

Lappeenranta University of Technology
School of Industrial Engineering and Management
Department of Software Engineering and Information Management

Nikolaos Paraschou

**DESIGNING A USER INTERFACE FOR GAME DEVELOPERS TO
ENTER GAME SPECIFIC INFORMATION**

Supervisors: Adjunct Professor, D.Sc. (Tech.) Jouni Ikonen
Associate Professor, D.Sc. (Tech.) Kari Heikkinen

ABSTRACT

Lappeenranta University of Technology

School of Industrial Engineering and Management

Department of Software Engineering and Information Management

Nikolaos Paraschou

DESIGNING A USER INTERFACE FOR GAME DEVELOPERS TO ENTER
GAME SPECIFIC INFORMATION

Master's Thesis

2014

72 pages, 26 figures, 7 tables

Supervisors: Adjunct Professor, D.Sc. (Tech.) Jouni Ikonen

Associate Professor, D.Sc. (Tech.) Kari Heikkinen

Keywords: User Centered Design, Usability Testing, User Interface, Games

Designing user interfaces for novel software systems can be challenging since the usability preferences of the users are not well known. This thesis presents a usability study conducted for the development of a user interface for game developers to enter game specific information. By conducting usability testing, the usability preferences of game developers were explored and the design was shaped according to their needs. An assessment of the overall usability of the final design is provided together with the main findings that include the usability preferences and design recommendations. The results showed that the most valuable usability preferences are quickness, error tolerance and the ability to constantly inspect the entered information.

ACKNOWLEDGMENTS

I would like to acknowledge:

- My primary supervisor, Jouni Ikonen, for giving me the opportunity to work on this project and guiding me throughout the implementation journey.
- My secondary supervisor, Kari Heikkinen, for his advice and guidance throughout the implementation journey.
- The lead developer of the Game Cloud, Janne Parkkila, for his invaluable guidance, advice and assistance whenever those were needed, throughout the implementation journey.
- Timo Hynninen, one of the core developers of the Game Cloud, for his guidance, advice and assistance.
- My family, for providing the required spiritual “fuel” to keep the “engines” running.
- My friends, here at Lappeenranta and down to Greece, for the same reason.

Lappeenranta, 7 May 2014

Nikolaos Paraschou

TABLE OF CONTENTS

1 INTRODUCTION.....	7
1.1 Objectives and research questions.....	9
1.2 Research methodology.....	9
1.3 Structure.....	10
2 USABILITY IN A NUTSHELL.....	12
2.1 How the usability discipline emerged.....	12
2.2 Usability engineering.....	13
2.3 User centered design.....	13
2.4 Understanding usability.....	15
2.5 Usability testing.....	19
2.5.1 Discount usability engineering method.....	20
2.5.2 Other usability engineering methods.....	21
2.5.3 Formative and summative usability tests.....	22
3 CASE STUDY: GAME CLOUD WEB UI.....	24
3.1 The Game Cloud.....	24
3.2 The development and testing process.....	25
3.2.1 Iteration 1.....	28
3.2.2 Iteration 2.....	31
3.2.3 Iteration 3.....	33
3.2.4 Iteration 4.....	34
3.2.5 Iteration 5.....	35
3.3 Participant recruitment.....	41
3.3.1 User profiles.....	41
3.3.2 Total number of participants.....	43
3.3.3 Background of selected participants.....	43
3.3.4 Number of participants per test.....	46

3.4 How the tests were conducted.....	47
3.4.1 Basic training.....	47
3.4.2 The testing process.....	48
3.4.3 Moderator role.....	50
3.4.4 Observer role.....	51
3.4.5 Debriefing.....	51
3.4.6 Test environment.....	52
4 RESULTS AND DISCUSSION.....	54
4.1 Overall usability of the UI.....	54
4.2 Main findings.....	56
4.2.1 Usability preferences.....	57
4.2.2 Wizard versus form.....	59
4.2.3 Design recommendations.....	63
4.3 Error rate.....	64
5 CONCLUSION AND FUTURE WORK.....	66
6 REFERENCES.....	68

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface
HCI	Human Computer Interaction
HTTP	Hypertext Transfer Protocol
LCU	Least Competent User
NFL	New Functionality List
REST	Representational State Transfer
UCD	User Centered Design
UI	User Interface
UIL	Usability Issues List

1 INTRODUCTION

An important factor in the success of a software system is the design of its User Interface (UI) in a way that users will encounter its usage as a gratifying experience. Software developers have always been challenged by the fundamental question of what design decisions should be taken to produce a UI that is efficient, effective, easy to learn, error tolerant and satisfying. In other words, how to produce a usable UI.

The benefits of usability, as identified in the literature, justify the rationale behind investing in it. Highly usable systems can have substantial economic and social benefits for users and employers. Such systems result in increased productivity for users and operational efficiency for organizations. By being easier to understand and use, the training and support costs for the organizations are reduced. At the same time, the overall user experience is improved with less discomfort and stress for the users. [1]

A remedy that comes to alleviate designers when it is not clear how to incorporate usability into a product is the User Centered Design (UCD). UCD represents the processes, methods, techniques and procedures for developing usable products. It is an iterative design approach that places the user at the center of the development process and employs various techniques to evaluate and measure the usability of the product [2].

There have been multiple case studies in the scientific literature, e.g. [3][4], demonstrating the application of UCD to achieve usability in various types of software systems. These studies have shown that applying UCD is an established

way of working to ensure that the final product will be usable. Presently, UCD is internationally endorsed as a best practice in systems design and development [5].

The case studied in this thesis is related to the development of the web UI of the Game Cloud with usability as a driving factor. The Game Cloud is a software system that allows game developers to store game specific information in order to achieve links between games. It operates as a service and offers various programming interfaces to be used by games for the exchange of information. Before integrating the interfaces to the source code of their games, the game developers have to enter game specific information into the system (e.g., items, achievements, events). This is achieved through a web UI.

A significant design problem is derived from the novelty of the Game Cloud which deprives the developers of the system from knowing what design decisions would ensure the usability of the UI, in terms of efficiency and effectiveness. The use case of entering the items, achievements and events of games into a software system through the use of a UI has not been practiced in the past. Thus, there is not any prior source to look for design recommendations. A further obstacle that amplifies the difficulty of the design task is the fact that the usability preferences of the game developers are not well known.

The use case of entering game specific information to the Game Cloud is fundamental for the proper functioning of the system. It is a prerequisite that must be accomplished before the Game Cloud can offer its complete set of services. Thus, the overall acceptance of the system is tightly coupled to the usability of the front-end (i.e., the web UI). If the game developers experience an unusable UI during their first encounter with the Game Cloud, they are very likely to reject the system.

1.1 Objectives and research questions

The research question to be answered in this thesis is the following: “*What usability preferences do game developers have from the web UI of the Game Cloud?*” The research question is supported by the following sub-question: “*What design decisions can improve the usability of the UI in terms of efficiency and effectiveness?*”

The thesis presents a usability study conducted for the development of the web UI of the Game Cloud. The employed evaluation technique was usability testing, the mostly renowned UCD technique. The information collected by the usability tests was analyzed and translated into a collection of main findings that include the usability preferences game developers have from the UI as well as design recommendations. The findings of this study can assist UI designers who would have to design similar products in taking the right design directions.

1.2 Research methodology

The development of the UI followed an iterative approach. After implementing the first prototype, exploratory usability tests were conducted to receive qualitative feedback from the users, expressing their preferences and feelings on the design. That feedback was used to fix potential usability issues and extend the prototype’s functionality according to the users’ preferences. This led to a new version of the prototype to be tested to the next iteration.

This cycle (i.e., conduct usability test, fix usability issues, add new functionality, test again) was repeated until the UI reached its pre-release state. In the pre-release state a different type of test was conducted that aimed to assess how usable the UI was at that point. In addition to qualitative data, the final test collected quantitative

data. Thus, it was possible to measure the usability of the final product.

Furthermore, the final usability test compared two different design approaches for one of the most critical and frequently used functions of the application (i.e., entering game items). The results of this comparison meant to assist in the selection of the most suitable UI design for the final product. Additionally, the feedback of the comparison provided valuable design recommendations directly from the users on how to further improve the design.

1.3 Structure

The rest of the thesis is structured as shown in figure 1. Chapter 2 reviews the literature associated to software usability. It discusses the concepts of usability, usability engineering, user-centered design and usability testing. Chapter 3 presents the case of this study, the Game Cloud. It describes in detail the applied design and development process of the web UI of the Game Cloud and justifies the decisions taken concerning the number and type of usability tests as well as the selection of test participants. Furthermore, it provides procedural details on how the usability tests were conducted. Chapter 4 presents the results of the study. It begins by assessing the overall usability achieved by the applied process and continues by discussing the main findings of the study. Finally, chapter 5 concludes the thesis and provides conceptions for future research.

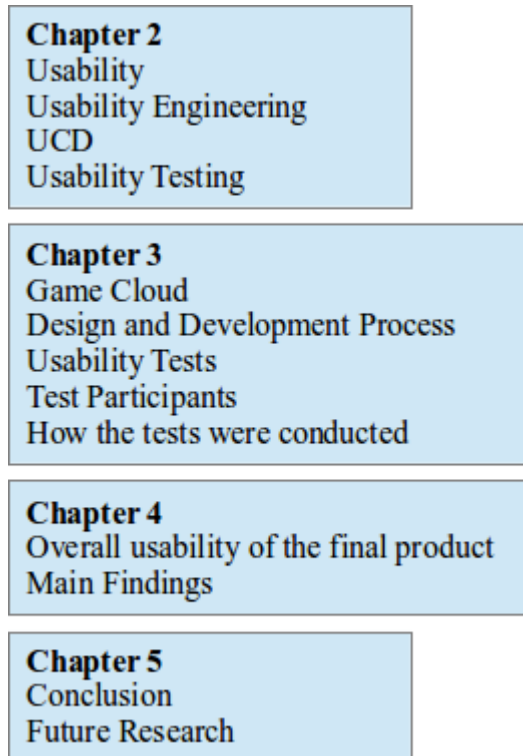


Figure 1: The structure of the thesis

2 USABILITY IN A NUTSHELL

This chapter introduces the reader to the concepts of usability, usability engineering, user-centered design and usability testing. A historical overview of the usability discipline is provided followed by definitions of the terms and techniques involved in the incorporation of usability to software projects.

2.1 How the usability discipline emerged

The scientific and industrial field of software usability is not a novel one. Instead, it has been in the focus of the research community for many decades. Turning back in time as early as 1959, one can find the concept of ergonomics for the computer being raised by Brian Shackel for the first time [6]. It is from these origins that usability started to slowly emerge [7].

According to Shackel, the first definition of usability was probably attempted by R B Miller in 1971 (cited by [8]) and it was based on “ease of use”. Following Miller’s paper, Shackel contributed a detailed formal definition in 1981 [9] which was modified by Bennett [10] to be incorporated later on by Shackel to his next formal definition [11][8]. Shackel’s latest definition was based on effectiveness, learnability, flexibility and attitude. Since then, multiple researchers and practitioners in the field have provided their own definitions for usability and discussed the subject extensively as shown in the official website of the User Experience Professionals Association [12].

Over the years, usability has earned its place among more traditional software quality attributes such as performance, reliability, and robustness. It is now considered a fundamental software quality. This progression was accompanied by

the introduction of a new field in the software development ecosystem to promote and ensure the incorporation of usability into software products. That field is known as Usability Engineering.

2.2 Usability engineering

Usability engineering was introduced to fill the gap between software engineers and human-computer interface designers [7]. The term was coined by usability professionals from Digital Equipment Corporation [13] who discussed concepts and techniques for planning, achieving, and verifying objectives for system usability [14]. Their formulation relied heavily on the works of Gilb [15][16], Shackel [9], Bennett [10], Carroll and Rosson [17], and Butler [18]. Further development to the subject was contributed by Whiteside and Holtzblatt [19].

The key concept behind usability engineering is the definition of measurable usability goals early in the development process and the repeated assessment of the defined goals during development to ensure that they are achieved [10][16]. Tyldesley [20] describes usability engineering as *“a process whereby the usability of a product is specified quantitatively, and in advance. Then as the product itself, or early ‘baselevels’ or prototypes of the product are built, it is demonstrated that they do indeed reach the planned levels of usability”*.

2.3 User centered design

In the broader world of Human Computer Interaction (HCI), usability engineering can be found under the name of User Centered Design (UCD) [2]. UCD is a design approach that represents the processes, methods, techniques, and procedures for developing usable products [2]. As Rubin and Chisnell point out, *“it (UCD) is the philosophy that places the user at the center of the process”* [2].

The initial launch of UCD was in 1986 under the name of User Centered System Design [7], by Norman and Draper [21]. Several definitions and understandings have been proposed over the years. According to Norman, UCD is “*a philosophy based on the needs and interests of the user, with an emphasis on making products usable and understandable*” [22]. ISO [1] defines UCD as an “*approach to systems design and development that aims to make interactive systems more usable by focusing on the use of the system and applying human factors/ergonomics and usability knowledge and techniques*”.

