

Master's Thesis

**Design and Development of a versatile  
psychological experimental environment for  
remote usability testing**

Stefan Mayr, Bakk.rer.soc.oec.

Institute for Information Systems and Computer Media (IICM)  
Graz University of Technology



Supervisor: Assoc. Prof. Andreas Holzinger, PhD, MSc, MPh, BEng, CEng, DipEd, MBCS

Graz, March 2010

# Masterarbeit

(Diese Arbeit ist in englischer Sprache verfasst)

## **Entwurf und Entwicklung einer vielseitig verwendbaren Remote-Usability Test-Umgebung**

Stefan Mayr, Bakk.rer.soc.oec.

Institut für Informationssysteme und Computer Medien (IICM)  
Technische Universität Graz



Betreuer: Univ.-Doz. Ing. Mag. Mag. Dr. Andreas Holzinger

Graz, März 2010

## **Abstract**

Conducting usability tests in the laboratory is usually a time consuming and expensive task, due to the fact that the experimental participants have to travel to the laboratory and have to perform the tasks in a sequential order. Consequently, most of these lab based usability tests are executed with a relative small sample size, thus not representing the real users of a widespread product or service. Moreover, during the test of a Web page, which is visited from all over the world it would be ideal to include people of diverse cultural background. The application of remote usability testing may close the gap between tests in the lab and tests in the field, e.g. at the workplace. These remote tests may be done in a synchronous and an asynchronous way, depending whether the test execution and the evaluation are done at the same time or if it is separated in time.

Within this master's thesis a tool for such a remote, asynchronous usability has been designed and developed. In contrast to existing tools on the market, this tool has an open interface for usability interested people to implement their own usability tests. This is achieved by a modular architecture which can be easily expanded such as writing a plug-in for a content management system.

The backend for managing the remote usability tests was developed on basis of a JavaScript framework which enables to create Web applications in the style of standard software programs. A virtual desktop with desktop symbols and content which is opened in windows looking and behaving similar as in every windows based operating system, generates a new user experience.

Besides of the technological aspects of the developed software and the tools used this master's thesis, it covers some possible business models which could be applied for generating revenues. As such online business models always need a way of direct online payment, some possibilities are discussed.

## **Kurzfassung**

Die traditionelle Durchführung von Usability Tests in Labors kann einen hohen Zeit- und Geldaufwand bedeuten, da alle Testpersonen direkt das Labor aufsuchen müssen, um an den Tests teil zu nehmen. Aus diesem Grund ist auch die Anzahl der teilnehmenden Personen meist auf eine kleine Anzahl beschränkt. Dieser Personenkreis repräsentiert – vor allem wenn eine weltweit erreichbare Webseite getestet werden soll – meist kaum die tatsächliche Personengruppe. Gerade in Bezug auf eine internationale Webseite sollten möglichst Personen aus allen verschiedenen Kulturkreisen in die Tests mit einbezogen werden. Da dies nahezu unmöglich ist, sollen Remote Usability Tests gerade diesen Nachteil ausgleichen. Bei diesen Tests sind die durchführenden Usability Experten räumlich von den StudienteilnehmerInnen getrennt. Diese können die Tests bequem von ihren jeweiligen Arbeitsplätzen ausführen.

Im Rahmen dieser Diplomarbeit wurde ein Werkzeug entwickelt, das die Durchführung solcher ‚Remote Usability Tests‘ ermöglicht. Im Gegensatz zu bestehenden Lösungen bietet die entwickelte Software eine offene Schnittstelle zur einfachen Erweiterung der angebotenen Usability Tests. Ähnlich wie Plug-Ins bei Content-Management-Systemen können so eigene Tests von Entwicklern hinzugefügt werden.

Der Administrationsbereich basiert dabei auf einem JavaScript Framework, das die Erstellung von Webapplikationen ermöglicht, die das „Look and Feel“ von normalen Softwareprogrammen haben. Die Inhalte der einzelnen Seiten bzw. Funktionen werden dabei in Fenstern, wie sie von jedem fensterbasierten Betriebssystem her bekannt sind, dargestellt. Alle Fenster werden dabei auf einem virtuellen Desktop mit Desktop-Symbolen dargestellt.

Neben den technischen Hintergründen der entwickelten Software wird im Rahmen dieser Diplomarbeit auch über mögliche wirtschaftliche Verwertung diskutiert. Verschiedene Geschäftsmodelle werden gegenübergestellt und die dafür notwendigen Online-Bezahl-Systeme beleuchtet.

## STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, 15<sup>th</sup> March 2010

.....

Stefan Mayr

## Danksagung

Mein Dank gilt all jenen Menschen, die mir während meinem Studium immer zur Seite gestanden sind. Im Besonderen möchte ich mich bei meiner Freundin Sabine bedanken, die mich in der Zeit der Erstellung dieser Arbeit immer unterstützt und auf viel gemeinsame Zeit verzichtet hat. Natürlich danke ich auch meinen Eltern, die mich immer motiviert haben, mein Interesse zu verfolgen. Ein großer Dank gilt auch meinen wahren Freunden, mit denen ich immer durch Dick und Dünn gehen konnte.

## Table of contents

<b>1. Introduction</b> .....	<b>1</b>
<b>2. Theoretical Background</b> .....	<b>2</b>
2.1. Usability.....	2
2.1.1. Usability according to the ISO 9421-11:1998 definition .....	2
2.1.2. Usability defined by Jacob Nielsen.....	3
2.1.3. The 5 dimensions of usability by Quesenbery .....	5
2.1.4. Summary of the different definitions.....	6
2.2. Web Usability .....	7
2.2.1. Types of web applications.....	7
2.2.2. Importance of Web-Usability .....	8
<b>3. Common testing in laboratory</b> .....	<b>9</b>
<b>4. Laboratory method categories</b> .....	<b>10</b>
4.1. General categories .....	10
4.2. Usability inspection categories .....	11
<b>5. Methods</b> .....	<b>12</b>
5.1. Usability Testing .....	12
5.1.1. Definition.....	12
5.1.2. Involved persons .....	13
5.1.3. Process .....	15
5.2. Cognitive Walkthrough.....	16
5.2.1. Process .....	17
5.3. Pluralistic Usability Walkthrough .....	18
5.3.1. Procedure .....	19
5.4. Heuristic evaluation .....	19
5.4.1. The heuristics .....	21
5.4.2. Process .....	22
5.5. Formal usability inspection .....	22
5.5.1. The six logical Steps of the formal user inspection .....	24
<b>6. Remote Evaluation</b> .....	<b>26</b>
6.1. Reasons for remote evaluation.....	26
6.2. Kinds of remote evaluation.....	27
6.2.1. Separated in place.....	27
6.2.2. Separated in place and time.....	27
6.2.3. Third party services .....	28

6.3.	Summarization .....	29
6.3.1.	Comparison of equipment used and quantity of data gathered .....	29
6.3.2.	Cost to collect and analyze data.....	30
<b>7.</b>	<b>Existing Tools .....</b>	<b>32</b>
7.1.	Moderated tools.....	32
7.2.	Unmoderated Tools .....	32
<b>8.</b>	<b>Possible commercialization.....</b>	<b>34</b>
8.1.	Classification.....	34
8.1.1.	Three classification options.....	34
8.1.2.	Classification of the developed software.....	34
8.2.	Electronic Payment Systems .....	36
8.2.1.	Online Credit Card Payment Systems.....	36
8.2.2.	Electronic Check Systems.....	37
8.2.3.	Smart Card based Systems.....	38
8.2.4.	Digital or Electronic Cash Systems .....	38
8.3.	Project specific requirements for an payment processor .....	39
8.4.	PayPal as a requirements fulfilling company .....	40
8.4.1.	Credit card acceptance.....	41
8.4.2.	Multinational usage.....	41
8.4.3.	Recurring payments .....	41
8.4.4.	Application Programmable Interface .....	42
8.4.5.	Micropayments .....	42
8.5.	PayPal integration options .....	43
8.5.1.	Website Payments Standard .....	43
8.5.2.	Express check out.....	43
8.5.3.	Direct payment.....	44
8.6.	Licensing Models .....	44
8.6.1.	Approximately costs.....	45
8.6.2.	Free usage .....	46
8.6.3.	Usage fee.....	48
•	Pay per use .....	48
•	Subscription.....	48
8.6.4.	License.....	49
<b>9.</b>	<b>Software Requirements.....</b>	<b>50</b>
9.1.	Introduction .....	50



9.1.1.	Purpose.....	50
9.1.2.	Intended Audience and Reading Suggestions .....	50
9.1.3.	Project Scope.....	50
9.1.4.	References.....	50
9.2.	Overall Description.....	51
9.2.1.	Product Perspective .....	51
9.2.2.	Product Features .....	52
9.2.3.	User Classes and Characteristics .....	55
9.2.4.	Operating Environment.....	55
9.2.5.	Design and Implementation Constrains.....	56
9.2.6.	User Documentation .....	56
9.2.7.	Assumptions and Dependencies .....	56
9.3.	System Features .....	57
9.3.1.	Register at the platform .....	57
9.3.2.	Buying credits .....	58
9.3.3.	Test creation.....	58
9.3.4.	Generate users .....	59
9.3.5.	Add user to test.....	60
9.3.6.	Delete users.....	60
9.3.7.	Email invitation.....	61
9.3.8.	View results .....	61
9.3.9.	Participating in an online test without being invited.....	61
9.3.10.	Accept an invitation for an usability test .....	62
9.3.11.	Declining an invitation for an usability test.....	63
9.3.12.	Request for user list removal .....	63
9.3.13.	Performing the test .....	63
9.4.	External Interface Requirements .....	64
9.4.1.	User Interfaces .....	64
9.4.2.	Hardware Interfaces.....	66
9.4.3.	Software Interfaces .....	66
9.4.4.	Communication Interfaces .....	66
<b>10.</b>	<b>Solution .....</b>	<b>67</b>
10.1.	Concept .....	67
10.2.	Conductors User Interface .....	67
10.2.1.	Conductors entry point .....	67

10.2.2.	Password generation.....	68
10.2.3.	Login process.....	69
10.2.4.	Server side form validation .....	69
10.2.5.	Form data transmission.....	70
10.2.6.	Email verification.....	71
10.2.7.	Email validation .....	71
10.3.	Email invitation.....	73
10.3.1.	Customized invitation links .....	73
10.3.2.	Email spooling .....	73
10.4.	Test user frontend.....	74
10.4.1.	Start screen (splash screen) .....	74
10.5.	Example test (Picker).....	75
10.5.1.	Background.....	76
10.5.2.	Settings.....	76
10.5.3.	Procedure .....	77
10.5.4.	Data submission .....	78
10.6.	Data model.....	78
10.6.1.	Scheme .....	78
10.6.2.	Central tables .....	79
10.7.	Custom model concept .....	81
10.7.1.	Directory structure .....	81
10.7.2.	Class file.....	82
10.7.3.	JavaScript logic .....	83
10.7.4.	Settings.....	83
10.7.5.	Help .....	84
10.7.6.	Endscreen .....	84
10.7.7.	Results .....	84
10.7.8.	Recognition of valid class files.....	84
<b>11.</b>	<b>External software components .....</b>	<b>86</b>
11.1.	MochaUI.....	86
11.1.1.	Windows design and behaviour .....	86
11.1.2.	Basic steps when setting up a MochaUI interface .....	88
11.1.3.	Further useful methods.....	89
<b>12.</b>	<b>Future Work.....</b>	<b>91</b>
<b>13.</b>	<b>Conclusion .....</b>	<b>92</b>

**Appendix..... |**  
A. MIT License..... |

## 1. Introduction

One of the biggest problems in executing formal experiments is the lack of proper test user, who can attend a test session at a specific time and a specific place. Therefore most experiments are done by only a couple of users in a special test environment. Such environments guarantee optimal test performance but don't guarantee the coverage of all demographic possible users. Specially hiring people of foreign countries is nearly impossible. Online experiments can overcome such limits by being accessible from everywhere, 24 hours a day. The disadvantage of having tests in an unsupervised way should be even by the bigger number of participants in the study.

Such online experiments need a special software system which enables people to set up a virtual test environment in a simple, easy to use way. The system itself should be designed to run quiet in the background and should not influence the test in any way. For example, timing tasks should not be blocked by heavy arithmetic operations.

As a lot of usability test services exist on the market, the goal is to develop a piece of software which is different to all available solutions. Therefore the platform itself should be designed like a virtual desktop, giving the user a complete new feeling of navigating within a webpage. This can be achieved by using an existing JavaScript framework as a basis. The second unique feature is an open interface for expanding the set of available usability tests.

Different options for generating revenue from the software exist which will be discussed in detail as well as the necessary online payment services which are necessary to generate this revenues in a direct and automated way.

## 2. Theoretical Background

### 2.1. Usability

Defining usability is not that easy as it might be. A common accepted definition does not exist even though a ISO standard exists since 1998 defining usability as: “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.”(Normungsinstitut, 1998)

As stated in the ISO standard, usability is not a characteristic of a user interface but more like a function of users using a product or system to achieve a goal with a set of attributes (Jokela et al., 2003). Generally usability might be seen as a factor of system quality in a human computer interaction system / situation.(Matera, Rizzo & Carughi, 2006)

#### 2.1.1. Usability according to the ISO 9421-11:1998 definition

The attributes stated in the ISO 9241-11:1998 standard are extended by some usability experts as stated below. To see the differences and/or the commons within the definition of the usability experts and the ISO norm it's important to clarify the meaning of these original ISO attributes by giving the original definitions(Normungsinstitut, 1998):

- **Effectiveness:** *the accuracy and completeness with which users achieve specified goals*
- **Efficiency:** *the resources expended in relation to the accuracy and completeness with which users achieve goals*
- **Satisfaction:** *freedom from discomfort, and positive attitude to the use of the product*
- **Context of use:** *characteristics of the users, tasks and the organizational and physical environments*
- **Goal:** *intended outcome*
- **Task:** *activities required to achieve a goal*

Especially the goals and the satisfaction attribute are very subjective ones. According to the purpose of the usage of a product or system, a goal could be to have a real output like a document written but also a hard to measure goal of enjoyment when playing a game. The usability of a product/system can vary a lot when it is used in different context/goal combinations. For example a novice and an expert may find a system more or less usable even

they have the same goal. If the system is designed for professionals with a lot of sophisticated features a novice may have problems understanding all the features. Also the other way round a professional may feel that the system less usable when he does not find professional tools in a system made for novices. Taking in consideration such preconditions may increase the usability of a product/system.(Normungsinstitut, 1998)

The effectiveness and the efficiency attribute are easier to measure. Effectiveness indicates if a system allows users to achieve their goals or sub goals in the manner of completeness and accuracy. If the goal is for example to reproduce a 2 page paper, the number of typing mistakes may be counted. A way to calculate the effectiveness is to divide the total number of words with the errors occurred.

Efficiency – in this definition – is calculated by taking the effectiveness factor and the used resources. Those resources may be psychological or physiological efforts as well as time, money or all other material used. For example let's imagine the situation when copying a sheet of paper. The efficiency could be calculated dividing the number of correct copies (effectiveness) by the used effort (time, costs).

### **2.1.2. Usability defined by Jacob Nielsen**

5 years before the ISO 9241-11:1998 was published Jacob Nielsen(Nielsen, 1993) stated a model of five factors, which should be present in a high usable system. These five factors differ from the ISO definition but are related in some manner.

- **Learnability:** means that users which have not used the product / system so far can learn the usage in a in a timely short manner. It's the most important usability attribute because the use of nearly every system has to be learned at the beginning. In some cases, the learnability can be increased by being consistent to older versions of the system or to similar products. The learnability also depends on the user target group the system is designed for. If the focus is on novice users, the system should be easy and fast to learn. On the other hand, systems designed for expert users sometimes afford training and those expert users need some time longer to get into a phase of productivity.

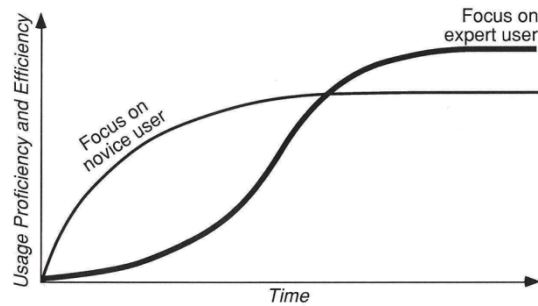


Figure 1 Learning curve according to Nielsen

- **Efficiency:** once the user know how to use the system, it should be possible to have a high level of productivity when working with the system. The efficiency of a system can only be measured when the user knows the system well and is in a steady-state level of performance. Otherwise the lack of experience of the user would falsify the system efficiency. This state is reached when the learning curve flattens (see **Error! Reference source not found.**).
- **Memorability:** when a user has used the system some time ago it should be no problem to use it again without relearning the usage. The memorability is especially important for casual users, which belong to the third group of users beside the novice and expert users. Casual users have already learned the usage of the system but do not use the system frequently or have temporarily stopped using the program, so the user interface has to be easy to remember to ensure a smooth restart.
- **Errors:** on the one hand the system should have a low error rate itself and on the other hand the error behavior should enable the user to easily recover from them. Minor errors are mostly corrected by the users while performing a task and need not to be counted separately because the slowdown is already included in the efficiency factor. More dangerous are errors, which are not discovered by the user, leading to faulty work product or result in a loss of the entire work of the user. Therefore such errors have to be counted separately.
- **Satisfaction:** this subjective factor indicates how pleasant it is for the users using the system. There exist some psycho physiological measures such as EEGs, pupil dilation and so on. But the overwhelming number of usability studies use questionnaires for that purpose which produce good results when using the average of all the user ratings.

### 2.1.3. The 5 dimensions of usability by Quesenbery

Another, nearly corresponding definition of the most important factors of usability is stated by Quesenbery (Quesenbery, 2002). Based on the ISO 9241 characteristics of usability (efficient, effective and satisfying), Quesenbery expanded the ISO definition to 5 dimensions:



Figure 2 The five usability dimensions according to Quesenbery

- **Effective**

It measures how effective people can reach their goals in means of completeness and accuracy of the results, which may be different due to the task they have to cover. Goals may not only be to complete a task and get an answer to a previous defined question (for example calculating the sum of two numbers) but may also be the joy of playing a game. In this case, a product is effective if the user enjoys it. Regardless to the usage, the result should be reached with a minimum effort.

- **Efficient**

Efficiency can be measured in time units and in total actions needed to complete a task. The way from the beginning to the completeness of a task differs in some cases, which have to be considered when measuring efficiency in context with usability. Typing in a keyword in a search engine and press the submit button is for example a very efficient task as it takes only some seconds. But if choosing a result among the returned suggestions is considered as a part of the searching task, it may take longer and be inefficient. So the user's perception should be included when defining a task.

- **Engaging**

As engaging is an indicator about how much the system is encouraging the user to interact with the system it's a very subjective factor of usability and therefore hard to measure. The visual design is one of the most important factors of an engaging system,



but also the language and the interaction style used play a part in a system that creates a positive user experience. For example, a 'related topics' link in a help system may help a user to find the correct answer to his problem and make the help system more easy and comfortable to use.

- **Error** **tolerant**  
A user friendly system should be tolerant to any error which may occur. This include errors which origin from bugs within the code or misunderstanding the users mental model as well as errors which are caused by mistakes made by the user. In any case the user must have the opportunity to recover from such errors so he can complete the task with success and have enough information to prevent such errors in future if he made a mistake.
- **Easy** **to** **learn**  
Easy to learn is divided into an initial learning phase and a deeper understanding of the system, so the user knows about all the capabilities of a product. Intuitiveness – which is important especially in the initial learning phase – is mostly achieved by a consistent and predictable interface. For example similar elements should always be on similar places or clicking on a certain class of buttons should always lead to the same behavior. Interfaces of different systems which are similar to each other should have similar interfaces so users can use knowledge which they learned by using another but similar system.

#### **2.1.4. Summary of the different definitions**

The context within a system/product is used is an important factor when talking about the usability of this system/product because usability is understood differently by different user target groups. End users will have another viewpoint than software developers for which usability describes the internal attributes of a system rather than the well usable features recognized by end users.(Abran et al., 2003)

But not only the skills of the user has to be considered when talking about a specific system, also the type of the task and the situation of use has influence on the three usability attributes defined in the ISO 9241-11:1998 standard (effectiveness, efficiency and satisfaction). (Dillon, 2001) The type of the task may be to play a game which will define the 3 attributes completely different as in the case of a software development task. And if the product/system has to be

used outdoors may influence the factors as well since there are many environmental factors like sunlight, noise which can make a system which has a good usability indoors unusable.

