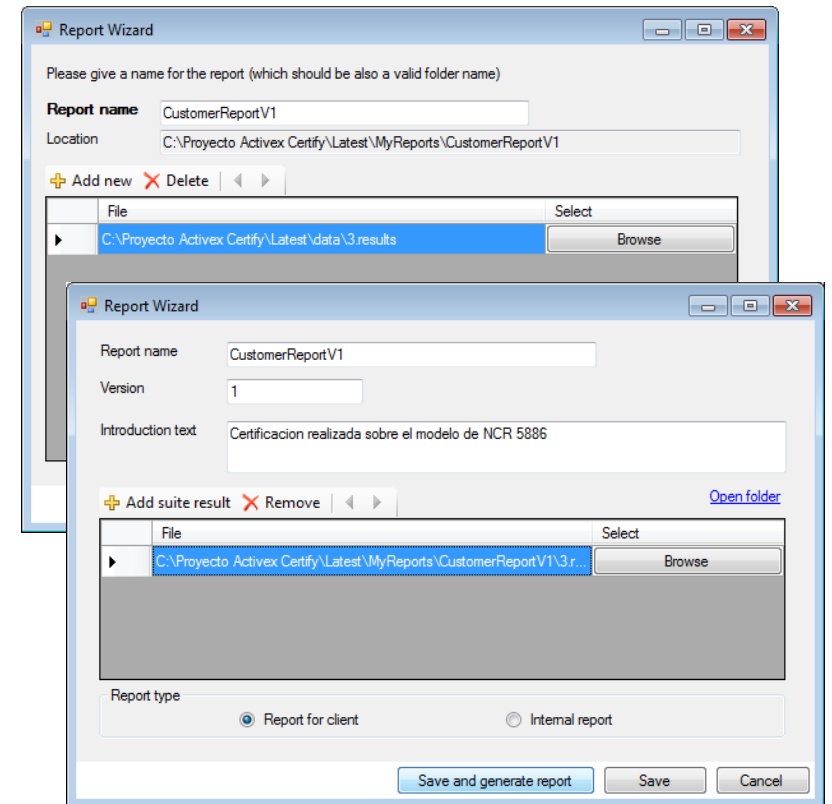
Report type ☒ Report for client ☐ Internal report