The design approach of UCD provides the necessary means for the development of products that meet the usability requirements of users. The success stories reported in [3] and [4] indicate its potential. A case that closely resembles the study of this thesis, is the development of a product called VirtualCenter 2.0 from VMWare [23]. VMWare introduced a new conceptual design for one of its virtualization systems. The concept of virtualization was so new that there was no precedent for how users would interact with such a system. In order to ensure that the users will be able to learn and use the product, the company applied UCD and managed to achieve the desired usability. Another successful case is the application of UCD principles to the development of IBM’s DB2 Universal ® Database [24].

Since the Game Cloud is considered to be a middleware system similar to the ones mentioned earlier (i.e., VirtualCenter and DB2), the application of UCD practices for the development of its web UI would provide the foundation for an easy-to-use, useful and engaging user experience. As Righi and Clow indicate, UCD can and should apply to the design of the middleware user experience [25].

Rubin and Chisnell emphasize three basic principles of UCD [2]. First, the design team should set an early focus on users and their tasks. The developers must be in direct contact with the users throughout the development process in order to collect information from and about users. The users' goals, tasks and needs should guide the development. Second, the usability of the product has to be evaluated and measured repeatedly throughout the development cycle. By doing so, valuable feedback will be returned to assist in driving and refining the design. Third, the process must be iterative. It should allow the shaping of the product through a repetitive cycle of design, test, redesign, and retest activities. The principles of UCD are further discussed by Gullisken et al [26] and ISO [1].

2.4 Understanding usability

Having introduced the most widely endorsed design approach that can be employed to achieve usability in software systems, that is UCD, it is now time to delve deeper into the notion of usability. The concept of usability is discussed in the literature with a number of attributes [2], quality components [27], or dimensions [28] that are used to define and measure it. Among others, some examples include efficiency, effectiveness, learnability, error tolerance, satisfaction and usefulness.

According to Rubin and Chisnell, in order for a UI to rightfully claim the title “usable”, it shall be describable by as many of the following attributes as possible: useful, efficient, effective, satisfying, learnable, and accessible [2]. These attributes are defined in [2] as follows:

- Usefulness assesses the user's desire to use the software at all. It refers to the extent to which the user is enabled by the software to achieve his or her

goals.

- Efficiency refers to the quickness with which the user can accurately and completely accomplish his or her goals by using the software. As such, efficiency is usually measured in time.
- Effectiveness concerns the degree to which the software behaves in the expected by the users ways and the ease with which users can use it to perform the tasks they intend.
- Satisfaction is an indicator of the user's feelings, opinions and perceptions of the software.
- Learnability refers to the user's ability to use the software to a certain level of competence after experiencing a predetermined amount and period of training. It may also refer to the ability of infrequent users to relearn the software after abstaining from its use for significant periods of time.
- Accessibility is a sibling of usability. It is about having access to the software which is required to accomplish a goal. Accessibility primarily concerns people who have disabilities. Nevertheless, making a UI usable for people with disabilities benefits people who do not have disabilities.

Rubin and Chisnell [2] rely on the following definition of usability: *“When a product or service is truly usable, the user can do what he or she wants to do the way he or she expects to be able to do it, without hindrance, hesitation, or questions.”* As the authors state to simplify the notion of usability in one sentence, *“in large part, what makes something usable is the absence of frustration in using it”*.

According to Barnum [29], one of the best-known definitions of usability is the one provided by ISO [1]: *“The extent to which a product can be used by specified*

users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.”

The definition of ISO [1] encompasses the three critical elements of specific users, specified goals, and specific context of use. What is meant by specific users is that usability is measured not with any users, but with the specific ones for whom the product is designed. Specified goals indicate that the users and the product share the same goals. In other words, the product represents the users' goals. Finally, the specific context of use signifies that the product is designed to be operated by users with certain characteristics, performing certain tasks, in a certain environment [1].

The same definition also focuses on the critical attributes of effectiveness, efficiency, and satisfaction which can be used to measure usability. Effectiveness measures how accurately and completely users can achieve their specified goals by using the product. Efficiency refers to the resources to be expended so that the user's goals can be achieved accurately and completely. Finally, satisfaction concerns the user's freedom of discomfort and positive attitudes while using the product. [1]

Quesenbery [28], a well-known usability consultant, defines software usability with five easy to remember dimensions which she calls the 5Es (Table 1):

Table 1: The five dimensions of software usability

Dimension	Description
Effective	Addresses whether the software allows the user to reach his or her goals completely and accurately.
Efficient	Concerns the speed with which the user's work can be done accurately.
Engaging	A software system is engaging when the user is interested in using it because it provides a pleasant and satisfying experience.
Error tolerant	Involves the software's ability to prevent errors and assist users in recovering from any errors that might occur.
Easy to learn	Refers to the extent to which the software can provide initial orientation to the novice user and guidance to deeper learning.

One of the leading specialists in the field, Jacob Nielsen, provides the following definition for usability [30]: *“It is important to realize that usability is not a single, one-dimensional property of a user interface. Usability has multiple components and is traditionally associated with these five usability attributes: learnability, efficiency, memorability, errors, satisfaction.”* This is how Nielsen describes the five quality components which are used in his definition [30][27]:

- Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?
- Efficiency: Once users have learned the design, how quickly can they perform tasks?
- Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?
- Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?

- Satisfaction: How pleasant is it to use the design?

To conclude the discussion concerning the meaning of software usability, a slightly less formal definition is presented: “*After all, usability really just means that making sure that something works well: that a person of average (or even below average) ability and experience can use the thing - whether it's a Web site, a fighter jet, or a revolving door - for its intended purpose without getting hopelessly frustrated.*” [31]

For the purpose of this study, the attributes of efficiency and effectiveness as defined by Rubin and Chisnell [2] will be used. To recapitulate their meaning, efficiency is the quickness with which the user can accurately and completely accomplish his or her goals by using the software and it is usually measured in time. Effectiveness is the degree to which the software behaves as expected by the users and the ease with which users can use it to perform the tasks they intend.

2.5 Usability testing

There is a wealth of techniques involved in implementing UCD. Each one has its own characteristics and is meant to be practiced at a different stage of the development process. Rubin and Chisnell [2] and Barnum [29] present in their books the major UCD techniques. According to the authors, usability testing is the mostly renowned one, a fact that is further supported by the Usability Professionals' Association 2009 Salary Survey [32].

Usability testing is a research tool [2] that involves users to evaluate a system in order to ensure that it meets usability criteria. It is defined by Dumas and Redish [33] as “*a systematic way of observing actual users trying out a product and*

collecting information about the specific ways in which the product is easy or difficult for them". According to Rubin and Chisnell [2], usability testing aims to inform the design, eliminate design problems and frustration and eventually improve profitability.

2.5.1 Discount usability engineering method

Since its beginnings until the 1990s, usability testing was formally conducted employing the methods of experimental design [29]. Consequently, it tended to be expensive, time consuming and rigorous. Later research set the foundation for more informal usability testing studies that can be highly effective. Nielsen determined that the highest cost-benefit value can be gained by testing no more than five users and by conducting as many small tests as possible [34]. Similar findings were published by Virzi [35][36] and Lewis [37]. They both found that small studies can uncover 80% of the usability issues from a test. According to Nielsen the number was 85%.

Presently, a widely advocated approach for practicing usability testing is through a series of quick tests with few participants, beginning early in the development process and following an iterative approach [2] (i.e., conduct test, list usability issues, apply fixes, re-test to verify the applied fixes and discover new issues). This approach to usability testing is known as discount usability engineering and it has been popularized by Jakob Nielsen [30][38][39]. A representative list of discount usability engineering practices includes scenarios, simplified thinking aloud, heuristic evaluation and card sorting.

Scenarios are simplified prototyping approaches that can be used to extract user feedback. As a usability testing approach, scenarios distill the system to the most

essential elements needed for valuable feedback. The system, although not fully functional or complete, can be used to elicit the users' opinions for some user-driven activity, i.e., a scenario. [39]

Simplified thinking aloud is an interview technique that is used to enhance the user feedback produced in usability tests. Users are prompted to think aloud while they are evaluating a prototype of the software, expressing what they are doing and what they expect from the system. [39]

Heuristic evaluation is an approach to improve usability while the design and development is still in progress and there are not any testable elements to be presented to the users [39]. With this approach the developers can apply to the design collections of usability principles that are known to have guaranteed usability success. Some examples of heuristics include [40]: a) maintain visibility of system status, b) enable users to rely on recognition instead of recall memory and c) help and documentation.

Card sorting is a technique that reveals the users' mental models of certain aspects of a software system. Each system feature or concept is placed on a card. Users are asked to group the cards in piles. Each group shall be labeled and it shall contain cards with features of similar characteristics. This technique is mostly useful when looking for ways to organize the system's functions into useful collections of menus. [41]

2.5.2 Other usability engineering methods

The scientific community and the usability practitioners are constantly trying to improve the existing usability engineering methodologies and form new ones. The

results of this endeavour can be reflected on publications concerning the matter, for example RITE [42]. RITE is a Rapid Iterative Testing and Evaluation method that aims to identify and fix as many usability issues as possible and to verify the effectiveness of the fixes applied to those issues in the shortest possible time [42]. The testing sessions in RITE are conducted with one participant only.

One of the primary concerns of the researchers while studying the usability engineering methodologies has been the merging of UCD with agile software engineering processes. Several studies have focused on the challenges involved in the incorporation of UCD practices into agile methods and have contributed various solutions. To meet the challenges of agile development, McGinn and Chang [43] propose the combination of RITE [42] with the approach to usability testing taken by Steve Krug [31][44]. Kane suggests that the combination of discount usability engineering methods with agile methods is feasible since they both share many similarities [45]. He considers the use of discount usability engineering with Scrum as a feasible strategy. Sy presented the adjustments her company had to do to the applied UCD methods in order to fit within an agile framework [46]. Constantine outlines a streamlined and simplified variant of the user-centered process that is readily integrated with agile methods [47].

2.5.3 Formative and summative usability tests

Rubin and Chisnell [2] and Barnum [29] describe two types of usability tests; formative and summative. Depending on the point of the development cycle at which a usability test is conducted, the objective and the methodology of the test can vary. Both of these types of usability tests were applied in this study and they are described in the following paragraphs.

Formative tests begin early in the development cycle when the product is still being defined and designed. The main objective of formative tests is to evaluate the usability of the design and provide feedback that will drive the designers in forming and refining the product. Typically, these tests require the test participant to think aloud while performing the tasks in order to capture his or her real sentiments. The data collected by formative tests are qualitative and express the users' preferences and feelings for the product.

Summative tests are targeted towards more complete versions of the design, typically midway into the product development cycle. The objective of summative tests is to examine and evaluate the usability of the product by collecting qualitative and quantitative data. The qualitative data provide similar feedback to that of the formative tests. The quantitative data act as performance indicators. With such measures, the designers can assess the usability of the product.

3 CASE STUDY: GAME CLOUD WEB UI

This chapter introduces the case studied in this thesis. It starts with a presentation of the Game Cloud and continues with the development and testing process. The iterations of the development cycle together with their implementation and testing activities are discussed with examples and illustrations. The discussion then evolves around the participants of the usability tests. The required user profile is presented and a number of issues related to the participants are discussed; for example, the total number of participants in the study, the background of the selected participants, and the number of participants per test. The chapter concludes by presenting procedural information related to the usability tests, such as the testing process, the roles of the moderator and the observers and the testing environment.

3.1 *The Game Cloud*

The Game Cloud is a cloud based platform that provides a set of services for game developers. From a technical perspective, it is a semantic, scalable, cloud data storage and analysis service. The services offered by the Game Cloud can be used to establish links between games for the exchange of game specific information. The purpose of doing so is to enable cross promotion between games as well as to provide valuable analytics to the game developers that would assist them in improving their games according to the players' needs.

Figure 2 depicts a high level architectural overview of the Game Cloud. Two different games, GameX and GameY, are connected to the Game Cloud through a REST API over HTTP. What the Game Cloud does is the establishment of the green link that connects the two games so that they can exchange game specific information. Before the Game Cloud can offer its full set of services, the game

developers have to model the information of their games into the system. For example, they have to enter the game's items, achievements, events and several other elements of the game. This is achieved through the web UI in the blue box. This web UI constitutes the main focus of this study.