Regardless which model of defining usability is used the factors are always context-bound and have to be weighted according to the situation. To measure usability you always have to know your users and the situation in which the system is used.(Quesenbery, 2004)

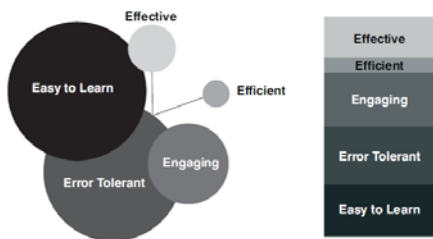


Figure 3 Embolyees usability needs [8]

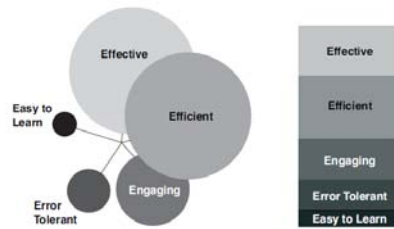


Figure 4 Experts usability needs [8]

## 2.2. Web Usability

### 2.2.1. Types of web applications

Web usage may be divided into three types of applications. From the beginning of the web there have always been the content driven and document orientated applications, which are commonly named "Web hypermedia applications".(Olsina, Covella & Ross, 2006) They are characterized by being structured by nodes which are accessible through links, anchors and access structures for navigation. All the data (content and layout) is delivered over the internet in form of HTML or XML formatted files.(Mendes, Mosley & Counsell, 2006)

The second has become a fast growing part in the web usage specially since of the development of asynchronous scripting languages (AJAX) and cascade style sheets (CSS). These both techniques allow "web applications", which are executed within a browser, to interact with the server in a way that all the data can be stored on the server and the client only acts as an input/output terminal to represent the results of the queries. This type of applications had an big impact in last view years. "All the cool, innovative new projects are online." (Garrett, 2005)

The third one are "Web software applications" which are more like standard 'stand alone' software but these applications rely on the Web or uses the Web's infrastructure for execution.(Mendes, Mosley & Counsell, 2006)

### **2.2.2. Importance of Web-Usability**

“Web applications are interactive, user-centered, hypermedia-based applications, where the user interface plays a central role”.(Olsina, Covella & Ross, 2006) That's the reason why web based applications differ from normal software where people buy a piece of software and pay it at the beginning of the usage. Bugs in traditional software often lead into new releases which are offer as separate products. On the internet it is essential that people which using the websites come back and reuse it. Web applications life from the returning traffic and therefore web application developer only see a return on their investment if they satisfy their customers' needs. If the usability of a site is poor web user often tend to switch to other similar sites due to a little site loyalty.(Offutt, 2002)

### **3. Common testing in laboratory**

Laboratory based usability evaluation is the traditional user interface evaluation and is always directly located at a laboratory where the usability experts prepare such tests in a way that every participant has the same preconditions about the test. So when inspecting a user interface, all participants do their tests on the same machine or an exact equivalent machine in the mean of same hardware and software. Also the operating system and all necessary settings are equivalent. The same preconditions are necessary to compare each of the test with all other of the series.

During the test, the evaluator and maybe also some operators who are responsible for the technical settings are also present in the same room as the user. (Hartson et al., 1996)

During the test session as much information is recorded as possible. Nowadays capturing the monitor screen is as well standard as recording the person in front of the computer to capture all the emotions, because not every problem is spoken out by the user.

Another way to be with the user at the same place at the same time is to move the lab equipment directly to the user at his common work location, so the user can use own equipment. But this way is rather expensive due to high transport costs.

In any case a set of representative user is selected by the evaluators to ensure that the future users will match the test users as good as possible. Due to the fact that every participant has to come to the lab or that the lab has to come to every single person only a small number of people are normally involved at such studies. (Tullis et al., 2002)

## 4. Laboratory method categories

### 4.1. General categories

There exists some different categorizations of evaluation methods, most of them divide the methods into very general classes like Nielsen and Molich (Nielsen & Molich, 1990) which state 4 different categories:

- Formally by applying some analysis techniques
- Empirically by having real users solving real tasks
- Automatically by some computerized procedures
- Heuristically by an interface check

Nielsen also states that people do not participate in empirical tests due to the time effort, the lack of expertise or simply because they dislike such tests. (Nielsen & Molich, 1990) Another problem is that such tests require an adequate number of test users which are difficult or expensive to recruit especially when different versions of a user interface have to be tested over a period of time. (Nielsen, 1994) So in real life most of the evaluations are done by heuristic evaluations where experts review the interface and state proposals for the improvement. (Nielsen & Molich, 1990)

Another categorization is stated by Wixon (Wixon & Wilson, 1997) who splits the usability evaluation methods into 5 classes:

- Formative / summative methods: while summative methods are used for evaluating existing systems, formative are used for identifying new ideas (Holzinger, 2001, Holzinger, 2003).
- Decision / discovery methods: whenever a decision within alternatives is needed (for example for different kinds of buttons) a decision method is used. In contrast, discovery methods help understanding user behavior, how they work and where problems exist.
- Formal / informal methods: formally described methods are used to show the connections between different facts from a theoretical point of view while evaluators often use them informally.

- User involvement: this class divides the methods into a category where users are involved in evaluation, analysis and design and a second one where no real users are needed.
- Component / complete methods: component methods only cover only one or some special parts of the evaluation process while complete methods cover all steps which are needed to complete the design from the usability point of view.

## **4.2. Usability inspection categories**

Also usability inspection methods may be divided into different categories. Virzi (Helander, 1997) found 4 possible classes:

- The characteristics of the judges: this categorization divides the usability inspection methods into two classes according to the type of the evaluator experiences in usability. The so called 'non experts' may have experiences in other domains but are not specially trained in usability science. Usability experts belong to the other class. Virzi names two reasons for using non experts: a limited budget which does not allow paying for usability experts or if experts are simply not available.
- Number of evaluators in a single session: when more than one evaluators work together on a usability project in a single session they can discuss and come to a single conclusion. On the other hand if only one evaluator works on the project, it results in different and independent opinions. So even only one evaluator is present at a single session, multiple sessions may be held with different evaluators. These evaluations also belong to the group with only one evaluator.
- Goals of the inspection: finding usability problems is not the only purpose of usability evaluation even if this is the most common reason. Other reasons may be the improvement of the learnability which can for example achieved by the cognitive walkthrough method.

## 5. Methods

In this chapter the most important usability evaluation and usability inspection methods are described. The participation of real users in the test marks the difference between these two categories.

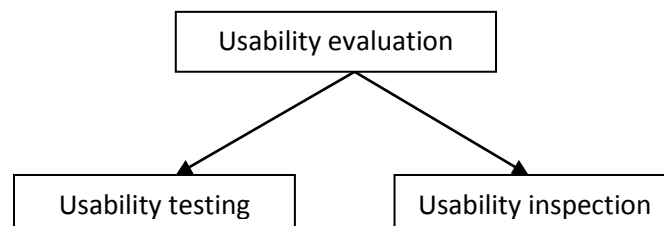


Figure 5 Usability evaluation methods

### 5.1. Usability Testing

#### 5.1.1. Definition

As Holleran defines, usability testing is “the gathering of information about the use of prototypes of software products from users who are not involved in the design of the products themselves”. (Holleran, 1991)

As usability testing gathers information directly from real users the method is indispensable in some kind, because it shows real user behavior and their exact problems with a specific interface. (Holzinger, 2005)

Within the pool of several usability testing methods, Thinking Aloud, questionnaires and field observations are the most common and Thinking Aloud is the most valuable because the developers get an idea ‘why’ user do something. (Holzinger, 2005)

For Dumas and Redish there exist a lot of variations of usability testing, but all of them have to have 5 common characteristics to be classified as a usability test (Dumas & Redish, 1999):

- **Improving the usability of a product is the primary goal** of every usability test even other goals may coexist. This primary goal distinguishes the usability test from other tests. Function tests for example focus on the confirmation that the prototype

matches the specifications stated in the software requirements document. The primary goal must not be stated in such a general way and can be specified more precisely. According to the state of the project, such a test can be used to test the design of the menus in an early development phase where only paper drawings of the interface exist. If the type of users is in focus, another primary goal can be to test an interface designed for novices about acceptance among expert users.

- **The participants represent real user:** only if those users participate in the test that will use the system in real life, the results are usable for achieving the primary goal. This precondition also excludes members of the development team because such tests would belong to another class, the usability inspection or expert reviews. Being separated from the development team is not enough to meet the definition exactly. Also the level of experiences of the users is important to have a representative sample of the end users.
- **The participants do real tasks:** understanding the users and their daily tasks is important when defining the tasks for the test. Beside this primary precondition the tasks should be carefully selected to cover also usability concerns which the developers want to test.
- **Observe and record what the participants do and say:** in a usability test you observe your participants by recording performance and comments during the test as well as getting their subjective opinion about the test and the product by questionnaires. This distinguishes the usability test from other test like surveys, where only the subjective opinion is tracked.
- **Analyze the data, diagnose the real problems, and recommend changes to fix those problems:** recording all the quantitative and qualitative data from the user and the own observations do not satisfy the requirements for a successful usability test. Only if the data is analyzed and used for improvement of the product, such a usability test can be called as successful.

### 5.1.2. Involved persons

The person or the group of persons who prepare and moderate the test sessions should be rather a usability expert than a member of the development team due to the fact, that developers are not always objective when testing their own team. Usability experts may be not familiar in the specific domain of the product but are well trained in preparing and conducting



tests as well as in analyzing the data gathered from the tests. They primary focus on the tasks and the users. (Dumas & Redish, 1999)

As stated in the section 5.1.1 the people acting as test persons should not belong to the real user group. Therefore characteristics like experience, skills, education, type of the job and demographic attributes should match as well as possible the real user group. Known people like college students, family or co workers should not be asked for attending the test. (Hansen, 1991)

The sample size often depend on external factory like money and time budget but also on the importance of getting statistical significant results and how many subgroups the testers need to satisfy their goals and concerns. (Dumas & Redish, 1999)

Keeping in mind that usability evaluation should lead to improved products by finding usability problems a number of people should be found that finds most of the usability problems and keeping the size due to the costs as small as possible.

Nevertheless an exact number of people needed for fulfilling the requirements for finding most usability problem does not exists. While some experts like Nielsen (Nielsen & Landauer, 1993) think that a low number is enough, other experts at other studies showed that more than Nielsen's 5 user are necessary to find 85% of the critical usability problems.

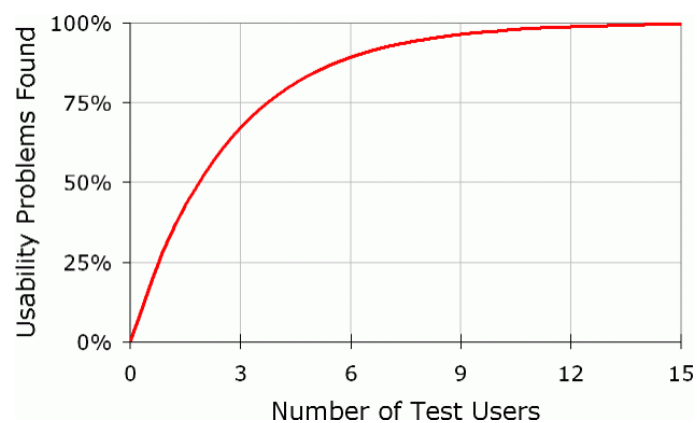


Figure 6 Number of required test users according to Nielsen

Spools Study (Spool & Schroeder, 2001) covered the usability evaluation of 4 websites. All of 4 tests showed that more users are necessary, especially 2 of the 4 tests showed that after 5 users only 35% of the usability problems were found. This low percentage is because of

nowadays websites where sometimes thousands of pages and diverse navigation options enable the user to solve a problem in a very personal way.

It should be mentioned that Nielsen's recommend such a low number only if there is a very homogeny group of users doing all nearly the same tasks.

### 5.1.3. Process

According to the definition and the characteristics of usability tests stated in section 5.1.1 a test is usually divided into 4 steps: (Dumas & Redish, 1999)

- Preparation of the test
- Conducting the test
- Analyzing the data
- Usage of the gathered data

Some of the most important factors for achieving useful results are fixed in the preparation phase of the test. The collected data may become gaping holes when details are overlooked in the preparation phase. (Kantner, 1994) So all involved people have to be selected carefully like the manager of the test and all other people for the associated tasks. The ideal person for the manager job is part or knows about different departments within a company. After being selected the manager should prepare the test by considering the following rules: (Kantner, 1994)

- **Recruiting the right participants** which match the "real-world" audience best
- **Defining the right tasks** which match expected and natural use of the product
- **Minimize activities:** only activities and variables which are needed to generate useful results should be part of the test
- **Same test conditions for every participant:** to generate useful data which is comparable with all other test sessions, each of the participants must have the same conditions.
- **Test the test:** to avoid problems during the test or when analyzing the data, all tests should be tested by a so called 'dry run'.

Every single test session should be well prepared too. Especially because external people – the test users – are involved and are sensible to handle. Nielsen (Nielsen, 1993) states the main ethical considerations within the 3 phases of a test session:

- **Before the test**, all the test equipment should be ready and tested when the user enters the room and the user has to be briefed about all details of the test, the fact that everything will be kept anonymous, the recorded data and that the user can stop the session any time. Especially the users must know that the system is tested, not the user.
- **During the test**, the whole atmosphere must be relaxed so the user feels comfortable. This includes that as less as possible persons take part in the test session and no interruption should happen. Especially no negative response should be given to the user for being slow or making mistakes.
- **After the test** a short debriefing should happen thanking the user for helping and guarantee once again the anonymous usage of the data collected.

A test session normally generates a lot of data like quantitative measurements (task completion times, ratings), background information about the test users, notes of the evaluators, recordings (audio/video) and maybe most important, a list of problems that has been growing over the test.

The goal should be to find the reasons for those problems which may also occur when the product is used in real life by the customers at their working place. The problems discovered at the tests are mostly only symptoms of larger problems. (Dumas & Redish, 1999) Therefore all different kinds of data collected have to be combined to analyze why the problems occurred.

## 5.2. Cognitive Walkthrough

The Cognitive walkthrough method was created for testing how easy user can perform tasks on systems without having instructions or help. Its focus is on the easy of learning (learning through exploration). (Polson et al., 1992) The method can be applied in every stage in the design process directly by the developers and it does not require a prototype or real users. Instead, it will help designers to see the interface from the users point of view and help indentifying future problems when real user interacting with the software. (Rieman, Franzke & Redmiles, 1995)

The theory underlying the method derives from Polson's CE+ theory of exploratory learning (Polson & Lewis, 1990). It describes the human computer interaction in a cycle of 4 steps:

- Users want to perform a task and want to achieve a certain goal (for example printing a page)
- The current page of the user interface is inspected by the user to identify all possible actions (buttons, links)
- The user decides which action seems likely to make progress toward the goal
- The user performs the selected action and waits for the system feedback

These 4 steps will be repeated by the user when the main goal can be reached by completing more subtasks.

### **5.2.1. Process**

Choosing the right tasks to test is the first step when setting up such a test. These tasks should match the reality as close as possible, so they will also be performed from real users at their working place. The set of tasks should also include some exercises which are made up of basic tasks, so the completion of these tasks can only be achieved by completing some subtasks. So the user has to follow a problem solving path rather than simply solving a one step problem. All these tasks should be formulated in a language which can be understood by novice users who never used the interface before. (Polson et al., 1992)

For each of tasks a correct sequence has to be stated to compare test results with the optimal way. If a system offers more than one way to achieve the goal, all alternative paths should be tested. The path through the correct sequence is normally stated by one of the developers. (Polson et al., 1992)

As throughout the test session preconditions about the user decisions have to be made it is very important to know about the future user in detail. Especially what goals they will have, what kind of ways they find hard or easy to work with and which experience they have on same or similar systems. (Polson et al., 1992)

Finding a reasonable story about why the user will choose the right action on every step of the correct sequence is the main task on the cognitive walkthrough process. But this is not a

prediction about user interaction; it is more like considering if whether the user will select each of the correct actions. (Rieman, Franzke & Redmiles, 1995)

Especially this test method where prediction of user behavior is essential, the affordance of the objects on the screen is rather important. Norman (Norman, 1999) defines affordance as the functions a screen object may have due to the implication a user give according to the visible appearance. A Button with the label 'close' affords for example to close something. Sometimes these labels can be misinterpreted due to the level of experience and a correct prediction of the user actions is not possible.

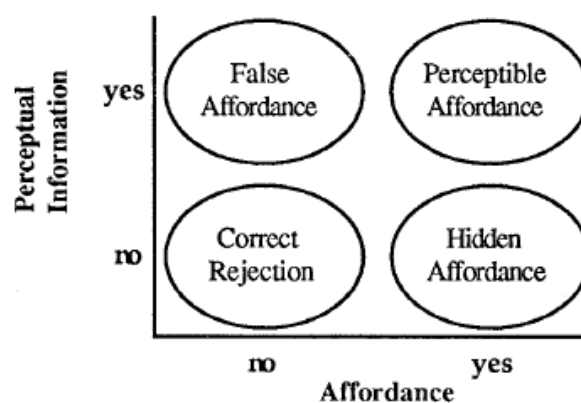


Figure 7: the relations between perceptual information and user affordance. Gaver (Gaver, 1991)

Figure 7 shows the connection between perceptual information (what the user sees) and the affordance (what the user thinks that a thing does). When an item on the user interface suggests a function and it really fulfills that function a perceptible affordance is achieved (upper right corner). Usability problems may occur when an item suggests a function that is not present at the system (upper left corner). Another problem may occur if the existence of an item is expected but is not present (lower right corner).

### 5.3. Pluralistic Usability Walkthrough

The Pluralistic Usability Walkthrough method is a combination of usability test and a usability inspection due to the fact that real users are as well participants as developers and usability experts. The method is defined by 5 characteristics: (Bias, 1994)

- **Three types of participants:** beside the usability experts, product developers and real, representative users participate in the test

- **Hard-copy panels:** every participant receives copies of the panels in the same order than they would appear on screen.
- **Assumption of the user role:** all three types of participants represent real user and their behavior. So the developers and the usability experts must act like normal user rather than as a part of the development team.
- **Write before speak:** before any discussion or any verbal communication all of the participants write down on the hard-copy panel the action they would take in pursuing the designated task online.
- **The order in the discussion:** the real users speak before all other to discuss each panel. Only after they have no further comments, the usability experts and the product developers may speak.

### 5.3.1. Procedure

After all of the participants got an overview about the testing procedure and the rules the product designers give a short briefing about the interface itself and the purpose of the product and its features. This briefing is intended to give the test participants the same precondition about the system knowledge as the end users are assumed to have.

As it is a step by step procedure only the first panel is inspected in the first round. First, all participants write down on the panel which action they would perform in the first step, on the specific screen (panel). The “right” answer is stated by the conductor after all participants have finished their answers before the discussion round start. As stated in the characteristics of this method, the real users are the first to speak while the others remain silent. Next, the developers can join the discussion and often explain why the current design was chosen.

In some cases a solution to found usability problems can be made directly within the discussion session.

The method affords courage of the real users because they have to state problems while the designers are present. On the other hand the designers have to be briefed to be thick-skinned because there could be many problems in the design of their products. (Bias, 1994)

## 5.4. Heuristic evaluation

Heuristic evaluation is a member of the usability inspection methods because experts review a user interface and come up with opinions about the good and the bad things of a certain part of the user interface. To decide what is 'bad' or what is 'good' the evaluators have some guidelines which are a collection of fundamental usability rules and are called 'heuristics'. Before this a short and comprehensive list of rules was stated such methods were not successful or were intimidating to the evaluators due to lists with thousands of rules. (Nielsen & Molich, 1990)

The method was developed to have an inexpensive usability inspection opportunity because due to the list of heuristics no or less usability experience is needed to perform such tests. But Nielsen (Nielsen, 1992) showed that usability experts are more successful in finding major and minor usability problems. Especially when the experts also had experiences in the domain the user interface was made for. Such 'double experts' found 61% of the major usability problems and 59 of the minor problems while novice evaluators only found 29% (major) and 21% on minor problems.