Save and generate report **Save** **Cancel**

Iteration 19. The wizard requires 2 steps

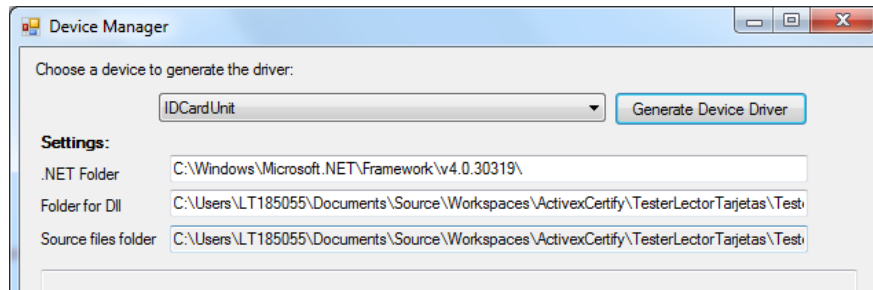


Iteration 17

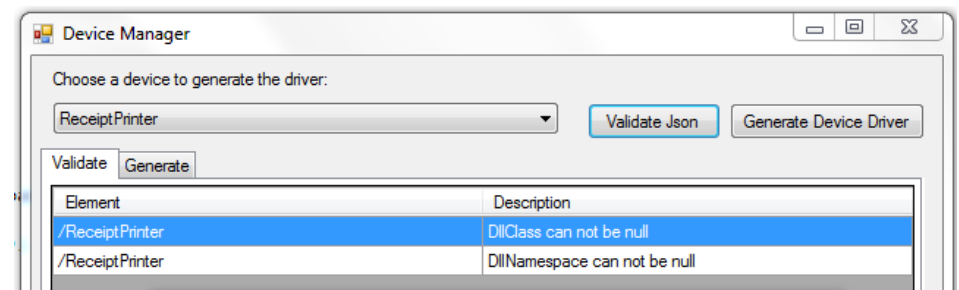


Iteration 19

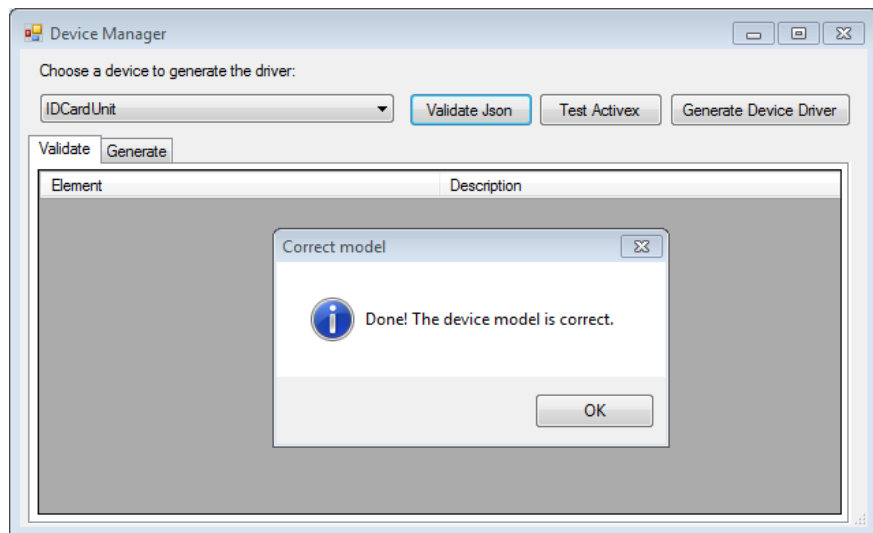
13.6 DEVICE MANAGER



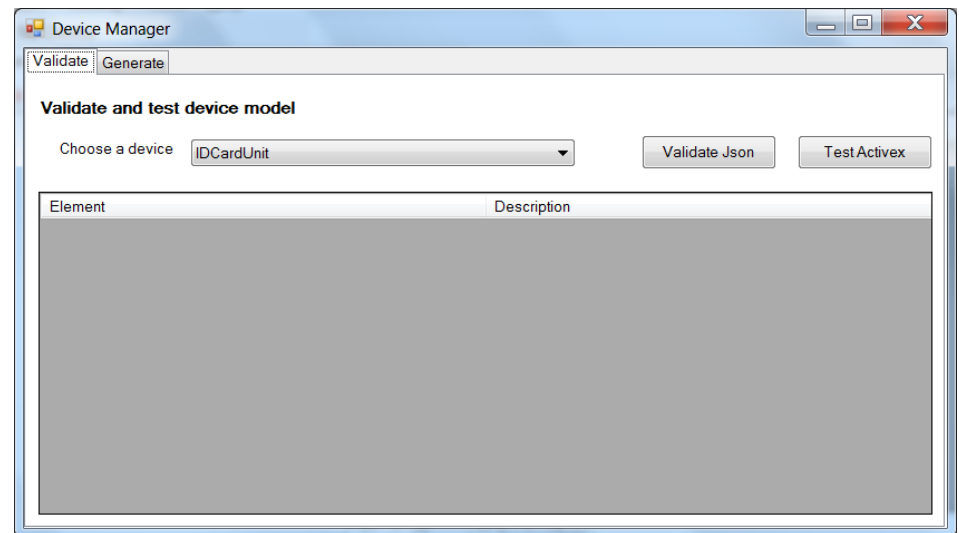
Iteration 7



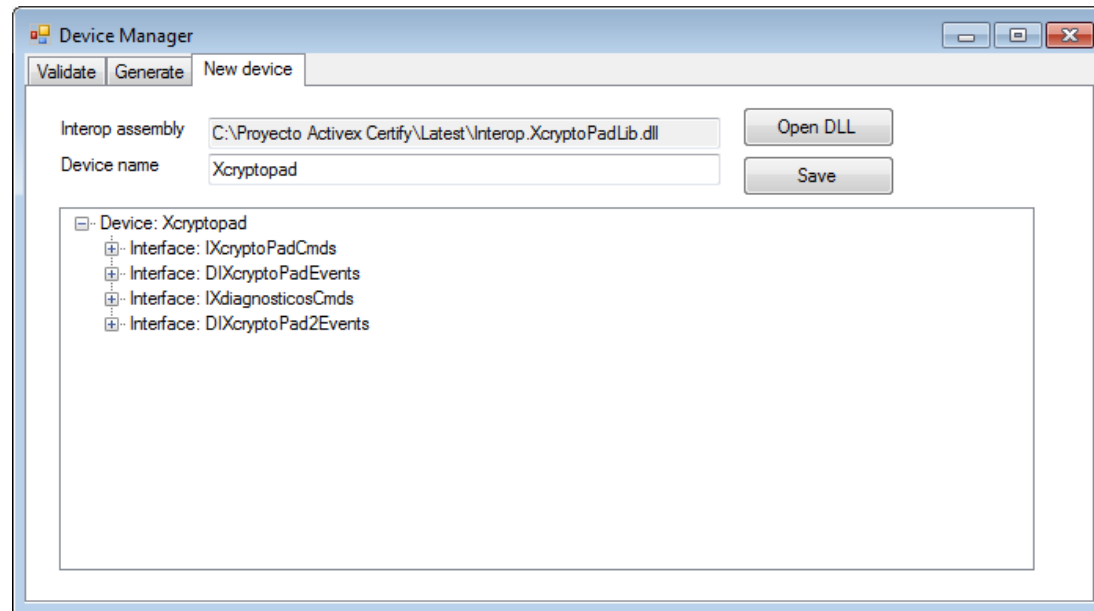
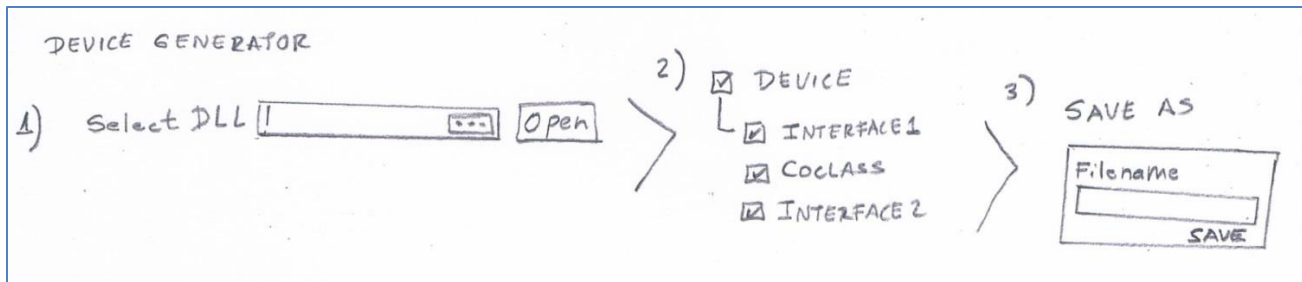
Iteration 8