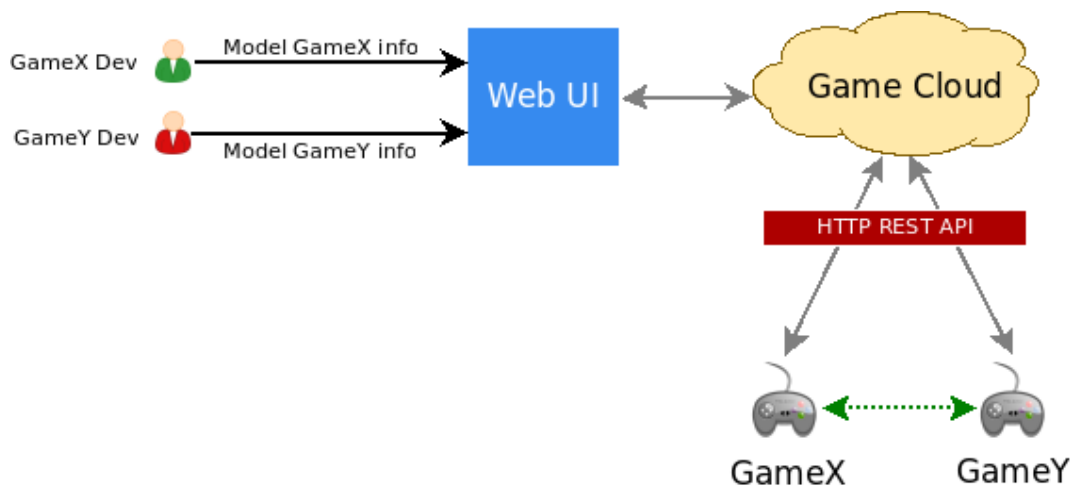


Figure 2: High level architectural overview of the Game Cloud

Given the fact that the Game Cloud is a novel product, the use case of entering game specific information had not been practiced in the past. As such, it was not clear what design decisions would ensure the usability of the web UI. Furthermore, the usability preferences of the game developers were not well known.

3.2 The development and testing process

Five usability tests were conducted to the web UI of the Game Cloud during the development cycle. The tests intended to uncover the usability preferences of the users and ensure that the final product would be shaped according to the users' needs. Usability testing started early in the development process and continued

until the first beta of the system was released. Figure 3 illustrates the development and testing process.

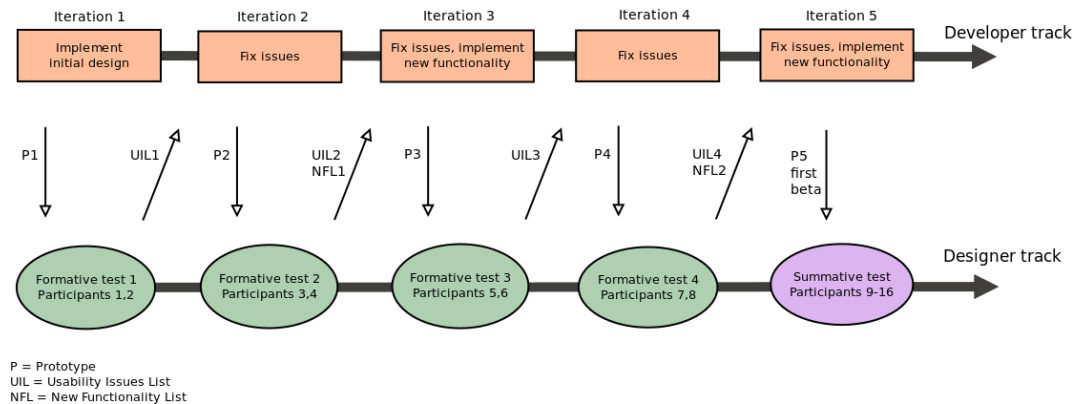


Figure 3: The development and testing process

As can be seen from Figure 3, the iterations consisted of two main parts: a) implementation activities on the developer track; b) testing activities on the designer track. In the beginning of each iteration, implementation tasks were taking place to develop and refine working prototypes of the UI (i.e., P1-P5). After the programming work was completed, the prototypes were undergoing usability testing to uncover potential usability issues. The debriefing session that followed after each test resulted in a Usability Issues List (UIL) that included the most critical usability issues discovered during the test. The UILs constituted the primary implementation work for the next iteration. In two occasions, at the end of iterations 2 and 4, an additional list was formed called New Functionality List (NFL). The NFL included implementation tasks that meant to increment the functionality and enhance the design of the UI during the next iteration.

Table 2 summarizes the implementation and testing activities that occurred during iterations 1 through 5 together with the outcome of the tests. The iterations and their activities are described more thoroughly in the following sections.

Table 2: Summary of implementation and testing activities per iteration (It.)

	Implementation	Testing	Testing Outcome
It.1	Design home page. Implement basic functions to enter games, items, achievements, events using wizards.	Formative test #1. Objective: Identify usability issues and collect users' opinions on the design.	UIL1 with 15 issues. Example: Absence of summary card in wizards. The users want to review a summary of the entered information before submitting.
It.2	Fix usability issues in UIL1 .	Formative test #2. Maintain the same testing tasks as in formative test #1. Objective: Verify the effectiveness of the applied fixes. Discover new usability issues.	UIL2 with 7 issues. Example: Users do not understand the "Description" step in the wizards. Inappropriate descriptions added. NFL1 with 6 tasks. Example: Create the view "My Games" so that the users can view the submitted information (i.e., games, items, etc)
It.3	Fix usability issues in UIL2 . Implement tasks in NFL1 .	Formative test #3. Modify testing tasks to include the new functionality. Objective: Verify the effectiveness of the applied fixes. Discover new usability issues.	UIL3 with 10 issues. Example: Users do not understand the meaning of the API calls returned by the Game Cloud after submitting an entry (i.e., item, event).
It.4	Fix usability issues in UIL3 .	Formative test #4. Maintain the same testing tasks as in formative test #3. Objective: Verify the effectiveness of the applied fixes. Discover new usability issues.	UIL4 with 5 issues. Example: The contents of the UI span great width. Need to move head left-right in wide screens. NFL2 with 15 tasks. Example #1: Use different coloring for the controls of different views (i.e., blue for games, red for items, etc). Example #2: Implement alternative design for the process of entering game items. Use a form.
It.5	Fix usability issues in UIL4 . Implement tasks in NFL2 .	Summative test. Modify testing tasks to include the new functionality. Objective: Verify the effectiveness of the applied fixes. Discover new usability issues. Measure overall usability of UI. Compare wizard and form in terms of efficiency and effectiveness.	Outcome reported in results and discussion section.

3.2.1 Iteration 1

The first prototype of the UI (P1) was created during iteration one and it contained the most basic functions to be offered by the system; the options to add games, game items, game achievements, and game events. Furthermore, a very draft design of the home page was provided. The home page contained essential navigation elements to allow access to the basic functions of the system.

The design approach that was followed in the first prototype was to provide UI elements capable of guiding the users step-by-step in the process of entering game specific information. The rationale behind this decision was based on the novelty of the use cases and the fact that they could become long and complicated to accomplish at later iterations. Novice users would find it hard to understand how to perform these tasks. To mitigate this problem, the use of wizards was employed. Figure 4 illustrates a screenshot of the wizard to enter new games in P1.

Figure 4: Wizard to enter a new game in PI

Wizards could effectively simplify the tasks by providing a pre-planned road map for the novel users to follow, thus sparing them the effort of figuring out the requirements of the tasks [48]. All users would have to do is follow the instructions of the wizards trusting that their goals will be achieved without problems. The wizards are there to provide guidance and support as well as protection from possible errors (e.g., invalid inputs). Even though the benefits of wizards to novice users were evident, it was still unknown how expert users would wish to accomplish the same tasks, if not with the oversimplified approach of the wizards. This was a question that could be answered through usability testing.

After the first prototype was completed, it was time to conduct the first formative test (lower left corner in figure 3) to explore the overall ease of use of the UI, identify potential usability issues and collect the users' opinions on the design.

This test, as well as every formative test that followed, was conducted with two participants and it was observed by the developers (i.e., of the Game Cloud). To collect the required qualitative data, the users were asked to perform a number of predefined tasks (e.g., enter a specific game, enter a specific game item, etc) and think aloud during the process. The tasks were designed in a way that would drive the users to use the wizards (e.g., enter a new game or a game item). Table 3 lists the questions the first formative test aimed to answer. These questions remained the same to all formative tests until the end of the study.

Table 3: Questions the usability tests aimed to answer

How do users feel about the overall look and feel of the UI? Is the UI clean? Which sections are not clean? Why?
How easily do users grasp the fundamental and distinguishing elements of the UI?
Which functions of the product are "walk up and use" and which will probably require either help or written documentation?
How easily can users add information about games?
How easily can users learn how to use the system by themselves? Is the provided help enough? Should the help be improved?

In the debriefing session that followed after the first formative test, the developers (i.e., of the Game Cloud) and the test moderator formed a Usability Issues List (UIL) with the most critical usability issues observed during that test (UIL1). They prioritized the issues by severity and agreed on their fixes. An example issue was the absence of a summary card in the wizards. A participant complained that it was not possible to review a summary of the entered information before submitting the wizard. To review and verify the information before submitting you had to go to the previous steps sequentially, a rather unpleasant and time consuming process. Figure 5 illustrates the problem.

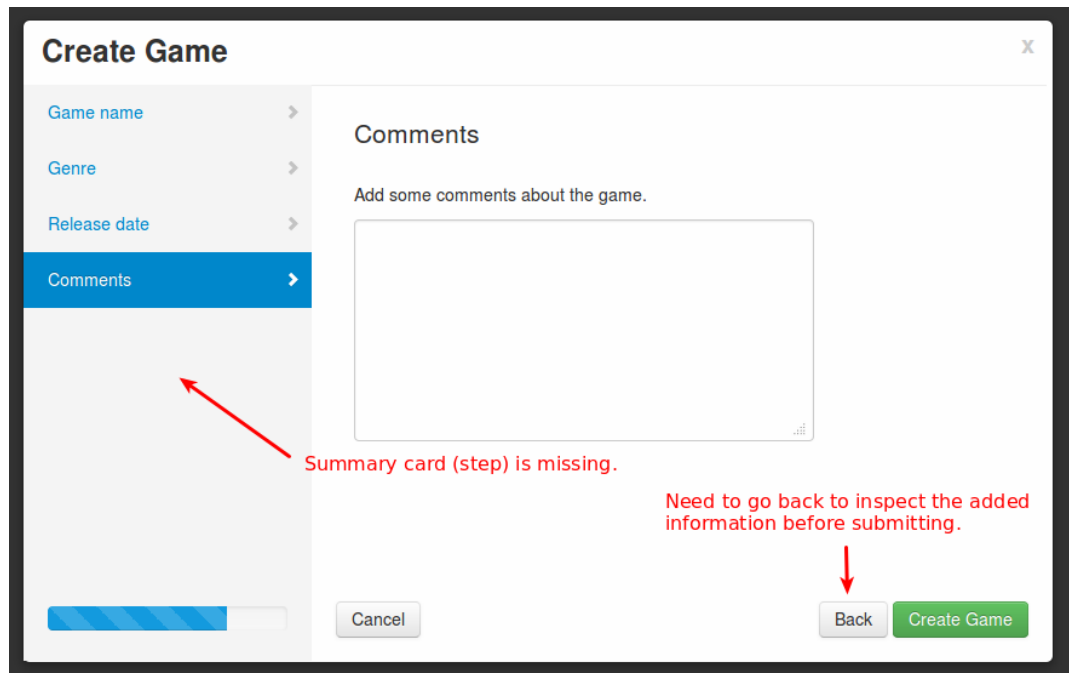


Figure 5: Summary card is missing from the wizard

3.2.2 Iteration 2

UIL1 constituted the main implementation artifact in iteration number two. The second iteration was entirely devoted to the implementation of fixes to the usability issues discovered in iteration one. The outcome of iteration two was an improved prototype (P2) that had to be retested by the second formative test.

The aim of the second formative test was twofold. First, it had to verify the effectiveness of the applied fixes and second, to discover new usability issues if any were introduced by the previous fixes. The tasks that the users had to perform in this test were exactly the same as the ones in the first test. Once again, by the end of the second formative test a list with prioritized usability issues and their fix

recommendations was formed (UIL2). An example issue was the fact that users did not understand the exact purpose of the “Description” step in the wizards and, as a result, they were always adding inappropriate descriptions. The objective of this step was to describe the entered element (i.e., game, item, achievement, event) in a way that would assist other game developers in understanding the element’s purpose. Figure 6 shows the description step in the wizard. As can be seen, there is not any help to guide the users in entering proper information in the description field.