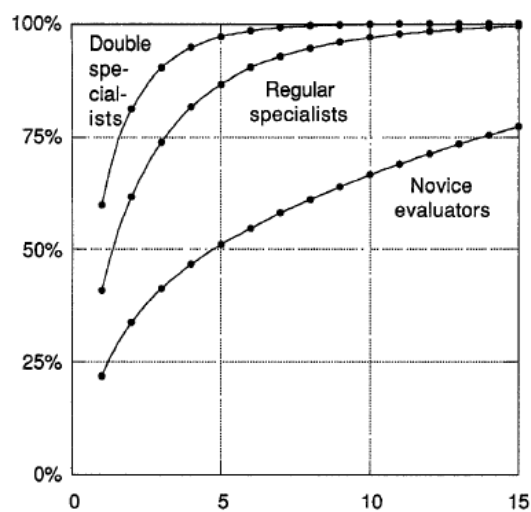


Figure 8 number of evaluator needed, Nielsen (Nielsen, 1992)

As shown in Figure 8 a lot of novice evaluators are needed to get 75% of the usability problems found while double specialists are much more efficient. Even when using regular specialists, a number of 9-10 should uncover more than 90% of the usability problems. Regardless to the level of experience a single evaluator tend to find much less problems than when having more tests and combining the result sets.

### **5.4.1. The heuristics**

The list of 9 heuristics was build upon the experience of Nielsen and Moloch which they gathered in years of teaching and consulting usability. According to Nielsen the list represents a huge number of usability problems because many of the problems can be summarized in those 9 classes. (Nielsen & Molich, 1990) Over the years Nielsen refined the original 9 heuristics to 10 similar usability rules: (Nielsen, 2005)

#### **1. Visibility of system status**

Within a appropriate time the user should be kept informed about the system status

#### **2. Match between the system and the real world**

The user interface and all messages should be in a language that can be easily understood by the users. This can be achieved by using real world terms, phrases or concept rather than sophisticated, technical ones.

#### **3. User control and freedom**

Users should always have to opportunity to cancel operations without any complicated procedures. Especially if they choose a system function by mistake.

#### **4. Consistency and standards**

Similar functions should be labeled and placed in a consistent way so a user can get handle a new screen without learning new things.

#### **5. Prevent errors**

An ideal system should have no errors at all even well understandable error messages are chosen. As ideal systems do not exist, prompt user before they enter a error-prone function.

#### **6. Recognition rather than recall**

All available options, actions and objects should be clearly visible and recognizable for the user, so he does not have to remember all behaviors within different screens.

#### **7. Flexibility and efficiency of use**

Special features for experienced users should be provided to speed up certain processes. These features should be hidden for the novice users.



## **8. Aesthetic and minimalist design**

Only really necessary information should be used on dialogs and screens.

## **9. Help users recognize, diagnose, and recover from errors**

Always useful error messages should be used, providing a solution for recovering from the error. Especially no error codes should be used since they are only important for developers.

## **10. Help and documentation**

Even when the system is easy and self explanatory a easy to use help system should be provided.

Some practical heuristics are e.g. also provided in 'Mobile computer Web-application design in medicine: some research based guidelines' (Holzinger & Errath, 2007) and 'Investigating Usability Metrics for the Design and Development of Applications for the Elderly' (Holzinger et al., 2008).

### **5.4.2. Process**

Even when more evaluators are chosen to perform a heuristic usability evaluation the tests are separated and they are not allowed to communicate as long as not all tests of all evaluators are finished. This should guarantee that other tests are not influenced.

Within a test session the evaluator may have an observer helping him to record the findings or if the evaluator has limited expertise in the domain. But in contrast to usability evaluation where real users are tested, the observer only records the findings of the evaluator and does not interpret the results.

Nielsen recommends performing such tests always at least twice because the evaluator decides by his own which and in which order he performs certain tasks. So he can get familiar with the system in the first run and when he knows about the whole, he can go into detail at every single step.

## **5.5. Formal usability inspection**

This usability inspection method is conducted by following a formal process for detecting and describing defects. Kahn (Kahn & Prail, 1994) defines the defects found with this method as "a

product characteristic that makes it difficult or unpleasant for users to accomplish tasks supported by the product.” This definition is close to that one of usability problems.

The formal usability inspection method has three characteristics: (Kahn & Prail, 1994)

- **A defect detection and description process:** structure to the design review process was provided to increase engineer recognition of defects. According to Kahn (Kahn & Prail, 1994) development team members do not spontaneously notice usability defects during design discussions, informal reviews and traditional inspections.
  - **Stepping through the task:** a task always includes a user profile which contains the level of education and the level of experience of the fictive user, so the inspector can understand the users’ perspective. The second part of a task includes a task scenario, containing a user goal, the starting point and an intermediate situation. For example a goal could be to print a page, a starting point within this example would be that the printer dialog is open and the intermediate situation could be that the wrong paper size is selected.
  - **Where to look:** The evaluators are supported by giving them a users’ task performance model (see Figure 9). Every of the 4 steps within this model (perceiving, planning, selecting and acting) are connected to a set of questions which helps to places where usability problems may occur.
  - **What to look for:** in difference to the ‘task performance model’ this section covers looking for heuristics (usability principles) to find usability problems.
  - **How to log defects:** the intent is to describe defects in a user-centered format rather than in a vaguely way (“I don’t like the ....”) or giving direct solutions (“the label should be changed...”).
- **An inspection team:** Besides supporting members like moderator or one for logging the findings, a number of 3-5 complete the team. These inspectors are selected to represent areas of knowledge (software, hardware, documentation, support and human factors engineering).
- **A structure within the usability lifecycle:** also the usability lifecycle is characterized by a formal procedure containing 6 logistical steps which will be explained below.

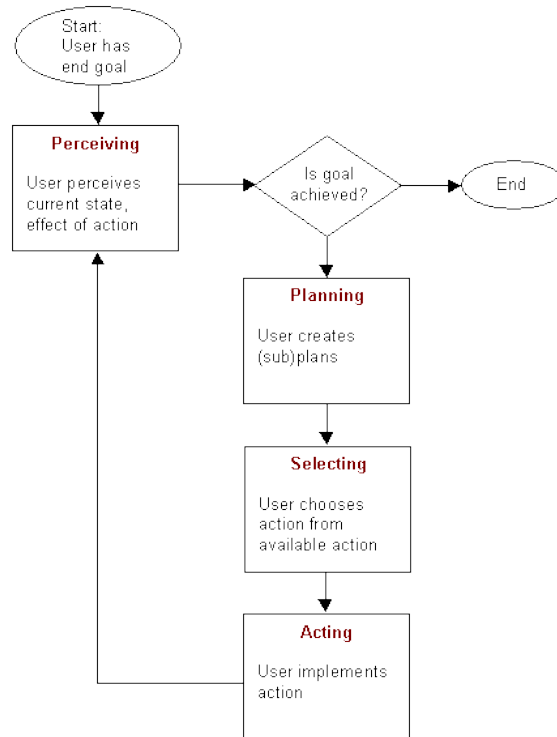


Figure 9 The task performance model (Kahn & Prail, 1994)

### 5.5.1. The six logical Steps of the formal user inspection

To ensure a complete and efficient method completion, 6 logical steps are defined:(Kahn & Prail, 1994)

- **Planning:** primary it's up to the moderator and owner to prepare all necessary information and organize a group of appropriate engineers.
- **Kickoff Meeting:** the intent of the kickoff meeting is to get know of each other and handing out the inspection packets. The content of these packets are discussed as well as the principle and details about the method.
- **Preparation:** after becoming familiar with the packages, single tasks are carried out by taking the role of a specific user and working through the task steps to complete the given scenario. In this phase all inspectors work on their own.
- **Logging Meeting:** in a moderated session, all the findings are presented by asking questions like "What does the user do next?" and "Will the user have any trouble here?" The found defects are recorded.

- **Rework:** all the findings of the logging meeting are used to find solutions to the reported problems in the rework phase. These solutions will also be implemented to improve the interface.
- **Follow-Up:** The collection, assessment and release of the inspection process results are in the focus of the follow-up phase. This completes the inspection.

## **6. Remote Evaluation**

In contrast to traditional laboratory-based usability evaluation, remote evaluation is defined as usability evaluation where users are separated from the evaluators whether in space and/or in time. The term remote always refer to the location of the user where local always means the place of the developers.(Castillo, Hartson & Hix) With this new kind of evaluation it is possible to evaluate an interactive system in a cost effective, fast and efficient manner and it combines empirical and analytic usability evaluation. This kind of evaluation requires a software tool which allow to observe and analyze the behavior of the user as well as for moderating the communication between the usability engineer and the user.(Christos et al., 2007)

### **6.1. Reasons for remote evaluation**

Specially since web applications became more powerful and cover a wide range of features, the classical testing in laboratory became more and more difficult due to the fact, that it is really hard to get a representative sample of the system users to the laboratory. (Hartson et al., 1996) Specially when the users are spread on diverse geographic areas or if the system has been designed for a special group of people which is hard-to-reach and/or decentralized and so hard to schedule for interviews at a single location.(Christos et al., 2007)

Beside the cost factor of transporting such a representative sample to the laboratory another big problem arises when testing remote applications: most of the web applications are available worldwide with numerous network and remote work settings and therefore it is nearly impossible to have direct observations in usability testing. (Hartson et al., 1996) By using remote evaluation it is guaranteed that users will test the product on a more realistic environment since they use their own computer with all the own settings and infrastructure. (Christos et al., 2007)

Also bringing the usability experts to the users at their normal work locations would lead to huge money consuming test setting (Holzinger & Leitner, 2005) because beside the travel costs of the people involved in the project all the usability equipment would have to be transported to every single place. Only a few places could be covered in such an experiment. (Baravalle & Lanfranchi, 2003)

So separating the users from the usability experts in time and/or location enlarges the number of potential test participants dramatically at lower costs since only some software and an internet connection is needed beside all the cost factors which are already present at every usability study. If a world-wide audience is used for a remote evaluation a larger and more diverse pool of participants is accessible and therefore leads to a higher amount of participants and therefore to more convincing results. (Christos et al., 2007)

## **6.2. Kinds of remote evaluation**

### **6.2.1. Separated in place**

When the usability experts are separated only in place a similar test situation as in lab can be achieved by bridging the distance between the lab and the user with modern communication methods. By using software tools which allow sharing the computer screen of the user with that one of the lab the usability experts can follow every step the user do when executing a task as well as measuring efficiency by taking the time for completing the tasks. If a bidirectional communication channel exists, the experts can interact with the user as they would do it in laboratory as well as recording the reactions of the user – for example with a video and/or audio-stream. (Christos et al., 2007)

To perform such tests a lot of software tools have to be installed on every user's computer. This leads into the problem that such software tools have to be as easy to install as possible. Beside the installation there should be high level of transparency of the software so the user understands and accepts the usage. (Christos et al., 2007)

These kind of remote tests are often called synchronous methods since the usability experts and the users are working on the tests at the same time. As there is nearly the same possibility as in standard lab tests, mostly the think-aloud method is used. When the remote users are part of the development team or if they are geographically dispersed members of the usability team, the tests are more likely a usability inspection or a walkthrough to find possible usability issues.(Andreasen et al., 2007)

### **6.2.2. Separated in place and time**

When the test itself and the evaluation of the results are done asynchronous two scenarios of getting feedback about the results exists. In the first case, the own behavior is reported by the

user after completing a task by giving feedback. (Christos et al., 2007) This is mostly done by answering questions which can be sent to the user by mail or by providing special online forms directly after a certain task is completed. When evaluating websites, the questions are mostly asked by displaying a pop up window directly within the browser as an overlay over the original site. (Castillo & Hartson, 2006)

A special method for acquiring high quality usability data is the critical incident method. Whenever such a critical error occur the user immediately records the error mostly by filling in a short problem description form together with recording a screen sequence clip. Both is locally stored and transmitted later to the usability lab where the experts can analyze the problem and give the results to the developers to redesign the part of the software. The quality of the results and the efficiency of gathering it depends the following facts (Castillo & Hartson, 2006):

- Real users work normally on their usual work, no test situation has to be created
- The work is done at their normal working environment
- The problems are reported by the users themselves
- The report is sent only a short period after the problem occurs
- There will be no interaction within the period of testing
- It's cost effective
- The problems are relative easy to translate into usability problem descriptions

The critical incidents method is a very cost and labor efficient method as most of the usability work is done by the users rather than the lab staff. (Andreasen et al., 2007)

In the second case a software tool collects all necessary data on the client side during the usage of the system and transfers the results to the usability lab on a defined point of time. The collected data can be analyzed in either an automatic or manual way. (Christos et al., 2007)

### **6.2.3. Third party services**

Design documents, software samples and/or prototypes may also be inspected by third parties. In this case the usability review itself is done in lab at the contracted company/institution but remote from the developer itself. Therefore it is due to the point of

view ‘third party usability inspection’ is categorized as an in lab evaluation or a remote evaluation. Nevertheless the remote expert group revises the transmitted software or document and gives feedback to the developers. As it is an inspection, no real user is involved but some usability experts.

If real users are involved in the usability evaluation process done by a third party, it is called ‘third party remote usability testing’. Like the previous method it’s local for the remote testers. Especially when a representative group of real users do the evaluation in the usability lab of the third party. (Castillo & Hartson, 2006)

### 6.3. Summarization

In a study of Castillo (Castillo, Hartson & Hix) several aspects according the quality, quantity and the costs for gathering the data are compared.

#### 6.3.1. Comparison of equipment used and quantity of data gathered

Among the discussed methods it is obviously that those methods which are lab based or use other technical equipment than the computer itself require most tools and gather the most output. These methods are all found in the upper right corner of Figure 10. All methods where the user collects the data himself or the data is collected automatically require less equipment but have a wide range of the amount of gathered data.

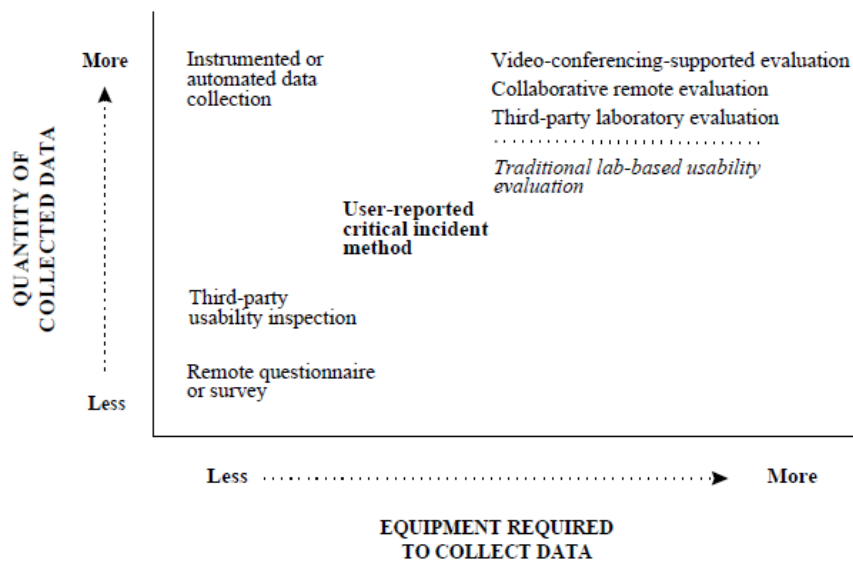


Figure 10 Equipment required to collect data (Castillo, Hartson & Hix)



### 6.3.2. Cost to collect and analyze data

The costs of a usability study can be divided into such afford paid for collecting the data and costs for analyzing the data gathered. (Castillo, Hartson & Hix) These costs include:

- equipment and technology
- time and effort (person-hours)
- external services (third party)
- training (for developers as well as for user)
- travel and time costs for users and developers (if they travel to the users site)

Methods which gather a lot of data are also seen to afford a lot of costs in analyzing the data. Most expensive ones are those which are similar to traditional lab based evaluation methods like all the video conferencing, collaborative remote evaluation and Third-party laboratory evaluation. This is also due to high equipment costs. (Castillo, Hartson & Hix)

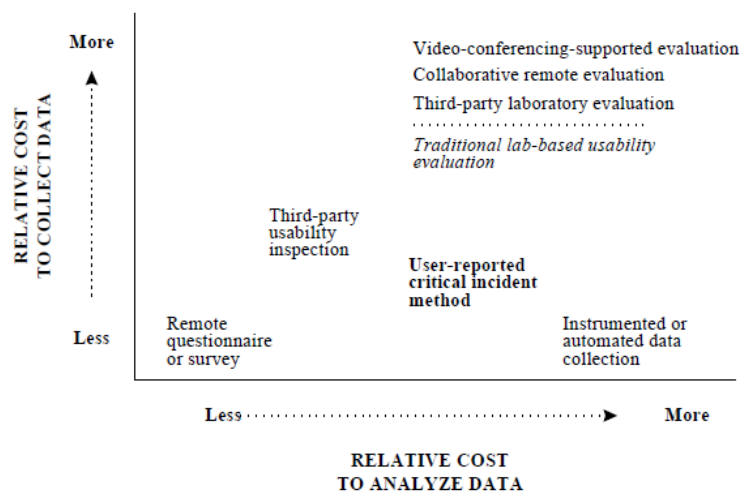


Figure 11 Costs to collect data and costs to analyze them (Castillo, Hartson & Hix)

Figure 12 shows the efficiency of each of the method by comparing the quality of the data with the total costs of gathering the data. A higher slope suggests higher efficiency (Castillo, Hartson & Hix). Depending on the budget and the level of quality needed a method can be chosen by this graphic. The most efficient usability evaluation is based on the remote questionnaires and surveys because of the low costs. The negative aspect of such questionnaires is to deliver only subjective feedback which is always hard to interpret. It also shows that high quality data efforts also most costs. The 'critical incident method' seems to have the best efficiency at relative low costs and a proper quality of the results.

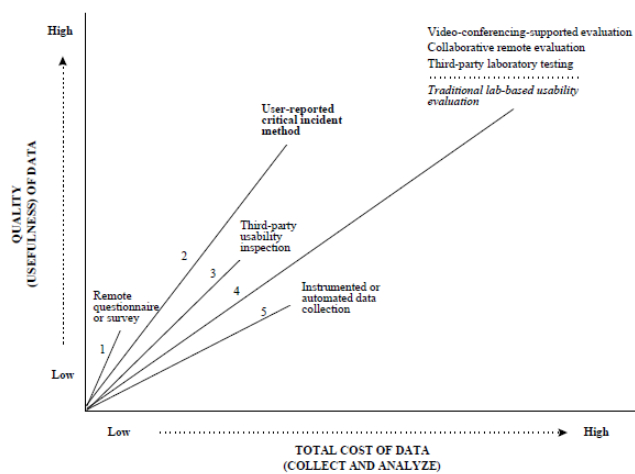


Figure 12 The relation of quality and costs (Castillo, Hartson & Hix)

## 7. Existing Tools

There are a lot of tools on the market but most of them are commercial ones with fees for the tests and/or inspections. Only some provide a trial, so most of the findings in this chapter are due to product description directly from the webpage of the providers.

### 7.1. Moderated tools

The service **UserVue** of TechSmith provides a one-to-one usability test tool where the whole test session can be managed. It provides a tool for inviting users and observers for joining a test session. There haven't to be any additional observers but there can be multiple. The tool (has to be downloaded) captures the screen of the user and provides a video afterwards which will be synchronized with all the verbal comments. The communication between the user and the facilitator is done over phone or internet chat.

**LiveLook** is a simple screen sharing tool with the advantage that no additional software has to be downloaded with the exception that Java is installed. As it is only a screen sharing tool, no audio or video recording (including the screen itself) is available.

**GoToMeeting** is a web conferencing system where screen sharing is the primary function of the tool. With only a small Java application has to be downloaded it is easy to start with. The voices of the participants are transmitted via VoIP (voice over ip).

**Skype** is primary a messenger service but due to the new features of video and audio conferencing it may be used to have user test sessions with multiple observers. Especially for international test series it is useful due to low international rates.

**OpenHallway** is a quite simple screen recording tool. With only a small Java application to be installed, it is very easy to join for users. After a registration, test scenarios can be created and a link is provided to be sent to the test users. As tasks for the users can be created, a short feedback of the users is also possible by filling in forms. The test session is only recorded so no live viewing is possible.