Iteration 13

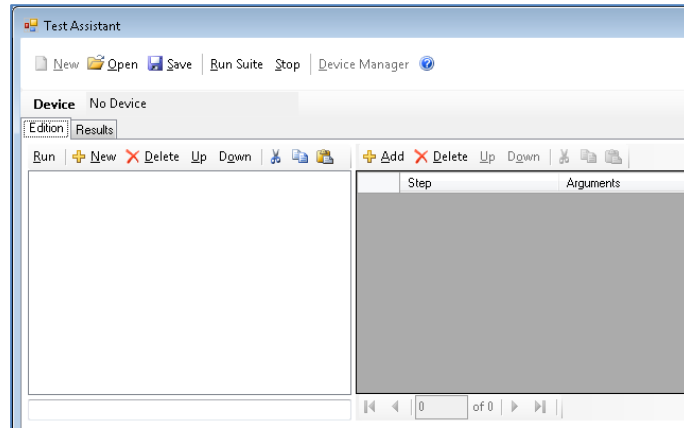


Iteration 15

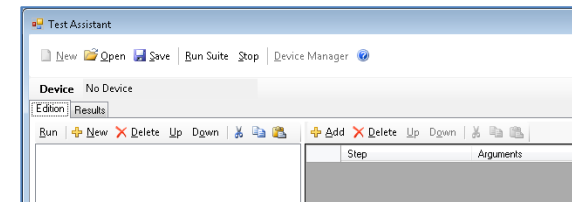
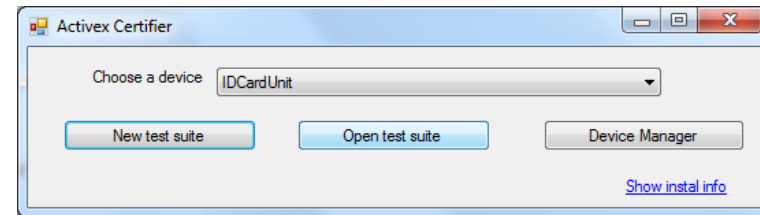


Iteration 20 – Generator of devices (paper prototype and implementation)

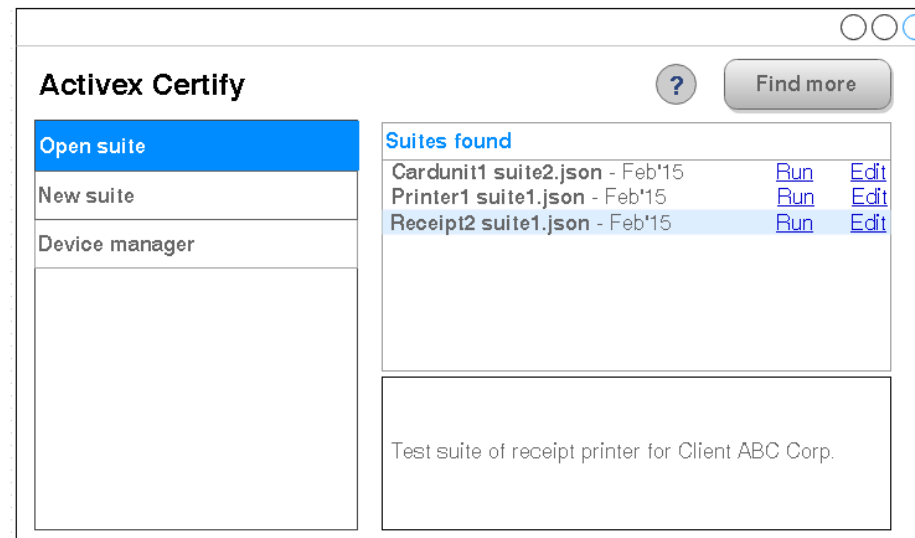
13.7 START SCREEN



Iteration 6



Iteration 7



Iteration 11. Wireframe proposal for starting screen. But it was not approved by users.

14 APPENDIX C. SUMMARY OF THE FIRST ETHNOGRAPHIC OBSERVATION

Date: 2014-12-04

Duration: 5 hours

User goal: Certification of the PIN Keypad

Participants: One specialist and one observer

Place: Customer facilities (basement)

The activities carried out were:

- 5) Observation (taking notes)
- 6) Summary of findings
- 7) Presentation of findings to and getting feedback from users
- 8) Review of requirements specification, process analysis and documentation according to the findings.

Below we provide the result of step 2 (in Spanish).

Entorno físico

Sótano amplio con 11 cajeros, mesas, y sillas disponibles. Sin WI-FI o puntos de red para visitas. Teléfono disponible. Toma corrientes limitadas.

El Cajero está preparado de la siguiente forma:

- Encendido y destapado en ubicación de fácil acceso.
- Puerto USB disponible.
- Pantalla no táctil y sin ratón. Pantalla sin película de privacidad.
- Teclado disponible pero sin lugar para apoyarlo. Se debe operar desde la parte superior del cajero.
- Razón de la homologación: cambio de sistema de operativo (Windows XP a Windows 7 pro)

Herramientas

- Portátil para editar el reporte de homologación (documento Word)
- Hoja de claves (llaves) de prueba
- Hoja con mapa de teclas del Pinpad
- Memoria USB con aplicaciones para las pruebas: WOSA tool, Container, Pinpad Test (tester Activex), etc. Los logs deben iniciar en blanco. Se conecta la memoria al cajero.
- El inventario del cajero está disponible en un fichero *inventario.ini*.
- Plantilla de homologación en blanco lista para edición en el portátil
- Móvil

Actividades de ejecución de tests

- Se lee la descripción de la prueba en el reporte (portátil) y luego se ejecuta usando Pinpad Test.
- Si hay un error: Ver detalles en el log; ejecutar otros comandos para tener más información (capabilities, status, etc.); ejecutar otras aplicaciones (WOSA tool, COM container, aplicaciones del cliente, etc.) para ver más detalles y corroborar si es un problema real, si es de ActiveX o si es de XFS, etc. También está la opción de preguntar al personal disponible y llamar para consultar con NCR. Es indispensable conocer el inventario del cajero y su historial.
- Para agilizar las pruebas y evitar repeticiones se agrupan pruebas en pequeños lotes que parten de estados iguales.