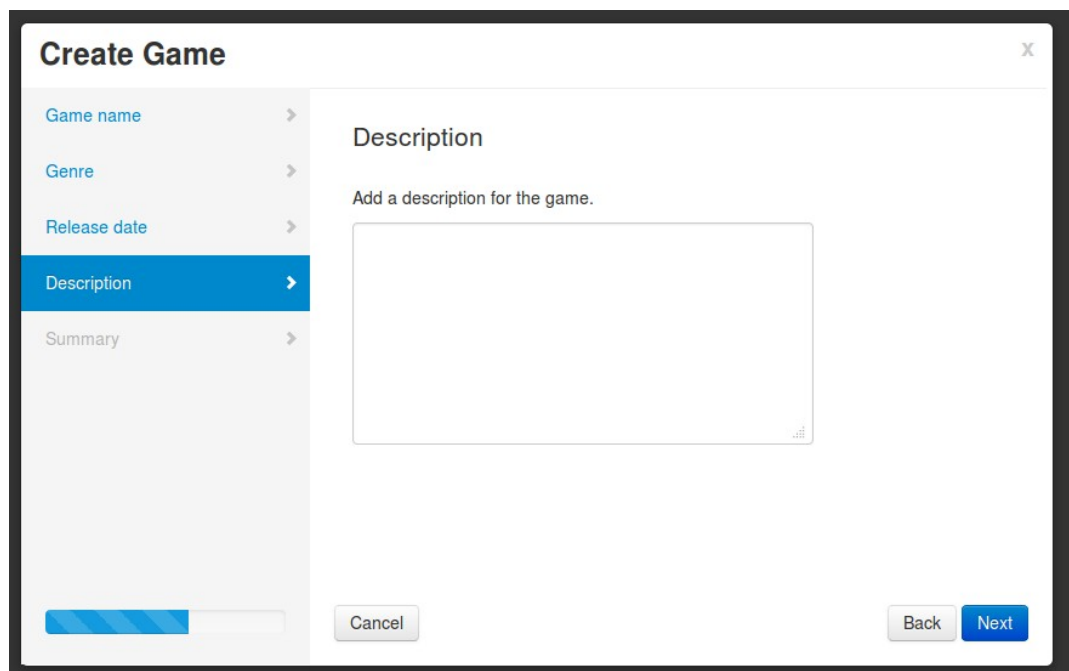
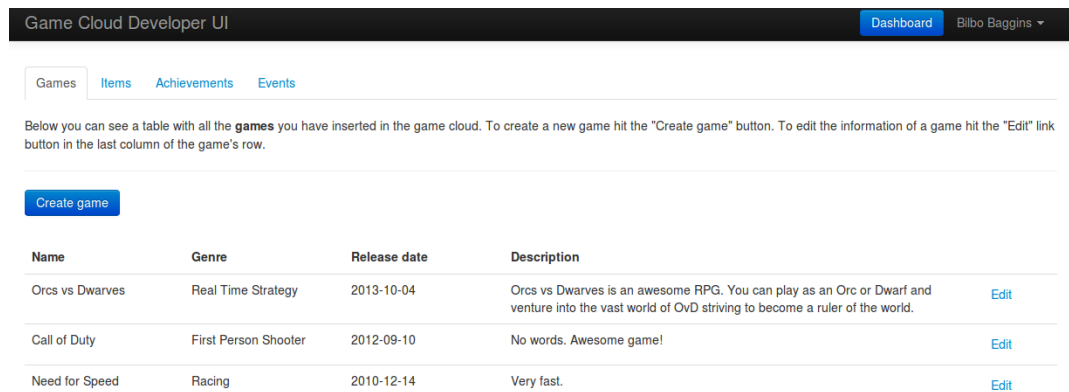


Figure 6: The description step in the wizard

At this point, after having conducted two formative tests (i.e., the first one to discover usability issues and the second one to verify the applied fixes), it was time to extend the functionality and enhance the design of the UI by implementing new features. Therefore, an additional list was formed called New Functionality

List (NFL) that included implementation tasks for that purpose (NFL1). An example task was the creation of a view called “My Games” (figure 7) in which the user would be able to view the submitted information (i.e., games, items, achievements, events). The two lists, UIL2 and NFL1, provided the implementation work for the third iteration.



Game Cloud Developer UI Dashboard Bilbo Baggins ▾

Games Items Achievements Events

Below you can see a table with all the **games** you have inserted in the game cloud. To create a new game hit the "Create game" button. To edit the information of a game hit the "Edit" link button in the last column of the game's row.

[Create game](#)

Name	Genre	Release date	Description	
Orcs vs Dwarves	Real Time Strategy	2013-10-04	Orcs vs Dwarves is an awesome RPG. You can play as an Orc or Dwarf and venture into the vast world of OvD striving to become a ruler of the world.	Edit
Call of Duty	First Person Shooter	2012-09-10	No words. Awesome game!	Edit
Need for Speed	Racing	2010-12-14	Very fast.	Edit

Figure 7: The view “My Games” in P3

3.2.3 Iteration 3

The development process progressed in a similar fashion. The outcome of the implementation work in iteration three was the third prototype (P3) which was tested in the third formative test by participants five and six. The tasks that the users had to perform in the third formative test were modified accordingly to allow the extraction of usability feedback related to the updated functionality and the updated design introduced in iteration three.

The debriefing session of the third formative test resulted in UIL3. One of the most critical issues in UIL3 was the fact that users did not understand the meaning

and purpose of the API calls returned by the Game Cloud after successfully submitting an entry (i.e., item, achievement, event). The API calls are meant to be used in the source code of the games to interact with the Game Cloud. The way in which they were presented to the users was not helping them understand their real purpose. Figure 8 illustrates the post-submission card in the wizard in which the API calls are presented to the user.

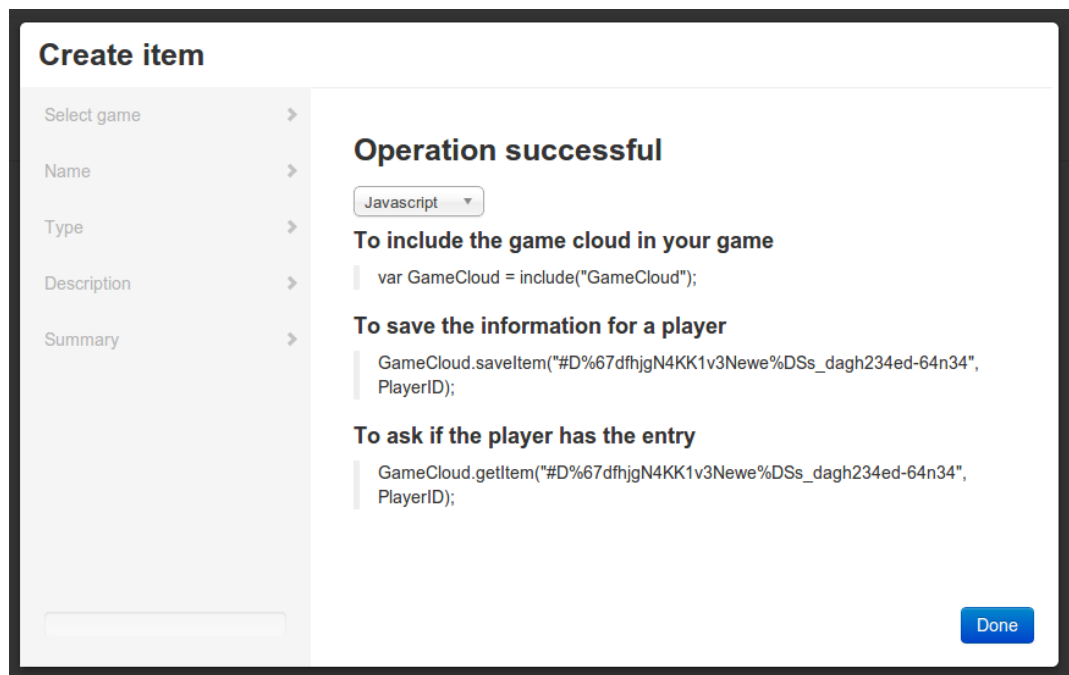


Figure 8: API calls card in P3 wizard

3.2.4 Iteration 4

Once again, the implementation work in iteration four was targeted in fixing the issues in UIL3. The result was prototype four (P4) which was tested in the fourth formative test. The tasks that the users had to perform in the fourth test were exactly the same as the ones in the third test. Similar to the second formative test,

the fourth formative test was primarily intended to verify the effectiveness of the fixes applied to the prototype in iteration four, in addition to discovering new issues.

3.2.5 Iteration 5

The implementation work of the fifth iteration was included in UIL4 and NFL2. UIL4 contained only minor issues. For example, a participant indicated that the contents of the UI span the full width of the screen. Using a wide screen, he had to move his head left and right to read the contents of the page. Figure 9 showcases the problem. The proposed fix was to confine the contents in a more suitable width.

Game Cloud Developer UI Dashboard Bilbo Baggins ▾

Games **Items** Achievements Events

Below you can see a table with all the **games** you have inserted in the game cloud. To create a new game hit the "Create game" button. To edit the information of a game hit the "Edit" link button in the last column of the game's row.

[Create game](#)

← The contents of the page span the whole width of the screen. →

Name	Genre	Release date	Description	
Orcs vs Dwarves	Real Time Strategy	2013-10-04	Orcs vs Dwarves is an awesome RPG. You can play as an Orc or Dwarf and venture into the vast world of OvD striving to become a ruler of the world.	Edit
Call of Duty	First Person Shooter	2012-09-10	No words. Awesome game!	Edit
Need for Speed	Racing	2010-12-14	Very fast.	Edit

Figure 9: Page contents span full screen width

A significant task in NFL2 was the redesign of the view “My Games”. The initial view included a tabbed pane with four tabs. The tabs were used to list the users’ games, items, achievements and events. To view the items of a specific game the user had to navigate to the items tab and select the game. The achievements and events tabs operated in a similar way. The new design proposed the creation of a

view that would list only the games as shown in figure 10. From there, a game could be selected redirecting the user to another view devoted entirely to the selected game's information. This second view would include a tabbed pane with three tabs for the selected game's items, achievements and events as shown in figure 11.

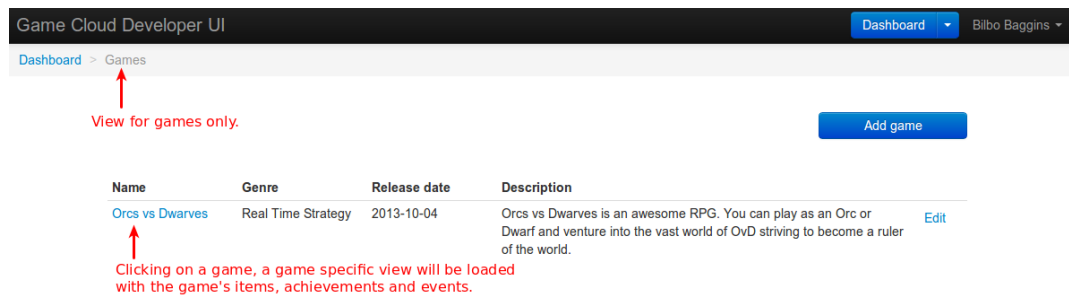


Figure 10: My Games view in P5 (early implementation phase in iteration 5)

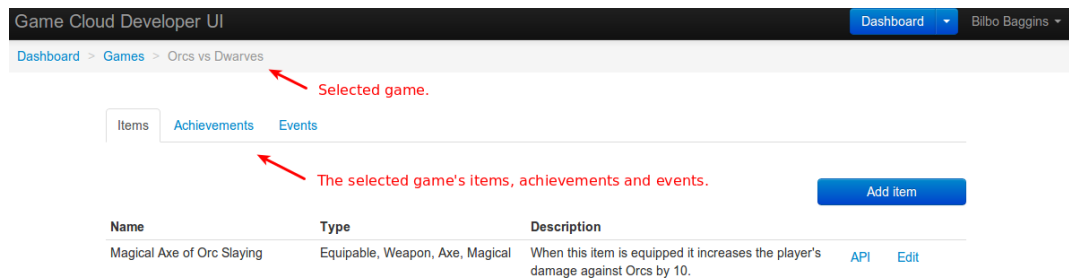


Figure 11: Selected game's view in P5 (early implementation phase in iteration 5)

Another worth mentioning enhancement included in NFL2 was the use of different coloring for the controls (i.e., buttons) of different views. Blue for games, red for items, green for achievements and orange for events. Figures 12, 13 and 14 illustrate this enhancement.

Game Cloud { game devs' console }

Dashboard > Games > Hammerfall506

Hammerfall506 RPG

Items | Achievements | Events

To add a new item click on the red **Add Item** button.
To get the **hash keys** of an item click on the item's **API** link button.

Search by Name or Type or Feature

Using red color for the buttons in the game items tab.