### 7.2. Unmoderated Tools

**Loop11** measures task times and task completion rates on evaluation sessions. The user is given certain tasks (which can be defined by the operator) which are stated at the top of the browser window. When the user completes a task it is told to the system by clicking a button, the same when the user fails. The screen behavior like mouse movements and clicks is not recorded.

**Clickatale** primary collects the user behavior by recording the mouse and keyboard interactions. There is no direct screen capturing but the system will generate videos of the user session by combining the mouse and keyboard data with pre captures of the screens. So the recording can take place without any software installation. It's a smart solution since only a few bytes of data (mouse and keyboard inputs along with time codes) have to be stored instead of whole videos. The online tool also reports about areas where the users pay attention to and provides heat maps of clicks and reports about form input behavior of the users (for example if a field of the form requires more time to be filled in than others).

**Userzooms** target group are usability experts who want to have a framework for their online researches. Different services may be done as for example prompt users to do online tasks, card sorts or surveys. It also includes a tool to invite user to join the test.

## **8. Possible commercialization**

### **8.1. Classification**

#### **8.1.1. Three classification options**

According to Mahadevan (Mahadevan, 2000) business on the internet can be categorized into three groups depending on the provided service and the way to communicate with the user.

In his definition business internet portals are focusing on building up communities to bundle potential customers of a service or product in one website. The members of the community may be influenced by increasing the image of the product or service. The goal is that money is being spent on those products later on time at other web pages or real offline stores. It is seen as a marketing instrument for the own brand or to generate revenue from displaying ads or being an affiliate of other companies.

Market Makers are the second group described by Mahadevan (Mahadevan, 2000) and are similar to the Portals except that they connect customers and service or product providers together. A very well known example of such an Market Maker is EBay (Ebay, 2010) where the platform provides the way how the business transaction is facilitated. Like portals, its power also arises by the size of the community.

Dealing directly with the customers is the main characteristic of the third group called product/service providers. Business transactions are facilitated without any third party (with exception of some services like payments). A typical example of a member of this group is Amazon (Amazon, 2010).

#### **8.1.2. Classification of the developed software**

All three business structures mentioned above would be possible for the software developed within this diploma thesis even the portal option is the least interesting because of the small number of involved people. It would be possible to divide the product into a free and a professional version where the aim of the free version would be to convince possible buyers of a professional license about the quality and features of the product. It is questionable if displaying advertisement on the screens of the test users would influence the test results and if this would be accepted by the test conductors.

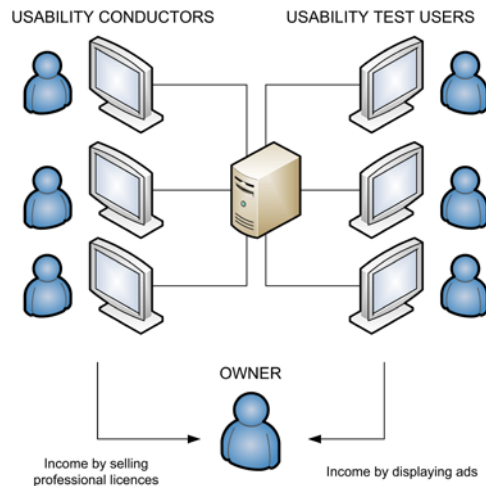


Figure 13 Acting like a portal

The second option acts as a market maker and connects usability professionals and website owners together. The platform would be a kind of a tool used by the professionals to do their business on the one hand and service for people who want to improve the quality of their websites. Especially here the main task in setting up this business would be to contact usability professionals on the one hand and build up a brand to be THE place for usability improvements of websites. Every usability professional would be able to implement own usability test methods to diverse from other conductors within the system. As having all competitors of the usability professionals using the same system may lead into problems, also this option is not chosen.

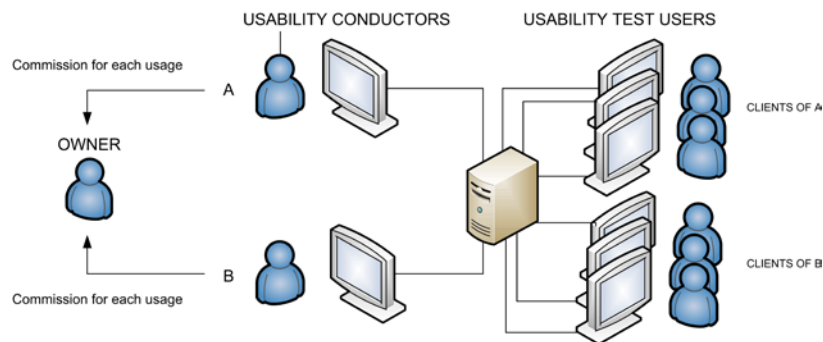


Figure 14 Acting like a Market Maker

The third option is to act as a product/service provider where this option may be split into a pure product provider and a pure service provider. As a service provider, the developed platform would be used only by the operating company and this company would offer usability

tests as a service. This would also mean to become more an usability professional rather than a software developing company which is not the intent of the writer of this thesis.

The last option is to act as a product provider and sell licenses of the software to usability test conducting companies. As this is the preferred way such licensing models are discussed in section 8.6.

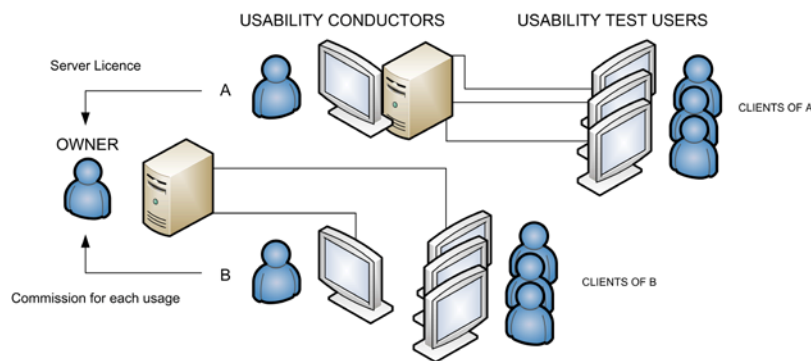


Figure 15 Lisencing models

## 8.2. Electronic Payment Systems

For a system where online services are bought and are intent to be used immediately an online payment system is required which allows to immediate approve the success of the payment and reduce the risk of a non-payment (Wright, 2002). In this case the buyer can immediately use the system and the seller can be sure that the payment process was successful. Most of the discussed electronic payments methods derive from popular offline payment methods like the credit card processing or the electronic check. The virtual money approach on the other hand are new systems with an big potential impact on e-business (Sumanjeet, 2009).

A lot of different categorizations exists dividing the online payment options into groups due to the way of payment, the technologies used or the amount. Nevertheless, they can be broadly divided into four general types (Sumanjeet, 2009) which will be discussed in the following sub sections.

### 8.2.1. Online Credit Card Payment Systems

When using a credit card based payment system, the merchant has to have an merchant account with one of the credit card companies or at least with one of so called credit card clearing houses. This companies are basically responsible for authorizing the credit card

payment by contacting the bank where the card was issued electronically and inform the merchant about the status immediately (Wright, 2002).

There are three possible ways how the confidential information – like the credit card number – is transferred over the internet. First of all the data can be entered in an online form on the merchant's website. The collected data is then sent to the e-payment service provider. Two security problems arise here for the customer: the credit card number is available for the merchant and the way how the data is transferred to the e-payment service provider may be not a secure connection. The second method is to forward the customer to the payment processors website where the data is keyed in. Afterwards the merchants system is informed about the status of the payment. Here the data is stored and/or processed by the service provider only and is not accessible to any third party.

The third option is to use a Secure Electronic Transaction (SET, an international standard) where all transferred data is digitally signed and encrypted by the customer with a bank's public key. So no one between the customer and the bank can read the transferred data because only the bank can decrypt the information by using their private key. The merchant is informed by the bank system about the successful/unsuccessful payment status immediately (Tygar, 1998).

Regardless of the option chosen the process for the customer is very easy since only the credit card details have to be entered in a form while all other process is handled in background (Sumanjeet, 2009). In contrast other payment systems sometimes require pin codes, special procedures or other information which vary on every payment.

### **8.2.2. Electronic Check Systems**

Like at the credit card processing system three parties are involved in the electronic check system process: the merchant, the customer and a clearing company (mostly called Automated Clearing House, short ACH). In contrast to credit card processing, the customer has to provide other data to the ACH (Wright, 2002).

The e-check issued by the buyer contains

- the name of the financial institution
- the payer's account number



- the name of the payee
- amount of the check (Sumanjeet, 2009).

Beside this payment information an equivalent of the written signature has to be provided to authenticate the check from the owner of the account. This is mostly a computed number (Sumanjeet, 2009).

Although the bank guarantees that the funds have been reserved for payments, the clearing process takes some days (Wright, 2002).

### **8.2.3. Smart Card based Systems**

Smart cards are basically credit card sized plastic cards with an embedded micro chip. This microchip may only consist some storage possibilities but also microprocessors for encryption and other inbuilt transaction processing capabilities which enable interacting with online merchant sites and authorizing payments without releasing confidential information on the internet (Chakrabarti & Kardile, 2002).

The disadvantage of such smart cards is that a card reader is needed and therefore the mobility of the method is not that high until electronic devices like laptops or mobile phones are able to read such cards. On the other hand the microchip adds a hardware protection level to the payment process rather than only software encryption protection (Chakrabarti & Kardile, 2002).

### **8.2.4. Digital or Electronic Cash Systems**

Digital cash systems are characterized by the fact that no real money is transferred between the customer and the merchant. Instead, digital cash is sent which was bought by the customer previously at the digital cash system company.

Centralized digital cash systems like PayPal (PayPal, 2010a) require that both of the involved parties (merchant and customer) have established accounts at their companies. When more companies (in most cases banks) are accepting the digital cash, the money can be transferred directly to an account at another bank (Wright, 2002). While centralized digital cash systems have the advantage that the electronic money never leaves the own system, special security concerns arise when the electronic money has to be sent over a network. Therefore such e-

money is normally digitally signed to prevent forgery and criminal use. Beside the signature, a digital watermark should prevent copying the money (Sumanjeet, 2009).

To cash out the received money the companies offer different ways. PayPal (PayPal, 2010a) for example offers the possibility to send the money to a bank account, so the conversion back from virtual money to real money can be done. Other such companies like AlertPay (AlertPay, 2010) offer more options like paper checks or sending money to credit cards.

In difference to other methods where actual funds in legal tender money are transferred for every transaction made, electronic money is issued by a company. As such electronic money is always linked to real money it is up to the issuing company if the value of the electronic money is shown in a virtual currency or in a real currency (Chakrabarti & Kardile, 2002).

### **8.3. Project specific requirements for an payment processor**

Regardless of the business model chosen a payment service provider should be chosen which fulfills at least the following requirements:

- **Credit-Card acceptance:** as more than 631 million valid credit cards are in circulation only in the United States (creditcards.com, 2010) the acceptance of those cards is essential
- **Multi-Currency:** as the platform should be available in all countries, the payment processor should support a multi currency support so the amount to pay should be displayed to the customer in the local currency or if that currency is not available at least in Euro.
- **Recurring payments:** especially for the subscription option a recurring payment option should be provided by the payment processor. The amount, the period (day, week, month and year) and the frequency should be selectable for each subscription. For example if the user should be charged €2 every two weeks the parameters would be: amount = 2, period=weeks, frequency = 2;
- **Withdrawal possibilities:** a withdrawal should be possible to a European bank account, regardless of the country where the company is based.

- **API:** the payment processor should provide a Application Programming Interface (API) which allows to perform most of the administrative tasks automatically (payment notification, make payments, set up subscriptions and other)
- **Micropayments possible:** to be flexible in pricing micropayments should be possible with an acceptable transaction fee.

If the payment processor requires a registration by the users (test conductors), the most accepted and spread payment processor should be taken to ensure that users feel only a low barrier to enter payment details. Even a unknown payment processor is cheaper, the lower barrier is even more important.

#### **8.4. PayPal as a requirements fulfilling company**

PayPal is considered to be the most important person-to-person payment system which made major payment innovations especially for small businesses (Sullivan, 2007). It was founded in 1998 by an US company located in San Jose (California) and acquired by EBay in late 2002 (Avaliani, 2004).

The benefits of using a wide spread and accepted payment systems can be summarized as followed (Williams, 2007):

- **Credibility:** as giving sensitive data like a credit card number to another person or company always affords trusting the third party, a trustworthy partner who lent the credibility to the small business is a big advantage to that company.
- **Security:** such a big company like PayPal is always a target for criminal attacks like unauthorized access to user accounts by a spoofing email tactic. The size of the company enables them to fight against such attacks and therefore to profile the company to a high security concerning business which satisfy the merchants need to have a reliable and secure partner when dealing with turnovers.
- **The checkout-experience:** as PayPal offers a very simple check out process in the standard payment flow which is well know by the existing customers of the payment company no proprietary solution has to be implemented. Such solutions are often painful to learn and – especially when usernames and passwords are needed for every solution – the acceptance may decrease. Besides that, existing PayPal clients skip the

“wallet grab” – the physical grab to the purse to get the credit card – which can prevent customers to finish the purchasing process.

#### **8.4.1. Credit card acceptance**

At the moment PayPal accepts all major credit cards like Visa, MasterCard, Discovery and American Express and those cards can be processed in different ways. First of all, using a so called “Virtual Terminal” enables small business to enter credit card details of their costumers by hand in a special form on the PayPal’s website. It’s an interesting option for businesses which sell over the telephone because their clients don’t have to have internet.

The second option is to use the direct pay method discussed in section 8.5.3 where PayPal acts more like a credit card clearing company rather than a e-payment service provider since PayPal is only involved in the authorization process and is not visible to the client.

And as a third option, the service can be used in its origin way, where the customer opens an account with PayPal and adds funds to it which can be used to pay for goods or services later in time.

#### **8.4.2. Multinational usage**

For a worldwide service it is especially important to provide a payment solution which is usable in many countries. This includes in providing payment related information in the language of the user as well as offering the possibility to charge in the local currency because it greatly increases the convenience of doing international business. With offering the possibility to accept payments in 23 different currencies and the support of many languages PayPal fits the requirements of such an worldwide service (Williams, 2007) .

On the other hand the merchant must be able to withdraw the earned money to a real existing bank account or to have other ways to get his money cashed out. As PayPal offers a direct bank withdrawal to local bank accounts in all major industrialized countries, also that requirement is fulfilled.

#### **8.4.3. Recurring payments**

PayPal offers the possibility to have recurring payments with an maximum of € 8.000,- per subscription payment (Williams, 2007). Like at standard one-time payments the platform

offers a button generator where all parameters can be set within a graphical interface. A corresponding HTML form code is also available, which makes the integration easy.

An interesting feature of the recurring payment method in PayPal is the “Subscription Password Management” where websites with a member only area can use PayPal to manage the subscriptions and auto generates usernames and passwords for the login of the merchant’s website. Like at the ‘Website payment standard’ (see 8.5.1) the PayPal server informs the merchant’s server about the payment and includes a randomly generated username/password pair. This username/password pair can be stored at the merchant’s web server to enable a user login.

#### 8.4.4. Application Programmable Interface

The first API was released by PayPal in the year 2004 (Williams, 2007) and enables a developer to access a wide range of the PayPal functionality from within own code / software. As in every API lifecycle (see Figure 16) the request is generated by the software of the API using platform and sent to the PayPal server. According to the function called and the parameters sent the server responds with the corresponding answer or an error by providing error codes. The response is used to perform further actions at the web shop’s website.

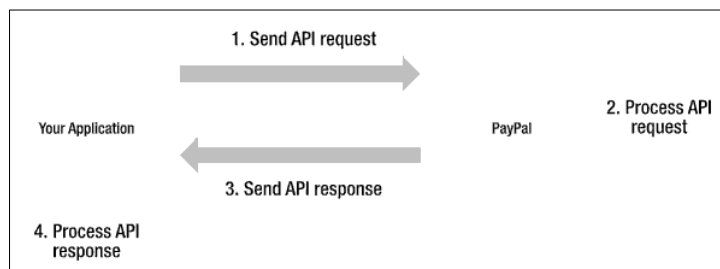


Figure 16: Lifecycle of an PayPal API call (Williams, 2007)

Beside the synchronous requests PayPal supports an asynchronous request in the payment process. The so called ‘Instant payment notification’ (IPN) will call a previous defined URL at the merchants site to inform the system about the status of a payment (see 8.5.1).

#### 8.4.5. Micropayments

To be adequate for processing micropayments, the payment processor has to fulfill two requirements. First of all the processing company must accept payments within that amount region and second, the fees for the micropayment have to be in an acceptable range.

Otherwise more fees than income would be generated. PayPal (PayPal, 2010b) offers special micropayment accounts where a fee of 5% of the payment amount plus USD 0.05 static fee is available. At a one dollar purchase amount basis this would effectively lead to 10% fees.

## 8.5. PayPal integration options

Depending how much PayPal is (visibility) involved in the checkout process at the merchants website, a division in 3 integration options can be made.

### 8.5.1. Website Payments Standard

This is the easiest way to integrate a payment solution to the merchant's website. PayPal offers a way where no programming skills are necessary to receive money from customers. A button generator can be used to create a 'buy now' Button, which can be insert onto a webpage by pasting the html code to the website.

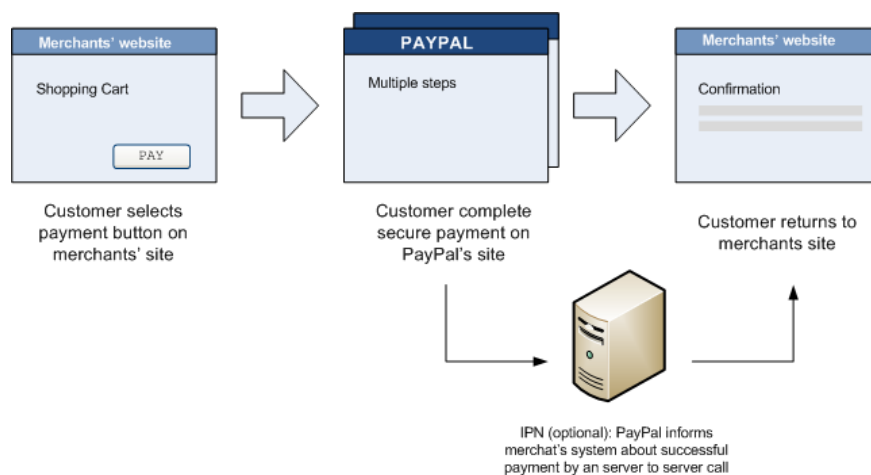


Figure 17: Website Payments Standard flow

With this option the entire payment process is done on the website of PayPal and the merchant is informed by an email about the success of the payment (see Figure 17). Optional a specified URL can be called by PayPal to inform the merchant's server about the state of the payment. This method is called 'Instant payment notification' (IPN) and is used to automate such payments where an immediate delivery (downloads, access ...) is necessary.

### 8.5.2. Express check out

Express check out requires API programming skills but enables the customer to finish the payment process on the merchant’s website (see Figure 18). At the beginning of the checkout process PayPal is only used to authorize a possible payment and to check the shipping address (the customers address stored at PayPal) so the user has not to type in this information again on the merchants website. It also enables to review the order once again on the online store before the payment process is started.

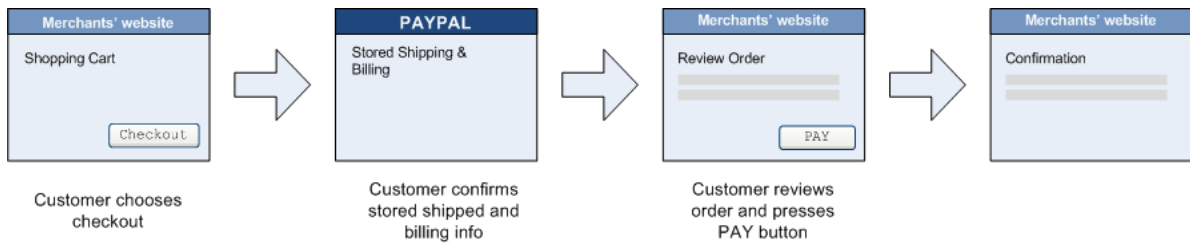


Figure 18: Express Checkout flow

### 8.5.3. Direct payment

Direct payment is the option when PayPal should be invisible to the customer. All data required to perform the credit card payment is sent to the PayPal server and the transaction is authorized in real-time.