Conclusiones

Sobre el desarrollo de las tareas:

- La ejecución de comandos en el cajero debe poder hacerse sin usar ratón o pantalla táctil.
- Es muy frecuente la consulta de logs, la búsqueda de texto en el log, y el reabrir el fichero para ver los cambios.
- El uso de varias herramientas (además de Pinpad Test) es común: Wosa tool, COM Container, bloc de notas, y la búsqueda de ficheros en el disco duro y en la memoria. Sumado a la falta de ratón hace que se tengan que abrir las mismas ventanas y ficheros varias veces. Lo cual es muy ineficiente.
- Hay pruebas que parten con el mismo estado inicial: Y por tanto se pueden agrupar. Sin necesidad de repetir la misma prueba varias veces (sin afectar el resultado de la prueba).
 - Por tanto la ejecución de las pruebas se basa en el orden del reporte pero no sigue el mismo orden estricto.
- Los resultados específicos de los comandos se escriben de forma resumida en el reporte Word, los datos específicos se dejan para ser copiados luego de los logs. Pero como el orden de ejecución varia bastante, hace que luego la tarea de conciliar datos sea más compleja y requiera más tiempo.
- El uso de Word para orientar la ejecución de la prueba en un portátil aparte tiene muchas limitaciones:
 - Necesidad de buscar enchufes (lo cual es tiempo adicional)
 - Es difícil navegar el documento hacia atrás o adelante (para buscar comandos similares, verificar pasos anteriores, pasos faltantes, etc.).
 - Al estar particularmente lento implicó tiempo adicional para buscar la causa del problema, lo cual fue un distractor importante e implicó retrasos por varios reinicios del ordenador.

Sobre los requisitos de la aplicación de pruebas:

- Eventos:
 - La captura de eventos debe mostrar cada evento con su nombre y los valores de todos sus parámetros (pueden ser varios parámetros).
 - Los mensajes de llegada de eventos se deben encolar: Esto permite por

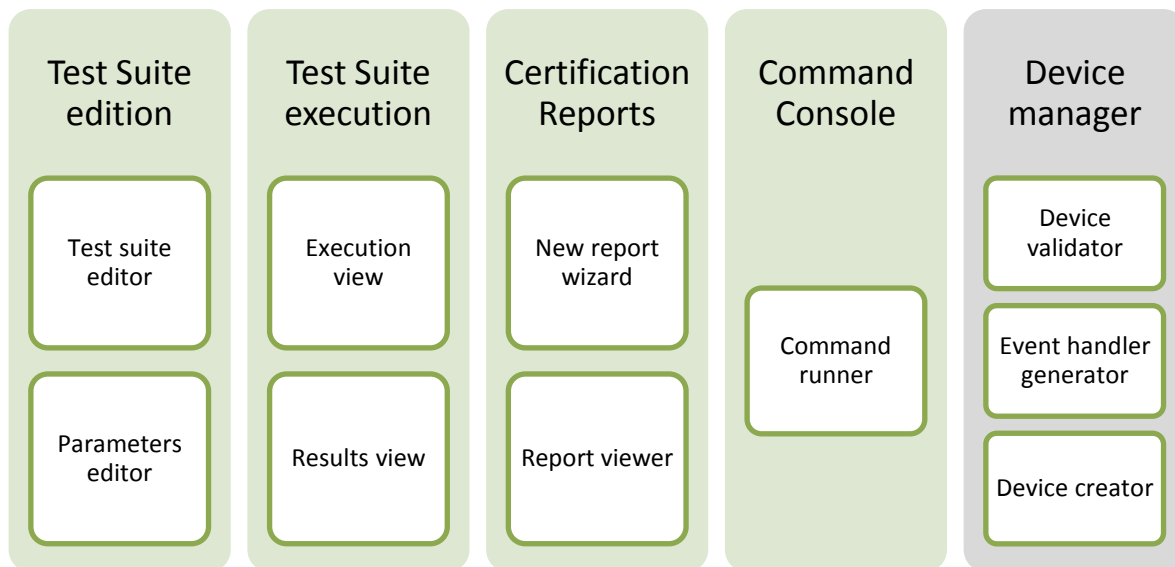
ejemplo que se puedan teclear 10 caracteres en el pinpad de forma continua, y al final confirmar los mensajes de llegada de los 10 eventos. Es decir, la aplicación no debe interrumpir la entrada de datos para pedir al usuario que confirme tecla por tecla.