Name	Type	Feature	Description	API	Edit
ShieldOfTwinnedDespair75	Shield, Armor, Equipment	OneHanded, Thunderforged		API	Edit
AncientMoguTowerShield75	Shield, Armor, Equipment	OneHanded, Thunderforged		API	Edit
GreatshieldOfTheGloaming75	Shield, Armor, Equipment	OneHanded, Warforged		API	Edit

Figure 12: Red color for the buttons in the game items tab (later implementation phase in iteration 5)

Green for the achievements

+ Add Achievement

Description

API Edit

Figure 13: Green color for the achievements

Orange for the events

+ Add Event

Description

API Edit

Figure 14: Orange color for the events

In an attempt to discover solutions to improve the efficiency of the fifth prototype, a second version of the UI was created for one of the most critical and frequently used functions; entering game items. The new version was implemented as a simple form replacing the wizard. Figure 15 illustrates the form. The input elements were laid out in two rows and two columns so that they could be visible without significant scrolling (even in smaller screens). To decrease the time required to enter great numbers of game items, a second submit button was

introduced with the label “Submit and add another”. By clicking this button, the game item would be submitted and the form would be emptied waiting for the next item submission.

The screenshot shows a web form titled "Add Item" with a blue header bar containing a "Help" button. The form is organized into four quadrants:

- Name:** A text input field with the placeholder "Enter the item's name without spaces" and an example "e.g. WoodenSword".
- Description:** A large text area with the placeholder "Add a description for the item". Below it is a note: "This is an important field. Consult [help](#) for more information and examples."
- Type:** A section with the instruction "Select the item's type. An item can have one or many types." and an "Edit" button. It contains a list of checkboxes: "Completion", "Vehicle", and "Equipment" (which is expanded to show sub-options).
- Feature:** A section with the instruction "Select the item's feature. An item can have zero, one or many features." and an "Edit" button. It contains a list of checkboxes: "TwoHanded", "Thunderforged", "Warforged", "OneHanded", "Heroic", "Flexible", and "Magical".

Annotations in red text with arrows point to specific elements:

- "2 rows" points to the vertical arrangement of the Name and Description fields.
- "2 columns" points to the horizontal arrangement of the Type and Feature sections.
- "Decreases the time required to enter great numbers of game items." points to the "Submit and add another" button.

At the bottom, there are three buttons: "Cancel", "Submit and add another", and "Submit".

Figure 15: Form to enter game items in P5

Prototype five was considered mature enough to be released as the first beta. At this point, a summative test was conducted in order to measure the usability of the prototype with quantitative data and compare the two versions (i.e., wizard and form), in terms of efficiency and effectiveness. Qualitative data were collected as well in a similar fashion as in the previous formative tests.

The summative test aimed to answer the following questions:

- How efficiently and effectively can users enter game items in the system?
- How do users feel about how long it takes them to complete the process of entering games, game items, and game achievements in the system, both in the perceived amount of time and the number of steps required?
- What obstacles do users encounter on the way to entering games, game items, and game achievements?
- Do users consult the online help when they encounter obstacles (for example not being sure what information to enter and how to proceed)?
- How helpful are the help contents?
- How easily can users navigate between different sections of the UI (e.g. from the dashboard to a specific item of a specific game)?

The quantitative measures collected by the summative test include:

- Number and percentage of tasks completed correctly
- Number and percentage of tasks completed incorrectly
- Number and percentage of tasks that failed to complete
- Count of errors of omission
- Time to complete each task

The tasks that the users had to perform are listed in table 4:

Table 4: The tasks of the summative test

Task	Description	Successful completion criteria
1	Login (credentials given on a sticky note, unique for each participant).	Login successful, dashboard page shown
2	Add to the Game Cloud the game Hammerfall501.	Game added. User should be able to see the game in the games table.
3	Add to the Game Cloud the achievement 501QuestsCompleted of the game Hammerfall501.	Achievement added. User should be able to see the achievement in the game's achievements table.
4	Copy the gain achievement hash of the achievement 501QuestsCompleted of the game Hammerfall501. Paste the hash in notepad.	Gain achievement hash pasted in notepad.
5	Add to the Game Cloud 10 game items (weapons) of the game Hammerfall501.	All game items added. User should be able to see the items in the game's items table.
6	Add to the Game Cloud another 10 game items (armor) of the game Hammerfall501 using the alternative user interface.	All game items added. User should be able to see the items in the game's items table.

The summative test was conducted with eight participants. Since it aimed at measuring the usability of the UI with quantitative measures, the sample size should be big enough to ensure statistically valid results. Each one of the 8 participants tested both versions of the prototype (i.e., wizard and form), one after the other. To account for the potential bias caused by the fact that the participants may learn to perform the tasks while testing the first version, the order of presentation of the versions was counterbalanced. For eight participants, some participants tested version A first (i.e., wizard), and others tested version B first (i.e., form). To negate the potential biasing effects, each version was performed in the first position as many times as it was in the last position, as shown in table 5.

Table 5: Testing order of the prototype versions per participant

Participant	Version
Participant 1	A, B
Participant 2	B, A
Participant 3	A, B
Participant 4	B, A
Participant 5	A, B
Participant 6	B, A
Participant 7	A, B
Participant 8	B, A

3.3 Participant recruitment

Regarding the selection of test participants for the tests, the main target group included professional and hobbyist game developers since the product is intended for game developers. Given the difficulty to schedule professional game developers, the testing sessions were primarily conducted with developers who create games as a hobby. The source of participants was the software engineering laboratory in Lappeenranta University of Technology. In addition to computer science researchers and professors, computer science students participated as well.

3.3.1 User profiles

A written profile of the target users of the system assists the developers and designers throughout the development cycle. Being able to reference an accurate picture of the user while designing and developing the system, the development team can design proper usability tests as well as take beneficial decisions concerning the design of the product [2]. The user profile of the end users of the Game Cloud is described by the following two personas.

Professional game developer

- Bob is a game developer working in a game development studio. His main role is that of a software engineer and he is working on the development of the company's games. Since it is a startup studio with a small number of employees, Bob participates in all the phases of the development process, from analysis and design to testing. Thus, he specializes in requirements engineering, analysis and design, programming and testing.
- The studio decided to use the services offered by the game cloud with one of its games. Bob was asked to use the UI of the Game Cloud to submit the game's information.

Hobbyist game developer

- Rob has an academic background in computer science. He is passionate about games and he is very keen to learn how to develop games. He puts a lot of effort on practicing and improving his game development skills by developing small games. He is doing so by participating to code camps and game development courses at the university and also by working on his own personal projects at home. He has already managed to publish one of his games to online game distribution channels.
- Rob wishes to use the game cloud out of pure exploratory interest. He wants to expand his game development knowledge by unveiling unexplored game development territories. He also wishes to use the game cloud in an attempt to promote his most valuable game titles by incorporating into them top-notch technological advancements.

3.3.2 Total number of participants

In total, 15 unique testers participated in 16 testing sessions. This means that one of the testers participated in two testing sessions. The reason behind this decision was the difficulty in finding suitable candidate participants with the required background (i.e., having been involved in the development of games as hobbyist or professional game developers). Having the same tester participate in more than one testing session is not a recommended practice and should be avoided. The tester would be biased, resulting to inaccurate feedback.

Knowing this issue, it was decided to re-test in the last testing session (i.e., summative test) with one of the testers who participated in the first testing session (i.e., first formative test), minimizing the bias likelihood. Given the fact that the UI had undergone radical changes between the first and the last test, the bias would be negligible.

Furthermore, the tasks in the summative test, especially those in which quantitative measures had to be collected, were designed in a way that would minimize any potential bias. For example, there was a task in which the users had to enter a number of game items into the system to measure how long it takes to complete the process of entering game items (i.e., task 5 and task 6 in table 4). The number of items to be entered was ten. If the participant was biased, then the time required to enter the first items would be similar to the time required to enter the last items (i.e., there would be no learning curve). However, no such thing was observed.

3.3.3 Background of selected participants

The first thing the participants had to do in the beginning of every test was to fill

in a background questionnaire. The questionnaire aimed to provide historical information concerning the background of the participants and their relation to the game development discipline. It would reveal how experienced the participants were in the domain of the tested software. Having that knowledge before the test was conducted would help them understand the behaviour and performance of the participants during the test.

The vast majority of the selected participants had been involved in the development of games as hobbyists (figure 16), whereas 4 had been involved in professional game development projects as well (figure 17). All participants, except one, stated that they spend time on playing computer games weekly. The majority of them devote 1 to 10 hours, whereas one declared 11 to 14 hours and another one more than 20 hours (figure 18). Consequently, the participants were well familiar with the concepts of games and the notions of game items, achievements and events.

Do you develop games as a hobby?

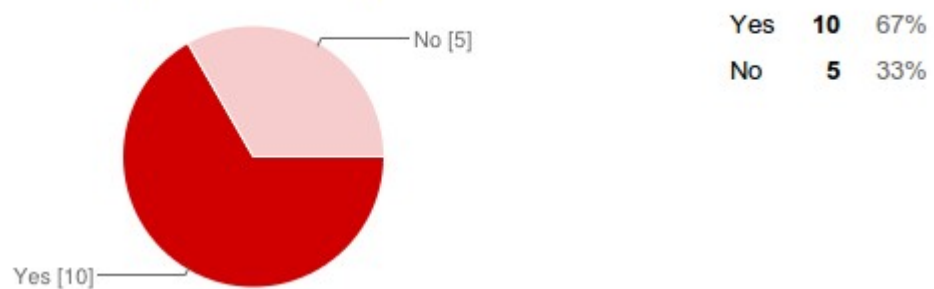


Figure 16: Hobbyist game developers

In how many professional game development projects have you been involved in?

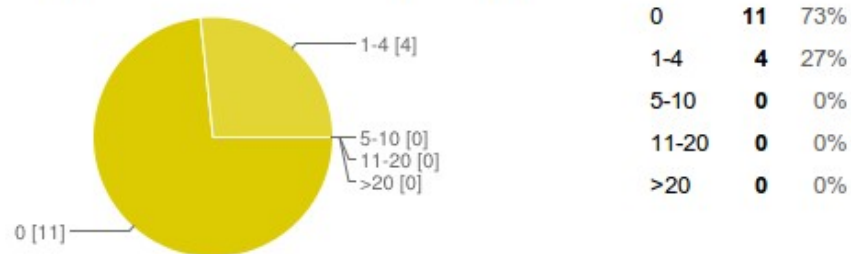


Figure 17: Professional game developers

How many hours per week do you spend on playing games?

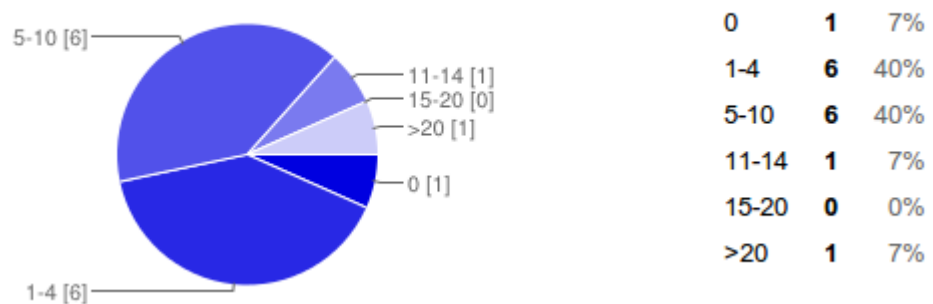


Figure 18: Number of hours per week the test participants spend on playing games

Three of the participants had never been involved in the development of games in any way and one of the three declared himself as a non-gamer. Those were the least competent users (LCUs) among all the participants of the tests. According to Rubin and Chisnell [2], an LCU is defined as “*an end user who represents the least skilled person who could potentially use your product*”. It is a good practice to include LCUs in usability tests since they are excellent indicators of the overall ease of learning of the product [2]. If the LCUs can successfully use the system, then it can be safely assumed that the target user groups are also able to perform similarly and even better [2].

3.3.4 Number of participants per test

Concerning the number of participants per formative test, various opinions have been expressed in the literature. Some studies propose three to five participants [34], while others only one [42]. The approach followed in this study lies in between the aforementioned suggestions.

The formative tests were conducted with two participants each. The reason for using two participants was two-fold. First, since the beginning of the project it was known that it would be difficult to find test participants with the required background (i.e., game developers). As a result, a decision was taken to distribute the limited resources in testers to more tests with fewer participants. Second, conducting the test with two participants instead of one would minimize the potential outlier effect. It was possible that a participant could provide false feedback. Especially in the case of this study where three of the participants did not much precisely the required user profile. A second testing session per test could act as a verifier to the findings produced by the first session, thus minimizing the outlier effect.

The summative test was conducted with eight participants. Due to the fact that the summative test aimed to collect quantitative measures, the sample size should be greater to ensure that the produced results will be statistically valid. Once more, the difficulty to schedule testers with the desired user profile prevented the team from testing with more than eight participants. However, number eight was a suitable option since it allowed the formation of two groups of four participants each, allowing the tests to be conducted in counterbalanced order to minimize the bias effect.

3.4 How the tests were conducted

Before conducting a usability test, a test plan has to be created. The test plan constitutes the foundation for the entire test [2]. It addresses every detail that can have an impact on the success of the test, e.g. the how, when, where, who, why, and what of the test. This section provides information related mostly to the how and where of the usability tests conducted in this study.