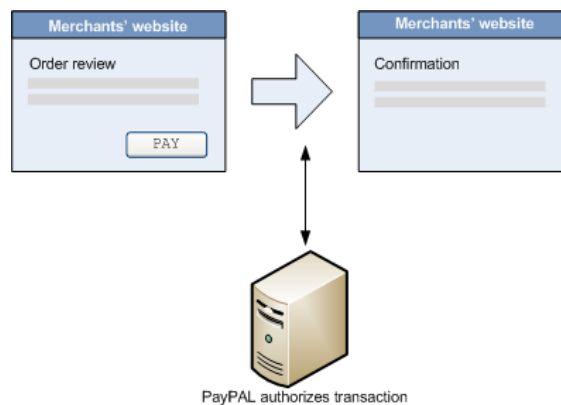


Figure 19: Direct Payment flow

For security reasons, all data is transferred over a Secure Socket Layer (SSL) and for avoiding fraudulent payments, the IP of the customer’s browser has to be also transferred to PayPal when making an authorization request.

### 8.6. Licensing Models

Regardless which licensing model is chosen when entering the market, the goal of each option should be to generate revenue in a direct or indirect way. Market strategies where revenue is obtained directly from the end-customer are covered by the subsections 8.6.3 and **Error! Reference source not found.** In contrast to those three models, another concept where the usage of the system itself is free of charge and revenues are generated in an indirect way is discussed in 8.6.2.

### 8.6.1. Approximately costs

To evaluate the direct revenue options an approximately cost overview about such an project should give a rough estimate about the amount needed to have enough turnover to cover the costs of life in an one man company. As all figures are simplified, it is assumed that the company is located in Austria and is a limited one (GmbH) and only one person is developing and selling the software or the service.

Such a calculation should illustrate if realistic prices can be achieved at the market even when low costs are considered. When the prices would be too high at this low-cost basis, the product is not sellable when higher costs are calculated too.

Gross turnover	6.666,-
Tax (20%)	1.111,-
Net turnover	5.555,-
Costs *)	2.000,-
Profit	3.555,-
Taxes **) – 43,75%	1.555,-
Personal profit	2.000,-




Table 1: simplified calculation of monthly turnover needed

So to achieve a personal profit of 2.000,- Euro for the one person in the company, a total gross turnover of about 6.600,- Euro has to be generated. In this calculation the taxes marked with \*\*) are calculated first by using the Austrian tax rate for the ‘Körperschaftssteuer’ (Wirtschaftskammer, 2006) of 25% and calculating the amount for the ‘Kapitalertragssteuer’ (25%) from the previous result. The costs consists of the rental fee for the office, costs for power supply, the server needed and other accumulated costs. The figures are taken from the Grizzly Werbeagentur GmbH where the author of this thesis is one of the CEOs.



### 8.6.2. Free usage

In the style of open source software models, this option is based on using the power of the open source community to improve the platform itself. Moreover, the users of the platform – in this case the usability conductors and/or their software developers – could be engaged to develop new testing methods which could be bound to special licenses. These licenses define the group of people who are allowed to use the ‘third party’ modules:

- **Personal usage only:** definitely the worst option, because this would not generate any benefit of new developed modules for neither the owner of the platform nor the other users.
- **Transfer of ownership:** by developing and uploading a module the developers agree that the platform owner receives the right of a non-restrictive usage (including modification, selling). This option has one big disadvantage: the correctness of the testing method cannot be assured except by time consuming tests of the platform owner. This would afford a well document and well written code which is also hard to guarantee.
- **Developing for the community:** all the developed modules are available for all other members of the community. This would have a nice side effect that the platform owner could take advantage of the other usability test conductors as testers and evaluators. This step would also afford a feedback system to report bugs or the make feature requests or to simply discuss the efficiency or the even the correctness of a given test. Bad designed or usability tests which generate wrong results are dangerous for the whole platform as the ‘quality-providing-company’ image could be damaged.

This model would allow to generate benefits in three different ways (Hars & Ou, 2001):

- **Revenues from related products and services (financial capital):** this would actually generate real revenues on a monetary basis by selling related products or services. In the case of a usability testing software especially commercial consulting, training, support and implementation services could be offered.

- **Human capital:** by having a great brain pool of developers, the skills, capabilities and knowledge of the own staff can be increased in the field of usability testing and in software development.
- **Self marketing:** as all parts of the platform can be branded with the own trademark, the owner of the platform will be associated with usability testing by all usability conductors using the service. Some side effects as being mentioned in scientific papers or being linked – and so improving the position in search results of search engines – can also lead to a good company profile.
- **Structural capital:** the already mentioned great brain pool could also generate revenues in a completely different way: methods could be developed and be patented so the position of the own company can be tightened even more in the marked. But also the other parts of the structural capital (see Figure 20) benefit from a structure similar to open source software. Especially to know current and potential customers – who maybe also have need for other related commercial products or services – is important to a successful market strategy.

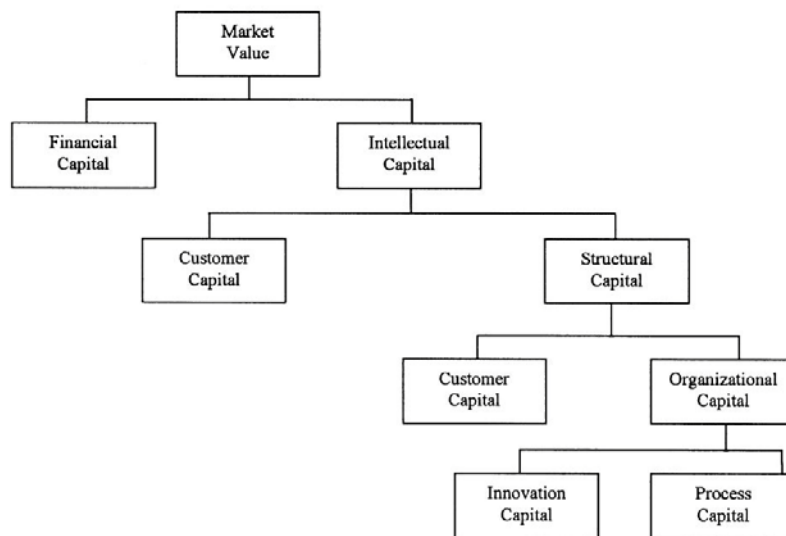


Figure 20: Skandia value scheme (Bontis, 2001)

### 8.6.3. Usage fee

- **Pay per use**

A usage fee can be connected to a period of time, a number of usages or the number of test users depending to what is more likely to be bought.

- **Period of time:** Due to the fact that conducting a test is a very time consuming task and it is not predictable when the test users will participate in the test, limiting to – let's say a week – could lead to an interruption of the test. This can be avoided by locking only the test in the conductors interface and not for the test users. From the economic point of view this would generate a benefit due to the fact that the conductors had to prolong their period of usage to view the test results.
- **Number of usages:** in this specific project, the number of usages is equal with the number of tests a usability conductor starts. From a system performance point of view it is not predictable how many test users will participate in a specific test and therefore how many system resources are used. The usage of resources should always be reflected in the price.
- **Number or test participants:** in comparison to the number of the test conductors, the number of participants is relatively high. They use the system resources in different ways: first, the storage of thousands of potential test users requires disk space for the database entries. Secondly the invitations sent to the users require also performance and bandwidth. And third, the tests itself are also resources consuming.

- **Subscription**

To bind the conductors to the platform for a longer time, subscriptions can also be offered to the end consumer. Some studies (Fishburn & Odlyzko, 1999) also showed, that people tend to prefer a subscription based purchase rather than buying on a per hit/usage basis. This seems to be valid even the subscription price is higher than that of the per usage price arrangement.

- **Mixture**

Of course both above mentioned options can be offered at the same time. People who want to try the service or only need a onetime test can choose the per usage payment option where other people who are using the platform more frequently can use the per subscription basis to avoid having to pay multiple times or be more flexible when – for example – more participants are invited than previously planned and paid for.

#### **8.6.4. License**

The economic benefit of using a license system where the whole platform is sold is to benefit from updates and support offered to the customers in future. The customer would be able to set up his own business which is totally branded with an own name and visually not connected to other usability experts (like it would be if all uses the same platform).

As PHP scripts are not being compiled (and therefore unreadable for humans) the disadvantage of giving the source code to a third party would be to host the system on own servers. The customer would be able to connect his own URL to his copy of the platform. So both – protecting the code and selling a whole platform – would be possible.

## **9. Software Requirements**

### **9.1. Introduction**

#### **9.1.1. Purpose**

This Software Requirements Specifications covers the requirements for an remote usability test tool which is completely new developed. As the system consists of a basic management tool for administer test sessions and separate modules for the individual test methods this document will specify beside the requirements of the basic management tool the interface specifications for the modules. The overall revision number of the project covered by this SRS is 0.0.1.

#### **9.1.2. Intended Audience and Reading Suggestions**

The main audience for reading these software requirements is software developers who will extend the prototype developed along with this diploma thesis. Secondly it should give an overview for potential investors about the principle structure and purpose of an online remote usability testing tools.

#### **9.1.3. Project Scope**

The goal of this diploma thesis is to implement a software tool which allows everybody to conduct usability tests with remote user in an easy way. The connectional design chosen offers two possibilities for the usability conductors. First of all, a predefined set of tests should be accessible so no programming skills are required to conduct such a test. On the other hand the platform should grand access to implement new usability tests by writing modules. In the style of a CMS (Content Management System) a mixture between simple usage and extensibility should open a new way to invent, implement and conduct remote usability tests.

#### **9.1.4. References**

Beside the information in this Software Requirement Specifications it is recommended to read also all other sections of the diploma thesis to get an overview about the necessary background of usability and possible commercial usage. Both are necessary to understand the

design of the software. Other references will be marked and can be found in the references section of this diploma thesis.

## **9.2. Overall Description**

### **9.2.1. Product Perspective**

The central element of the developed software is the “remote usability lab” platform where all user groups start and execute their specific tasks. For those user groups which have to register and login to the platform to use it a database keeps the login information and the access rights for the different areas. As this software should be used in a commercial context in future, the database also contains information about available commercial packages which can be bought by usability conductors as well as information about successful purchases. Separated from the first database, a second and independent database for storing the results exists.

Figure 21 shows the main user groups and their access to different areas of the platform. The developer area is only accessible by the system administrator in this early version because new modules are due to be uploaded manually. In a further version an upload area should provide the possibility to upload modules which are verified against specifications and available for the usability conductor for which the module was developed only.

The test users will never get in touch with the main platform itself as they only access the test area where the actual test is performed. So the system has to provide a way to participate in tests without having to register for it.

The group of usability conductors has access to the backend of the platform where all settings for a usability test can be made. This area also includes a payment option for buying credits for commercial tests.

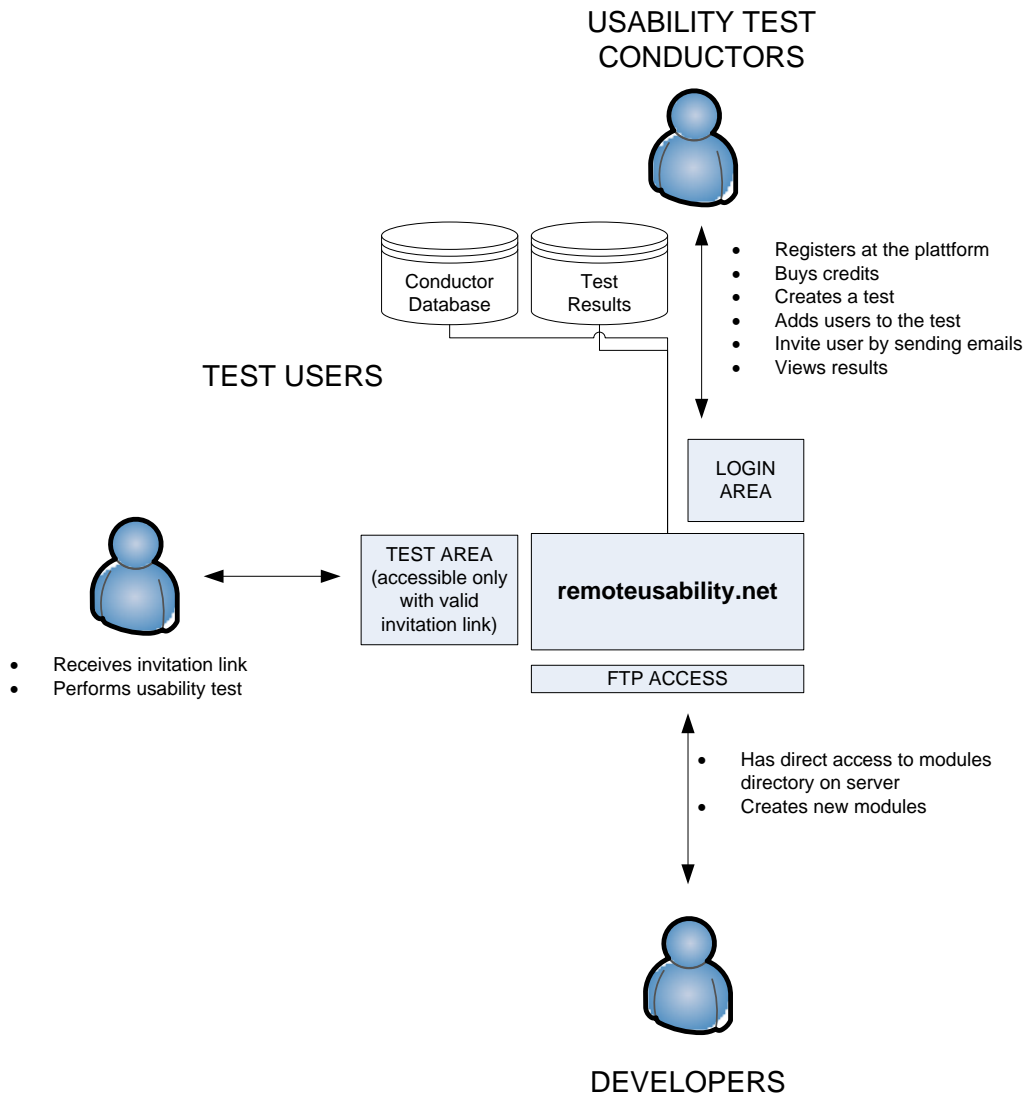


Figure 21 System overview

### 9.2.2. Product Features

As the platform will be used by three different user groups, the product features are also divided into those groups. As this version of the software does not allow uploading new test modules within a user account, the developer group functions are not mentioned here.

- **Usability test conductors**

- **Register at the platform**

A usability test conductor can register on the platform by giving detailed information about the company the tests are conducted for. As it is possible to

send email invitation to other people the registration process needs at least a email verification to avoid fake registrations.

- **Buying credits**

Predefined packages can be bought by the user by selecting the desired package. The user will be redirected to PayPal to complete his purchase. The system will receive the status of the payment by an 'instant payment notification' through PayPal and unlocks the system so the usability test conductor can generate new test sessions.

- **Test creation**

The available test modules will be shown to the user where he can select one of them as part of his usability test. After the type of the test is chosen, test specific parameters can be set by clicking a corresponding symbol.

- **Generate users**

The system will allow the usability conductor to add email addresses of potential test users to his account by copying them into a text field. The system will check against duplicate entries.

- **Add user to test**

The email addresses previously entered can be assigned to one or more tests by clicking a corresponding button at the tests overview screen. All of the entered email addresses will be added.

- **Delete users**

Previously entered user email addresses can be deleted by entering the email address in a search field. As the email address can only be stored one time, no result list will be presented. Instead a warning message (informing about the deletion process) appears which has to be confirmed.



- **Email invitation**

By clicking the 'invite now' button, all potential test users will be informed about the test. They will receive a unique link for joining the test. Beside clicking on this link and performing the test, the test user does not need to perform any other action.
- **View results**

The results can be viewed by clicking on the corresponding button at the running tests overview. How the results are presented is up to the implementation of the corresponding test module and not limited or specified by the main system.
- **Test users**
  - **Participating in an online test without being invited**

To promote a remote usability test without knowing the email addresses of potential test users, the usability conductor can post a link to the test to any webpage. If a visitor of that page decides to take part in the test, short email verification takes place.
  - **Accepting the invitation link**

As the link contains a unique id to identify the user and the test, no special input or action by the user is needed. So a smooth and uncomplicated test joining mechanism should be guaranteed.
  - **Declining the invitation link**

The user can also decline the invitation by clicking a corresponding second link in the mail or by ignoring the email itself. The link should provide feedback information for the usability conductor.
  - **Request for user list removal**

If a user does not want to receive further invitations, clicking on the third link in the email will remove the email address from the address list. This will be done system wide due to a restrictive non spam policy.
  - **Performing the test**

The user can perform the test according to the test implementation of the assigned test module.

### 9.2.3. User Classes and Characteristics

- **System administrators:** the system administrators are the representatives of the owner of the system. As a licensing model exists where the software is sold and installed on a third party server, more people than the system administrator of the origin platform may exist. This group of people has to have basic skills of installing such systems on the target platform. Special programming skills are not mandatory but they should be familiar with simple database administration.
- **Usability conductors:** the system is build to provide a set of test methods to usability conductors which have no programming skills at all. This group only has to manage the test by using the user interface in the login area.
- **Test users:** the skills and the computer experience of the test users may vary a lot because they are invited by the test conductors. Therefore the process of joining a test has no barriers like subscription.
- **Developers:** in the first version this group may be the same as the system administrators. However, each of the system administrators may grant access to the modules area to a group of developers. Programming skills are mandatory.

As the usability conductors may have less or none experience in using an online application, the focus of usability lies directly on this group.

### 9.2.4. Operating Environment

The remote usability web application will run on all major operating system where an Apache Web Server and a MySQL Server can be installed. For the programming logic PHP 5.0+ is required with an enabled mail() functionality.

On the client side, all major web browsers which support JavaScript, Cookies and XMLHttpRequest object. This include

- Mozilla Firefox 1.0 and above
- Netscape version 7.1 and above
- Apple Safari 1.2 and above
- Microsoft Internet Explorer 5 and above
- Konqueror
- Opera 7.6 and above

#### 9.2.5. Design and Implementation Constrains

- **Limitations to the Browser:** Both, the usage of the web application itself and participating in a usability test requires to have JavaScript and Cookies enabled. Furthermore the XMLHttpRequest object must be supported by the browsers of all users.
- **Limitation of the server:** as it is a single server architecture with no load balancing possibilities the usage of the system is limited by the available memory of the used web server.
- **Unmoderated usage of the mail() function:** as the amount of email addresses which can be added to a test is not limited and the system cannot check if the recipients want to receive the mails, a probably spam issue could arise.
- **Module validation:** because of the open architecture of the module implementation interface, implemented modules should be validated before they are used. Especially when the development is done by a third party.

#### 9.2.6. User Documentation

As the platform developed within this diploma thesis is thought as a prototype for a further, more powerful version, no special documentation is shipped along with the software except this thesis.

#### 9.2.7. Assumptions and Dependencies

The graphical user interface for the usability conductors will be build to imitate a desktop look and feel. As the development of such an Ajax based virtual desktop framework would go

beyond the scope of this thesis, the existing MochaUI (web applications user interface library) is used. The current version is published under the MIT License which allows also the commercial usage.

MochaUI itself is build upon the MooTools javascript framework, also released under the Open Source MIT license.

We assume that this license model will be kept on both releases.

### 9.3. System Features

#### 9.3.1. Register at the platform

Usability conductors have to register at the platform because payments, test-users and test-results have to be assigned to a user account. The necessary data is provided by filling in a registration form.