- Formulario de entrada de parámetros de comando:
 - Valores por defecto (ej.: flags activados por defecto).
 - Soportar parámetros con valores tipo “flags”
 - Soportar parámetros con valores de selección múltiple
 - Botón para mostrar ultima ventana abierta (ultimo comando usado con últimos datos usados)
- Logs
 - Debe trazarse el máximo nivel de detalle posible con posibilidad de filtrar por nivel (para evitar cambiar la aplicación para ver más detalles de un dato, un tipo de error, etc.).
- Posibles herramientas adicionales:
 - Notas de datos usados frecuentemente: Clave maestra, clave de pin, dato a encriptar/desenciptar (usado en pruebas de todos los algoritmos soportados)
 - Imagen con mapa de teclas de pinpad de todas las marcas soportadas

15 APPENDIX D. SOFTWARE ARCHITECTURE

15.1 APPLICATION FEATURES

The application is composed by five features: Edition, Execution, Reports, Command Console, and Device Manager. These features are composed by the functional modules shown in the figure below:



The responsibilities of these modules are as follows:

- Edition of test suites: The test suite editor allows the modeling of test cases. While the parameters editor is a runtime generated editor for editing test steps on any selected device (ActiveX component).
- Execution of test suites: The execution view allows the execution of Test Suite models through a generic interpreter (Runner) of test step actions. The results view is the UI component for visualizing and editing test case results.
- Command console: The command runner is a separate module for running only commands on demand. This component is built on the interpreter of test steps and implements its own UI view.
- Certification reports: The Report Wizard allows the composition of several Test suite results models into a single report model. The report viewer is the HTML content generated by the wizard through code templates.

15.2 STAKEHOLDERS CONCERNS

The main architectural design decisions were done according to the following concerns:

- **Compatibility:** The application must be compatible with every operating system supported by the ActiveX middleware (AX). Actually this corresponds to WXP and W7 versions.
- **Maintainability:** The application must support always the latest AX version (that is, any future version). And this should not imply programming effort.
- **Extensibility:** If new devices (ATM peripherals) are supported by the AX, the application must allow the addition of the new devices. Again, this should not imply programming effort.

To find solutions, we followed the Model-driven development paradigm along with several Gang of Four design patterns.

15.3 MODEL-DRIVEN DEVELOPMENT (MDD)

The model-driven development (MDD) paradigm allows us to build an application around high-level models, which are technology independent. The main idea is to define the core data entities as models, so the application functionalities are defined in terms of model interpreters: to perform queries and manipulate models.

Because models are defined in business terms, not in technology-specific terms, users can create, modify, analyze and execute models without requiring programming effort.

Extensibility. ActiveX Certify has two core business entities: Test Suites and Devices. The catalog of devices is known at compilation time. But it needs to be extensible (after application deployment) to accept new devices in the future.

On the other hand, Test Suites are *documents* or *scripts* that are written by the user to test a specific Device (depending on the AX installed in the ATM). Thanks to MDD, we can create Test suites that references any Device model, despite it was created after compilation.

Multiplatform execution: Through a model interpreter, execution of test suites is done in high-level steps which are translated to concrete function calls at runtime through reflection. This requires dynamic library loading, which is straightforward in .NET, and avoids the use of code generation and compilation. This way, the application is multi-platform (or platform independent), where the platform is the specific version of AX.

The following picture shows the models of Test Suites and Devices along with different classes required for processing them.

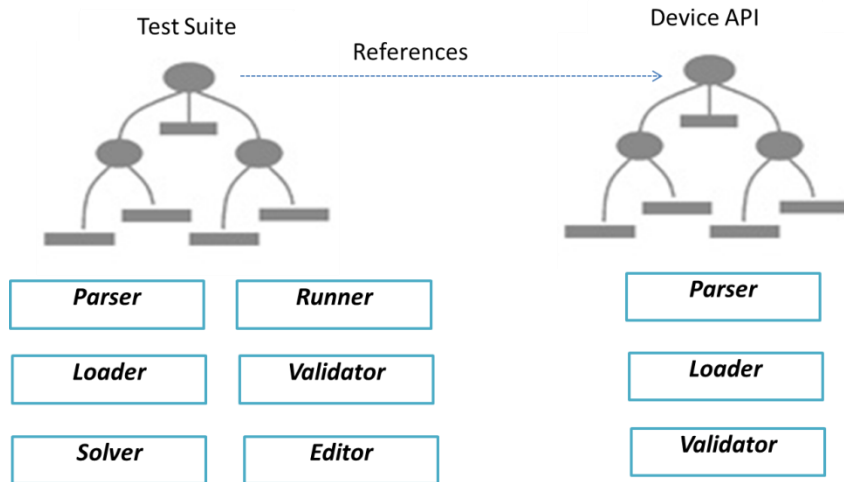


FIGURE 57. Test suite and device models (functionalities are implemented through model interpreters)

The classes required for model manipulation and interpretation are the following:

Parser and loader. Models need to be persisted and loaded. We selected the JSON format, because is human-friendly and it is fully supported in the development framework (.NET). Below we show the elements and the process around model loading and persistence.

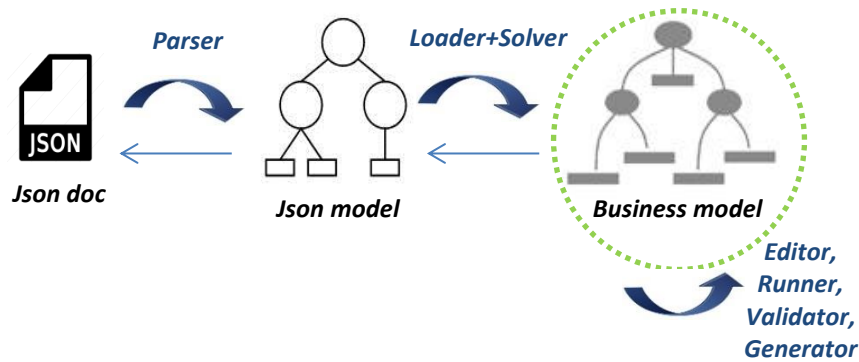


FIGURE 58. Parsing and loading of Test Suite models

Name Solver. Test suite and device models are fully decoupled so a test suite uses URIs to refer to device elements (as the syntax for full name scoping). These cross-references are solved at runtime by a Name Solver. An example of a Command Test Step is:

Command: /CardUnit/DeviceCapabilities

Where, the syntax is: *"/Device/Method"*.