3.4.1 Basic training

Given the fact that the majority of the testers who participated in the tests were not familiar with the Game Cloud, some basic training had to be conducted before the tests started to ensure minimum expertise. Without this training the testers would feel confused since they would be interacting with a completely unknown system. Not being aware of the exact purpose of the system and the problems it aims to solve, a participant would most likely feel as the “wrong” person at the “wrong” place.

To mitigate this issue and ensure minimum expertise for the participants, a single page training script was created. The information included in the script was meant to introduce the Game Cloud to the testers and establish a scenario in which they would play an important role. For example, the scenario was placing the user in the development team of a game company as a software developer. It continued by assigning a task (originating from the company’s boss) to use the Game Cloud for one of the company’s games. It then progressed by introducing the purpose and the main functions of the system. Extra care had to be taken when preparing the training script to ensure that it would not reveal any information that could bias the users.

The training script was handed to the participants when they agreed to participate to the tests, one or two days before the testing sessions. The participants were instructed to read it and note any questions they might have to be discussed on the testing day.

3.4.2 The testing process

The testing process involved the following activities:

- Pre-test arrangements
- The tasks
- Post-test arrangements

After the basic training was completed, the pre-test arrangements phase was commencing. This phase included three actions; 1) the test moderator was reading the test script to the participant, 2) the participant was asked to review and sign a recording permission agreement, 3) the participant was asked to fill in a background questionnaire.

The test script is a communications tool meant to be read verbatim to the participant. The purpose of the test script is to describe what will happen during the test session and emphasize the fact that the system, not the participant, is being tested [2]. The reason for reading it verbatim is to ensure that the moderator will always read the same information to all the participants, avoiding the disclosure of potentially biasing information to different testers. The recording permission agreement meant to guarantee that the participant had no objection in being recorded in the context of the conducted study. The background questionnaire aimed to reveal historical information concerning the background of the participant in the domain of the tested software. That information would provide

better understanding of the behaviour and performance of the participant during the test.

Following the pre-test arrangements phase, the actual testing tasks were conducted. There is a difference in this phase between the formative and the summative tests. In the formative tests, the tasks were given in printed form to the participant by the test moderator who had an active role in the testing session. The moderator was reading aloud the task scenario to ensure that the participant had a clear grasp of it before commencing its execution. While performing the task, the participant was prompted to think aloud to reveal as much of his thoughts and feelings about the system as possible. The moderator could interact with the participant, seeking for clarifications or providing assistance where absolutely needed.

In the summative test, the process was completely automated. The test was conducted with specialized usability testing software that guided the participant automatically. It was this software that was providing the tasks to the participant and it was the participant's call to decide when a task starts and when it ends. The role of the moderator was restricted to observation without any interaction to ensure that the participant would not be slowed down in any way. For the same reason, the participant did not have to think aloud. One of the primary objectives of the summative test was the collection of quantitative measures. In order to ensure the quality of the data, the test participants had to perform the tasks uninterrupted. At certain times during the test, the participants were presented with surveys (i.e., post-task surveys) which they had to fill in. This was particularly the case after tasks 5 and 6 (see table 4) where the users had to evaluate the efficiency of the wizard and the form.

During the post-test arrangements phase the participants had to fill in a post-test questionnaire. The purpose of the post-test questionnaire was to collect qualitative data that would reveal the opinions and feelings of the users about the system's usability. The same questions were asked of each individual in all the tests. Following the submission of the post-test questionnaire, there was a final discussion between the participant, the moderator and the observers (i.e., if the observers were present). Any particular problems that came up for the participant during the test or any clarifications needed by the moderator and the observers were discussed and sorted out.

3.4.3 Moderator role

The test moderator was present in all the testing sessions, both formative and summative. He was responsible to introduce the session, observe the participant performing the tasks, take notes and record the participant's behaviour and comments. If required, he was assisting the tester in cases where the last one was not able to proceed. The test moderator had to abide by the golden rules of moderating as discussed by Dumas and Loring in [49].

In the formative tests, the moderator was responsible to hand the printed tasks to the participant. Before starting a task, the moderator was reading it through to ensure that the participant understands it completely. While the tasks were in progress, the moderator could interact with the participant to extract useful feedback. Since the formative tests were exploratory, the moderator sometimes asked unscripted follow-up questions to clarify the participant's behaviour and expectations. He was also probing the user to think aloud when that was not happening.

In the summative tests, the moderator's role was slightly restricted. Since the main

aim of the test was to gather quantitative measures, the moderator had to minimize the interaction with the participant as much as possible. That would ensure the quality of the quantitative measures since the participant would be able to accomplish the tasks without being interrupted.

3.4.4 Observer role

The formative tests were observed by the core developers of the system. The observation was local with the observers being present in the testing room. They were situated behind the participant in safe distance that allowed them to observe and listen to the participant without burdening him with extra stress. The observers were observing the participant's actions through their computers. The software running in the participant's computer was sharing the audio, video and screen of the participant's computer to the observers' computers. The same software allowed the observers to mark their observations easily. At the end of the testing session, the observers were participating in the debriefing session with the moderator. The observers were not present in the summative testing sessions.

3.4.5 Debriefing

At the end of every formative testing session, the moderator and the observers participated in a debriefing session. The debriefing session had a double purpose; a) to create a prioritized list with the most serious usability problems the participants encountered while using the prototype, b) to decide what fixes shall be applied to the usability issues. The outcome of the debriefing session was a UIL.

The debriefing was occurring immediately after the testing session while what had happened was still fresh in everyone's mind. After the debriefing, the moderator

was forming a final report, a small summary of the testing session. The final report included: a) what was tested, b) the list of tasks the participants did, c) the UIL.

The summative testing sessions were not followed by debriefing sessions. In the context of this study, the fifth iteration was the last one in the development cycle. As such, the primary objective of the final summative test was to measure the usability of the UI. The formation of a new UIL was not of primary importance.

3.4.6 Test environment

The testing sessions occurred in the usability testing laboratory of the IT department at Lappeenranta University of Technology. The participants were using a desktop PC with Windows. The prototypes were loaded in Google Chrome. No other distracting software was running on the computer. A specialized usability testing software was employed to monitor the testing process. The software was MORAE. In addition to MORAE, google forms were employed in the early formative tests to create questionnaires and collect qualitative data. MORAE was configured to:

- Record audio and video during the testing sessions and share the recordings to the observers' computers.
- Capture and share the participant's screen to the observers' computers.
- Allow the observers to mark their observations easily and in real time while the test was in progress.
- Allow the test moderator and the observers to playback and analyze the test and the observations in the debriefing session.
- Provide surveys at predefined times during the test to collect qualitative data (e.g., background, post-task and post-test questionnaires).

- Provide all quantitative measures (e.g., time to complete task, error rate, success rate, etc).
- Specifically in the summative testing sessions, MORAE was configured to operate in auto-pilot (i.e., unmoderated) mode. The software assumed the role of the moderator, automatically presenting the tasks and the surveys to the participant.
- Allow the usability researcher to analyze the quantitative measures and create reports with the main findings of the study.

4 RESULTS AND DISCUSSION

The aim of this chapter is to present the results produced by the usability tests. First, the overall usability of the UI is examined. Then, the main findings of the study are discussed, divided into three groups: 1) usability preferences; 2) specific findings related to the comparison of the wizard and the form; 3) design recommendations. Finally, the chapter discusses the most critical errors that happened during the summative test and attempts to investigate their causes.

4.1 Overall usability of the UI

Following the completion of each test, the users had to fill in a post-test questionnaire which aimed to capture their opinions concerning the overall usability of the UI. The majority of the questions were in the form of a scale, inquiring users to evaluate certain aspects of the UI. For example, a statement like *“the process of entering a game is easy”* had to be evaluated with a score between 1 (strong disagreement) and 4 (strong agreement). The analysis of the qualitative data indicates that the users evaluated the overall usability of the UI with better scores in subsequent tests. Figure 19 depicts the improvement graphically. The final score assigned to each test was calculated as the average of the scores of all the test questions. The score of each test question was the average of the scores assigned to that question by all the participants of the test.

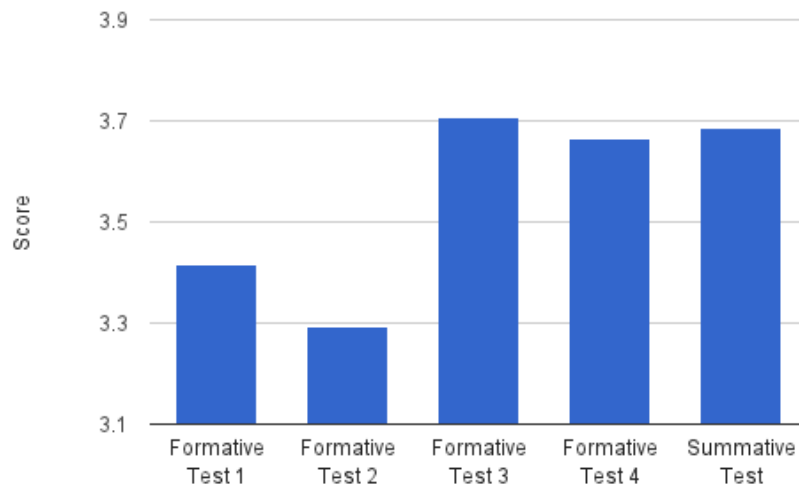


Figure 19: Average score of the UI per test

Table 6 lists some distinctive questions showcasing how the scores improved in subsequent tests. The top number in each cell is the average of the scores assigned to the question by the participants (i.e., numbers in parenthesis at the bottom of each cell). The overall picture drawn from these results indicates that the product has gradually improved throughout the development cycle in accordance to the preferences of the users. However, it can be observed that in some cases the scores were higher in earlier tests and then dropped notably before increasing again. An explanation to this development could be the fact that some of the participants did not match precisely with the required user profile. It might have been harder for them to understand various concepts of the system. As a consequence, they provided more strict evaluation.

Table 6: Questions showcasing the improvement of scores in subsequent tests

	Formative Test 1	Formative Test 2	Formative Test 3	Formative Test 4	Summative Test
The different functionalities offered by the UI are easily understood.	3 (4, 2)	2.5 (2, 3)	3 (3, 3)	3.5 (4, 3)	3.625
The process of entering a new entry (game, item, achievement) provides enough help and guidance along the way.	3.5 (4, 3)	2.5 (2, 3)	3.5 (4, 3)	3.5 (3, 4)	3.75
It is easy to learn how to use the system by myself.	3 (3, 3)	3.5 (4, 3)	3.5 (4, 3)	4 (4, 4)	3.875
I would use the system in the future.	2.5 (3, 2)	3 (4, 2)	3.5 (3, 4)	3.5 (3, 4)	3.625

4.2 Main findings

The information collected by the usability tests was analyzed and translated into a collection of main findings which are presented and discussed in this section. Table 7 summarizes the findings in three columns: a) usability preferences; b) wizard vs form; c) design recommendations. The usability preferences constitute basic preferences the game developers have from the UI in terms of usability. The findings in the second column are related to the comparison between the wizard and the form, the two design approaches that were the focus of the summative test. The design recommendations column includes suggestions on how to improve the UI according to the perspective of the users. The recommendations focus on the improvement of the process of entering game specific information, in terms of efficiency and effectiveness.

Table 7: The main findings

Usability preferences	Wizard vs form	Design recommendations
The users appreciate and demand quickness.	A form is faster than a wizard in the process of entering game items.	An automated approach to the process of entering game items would be the best solution (i.e., uploading a file that contains the items or entering the items directly through the source code of the games).
The users favor error tolerance.	A form is more error prone and “mentally challenging” than a wizard in the process of entering game items.	The efficiency of the form and the wizard could be improved by allowing duplication of similar items.
The users favor panoramic designs while entering game specific information that allow them to constantly inspect the entered information before interacting with the back-end (i.e., before submitting). They appreciate the feeling of inspection and control.	It is easy to learn how to use the system to enter game items using a wizard or a form.	The wizard could be improved by removing unnecessary steps and reducing the number of clicks required to enter a game item.
The users favor the adoption of different colors to distinguish collections of similar elements.		
The users seem to ignore help even when they need it.		

4.2.1 Usability preferences

The users appreciate and demand quickness. As experienced programmers, the users are accustomed to performing tasks rather quickly and they expect

technology to assist them in that respect. It was obvious that when asked to enter 10 game items (provided in a table in a simple web page), they employed all sorts of techniques to accomplish the task in the quickest way possible (e.g., using keyboard shortcuts, splitting the screen to have a concurrent view of the game items and the Game Cloud UI, copy-pasting). Nonetheless, they were not happy with the overall quickness. In many occasions, they were exhaling in frustration for not being able to accomplish the task even more quickly.

The users favor error tolerance. They demand and expect from the system to capture all errors they might cause while entering data, and in several occasions to fix the errors as well. They want the errors to be reported swiftly and in the places where they occur (i.e., to the UI elements that trigger the errors).