Priority	Trigger	Precondition	Actors
Essential	Selecting 'Open Account' on homepage	Guest is on the homepage	Guests

#### Main path:

1. The user selects 'Open Account' on the homepage
2. The user fills in the registration form
3. The user hits 'Send'-Button
4. The system confirms the registration through a message
5. The system sends an email to the client containing an approval link
6. The user clicks on the approval link

#### Exceptional path:

- 4a The system informs the user about erroneous input
- 4a1 The user is returned to step 2

**Post condition:** the user is registered; the user is on the homepage (virtual desktop without any opened windows)

### 9.3.2. Buying credits

Priority	Trigger	Precondition	Actors
Essential	Selecting 'Buy credits' on homepage OR trying to create a test session	User is logged in, no packages are bought	Registered users

#### Main path:

1. User clicks on 'Buy credits'
2. User selects the package he want to buy from within a package list (radio buttons)
3. User clicks on 'Buy now'
4. User is redirected to PayPal
5. User completes PayPal payment
6. User returns to homepage
7. System shows success message

#### Alternative path:

- 1a The user tries to create a new test by clicking on 'Create new Test'
- 1b The user is presented the same screen as in 2

#### Exceptional path:

- 5a User cancels payment or payment is unsuccessful
- 5b User returns to homepage
- 5c System shows failure message

**Post condition:** the credits are stored in database; the user is on the homepage (virtual desktop without any opened windows)

### 9.3.3. Test creation

Priority	Trigger	Precondition	Actors
Essential	Selecting 'Create new Test' on homepage	User is logged in, packages are bought	Registered users

**Main path:**

1. User clicks 'Create new Test' on the homepage
2. The system opens a virtual window with a list of all available tests
3. The user selects one of the test by checking one of the radio buttons
4. The users completes the test selection by hitting the 'OK' button
5. The system displays the test overview with all created tests in the previously opened window

**Post condition:** The test is stored in the database; the user is on the main screen with the opened test overview window.

**9.3.4. Generate users**

Priority	Trigger	Precondition	Actors
Essential	Selecting 'Add new test users' on homepage	User is logged in	Registered users

**Main path:**

1. The user selects 'Add new test users' on the main screen
2. The system opens a new virtual window on the main screen with two text input fields
3. The user writes a name for that group into the first text input field
4. The user writes the email addresses into that text input field, separated by a white space (or pastes previously copied email addresses into that field)
5. The user clicks on the 'ADD' button when finished

**Exceptional path:**

- 5.1a The user inserts no email addresses or string with an invalid email address format.
- 5.1b The system alerts the error by a window with a message ('no email addresses entered' or a message stating which addresses could not be stored.)
- 5.1c The system closes the virtual window
- 5.2a The user has entered a group name that already exists
- 5.2b The system alerts the error by a window with a message ('name already in use, please choose another one')

**Post condition:** The email addresses are stored in the database; the user is on the main screen

### 9.3.5. Add user to test

Priority	Trigger	Precondition	Actors
Essential	Selecting 'add test users' on the test overview window	User is logged in and the test overview window is opened	Registered users

#### Main path:

1. The user selects 'add test users' on the test overview window
2. The system shows all user groups previously entered
3. The user selects those user groups which should take part in the test by checking the corresponding check box
4. The user hits the 'add' button
5. The system closes the selection window and displays the number of participating users in the corresponding row in the test overview window

#### Exceptional path:

- 4a the user hits the 'add' button without having selected a email group
- 4b the system displays an error message ('please select at least one group or press cancel')
- 4c the user remains on the group selection screen

Post condition: The groups are assigned to the test; the test overview window is opened.

### 9.3.6. Delete users

Priority	Trigger	Precondition	Actors
Optional	Selecting 'Remove Test User' on homepage	User is logged in	Registered users

#### Main path:

1. User selects 'Remove test user' on the main screen
2. The system opens a window with a text input field on it
3. The user inserts all email addresses into that text field which should be removed

- The user presses the 'REMOVE' button

**Post condition:** The user is removed from the database; test results of the user are not removed from previous test sessions; the user is back on the main screen.

### 9.3.7. Email invitation

Priority	Trigger	Precondition	Actors
Essential	Selecting 'send email invitation' on the test overview window	User is logged in and the test overview window is opened	Registered users

**Main path:**

- The user selects 'send email invitation' on the test overview window
- The system pop ups a virtual window asking the user to confirm the send request
- The system sends the emails

**Post condition:** The invitations are being sent and the user is back on the test overview window; the send invitation button is disabled as well as the add test user button.

### 9.3.8. View results

Priority	Trigger	Precondition	Actors
Essential	Selecting 'view results' on the test overview window	User is logged in and the test overview window is opened	Registered users

**Main path:**

- The user selects view results at the corresponding test on the 'test overview window'
- The system displays the results
- The user closes the window by clicking the 'close window' button

**Post condition:** The user is back on the main screen.

### 9.3.9. Participating in an online test without being invited

Priority	Trigger	Precondition	Actors
Optional	Clicking on the a link pointing to the test	none	Guest



**Main path:**

1. The guest clicks on the link which points to the test
2. The system provides a screen where the user has to state his email address
3. The guest enters the email address
4. The guest clicks 'SUBMIT'
5. The system adds the user to the test
6. The system sends an invitation link to the user
7. The user clicks on this invitation link
8. The system opens the test

**Exceptional path:**

- 5a The system recognizes the user as being already member of the test
- 5b The system states this to the client by an error message ('your email address was already used for this test')

**Post condition:** The user is on the test screen.

**9.3.10. Accept an invitation for an usability test**

Priority	Trigger	Precondition	Actors
Essential	Clicking on the invitation link in the email	Invitation was sent to the users email adress	Guest

**Main path:**

1. The guest clicks on the invitation link
2. The system opens the test

**Exceptional path**

- 2a The system recognized the invitation link as already been used
- 2b The system alerts the error to the user ('you have already jointed this test, delete previous results and start over?')
- 2c The user selects 'YES'
- 2d The system opens the test

**Post condition:** the user is on the test screen and the first task is presented.

### 9.3.11. Declining an invitation for an usability test

Priority	Trigger	Precondition	Actors
Essential	Clicking on the decline link in the email	Invitation was sent to the users email adress	Guest

#### Main path:

1. The user clicks on decline test in the email which was sent during the invitation process
2. The system marks the user as having the invitation declined
3. The system shows a confirmation message on screen ('you successfully declined this test')

**Post condition:** the user is marked and remains on the confirmation screen.

### 9.3.12. Request for user list removal

Priority	Trigger	Precondition	Actors
Essential	Clicking on the remove link in the email	Invitation was sent to the users email adress	Guest

#### Main path:

1. The user clicks on the removal link in the email which was sent during the invitation process
2. The system marks the user by setting the email address on the black list
3. The system shows a confirmation message on screen ('you have been successfully remove from the invitation list and you will not receive any further invitations')

**Post condition:** the user is marked and remains on the confirmation screen

### 9.3.13. Performing the test

Priority	Trigger	Precondition	Actors
Essential	Clicking on the join link in the email	Invitation was sent to the users email adress	Guest

**Main path:**

1. The user clicks on the join link in the email which was sent during the invitation process
2. The system marks the user's email address as being used
3. The system opens the corresponding test

**Post condition:** the user is on the first page of the test.

## 9.4. External Interface Requirements

### 9.4.1. User Interfaces

The system will provide two different user interfaces. The first one will be used to perform the usability tests.

First of all, the welcome message informs the user about the test and what to do during the test session. The message window with the start link is vertically and horizontally centered to have always the same start position (see Figure 22).

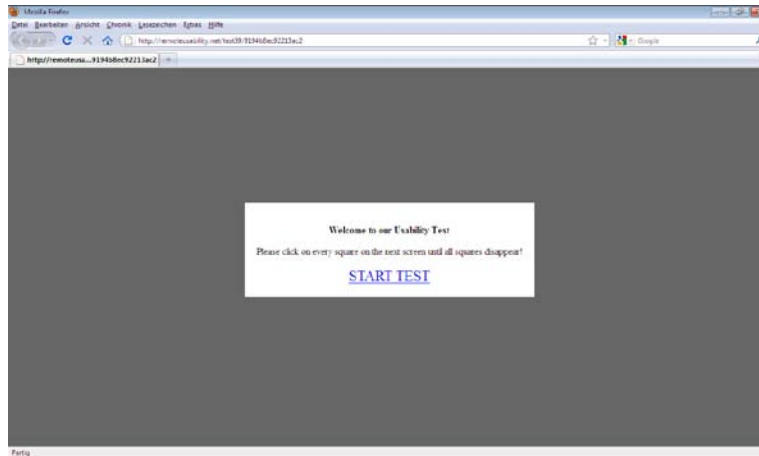


Figure 22: Usability test, start screen

As user may have different screen sizes, the user interface for the tests itself will basically consist of an empty area and a frame outside the centered area to simulate a standard screen size (see Figure 23).

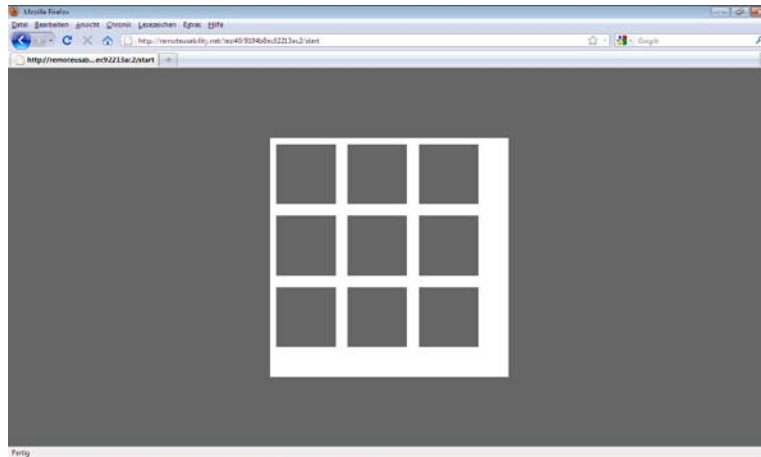


Figure 23: Usability test, test screen

The second user interface will be a virtual desktop for the usability test conductors to login into their accounts and generate and manage the tests. The main menu entries are accessible either by the drop down menu at the top of the window or the desktop symbols on the screen. Clicking on one of the menu entries will not lead to a reload of the page but opens a virtual window by displaying a div element shaped like a window (see Figure 24).

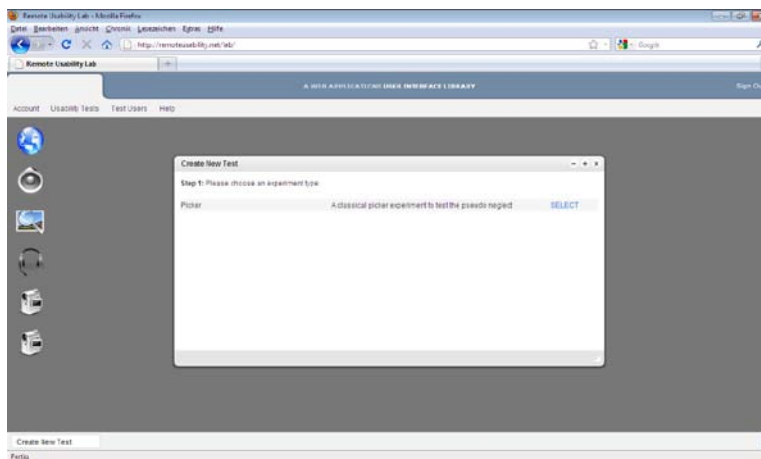


Figure 24: Test conductor interface with opened window

All windows will support the well know features of maximize, minimize and close by providing the according buttons in the upper right corner. Some windows may not be closed due to the

need of fulfilling a compulsory task (for example the login window). In this cases the corresponding button disappears.

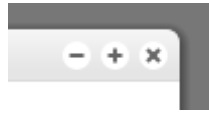


Figure 25: Window buttons

#### 9.4.2. Hardware Interfaces

For the usability conductors a minimum screen size of 800x600 is required due to the virtual desktop design which requires some space for the menu bar and the status bar at the bottom. The rest of the system does not require any other special hardware with the exception of future modules which may include other hardware like webcams or microphone.

#### 9.4.3. Software Interfaces

The system is build upon the MooTools (Proietti, 2010) JavaScript framework in the version 1.2.2. This framework is published under the MIT License (see page I) which guarantees the free usage even in a commercial context.

For the usability conductors interface another software framework will be used. The MochaUI (Houston, 2010) is designed for providing basic features of a web application like drawing windows to the screen as well as handling events such as closure, clicking and so on. The MochaUI tools are also available under the MIT license and the version 0.9.6 is used for the conductor's user interface.

#### 9.4.4. Communication Interfaces

Most of the data which is transferred between the client and the server is transmitted asynchronous by using the browser's XMLHttpRequest object. Therefore only web browsers which support this object are usable with this software. Beside the version of the browser, JavaScript has to be enabled as the system does not support an alternative interface for users without JavaScript.

## **10.Solution**

This section covers the information about the way chosen to fulfill the software requirements specifications and some differences to the specifications due to findings within the development process. Especially some technical concepts are illustrated in this section.

### **10.1. Concept**

The software is basically divided into three different parts. The first one provides a virtual lab for the usability experts. As the main URL (<http://www.remoteusability.net>) is reserved for explaining the service offered to potential customers, the subdirectory “/lab” offers the entry point for managing all usability tests. As users are used to have internet addresses with the prefix www, the usage of a subdomain like lab.remoteusability.net was not chosen.

The second part covers the process of conducting usability tests from the usability test user’s point of view. Users can participate in a usability study whether by following a link stated on some page leading to the test or through an individual link sent by email to the user.

The third part is for expanding the system in the mean of adding new usability test classes. As this is more the domain of psychologists the system is designed to be extendable by adding new modules to the system. All valid modules are recognized automatically and be available to the usability conductors. It is designed to use a corporation between software developers and psychologists to achieve scientific valid tests rather than developing tests with a software developer’s knowledge only.

### **10.2. Conductors User Interface**

#### **10.2.1. Conductors entry point**

Opening the website located at <http://www.remoteusability.net/lab> will open the entire web application and also open an overlay preventing the user to use any of the provided functions without the login or registration. A semi transparent overlay darkens the whole screen while a window with the functionality is in front of all other objects. This will give a visual hint to the user that the page is locked until a login or registration is performed.

On the left side of the window, a login form is provided, asking for the email address and the password, which will be auto generated during the registration process. For all new usability conductors who don't have an account at remoteusability.net the right side provides a registration form. The following data has to be provided by the user to successful finish the registration process:

- Company name
- Street
- City
- Postcode
- Nation
- Email-Address

All data is checked for valid input when the form is submitted. As all other data, also this form is sent to the server by an Ajax call. The validation of the entered data is done on the server side (see 10.2.3), providing a status flag in the response.

### 10.2.2. Password generation

Passwords are generated by the system rather than through user input. This is due to the fact that user generated passwords are weak and forgotten a lot (Florenco & Herley, 2007). Therefore passwords are generated by the system and consist of 8 characters as a combination of:

- Uppercase characters (A-Z , 26 characters)
- Lowercase characters (a-z, 26 characters)
- Numbers (0-9, 10 characters)

The bit strength of the password (Florenco & Herley, 2007) can be calculated as

$$\log_2((\textit{number of possible characters})^{(\textit{password length})})$$

and therefore

$$\log_2(62)^{(8)} = 47.634$$

in our case. As the average password bit strength found in the study of Florencio is about 40, the system generated passwords are about 20% stronger.

### **10.2.3. Login process**

As most of the data stored in the platform is confidential, the unauthorized access to this data has to be avoided. Handling the Login only through JavaScript would be possible (for example by setting Cookies). But as the server can't control parts of the software which are executed on the client's computer, the door for manipulating user sessions is widely opened. The login data is therefore transmitted to the server and evaluated. A temporary password is set in the session variables. This password is used in all future requests to check the permission of the requests.

The reason for some bugs during the development of the software was the lost of the session variables because the session management has to be started in every PHP script which wants to access those variables. It is important to include the 'session\_start();' call in every page.

### **10.2.4. Server side form validation**

As all JavaScript code is available and executed on the client's computer, the manipulation of the validation code and the transmission of manipulated data could be possible. To check the data entered in a form at the platform (not only into the registration form) it is sent to the server as it is. The code returned is received by the calling Ajax script and evaluated according to the flag transmitted.



```

removeTestUser = function(form) {

    var req = new Request({
        url: "data/removeuser.php",
        data: "&rtxtemail="+form.rtxtemail.value,
        onSuccess: function(txt){
            $('registererror').set('text', txt);
            var response = txt.split("|");
            if (response[0]=="OK") {
                $('registration').set('html', response[1]);
            } else {
                if ($(response[0]) != null) {
                    $('registererror').set('text', response[1]);
                    $(response[0]).focus();
                }
            }
        },
        onFailure: function(){
            $('registererror').set('text', 'Could not reach server! ');
        }
    });
    req.send();
    return false;
}

```

Figure 26: Example Ajax request

A proprietary protocol was chosen to inform the Ajax application about the status of such verifications:

- ['OK' || 'ERROR MESSAGE DIV NAME']|[' MESSAGE']

The first field indicates whether the request was successful or which DIV should be updated with the error message. The second contains a proper error message in the case the request was unsuccessful.

#### 10.2.5. Form data transmission

The data of the form is sent by fetching the submit event of a form. It is important to return 'false' back to the form event handler to avoid a real submission to the URL stated in the action property of the form definition. This would be also possible by adding an onClick event handler to the submission button but in that case the data of the form cannot be accessed in an easy way. In the second case, the name of the form has to be provided as a parameter to the onClick handler method and within this method, the corresponding DOM object has to be obtained by searching the DOM tree. In the first case, the form itself can be passed to the handler by stating the 'this' object which represents the form.

```
<form id="loginform" action="#" onsubmit="return checkLogin(this);">
```

Figure 27: Example form definition for handing over the form to a JavaScript function

If a system should support JavaScript and non JavaScript user, the usage of a valid action entry can be used to enable the correct action as stated in the attribute as the 'onsubmit' statement will not be executed in that case.

#### 10.2.6. Email verification

In contrast to the Software Requirements Specifications the use of extra email verification was cancelled because of the fact that sending the password to the client fulfills the same functionality as sending a ping string to the same email address. The knowledge of the system generated password verifies the email also.

Having only one email sent to the usability conductor reduces also the steps of the registration process which enables a quicker and safer way because sending more than one email with instructions always has disadvantages like being misunderstood or not received (spam filters).

The usage of free email addresses like Yahoo, Google Mail or GMX should be forbidden in a future version to avoid having registrations with fake email addresses.

#### 10.2.7. Email validation

The email verification discussed in section 10.2.6 should ensure that only the owner of the email address uses it and therefore the no other person can use it for the registration process.

#### Syntax

Email validation on the other hand is to ensure that an email address is valid in different senses. First of all, the syntax of the email address can be checked to avoid people entering some irrelevant characters into such a field due to the fact that they enter the wrong information in the email field because they mixed up the form input fields or if someone tries to enter invalid information to access the service without giving their real email address.

To achieve a syntax check of the email address, regular expressions are used as they are also well supported by the PHP scripting language.

As it is not possible to check an email address syntax with one single regular expression due its possible nested structure (Kruse, 2007) a combination of regular expressions is used to verify a email address.

Mainly such a validation checks three parts:

- Protocol (“mailto:”)
- User part
- Domain part

These three parts may be combined as “mailto:user.part@domain-part.com” where the protocol part may only consist of “mailto:” and is not necessary for a valid email address.

The user part must only consists of the characters A-Z, a-z, 0-9, an underscore, a dot and a hyphen. At begin of the user part there has to be an alphanumeric character, so the underscore, dot and hyphen are not allowed.

As stated by Kruse (Kruse, 2007) a host name may consist of n sub domains, a domain name and a top level domain.

- consist only of the characters a-z, A-Z, 0-9, a dot, an underscore and a hyphen,
- start with a-z, A-Z, 0-9
- end with an dot

and the domain name follows the same except that a valid email address can exists without a subdomain where the domain always have to be stated.

The top level domain (.de, .com, .org, .mobi ...) is a string consisting of 2 – 6 characters where the ‘museum’ top level domain is the longest official top level domain.

### **Prove of existing Mail Exchange Resource Records (MX-RR)**

Even if the syntax of the email address is valid, the email address or the whole domain name may not exist or don’t offer an email interface. As every outgoing email server first checks the Mail Exchange Resource Records of the domain part of an email address to evaluate which mail server is responsible for the given domain name, it is also a good idea to perform the same check to validate an email address. Because if no such MX Resource Records exists, no

regular email server could deliver the mail. Fortunately PHP offers a way to perform such request easily:

```
bool getmxrr ( string $hostname , array &$mxhosts [, array &$weight ] )
```

If MX records for the domain in the \$hostname variable exist the getmxrr function will return true and fill the \$mxhosts variable with the found records.