Runner. This class is responsible for the multi-platform execution of test steps, as explained in the previous section.

Editor: This class allows the edition of Test Suites by manipulating the model.

Validator and Generator: Classes required for supporting the extensibility of new Device models. The Generator performs code generation internally to add support for event subscription to new devices.

15.4 DATA MODEL

The business entities that encompass the core application concepts which the business logic is built upon are: *Devices, Test Suites, Test Suite Result, and Test Suite Reports.*

- A **Test Suite** is a container of test cases modeled by the user. A Test Suite is associated to a financial device (peripheral).
- A **Test Case** is the execution unit of test suites, which is composed by actions (test steps).
- A **Device** is a model that describes the ActiveX API that controls a physical device. Test cases test these API functions.
- A **Test Suite Result** is a detailed execution trace of a Test Suite, containing the results of the test cases executed by the user.
- A **Test Suite Report** is a summary of a Test Suite Result, containing aggregated data of the result of a Test Suite.
- A **Report Group** corresponds to a Certification Report. It groups reports of several devices (different Test Suite Reports)

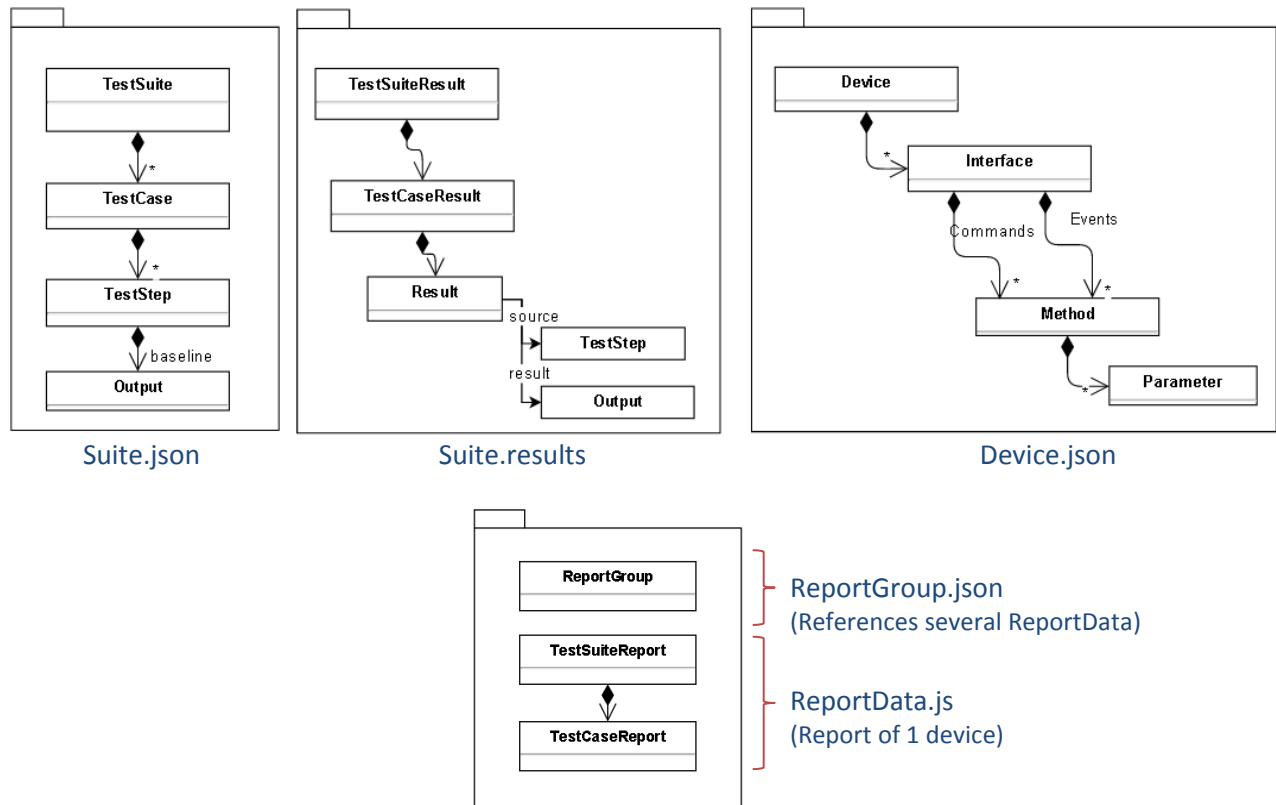


FIGURE 59. Class diagram with Test Suite, Device, TestSuiteResult and TestSuiteReport entities. Each entity is persisted in Json files (with different filename extensions).

In order to model a test case –where we like to test a device command- we need three things:

- The starting device state
- Execution of the device command under test
- Assertion of the command result (against the expected result)

For modelling the device state (A), we need to prepare the device through previous commands and possibly we require user intervention. ATM peripherals are input and output devices that need to interact with the user in an expected or unexpected way. For communicating instructions to the user we need test steps for displaying user messages and also test steps for accepting user input. For command execution (B), we just need to a test step that runs a command with the desired parameters. Finally, for allowing assertions (C), we defined that a command step has an expected value. So at runtime, we check the output (a string) against the expected result (another string).

Moreover, a device uses inter-application notifications through events. We need to trace what events were received chronologically. In test cases, we also need to check that the events received match with the events expected –according to the API specification.