The users favor panoramic designs while entering game specific information that allow them to constantly inspect the entered information before interacting with the back-end (i.e., before submitting). They appreciate the feeling of inspection and control. Some observations indicated that one of the factors that made the users happier while using the form was the fact that it allowed them to view all the entered data at the same screen, without having to scroll or switch screens. This was not the case with the wizard where they had to go back and forth between steps to inspect the entered information. This finding is further supported by a request made by one of the participants to include a summary screen in the wizard right before submission. The summary screen should present the entered data and explain what will be sent to the back-end precisely.

The users favor the adoption of different colors to distinguish collections of

similar elements. Employing a unique color for the main buttons in the views of games (blue), items (red), achievements (green) and events (orange) makes it considerably easier for users to mentally classify the basic notions involved in the system (i.e., games, items, achievements, events) and be able to remember and distinguish them. It also provides a visually appealing experience.

The users seem to ignore help even when they need it. Although the UI included a help system with highlighted and clearly visible access points, the users seemed to ignore it even when they did not know how to proceed or what sort of information to enter. Moreover, in cases where the help was provided as instructions or guidelines and it was visible by default near the UI elements the users were interacting with, even then the users were not consulting it. What they tended to do was looking for a solution to the problem by trial and error. After failing a number of times, they slowed down and started inspecting the page more carefully. Only then they went through the instructions. In some cases they never consulted help, resulting in wrong information input. Perhaps this behaviour was due to the fact that game developers are experienced programmers, accustomed to resolving hindrances by themselves.

4.2.2 Wizard versus form

A form is faster than a wizard in the process of entering game items. This observation is supported by qualitative and quantitative data. In multiple occasions, the users expressed their preference to the form over the wizard due to its quickness. Furthermore, the final summative test in which the wizard and the form were compared in terms of efficiency, clearly indicates the prevalence of the second over the first. Figure 20 supports this claim. The users had to spend almost one minute more to complete the task with the wizard than with the form.

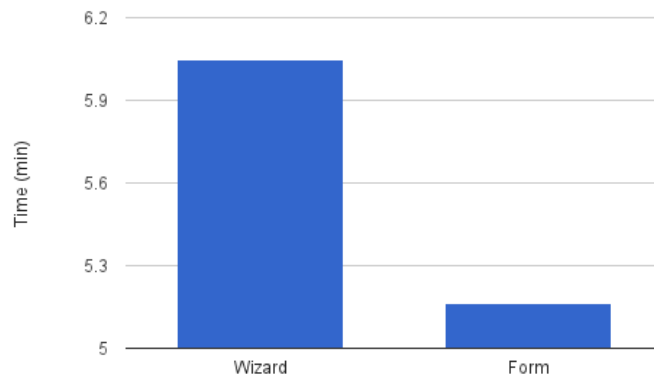


Figure 20: Average time spent to enter 10 game items with the wizard and the form.

A form is more error prone and “mentally challenging” than a wizard in the process of entering game items. Even though the form was reported to be considerably faster than the wizard, it was also more “mentally challenging” as one of the participants indicated. The users had to pay more attention to avoid error situations using the form. A possible explanation to this issue could be the way in which the input elements were laid out in the form. Since all the input elements were constantly at the user’s view, there was the risk that during the copy-pasting process - which was rapidly recurrent and dull - some data could be pasted to the wrong input field. The user had to look carefully in the form to locate the proper input field. This was not the case with the wizard since it was guiding the users as to what information should be added next.

It is easy to learn how to use the system to enter game items using a wizard or a form. It did not take the entry of more than two items before the users learned the process. Figure 21 depicts the average time spent per item when the users had

to enter 10 game items using the wizard and the form. It can be seen that after entering items A and B, the time required to enter more items does not vary significantly. Figure 21 also supports the statement that the form allows the users to enter the items faster than the wizard.

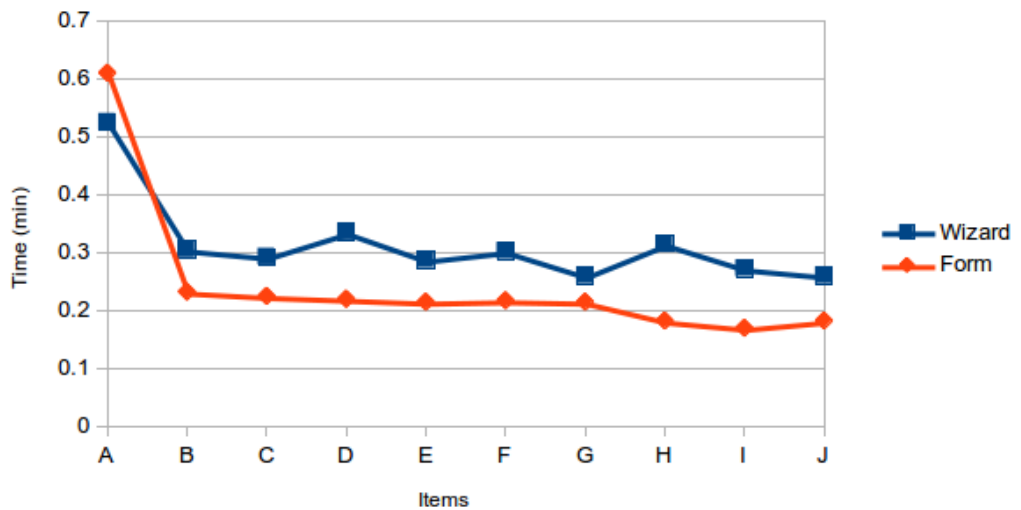


Figure 21: Average time spent per item; entering 10 game items with the wizard and the form.

The average values for the wizard in figure 21 were calculated only by the four participants who tested first with the wizard. The time spent by those participants to enter the game items is presented in figure 22. Similarly, the average values for the form were calculated only by the four participants who tested first with the form (figure 23).

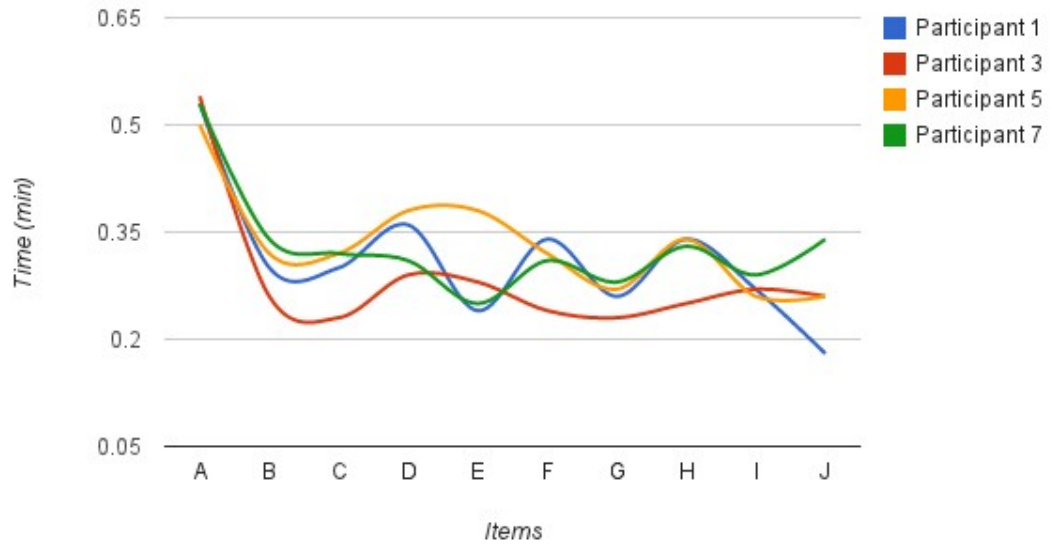


Figure 22: Time spent per item to enter 10 game items using the wizard; by the four participants who tested first with the wizard

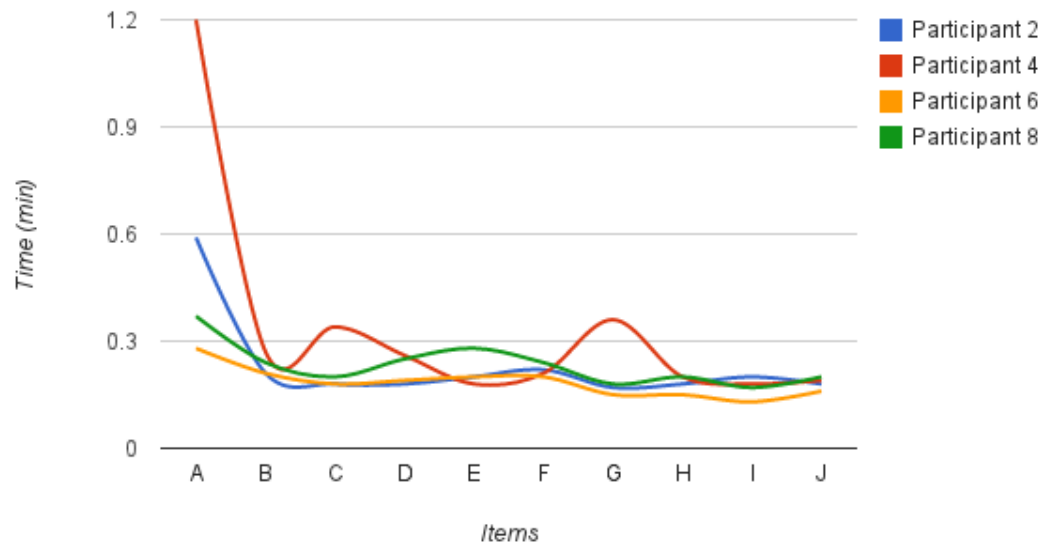


Figure 23: Time spent per item to enter 10 game items using the form; by the four participants who tested first with the form

4.2.3 Design recommendations

An automated approach to the process of entering game items would be the best solution (i.e., uploading a file that contains the items or entering the items directly through the source code of the games). Many participants reported that they would prefer the system to provide import functionality, allowing the submission of great numbers of game items by uploading a file that contains the items (e.g., spreadsheet, xml, json). Another widely supported suggestion was the option to submit the game items programmatically, directly through the source code of the games. In both cases, participants declared that the option to review and edit the items after the import operation completes constitutes a necessity.

The efficiency of the form and the wizard could be improved by allowing duplication of similar items. Some participants reported that the option to duplicate a game item was very desirable. Given the fact that the game items the users had to enter to the system shared many common characteristics, some users were wondering why a “Duplicate Item” option was not available.

The wizard could be improved by removing unnecessary steps and reducing the number of clicks required to enter a game item. What irritated the participants most when they were entering game items with the wizard was the number of clicks they had to do until the process was completed. Indeed, the wizard presented many steps some of which were not providing any real value to the process (e.g., showing the API calls after completing a submission successfully). Furthermore, the wizard was closing after a successful submission (i.e., the view with the list of entered items was presented) forcing the users to reopen it to add a new item, thus demanding more clicks.

4.3 Error rate

Altogether, the tasks were performed without significant errors. Figure 24 shows the error rate per task. As can be seen from the figure, most errors happened in task 3.

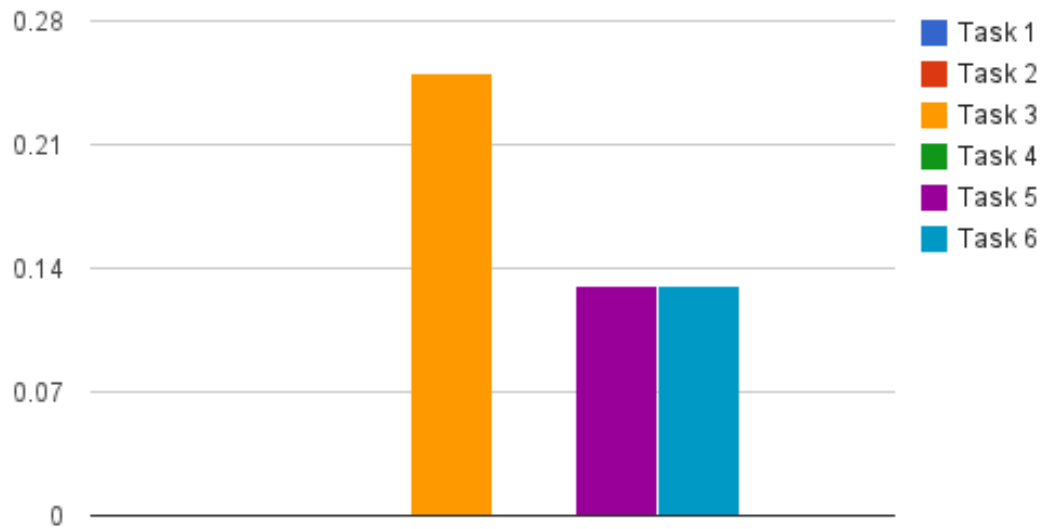


Figure 24: Error rate per task

Task 3 was asking the users to add to the Game Cloud the achievement 501QuestsCompleted of the game Hammerfall. What the users had to do was to navigate to the Achievements tab of the game Hammerfall and click on the green “Add Achievement” button to add the achievement (figure 25). However, two of the participants added the achievement as a game item by clicking on the “Add Item” button in the Items tab (figure 26). A possible explanation to this error could be the fact that the users did not understand the difference between a game item and a game achievement. Furthermore, the users did not consult the instructions provided by the UI in the blue frame (right on top of the items and achievements tables).