### **10.3. Email invitation**

#### **10.3.1. Customized invitation links**

Potential test users can be invited to participate in a usability test by sending them invitations through email. On the one hand afford for a user to participate in a test should be as low as possible and therefore no login or other identification should take place with the except that personal information is needed in a usability test. On the other hand the system needs to know who is trying to start a test as most of the tests are limited to be used only once per user. Therefore the invitation link, which is sent to the email address of the potential test participant, is unique and follows the following format:

```
http://www.remoteusability.net/test[testnumber]/[unique_user_id]
```

The “[testnumber]” represents the real unique test id in the database. In contrast the “[unique\_user\_id]” is a random generated key with a 16 characters length. This will help to avoid people guessing the unique link and perform the tasks without permission.

As PHP offers a method for generating such unique ids without a database lookup to check for duplicate entries, no proprietary function was implemented. The PHP internal method is based on the system time. In the worst case, the same id is generated within a millisecond as this is the smallest fraction of the PHP time function. To be sure, a prefix of an 3 digit random number is added to the numeric id generated by the PHP “uniqid” method.

#### **10.3.2. Email spooling**

Sending a huge number of emails can be a problem as the maximum execution time of a PHP script is mostly limited. Therefore the email addresses are stored into a spooler represented by a table in the database after the usability conductor starts the campaign.

As the process is asynchronous the user has not to wait until the sending finishes and can work on other issues within the application.

A “cronjob” was set up to call the PHP script which is responsible for sending the invitation emails at a 2 minute basis. Every time the script is called, 10 emails are sent. This amount can be increased but is held low at this time since the system is running on a shared server.

On Linux operating systems a cronjob can be easily set up by running crontab with the parameter `-e` for editing. Every line in the crontab settings represents a task which is executed on the specified time. Syntax:

- `m h D M DW path/to/command arg1 arg2`

where m = Minute (0-59), h = Hour (0-23), D = Day (1-31), M = Month (1-12), DW = Day of week (0-7, where 0 represents Sunday). Every value can be set to \* what will represent all possible values.

For calling the email invitation script the following cronjob settings were added:

```
*/2 * * * * /usr/bin/php /srv/www/vhosts/remot usability.net/httpdocs/emailcron.php
```

## **10.4. Test user frontend**

### **10.4.1. Start screen (splash screen)**

The idea behind the horizontally and vertically centered welcome message on the start screen is to force the user to move the mouse to the center of the screen before the test starts. This will ensure that everyone has the same start position regardless of the screen size used. The test only starts when the user clicks on the corresponding start link. The splash screen is also used to explain the rules of the test to the client.

As it may look simple to center a DIV on screen the reality is different. A cross browser solution has to ensure that the dimensions of the screen are read correctly, otherwise it will lead to errors as some functions or JavaScript class properties simply don't exist on different browser versions or brands.

Basically centering is done by getting the width and the height of both involved elements: the available screen and the object which has to be centered. Subtracting the DIV height from the

screen height will result in the total available margin for above and below the DIV. Therefore it has to be divided by 2. The final result can be used to set the DIV's top property to move it down the screen to the proper position. The same procedure with the corresponding width values instead of the height values works of course for horizontally centering the screen.

As the Internet Explorer is always a bit different than other browsers, a special work around for older Internet Explorer versions has to be done.

Nearly all major browsers support the `window.innerWidth` and the `window.innerHeight` property to receive the dimensions of the available screen. This is valid except for the Internet Explorer of Microsoft. If the document type declaration is set to a mode that forces the browser to switch to the 'standards compliant mode' the '`document.documentElement.clientWidth`' and '`document.documentElement.clientHeight`' properties have to be used. If not, IE delivers only correct values in the `document.body.clientWidth` and `document.body.clientHeight` properties.

```
function getWindowDimensions() {
    var wWidth = 0, wHeight = 0;
    if( typeof( window.innerWidth ) == 'number' ) {
        //Non-IE
        wWidth = window.innerWidth;
        wHeight = window.innerHeight;
    } else if( document.documentElement && ( document.documentElement.clientWidth ||
document.documentElement.clientHeight ) ) {
        // IE in 'standards compliant mode'
        wWidth = document.documentElement.clientWidth;
        wHeight = document.documentElement.clientHeight;
    } else if( document.body && ( document.body.clientWidth || document.body.clientHeight ) ) {
        // IE compatible
        wWidth = document.body.clientWidth;
        wHeight = document.body.clientHeight;
    }
    var ret = new Array();
    ret['width'] = wWidth;
    ret['height'] = wHeight;
    return ret;
}
```

Figure 28: Cross browser solution for receiving window dimensions

## 10.5. Example test (Picker)

This section will cover the frontend components of the sample Picker usability test to illustrate how such test may work on the client's side.

### **10.5.1. Background**

Bettina Diekamp et al. (Diekamp et al., 2005) conducted an experiment with birds to evaluate the theory that they tend to prefer objects on the left side when they have the opportunity to choose between alternatives spread from left to right.

It was shown that the birds really displayed a clear bias to the left hemisphere but due to some anatomic differences the test should be repeated with humans. Especially when talking about web usability it would be good to know if people tend to draw more attention to items on the left or on the right side (Diekamp et al., 2005).

One opinion is that - due to the structure of the brain – information is better and faster processed if it is transmitted directly to the right half of the brain. As the right hemisphere is responsible for processing visual input and information received by the left visual field is processed in the right hemisphere, images and navigation should be placed on the left side of a webpage (Weiland & von Gizycki, 2002).

Other state (Kalbach & Bosenick, 2003) that there is no significant difference between the right or left placement of web navigation elements.

Therefore a study about an evaluation about the so called 'visual pseudo neglect' among a huge number of participant could be interesting. Especially because such tests are normally conducted by animals or a low number of people (sample size).

### **10.5.2. Settings**

The possibility to change all parameter of the test enables the usability test conductor to adapt the test to a special purpose. For example, to a group of people (elderly people) could be confronted with bigger items while another group is confronted with small but more items. The parameters which can be set by the conductor are:

- Number of items to display
- Item width
- Item height

### 10.5.3. Procedure

These items represent the seed in the bird pick experiment discussed in section 10.5.1 and have to be 'picked' away by clicking on them.

Technically these items are DIV elements with the dimensions given by the individual test settings. A click handler is added to them by

- selecting them by name:

```
var x = getElementsByClass("picker");
```

- and iterating through the found elements and assigning the handler:

```
for (var i = 0; i < x.length; i++) {  
    x[i].onclick = new Function("alertme("+x[i].id+", "+x.length+"");  
}
```

In this case, every time when such an element is clicked, the function "alertme" is called with the parameters id and length which represents the id of the current clicked item (for accessing it later when processing the click) and the overall number of items. The number of items is passed to the function because of design constrains. In further experiments more than one picker field could be on screen so it is a smart way to inform the function about the whole size of the picker field.

The function called by the click handler has 3 tasks to perform. First of all it will make the clicked item invisible so it disappears from screen. Second task is to check if all items are clicked away and so if the test is finished. In the case that all items are picked away, it calls a function which sends the results to the server (third task).

```
function alertme(id,numitems) {  
    adds.push(id);  
    document.getElementById(id).style.visibility='hidden';  
    if (adds.length==numitems) {  
        sendResult(JSON.encode(adds));  
    }  
}
```

As the variable 'adds' stores the ids of the items (which is a number according to the placement) the result contains the sequence in which the items were clicked. That is exactly what was done in the bird pick experiment.



#### **10.5.4. Data submission**

As results of usability experiments may consist of different data types the results are converted to the JSON format where

*“JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.” (json.org, 2010)*

The advantage to convert the results to JSON is that the sender and the receiver are independent from the format of the data and that JSON converting tools for encoding and decoding objects are available in both, JavaScript and PHP.

In the specific case of the picker test, an array holds the ids of the clicked items, where the order in the array reflects the order of the clicks. Therefore serializing it with JSON results in a string like this example:

```
["0","1","2","5","4","3","6","7","8"]
```

Item 0 was clicked first, followed by items 1, 2, 5 and so on. This makes it easy to process it on the server where the data could be converted back to an object. But in this special case, the string is directly stored into database as it is an efficient way to store it in the case of a heavy usage. This is because only one step has to be done on server: the storage.

#### **10.6. Data model**

All data necessary for the web application is stored in one MySQL database. If the application goes live and a heavy usage is forecasted, the results tables should be moved to another database to reduce server load. The other data may remain in one database since the number of test conductors is relatively low compared to the number of test users.

##### **10.6.1. Scheme**

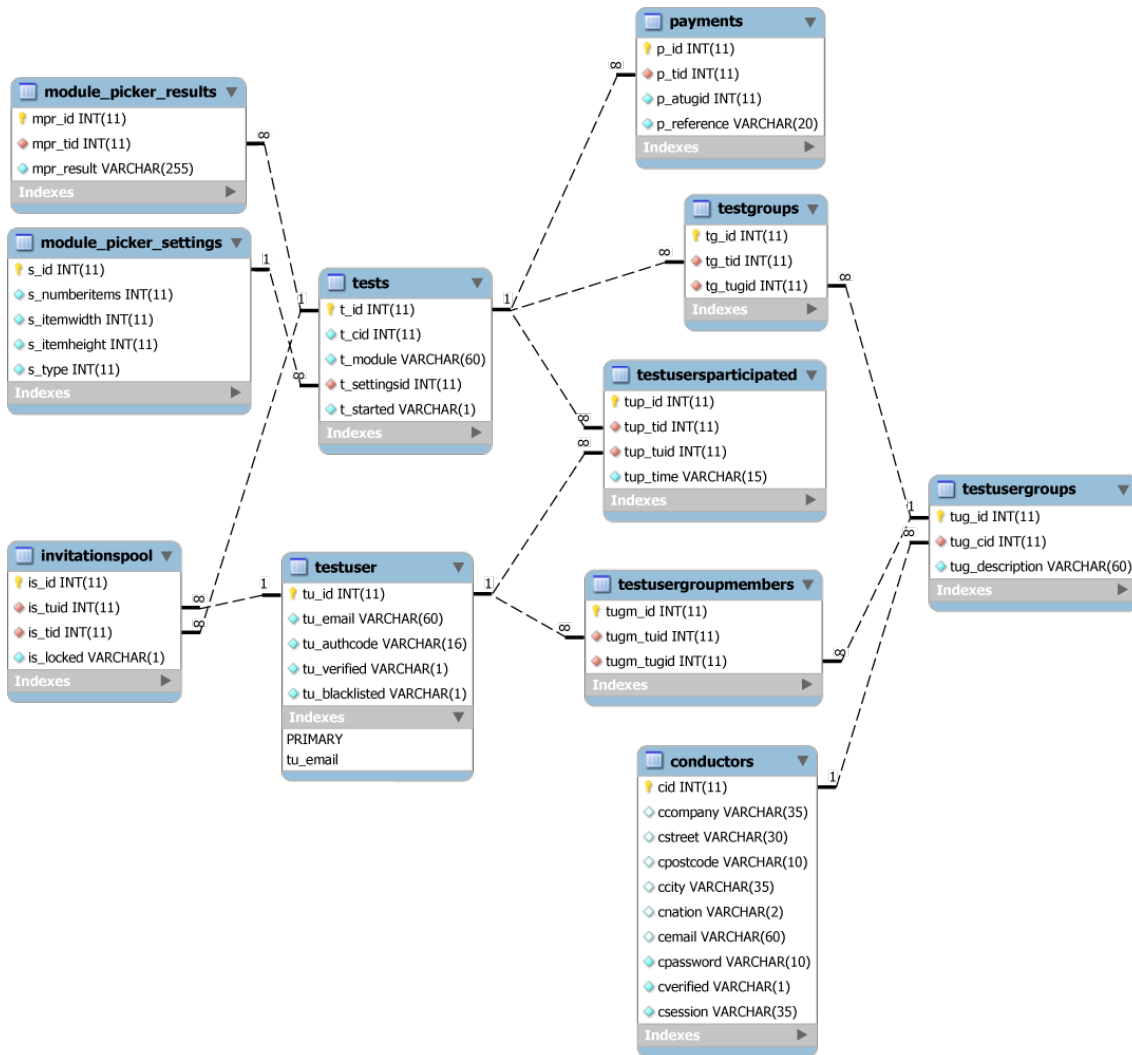


Figure 29: database relation model

### 10.6.2. Central tables

- Table conductors:** this table stores all information necessary to identify a test conductor. It includes the username (= email address) and the password as well as the session password which is used to check the login status when a page is requested. All other data stored in this table is entered by the conductor at the registration process and may be altered through the 'edit profile' option in the main menu.
- Table 'testuser':** as the same test user may be added to the system by different conductors, the 'testuser' table and therefore the email addresses are completely separated from the conductors. With this design it is also possible to have the same

authentication code, valid for all usability tests. Furthermore it enables to blacklist a email address system wide. This guarantees that the email address is not used in future even a usability conductor adds the email address again.

- **Table ‘testusergroupmembers’:** this table connects all email addresses (test user) with the test user groups which can be generated by a conductor. This table is used to get all email addresses for a given test user group id and to make the test user independent from the available test user groups.
- **Table ‘testusergroups’:** this table actually represents the single conductor’s test user groups which the conductor has created in the admin panel.
- **Table ‘tests’:** the test table holds all data needed to start a usability test. Beside the owner of the test (conductor id) the name of the module which represents what kind of test module should be loaded during the test is stated. As different test modules may have different types of settings, those settings are stored in another table and only linked to the test.

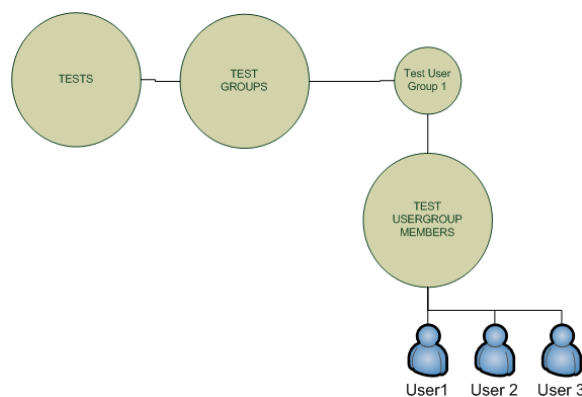


Figure 30: relations of the test user tables

So a test can contain one or more test user groups. This relation is stored in the ‘testgroups’ table. A test user group on the other hand consists of one or more test users. Therefore a table for holding this relation was introduced (‘testusergroupmembers’).

- **Table invitation pool:** this table holds all email addresses and information to the corresponding tests so an invitation mail with the invitation link can be generated and sent to the user.

## 10.7. Custom model concept

The software was designed to be extendable by psychological specialists. As software developers are well educated in informatics the field of generating psychological tests affords a lot of domain specific knowledge. Therefore the system has an open interface for adding such tests in an easy way to the platform.

### 10.7.1. Directory structure

Modules added to the system are recognized automatically due to a mechanism which scans the modules directory for new added modules. To be recognized as a module, the naming of the subdirectory and the PHP file holding the module class is important. It is also important to mention that all names are case sensitive.

The directory is structured as followed:

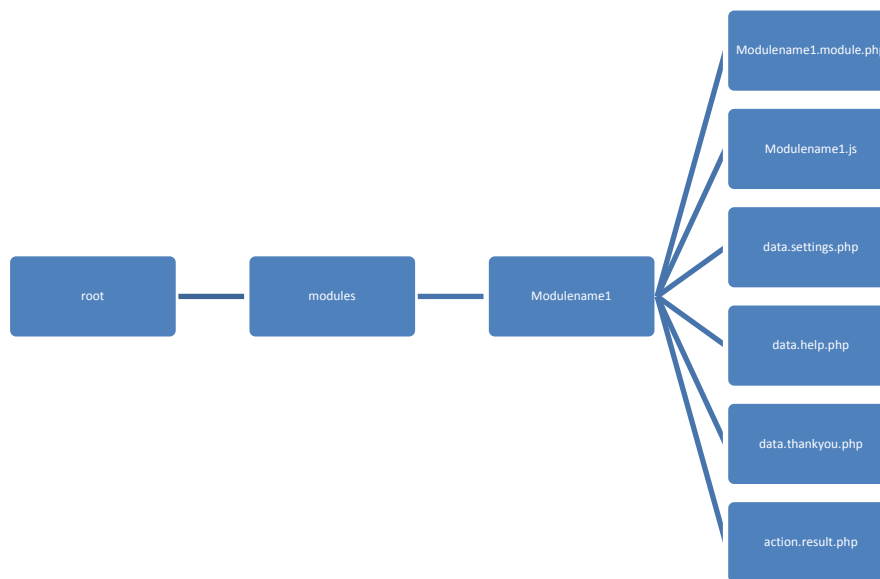


Figure 31: custom modules hierarchy

As shown in Figure 31 each module is located in its own directory, which is a subdirectory of the modules folder. It is part of the naming convention that those directories always start with an uppercase letter as well as the files within the module folder.

All files shown in the figure are necessary. While files starting with 'data' are necessary to display some static data (input forms, welcome messages and good bye messages) the files labeled with 'action' perform some operations like calculations or storages.

### 10.7.2. Class file

Each module has to have a class file named with the module name as prefix and the extension “.module.php”. This class extends the super class ‘modules.class.php’ which ensures that all classes have the required methods defined. The super class also handles the database access by providing a member variable with an instance of the database class.

The following necessary class methods are defined but unimplemented in the super class. The implementation has to be done in the corresponding module class.

- `Install()`: this method is responsible for creating the necessary tables in the database which are needed for the module to store results and other module specific data.
- `Uninstall()`: this will remove all tables from database which are generated during the installation process.
- `storeSettings($array)`: used to store the settings which a usability conductor enters during the test setup process.
- `storeResults($testid, $data)`: will be called when a test is completed. The variable `$testid` will be used to add the results to the right instance of a test class while the `$data` variable holds the results.
- `getSettings($settingsid)`: reads the settings from database and stores the result into member variables.
- `drawGamePad()`: as the test area is called game pad, it’s implementation should draw the test itself to the screen.

As mentioned the methods above have to be implemented by each of the added modules. The next section contains methods which should be overridden by the child classes:

- `getName()`: returns the name of the module.
- `getDescription()`: returns a short description about the module. This description is shown to the usability conductor when he creates a new instance of a test.
- `getVersion()`: returns the current version of the module.

All other methods are final methods not available for overriding. Most of them are responsible for drawing the test screens. As all of them are called automatically, the developers of new modules do not need to call them and know them in detail.

### 10.7.3. JavaScript logic

Most of the client-side functionality discussed in section 75 is implemented in the module's JavaScript file. As this is completely up to the developer of new modules, there are no formal requirements. Possible usages are

- Handling any type of events: as nearly every usability test has to measure something which can be evaluated afterwards, events have to be caught to enable such measurements. Some examples of events:
  - mouse events like clicks, double clicks and movements
  - key events like key up, key down
  - form field events like on change
- Manipulation of displayed objects: depending on the type of the test itself, properties of displayed objects may be altered. Some examples for such properties:
  - visibility
  - color
  - text decoration
  - position
  - size
- Transmission of data and handling the server response: all data which has to be transmitted for later evaluation or for continuing the test. Some examples for such requests:
  - Transmission of test results (time, coordinates, way covered)
  - Event handling which immediately need response from the server (next level, decisions according to user input)

This script is included in the test page automatically at the end of the page. Nevertheless the call of any initializing function within the JavaScript should be added to the onLoad() function which guarantees that all DOM objects are already available for manipulation.

### 10.7.4. Settings

The data.settings.php file contains the form fields for selecting all necessary options for a specific test. The form tag itself must not be present as it is automatically insert to ensure that the data is sent to the right script. It may consist of any valid form input field like text fields,

drop down menus, radio buttons, check boxes and so on. It is completely up to the developer to design the settings form.

This settings page will be displayed as the second screen after the usability conductor creates a new test and just before the conductor selects which user groups should be added to participate in the test.

#### **10.7.5. Help**

The data.help.php file should contain some short information about how the test has to be executed, so the test user knows what is to do during the test.

It should be as short as possible since the whole information is centered on screen and is not designed to have pages of instructions.

#### **10.7.6. Endscreen**

The data.thankyou.php file contains last word for the usability testers. Mostly used for thanking them for participating.

#### **10.7.7. Results**

Results are calculated and displayed through the action.result.php file. It is up to the developer if the results are generated every time the usability conductor clicks on the corresponding 'display results' button or if they are calculated once and so only read and displayed.