In summary, we defined that *Test Steps* can be user inputs, user messages, commands execution, and events waiting. The class model is shown below:

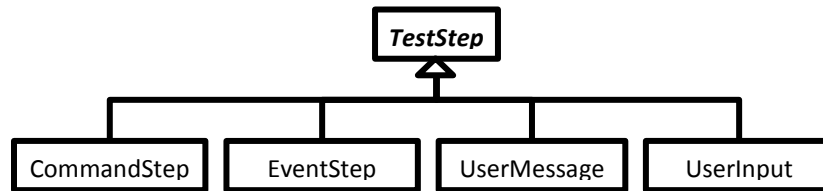


FIGURE 60. Test step class diagram

15.5 DETAILED SOFTWARE DESIGN

In the figure below we provide a class diagram with the main application classes and the design patterns applied.

First, to implement the GUI we used the Model-view-controller (MVC) pattern. The view is a class that inherits the Form class (which is the class for implementing windows in Windows Forms .NET). User interface actions are designed with the Command pattern.

The controller is a singleton class that receives UI requests and delegates the tasks to the Model classes, which implement the business logic.

Model classes are mainly the *TestSuiteEditor* and the *TestSuiteResultEditor*. The first implements the functionalities for the Test suite edition feature, while the second implements the functions of the Test suite execution feature.

Second, to manage properly the state of the application and allow or disallow user actions in the Models, the Controller and the View, we introduced a state machine. The state machine is composed by two state machines, the first for the editor and the second for the execution module.

The execution of test cases is performed through an iterator of test steps. So each step can be executed with a common Runner. The Runner is a *visitor* that is able to run any test step, independently of its type.

The event controller is used by the runner to execute Event test steps. The ActiveX Runner is used for running Command test steps. And the View is able to execute user test steps (messages and inputs).

The event controller implements the *observer* pattern to get notified whenever an AX event is produced. And the Default Assembly Runner is the concrete class that is able to dynamically load an assembly (dll) and run any function on it through reflection.

The Name Solver is a class for querying the device catalog (as explained before). So it is a singleton.

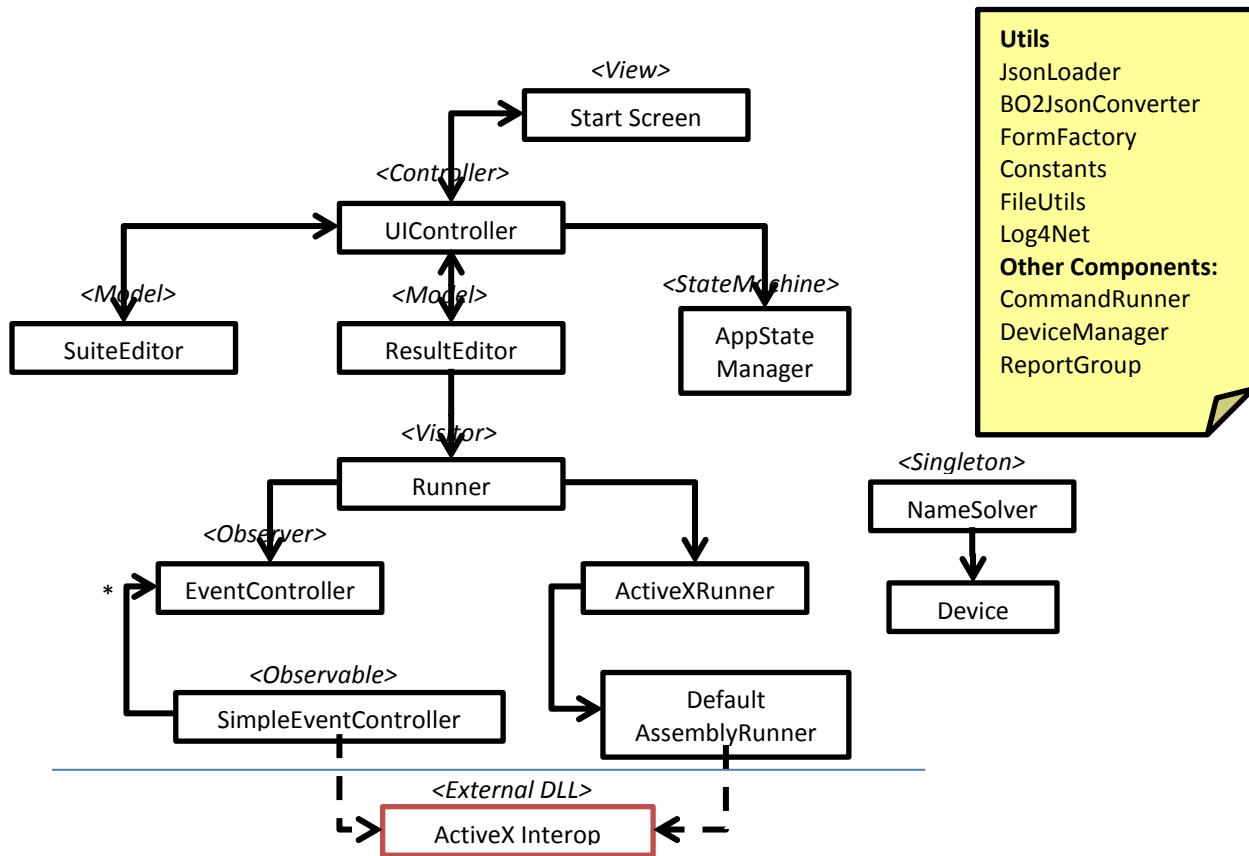


FIGURE 61. Main application classes and the design patterns applied

Finally, the application has other modules and classes such as the GUI Command Runner, the Device Manager and the Reports module. However, most of them are built on top of the mentioned core classes.