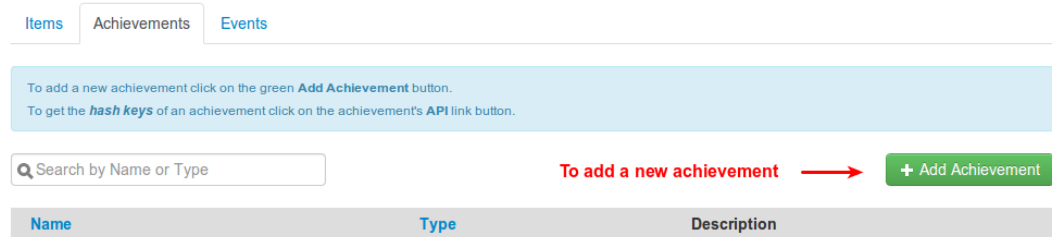


Figure 25: How to add a new achievement

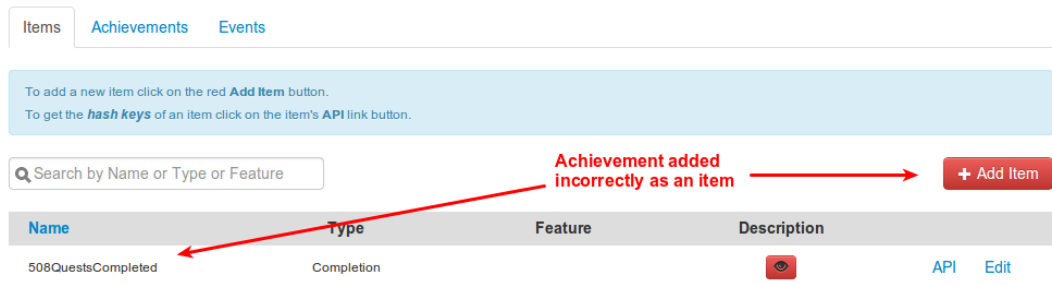


Figure 26: Achievement added incorrectly as an item

The errors that happened in tasks 5 and 6 were trivial copy-paste errors. Tasks 5 and 6 were asking the users to enter 10 game items using the wizard and another 10 game items using the form. The items were given in a table in a simple web page. The method employed by the participants to accomplish these tasks was copy-pasting. In some occasions, the users copied the wrong values. This was most likely due to the fact that the copy-pasting process was rather dull, it had to be repeated numerous times and the users wanted to finish it quickly.

5 CONCLUSION AND FUTURE WORK

This master thesis presented a usability study conducted for the development of a web UI for game developers to enter game specific information. The employed evaluation technique was usability testing. The information collected by the tests was analyzed and translated into a collection of main findings that include the general usability preferences game developers have from the UI as well as design recommendations. The outcome of this study could prove useful to UI designers who would have to design similar products in taking the right design directions.

The results showed that the most valuable usability preferences are quickness and error tolerance. Game developers favor UI elements that allow them to constantly inspect the entered information without having to scroll or switch screens. The use of different colors to categorize and distinguish collections of similar elements is desirable. The help provided by the system was rarely used since the users were first trying to resolve any hindrances by themselves.

The comparison of two different design approaches for one of the most critical and frequently used use cases (i.e., entering game items), a form and a wizard, showed that the users preferred the form. Their justifications included quickness, fewer number of clicks and broader inspection and control capabilities of the entered data from a single screen. On the side of the drawbacks, what was attributed to the form was the fact that it was more prone to errors and more “mentally challenging” compared to the wizard.

However, as the majority of participants stated, the best design approach for entering a great number of game items would be the implementation of import

functionality that would allow the game developers to enter the items either programmatically (i.e., directly through the source code of the games) or by uploading a file in a predefined format containing all the items (i.e., spreadsheet, xml, json).

For future research, the usability of the proposed design recommendations could be examined. Furthermore, the findings of this study could be verified by using other evaluation techniques (e.g., heuristic evaluation). Finally, it would be of interest to research the reasons that lead game developers to ignore the provided help, as well as to search for answers on how to improve the help system to provide assistance when the users need assistance but they have not yet realized it.

6 REFERENCES

- [1] “ISO 9241-210:2010 Human-centred design for interactive systems.” 2010.
- [2] J. Rubin and D. Chisnell, *Handbook of usability testing how to plan, design, and conduct effective tests*. Indianapolis, IN: Wiley Pub., 2008.
- [3] C. Righi and J. James, *User-centered design stories real-world UCD case files*. Amsterdam; Boston: Elsevier/Morgan Kaufman, 2007.
- [4] P. Sherman, *Usability Success Stories: How Organizations Improve By Making Easier-To-Use Software and Web Sites*. Gower Publishing, Ltd., 2012.
- [5] J.-Y. Mao, K. Vredenburg, P. W. Smith, and T. Carey, “The State of User-centered Design Practice,” *Commun ACM*, vol. 48, no. 3, pp. 105–109, Mar. 2005.
- [6] B. Shackel, “Ergonomics for a computer. Design.,” in *Ergonomics for a computer*, 1959.
- [7] X. Faulkner, *Usability engineering*. Houndmills, Basingstoke, Hampshire: Palgrave, 2000.
- [8] B. Shackel, “Human factors and usability,” in *Human-computer interaction*, J. Preece, Ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 1990, pp. 27–41.
- [9] B. Shackel, “The concept of usability,” in *Visual Display Terminals: Usability Issues and Health Concerns*, 1981.
- [10] J. Bennett, “Managing to meet usability requirements,” in *Visual Display Terminals: Usability Issues and Health Concerns*, 1984.
- [11] B. Shackel, “Ergonomics in Design for Usability,” in *Proceedings of the Second Conference of the British Computer Society, Human Computer Interaction Specialist Group on People and Computers: Designing for Usability*, New York, NY, USA, 1986, pp. 44–64.

- [12] UXPA, “Definitions of User Experience and Usability.” [Online]. Available: <https://uxpa.org/resources/definitions-user-experience-and-usability>. [Accessed: 04-Mar-2014].
- [13] M. Good, T. M. Spine, J. Whiteside, and P. George, “User-derived Impact Analysis As a Tool for Usability Engineering,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 1986, pp. 241–246.
- [14] M. B. Rosson and J. M. Carroll, *Usability engineering scenario-based development of human-computer interaction*. San Francisco: Academic Press, 2002.
- [15] T. Gilb, “Design by objectives,” Unpublished manuscript. Available from the author at Box 102, N-1411 Kolbotn, Norway., 1981.
- [16] T. Gilb, “The ‘impact analysis table’ applied to human factors design,” in *First IFIP Conference on Human-Computer Interaction*, London, 1984, vol. 2, pp. 97–101.
- [17] I. B. M. C. R. Division, J. M. Carroll, and M. B. Rosson, *Usability Specifications as a Tool in Iterative Development*. Defense Technical Information Center, 1984.
- [18] K. A. Butler, “Connecting Theory and Practice: A Case Study of Achieving Usability Goals,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 1985, pp. 85–88.
- [19] J. B. Whiteside and K. Holtzblatt, “Usability engineering: our experience and evolution,” in *Human-Computer Interaction*, 1988.
- [20] D. A. Tyldesley, “Employing Usability Engineering in the Development of Office Products,” *Comput. J.*, vol. 31, no. 5, pp. 431–436, Jan. 1988.
- [21] D. A. Norman and S. W. Draper, *User centered system design: new perspectives on human-computer interaction*. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1986.
- [22] D. A. Norman, *The design of everyday things*. New York: Basic Books, 2002.

- [23] D. McCusker and K. Guzik, "CASE 12 - User-centered design for middleware," in *User-Centered Design Stories*, C. Righi and J. James, Eds. San Francisco: Morgan Kaufmann, 2007, pp. 241–267.
- [24] R. Sobiesiak, R. J. Jones, and S. M. Lewis, "DB2 Universal Database: A Case Study of a Successful User-Centered Design Program," *Int. J. Hum.-Comput. Interact.*, pp. 279–306, 2002.
- [25] C. Righi and A. Clow, "Programmers are People, Too: Applying User-Centered Design to Middleware." TaskZ.com article, Feb-2004.
- [26] J. Gulliksen, B. Göransson, I. Boivie, J. Persson, S. Blomkvist, and Å. Cajander, "Key Principles for User-Centred Systems Design," in *Human-Centered Software Engineering — Integrating Usability in the Software Development Lifecycle*, A. Seffah, J. Gulliksen, and M. C. Desmarais, Eds. Springer Netherlands, 2005, pp. 17–36.
- [27] J. Nielsen, "Usability 101: Introduction to Usability." [Online]. Available: <http://www.nngroup.com/articles/usability-101-introduction-to-usability/>. [Accessed: 22-Apr-2014].
- [28] W. Quesenbery, "Balancing the 5Es: Usability," *Cut. IT J.*, vol. 17, no. 2, Feb. 2004.
- [29] C. M. Barnum, *Usability testing essentials ready, set... test!* Amsterdam; Boston: Morgan Kaufmann Publishers, 2011.
- [30] J. Nielsen, *Usability engineering*. San Francisco, Calif.: Morgan Kaufmann Publishers, 1994.
- [31] S. Krug, *Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition*, 2nd edition. Berkeley, Calif: New Riders, 2005.
- [32] UPA, "UPA 2009 Salary Survey Public Version," 2009.
- [33] J. S. Dumas and Redish, *A practical guide to usability testing*. Exeter, England; Portland, Or.: Intellect Books, 1999.
- [34] J. Nielsen, "Why You Only Need to Test with 5 Users." [Online]. Available: <http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>. [Accessed: 28-Mar-2014].

- [35] R. A. Virzi, "Streamlining the Design Process: Running Fewer Subjects," *Proc. Hum. Factors Ergon. Soc. Annu. Meet.*, vol. 34, no. 4, pp. 291–294, Oct. 1990.
- [36] R. A. Virzi, "Refining the Test Phase of Usability Evaluation: How Many Subjects is Enough?," *Hum Factors*, vol. 34, no. 4, pp. 457–468, Aug. 1992.
- [37] J. R. Lewis, "Sample sizes for usability studies: additional considerations," *Hum. Factors*, vol. 36, no. 2, pp. 368–378, Jun. 1994.
- [38] J. Nielsen, "Big Paybacks from 'Discount' Usability Engineering," *IEEE Softw*, vol. 7, no. 3, pp. 107–108, May 1990.
- [39] J. Nielsen, "Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier," 01-Jan-1994. [Online]. Available: <http://www.nngroup.com/articles/guerrilla-hci/>. [Accessed: 22-Apr-2014].
- [40] J. Nielsen, "10 Usability Heuristics for User Interface Design," 01-Jan-1995. [Online]. Available: <http://www.nngroup.com/articles/ten-usability-heuristics/>. [Accessed: 22-Apr-2014].
- [41] J. Nielsen, "Usability Testing for the 1995 Sun Microsystems' Website." 25-May-1995.
- [42] M. C. Medlock, D. Wixon, M. Terrano, R. Romero, and B. Fulton, "Using the RITE method to improve products: A definition and a case study," *Usability Prof. Assoc.*, 2002.
- [43] J. J. McGinn and A. R. Chang, "RITE+ Krug: A Combination of Usability Test Methods for Agile Design," *J. Usability Stud.*, vol. 8, no. 3, pp. 61–68, 2013.
- [44] S. Krug, *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*, 1 edition. Berkeley, CA: New Riders, 2009.
- [45] D. Kane, "Finding a place for discount usability engineering in agile development: throwing down the gauntlet," in *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, 2003, pp. 40–46.
- [46] D. Sy, "Adapting usability investigations for agile user-centered design," *J. Usability Stud.*, vol. 2, no. 3, pp. 112–132, 2007.

- [47] L. L. Constantine, "Process Agility and Software Usability - Toward Lightweight Usage Centered Design," *Repr. Inf. Age AugustSeptember 2002 Revis. Expand. Version Column Manag. Forum Softw. Dev. Vol 9 No 6 June 2001*, 2002.
- [48] J. Tidwell, *Designing interfaces*. Sebastopol, CA: O'Reilly, 2011.
- [49] J. S. Dumas and Loring, *Moderating usability tests principles and practices for interacting*. Amsterdam; Boston: Morgan Kaufmann/Elsevier, 2008.