This file is included every time the usability conductor clicks on the 'show results' link within the list of existing (and running) usability tests.

#### **10.7.8. Recognition of valid class files**

As mentioned above a module class file is valid when it is a child class of the 'UsabilityGame' super class as all necessary methods are predefined in this super class. Therefore the following procedure for recognizing available usability test modules was chosen:

1. Get all files which match the naming conventions:

```
$dir = ROOT_PATH.DIRECTORY_SEPARATOR."modules";  
if ($handle = @opendir($dir)) {  
    while (($file = readdir($handle)) !== false) {  
        if (@is_file("$dir/$file/$file.module.php")) {
```

```
include_once("$dir/$file/$file.module.php");
    }
}
closedir($handle);
}
```

The modules folder is scanned for files matching the syntax: "\$dir/\$file/\$file.module.php" where \$dir contains the path to the modules folder and the \$file variable contains the name of the module. If such a file is found, it is included so an instance of this class can be generated.

2. The classes are filtered so only such classes are taken which are subclasses of the 'UsabilityGame' class:

```
$res = array();
foreach (get_declared_classes() as $class) {
    $parentclass = strtolower(get_parent_class($class));
    if ($parentclass == 'usabilitygame') {
        $result[] = strtolower($class);
    }
}
sort($result);
return $result;
```



## **11.External software components**

### **11.1. MochaUI**

MochaUI (Houston, 2010) is a framework designed to easy build up user interfaces for different usages. As mentioned on the developers website, the main focus of the software is on

- Web applications
- Virtual desktops
- Enriching normal websites
- Widgets
- Modal dialogs and standalone window

As the current stable release (0.9.5) has some problems with the Internet Explorer 8 it is recommended to check out the latest version from the SVN repository. The software developed along with this diploma thesis was build up the version “0.9.6 development” and works with the latest Firefox Browser (available Version 3.6) and Internet Explorer (available Version: 8.0.76)

#### **11.1.1. Windows design and behaviour**

The main feature of the MochaUI framework is definitely the easy generation of nice looking windows which support all functionality of normal software program windows. Beside the features mentioned below MochaUI supports a lot more methods and parameters to style and set the behavior of the windows but they are only used in special cases.



Figure 32: a MochaUI window

- **Title bar:** the title bar consists basically of the page title which is shown on the left side and the title bar buttons to minimize, to maximize and to close the window. It is up to the developers which of this buttons are shown. By setting the minimizable, maximizable and/or closable attributes to false or true, those buttons appear or disappear. This can only be done within the initialization of the windows.
- **Windows resizing:** if the resizable attribute is set to true, the window can be resized not only by using the title bar buttons but also by grasping one of the borders or one of the corners of the window. This is a well know feature of all windows based operating systems. When defining the `resizeLimit` attribute a minimum and/or maximum value for resizing the window is set.
- **Content loading:** according to different loading method options the content can be loaded basically in two different ways.
  - **As part of the current page:** only the HTML code of the windows 'content div' is replaced and therefore all scripts and style sheets are also valid for the new loaded content.
  - **Within an iframe:** a new generated iframe is placed as a child of the windows 'content div'. The loaded pages have to have their own style sheet definitions and scripts as there is no access to the parent's definitions.

- **Source of the content:** when the content is loaded in an iframe, the source of this iframe has to be a valid URL. Therefore external and internal pages can be loaded. The same is true when using the XHR loading option which means that the content is loaded via the XMLHttpRequest object. If the HTML code for the window is already known or directly created by a JavaScript function, the HTML loading option can be used where only the HTML code is handed over. So no URL has to be stated in that case.
- **Modal or normal:** modal windows are used to force the user to react to the opened window before the other functionality is available again. This is normally achieved by darkening the whole screen and displaying the modal window in front of all other objects. Such windows are normally used when user have to do a compulsory step or when the focus of the users attentions should be drawn to a special part of the screen. A classic usage for a modal window is when a user needs to login at the beginning. Here the modalOverlayClose attribute should be set to false because otherwise the window can be closed by clicking on any area outside the window. The last mentioned option for closing windows is normally used when the modal window is only used to display a special content (image galleries, videos) – therefore the modalOverlayClose attribute should be set to true.

### 11.1.2. Basic steps when setting up a MochaUI interface

As MochaUI is build upon the Mootools JavaScript framework (Proietti, 2010) the corresponding JavaScript files have to be added before all other project JavaScript files. This include the core and the more file of Mootools as MochaUI needs both of them. Both files can be downloaded at the Mootools webpage in a compressed form so they need less bandwidth.

After adding the Mootools scripts to the page, also the MochaUI.js file has to be included in the page. When done, the head section of the HTML page should include the following lines:

```
<script type="text/javascript" src="scripts/mootools-1.2.2-core.js"></script>
<script type="text/javascript" src="scripts/mootools-1.2.2-more.js"></script>
<script type="text/javascript" src="scripts/mocha.js"></script>
```

With the exception that the version numbers of the Mootool files may vary.

The actual design of the own application is normally done in a separate JavaScript file. The first step is to initialize all windows by defining functions which generate those windows. This does not mean that those windows are displayed or the content of the windows are loaded but that the functions which can generate those windows are defined.

```
initializeWindows = function(){
    MochaUI.ajaxpageWindow = function(){
        var myWindow = function(){
            new MochaUI.Window({
                id: 'mywindow',
                title: 'My Window',
                loadMethod: 'xhr',
                contentURL: 'pages/lipsum.html',
                width: 340,
                height: 150
            });
        }
    }
}
```

Figure 33: a sample window generating function

As you can see in at Figure 33 such a function includes the call of the MochaUI.Window constructor along with all parameters which the user want to set. Of course, the windows and their creating function may be defined at any other place of the code which is reachable from a calling event handler but it is common to bundle this functions in another function so a single call within the window’s domready event handler will define all other functions.

This guarantees that any of the function calls will be stated only if the whole page is loaded and no access to a non existing DOM element is done.

```
window.addEvent('domready', function(){
    initializeWindows();
});
```

Figure 34: a sample window generating function

**11.1.3. Further useful methods**

Two other useful methods should be mentioned. First of all a way for opening different windows during the usage of the system has to be defined. This is done by adding a click event handler to a DIV object in the DOM. The \$ function defined by Mootools is used to select a specific DIV element by the id of it. When you use for example \$('test'), the DOM tree will be searched for an element with the id == 'test' and a reference to this element is returned. So it is possible to add any valid method call directly behind the \$ function:

```
if ($('#ajaxpageLinkCheck')){
    $('#ajaxpageLinkCheck').addEvent('click', function(e){
        new Event(e).stop();
        MochaUI.ajaxpageWindow();
    });
}
```

Figure 35: adding a click handler to a DOM object

As the execution of JavaScript is always stopped in browsers when an error occurs it is always a good idea to avoid making such failures. Therefore the existence of the DOM element should be checked before any access to this DOM element is tried. Even when the DOM element is hard coded in the source, it can be removed by JavaScript and a call like in Figure 35 but without the if condition would lead to the access to a null object.

The second method which should be part of this basic introduction to MochaUI is the `updateContent` method which is used for changing the content of an existing window. This is useful when having a multiple step process which should be covered in a single window. A next button or a link to the next step can then be used to trigger the update function which can basically contain the same attributes as when calling the constructor of the `MochaUI.Window` class.

```
showPingForm = function(cid) {
    MochaUI.updateContent({
        element: $('#login'),
        loadMethod: 'xhr',
        url: 'pages/pong.php?cid='+cid,
        title: 'Email Verification',
    });
}
```

## 12.Future Work

As this piece of software was always thought to be a first prototype of an virtual and remote usability lab, a lot of things may be added to a commercial version. First of all the platform should be multilingual for both, the usability experts and the test clients. Especially for the test users, a logic to select a proper language and to show the test instructions in their language would be a great help for them. Less important is to offer multi language support for the test conductors backend because it is assumed that most of them speak at least English.

A detailed market research should be done to decide which business model should be chosen. As discussed in section 8 several different models are available but require to adapt the system logic to process such business flows. For example if the fee is calculated according to the number of test participants, the test conductor must be informed on the right place and a payment option must be provided. If in contrast a membership fee is charged, the payment process has to start already at the login or registration – just before the usability conductor can start working with the platform.

The system should be further checked for security holes which enables other users to access some data without permission. As the whole application is based on Ajax, a lot of the logic is available on the clients side. Therefore the safety of the data has to be validated.

Some common community and user management features should be implemented as having a lost password function as well as the opportunity to change the email address of the conductor. Changing the address would require to verify the ownership of this address by sending a confirmation link which has to be clicked before the new email address is activated.

## **13.Conclusion**

Many usability software is available and also a small number of online services for usability testing can be found on the internet. But these test are always predefined and can – in the best case – be adapted by setting some parameters. The prototype developed in this thesis is different because it is not a service provided to usability specialists but offers the possibility to extend the system in an unlimited manner. How this test modules look like and how they behave will not be decided by the system owner but by the owner's clients. As it is up to the business model chosen, the owner may be a single company or the system may be distributed to run on separate servers independent.

The openness of the system on the other hand requires always a software developer who implements the new test modules. But this is only needed if the test conductor wants to have something new which is not present in the system. If it is present, creating a new usability test is very easy and done within 2 minutes.

## Appendix

### A. MIT License

*“Copyright (c) <year> <copyright holders>*

*Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:*

*The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.*

*THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.” (OSI, 2010)*



## Literature

- Abran, A., Khelifi, A., Suryan, W. & Seffah, A. (2003) Usability meanings and interpretations in ISO standards. *Software Quality Journal*, 11, 4, 325-338.
- AlertPay (2010), Online available: <http://www.alertpay.com>, last access: 2010-02-19:
- Amazon (2010), Online available: <http://www.amazon.com>, last access: 2010-02-18:
- Andreasen, M., Nielsen, H., Schrøder, S. & Stage, J. (2007) What happened to remote usability testing?: an empirical study of three methods. *Proceedings of the SIGCHI conference on Human factors in computing systems*. San Jose, California, USA, ACM New York, NY, USA.
- Avaliani, A. (2004) Successful E-Business Systems-Paypal. *Arxiv preprint cs/0412011*.
- Baravalle, A. & Lanfranchi, V. (2003) Remote Web usability testing. *Behavior Research Methods Instruments and Computers*, 35, 3, 364-368.
- Bias, R. (1994) The pluralistic usability walkthrough: Coordinated empathies. *Usability inspection methods*. John Wiley & Sons, Inc. New York, NY, USA, 63-76.
- Bontis, N. (2001) Assessing knowledge assets: a review of the models used to measure intellectual capital. *International Journal of Management Reviews*, 3, 1, 41-60.
- Castillo, J. & Hartson, H. (2006) Remote Usability Testing Methods a la Carte.
- Castillo, J., Hartson, H. & Hix, D. Remote usability evaluation at a glance. *Disponível por WWW em [http://hci.ise.vt.edu/~josec/remote\\_eval/docs/TR\\_remote\\_evaluation.pdf](http://hci.ise.vt.edu/~josec/remote_eval/docs/TR_remote_evaluation.pdf) (mar. 1998)*.
- Chakrabarti, R. & Kardile, V. (2002) *The Asian manager's handbook of e-commerce*. Tata McGraw-Hill.
- Christos, F., Christos, K., Eleftherios, P., Nikolaos, T. & Nikolaos, A. (2007) Remote Usability Evaluation Methods and Tools: A Survey. *11th Panhellenic Conference on Informatics*. Patras, Greece.
- creditcards.com (2010), Credit card statistics. Online available: <http://www.creditcards.com/credit-card-news/credit-card-industry-facts-personal-debt-statistics-1276.php#card-circulation-issuer>, last access: 2010-02-19:
- Diekamp, B., Regolin, L., Güntürkün, O. & Vallortigara, G. (2005) A left-sided visuospatial bias in birds. *Current Biology*, 15, 10, 372-373.

- Dillon, A. (2001) Beyond usability: Process, outcome, and affect in human computer interactions. *Canadian Journal of Information and Library Science*, 26, 4, 57-69.
- Dumas, J. & Redish, J. (1999) *A practical guide to usability testing*. Intellect Books.
- Ebay (2010), Online available: <http://www.ebay.com>, last access: 2010-02-18:
- Fishburn, P. & Odlyzko, A. (1999) Competitive pricing of information goods: Subscription pricing versus pay-per-use. *Economic Theory*, 13, 2, 447-470.
- Florencio, D. & Herley, C. (2007) A large-scale study of web password habits. *Proceedings of the 16th international conference on World Wide Web*. Banff, Alberta, Canada, ACM.
- Garrett, J. (2005) Ajax: A new approach to web applications.
- Gaver, W. W. (1991) Technology affordances. *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*. New Orleans, Louisiana, United States, ACM.
- Hansen, M. (1991) Ten steps to usability testing. *Proceedings of the 9th annual international conference on Systems documentation*. Chicago, Illinois, United States, ACM.
- Hars, A. & Ou, S. (2001) Working for Free?—Motivations of Participating in Open Source Projects.
- Hartson, H., Castillo, J., Kelso, J. & Neale, W. (1996) Remote evaluation: the network as an extension of the usability laboratory. *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*. Vancouver, British Columbia, Canada, ACM.
- Helander, M. (1997) *Handbook of human-computer interaction*. Elsevier Science Inc.
- Holleran, P. (1991) A methodological note on pitfalls in usability testing. *Behaviour & Information Technology*, 10, 5, 345-357.
- Holzinger, A. (2001) *Basiswissen Multimedia Band 3: Design. Entwicklungstechnische Grundlagen multimedialer Informationssysteme. Das Basiswissen für die Informationsgesellschaft des 21. Jahrhunderts*. Würzburg, Vogel Buchverlag.
- Holzinger, A. (2003) *Multimedia Basics, Volume 3: Design. Developmental Basics of Multimedia Information Systems. The fundamentals for the Information Society of the 21st Century. Translation of the original German First Edition (with assistance from Dawn E. Carmichael and Gig Searle)*. New Delhi, Laxmi-Publications.

- Holzinger, A. (2005) Usability engineering methods for software developers. *Commun. ACM*, 48, 1, 71-74.
- Holzinger, A. & Errath, M. (2007) Mobile computer Web-application design in medicine: some research based guidelines. *Universal Access in the Information Society*, 6, 1, 31-41.
- Holzinger, A. & Leitner, H. (2005) Lessons from Real-Life Usability Engineering in Hospital: From Software Usability to Total Workplace Usability. *Empowering Software Quality: How can Usability Engineering reach these goals*.
- Holzinger, A., Searle, G., Kleinberger, T., Seffah, A. & Javahery, H. (2008) Investigating Usability Metrics for the Design and Development of Applications for the Elderly. *Lecture Notes in Computer Science LNCS 5105*. Heidelberg, Springer.
- Houston, G. (2010), MochaUI - User interface framework. Online available: <http://www.mochau.com/>, last access: 2010-02-28:
- Jokela, T., Iivari, N., Matero, J. & Karukka, M. (2003) The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. *Proceedings of the Latin American conference on Human-computer interaction*. Rio de Janeiro, Brazil, ACM.
- json.org (2010), Introducing JSON. Online available: <http://www.json.org/>, last access: 2010-03-01:
- Kahn, M. & Prail, A. (1994) Formal usability inspections. *Usability inspection methods*. John Wiley & Sons, Inc., 141-171.
- Kalbach, J. & Bosenick, T. (2003) Web page layout: a comparison between left-and right-justified site navigation menus. *Journal of Digital Information*, 4, 1, 153-159.
- Kantner, L. (1994) Techniques for managing a usability test. *IEEE Transactions on Professional Communication*, 37, 3, 143-148.
- Kruse, C. (2007), Programmier technik: E-Mail-Kontrolle. Online available: <http://aktuell.de.selfhtml.org/artikel/programmier technik/email/>, last access: 2010-02-16:
- Mahadevan, B. (2000) Business models for Internet-based e-commerce: An anatomy. *California management review*, 42, 4, 55-69.
- Matera, M., Rizzo, F. & Carughi, G. (2006) Web usability: principles and evaluation methods. *Web Engineering*, 143-180.

- Mendes, E., Mosley, N. & Counsell, S. (2006) The Need for Web Engineering: an Introduction. *Web Engineering*, 1–27.
- Nielsen, J. (1992) Finding usability problems through heuristic evaluation. *Proceedings of the SIGCHI conference on Human factors in computing systems*. Monterey, California, United States, ACM New York, NY, USA.
- Nielsen, J. (1993) *Usability engineering*. Morgan Kaufmann.
- Nielsen, J. (1994) Usability inspection methods. *Conference companion on Human factors in computing systems*. Boston, Massachusetts, United States, ACM New York, NY, USA.
- Nielsen, J. (2005) Ten usability heuristics. Retrieved November, 29, 2005.
- Nielsen, J. & Landauer, T. K. (1993) A mathematical model of the finding of usability problems. *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*. Amsterdam, The Netherlands, ACM.
- Nielsen, J. & Molich, R. (1990) Heuristic evaluation of user interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*. Seattle, Washington, United States, ACM.
- Norman, D. A. (1999) Affordance, conventions, and design. *interactions*, 6, 3, 38-43.
- Normungsinstitut, Ö. (1998) Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten, Teil 11. 26.
- Offutt, J. (2002) Quality attributes of Web software applications. *IEEE software*, 25-32.
- Olsina, L., Covella, G. & Ross, G. (2006) *Web Quality*. Springer Berlin Heidelberg.
- OSI, O. S. I. (2010), The MIT License. Online available: <http://www.opensource.org/licenses/mit-license.php>, last access: 2010-02-28:
- PayPal (2010a), Online available: <http://www.paypal.com>, last access: 2010-02-19:
- PayPal (2010b), Micropayments. Online available: [https://www.paypal.com/IntegrationCenter/ic\\_micropayments.html](https://www.paypal.com/IntegrationCenter/ic_micropayments.html), last access: 2010-22-02:
- Polson, P. & Lewis, C. (1990) Theory-based design for easily learned interfaces. *Human-Computer Interaction*, 5, 2, 191-220.

- Polson, P., Lewis, C., Rieman, J. & Wharton, C. (1992) Cognitive walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of man-machine studies*, 36, 5, 741-773.
- Proietti, V. (2010), MooTools. Online available: <http://www.mootools.net/>, last access: 2010-02-28:
- Quesenbery, W. (2002) Dimensions of usability. *Content and Complexity, Erlbaum*.
- Quesenbery, W. (2004) Balancing the 5Es of Usability. *Cutter IT Journal*, 17, 2, 4-11.
- Rieman, J., Franzke, M. & Redmiles, D. (1995) Usability evaluation with the cognitive walkthrough. *Conference companion on Human factors in computing systems*. Denver, Colorado, United States, ACM.
- Spool, J. & Schroeder, W. (2001) Testing web sites: five users is nowhere near enough. *CHI '01 extended abstracts on Human factors in computing systems*. Seattle, Washington, ACM New York, NY, USA.
- Sullivan, R. (2007) Risk Management and nonbank participation in the US retail payments system. *ECONOMIC REVIEW-FEDERAL RESERVE BANK OF KANSAS CITY*, 92, 2, 5.
- Sumanjeet, S. (2009) EMERGENCE OF PAYMENT SYSTEMS IN THE AGE OF ELECTRONIC COMMERCE: THE STATE OF ART. *Asia Pacific Journal of Finance and Banking Research Vol*, 3, 3.
- Tullis, T., Fleischman, S., McNulty, M., Cianchette, C. & Bergel, M. (2002) An empirical comparison of lab and remote usability testing of Web sites.
- Tygar, J. D. (1998) Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce. *Proceedings of the 24rd International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc.
- Weiland, S. & von Gizycki, V. (2002) 3 Wahrnehmungspsychologische Erkenntnisse im Web-Design. *Usability: Nutzerfreundliches Web-design*, 33.
- Williams, D. (2007) *Pro PayPal E-Commerce*. Friends of ED.
- Wirtschaftskammer, Ö. (2006) Körperschaftssteuer (KöSt).
- Wixon, D. & Wilson, C. (1997) The usability engineering framework for product design and evaluation. *Handbook of human-computer interaction*, 2, 653-68.
- Wright, D. (2002) Comparative evaluation of electronic payment systems. *Infor-Information Systems and Operational Research*, 40, 1, 71-85.