15.6 DEPLOYMENT VIEW

The software implementation is a standalone desktop GUI application for MS Windows. The application –once it is deployed- consists on several assemblies:

- ActiveX Certify.exe: The application entry point
- ActionParametersForms.dll: The assembly that dynamically generates input forms for AX commands.
- Devices.dll: The assembly whose code is automatically generated at maintenance time each time a new version of the AX platform is released or a new Device is supported.

- And the set of interop assemblies for invoking registered ActiveX components: These libraries are automatically generated for bridging .NET applications and ActiveX components.

All the deployment artifacts are installed in the ATM. The basic requirement for execution is the .NET Runtime 4.0 or higher, which works on Windows XP SP3 and later Windows versions. The figure below shows the deployment view.

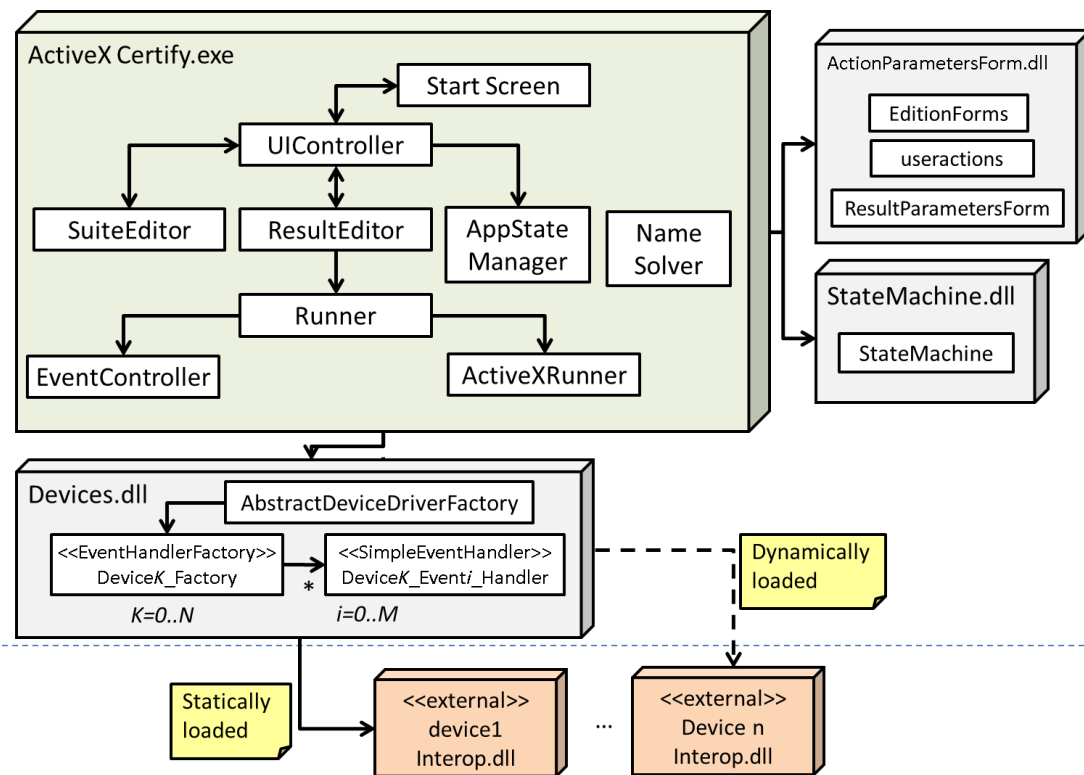


FIGURE 62. Deployment view (executables and assemblies)

16 APPENDIX E. SATISFACTION QUESTIONNAIRE

After each usability laboratory test we measured user satisfaction by using the UEQ questionnaire¹⁰ where we asked the user to evaluate the product through 26 questions. Each question presents two opposite qualities in a scale of 7 points. The results were collected and analyzes through the data analysis tool that comes along with it. This questionnaire measures the following six user satisfaction categories:

Attractiveness: <i>Overall impression of the product. Do users like or dislike is?</i>
Perspiciuity: <i>Is it easy to get familiar with the product?</i>
Efficiency: <i>Can users solve their tasks with the product without unnecessary effort?</i>
Dependability: <i>Does the user feel in control of the interaction?</i>
Stimulation: <i>Is it exciting and motivating to use the product?</i>
Novelty: <i>Is the product innovative and creative?</i>

Table 9. User satisfaction categories measured by the UEQ questionnaire.

The paper delivered to users is reproduced in the next page (in Spanish).

¹⁰ <http://www.ueq-online.org>

Por favor denos su opinión

Con el fin de evaluar el producto, por favor, rellene el siguiente cuestionario. Se compone de pares opuestos de las propiedades que pueden tener el producto. No hay "correcto" o "incorrecto" como respuesta. Su opinión es personal.

Ejemplo:

atractivo	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Feo
-----------	-----------------------	----------------------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----

Por favor, marque sólo un círculo por línea.

	1	2	3	4	5	6	7		
desagradable	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	agradable	1
no entendible	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	entendible	2
creativo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	sin imaginación	3
fácil de aprender	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	difícil de aprender	4
valioso	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	de poco valor	5
aburrido	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	emocionante	6
no interesante	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	interesante	7
impredecible	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	predecible	8
rápido	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	lento	9
original	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	convencional	10
obstrutivo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	impulsor de apoyo	11
bueno	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Malo	12
complicado	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Fácil	13
repele	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Atrae	14
convencional	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Novedoso	15
incómodo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	cómodo	16
seguro	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	inseguro	17
activante	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	adormecedor	18
cubre expectativas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	no cubre expectativas	19
ineficiente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	eficiente	20
claro	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	confuso	21
no pragmático	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	pragmático	22
ordenado	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	sobrecargado	23
atractivo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	feo	24
simpático	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	antipático	25
conservador	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	innovador	26