

# MASTER'S THESIS

---

## USABILITY TESTING OF MOBILE APPLICATIONS FOR CHILDREN

---

Ferdinand R. Knapitsch-Scarpatetti-Unterwegen  
Graz, October 15, 2012

Supervisor: Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Slany

*Institute for Software Technology  
Graz University of Technology*



# MASTERARBEIT

(DIESE ARBEIT IST IN ENGLISCHER SPRACHE VERFASST)

---

## USABILITY TESTEN VON MOBILEN ANWENDUNGEN FÜR KINDER

---

Ferdinand R. Knapitsch-Scarpattetti-Unterwegen  
Graz, October 15, 2012

Betreuer: Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Slany

*Institut für Softwaretechnologie  
Technische Universität Graz*



TO MY SISTER

# Abstract

The use of internet-enabled mobile phones is raising rapidly around the globe and this trend is not limited to adults. Children are increasingly confident users of mobile technologies, and the number of children with their own mobile phone is rapidly increasing in industrialised countries. Developing usable applications relies on knowledge in the field of human-computer interactions with the focus on user-centred design. The user-centred design process should take place during the whole development process and should be repeated iteratively. When designing user interfaces for children, children should be involved in the user-centred design process.

Despite this growth in the number of users and mobile phones, less is known about how children use mobile applications. Children as users of modern technologies challenge software developers and user interface designers in ways that are different from adults. This work concentrates on the fundamentals of the user-centred development process particularly with children, and furthermore on usability testing of mobile applications with children with the example of the Catroid on-device visual programming environment.

Several questions arose before the first usability tests of Catroid were performed with children: how to do usability testing when children are participating in such a test? What attention should be paid to address the needs of children during such a test? What challenges arise for supervisors of usability tests with children? The main goal included the requirement of obtaining feedback about the application and a global assessment of the system's usability.

## **Keywords**

Usability engineering, testing, Catroid, children, mobile computing, methodologies

# Kurzfassung

Die Verwendung von Internet-fähigen Mobiltelefonen auf der ganzen Welt steigt rasant an, und dieser Trend ist nicht nur auf Erwachsene beschränkt. Kinder werden immer sicherere Nutzer von mobilen Technologien, und die Zahl der Kinder mit eigenem Handy wächst in den industrialisierten Ländern stetig. Die Entwicklung benutzerfreundlicher Anwendungen beruht auf Wissen aus dem Gebiet der Mensch-Computer-Interaktion, mit Fokus auf Benutzer-zentriertem Design. Der benutzerzentrierte Designprozess sollte während des gesamten Entwicklungsprozesses stattfinden und iterativ wiederholt werden. Bei der Gestaltung von Benutzeroberflächen für Kinder sollten Kinder in den benutzerzentrierten Designprozess einbezogen werden.

Trotz dieses Wachstums der Zahl von Benutzern und Mobiltelefonen ist wenig darüber bekannt wie Kinder mobile Anwendungen benutzen. Kinder als Nutzer von modernen Technologien sind eine Herausforderung für Software-Entwickler und Designer von Benutzeroberflächen, denn die Anforderungen von Kindern unterscheiden sich von jenen der Erwachsenen. Diese Arbeit befasst sich mit den Grundlagen des Benutzer-zentrierten Entwicklungsprozesses im speziellen für Kinder und darüber hinaus mit dem Testen der Benutzbarkeitstauglichkeit von mobilen Anwendungen für Kinder anhand des Beispiels Catroid, einer visuellen Programmierumgebung für die Verwendung auf mobilen Geräten.

Bevor die ersten Usability-Tests mit Kindern an Catroid durchgeführt wurden, ergaben sich mehrere Fragestellungen: wie werden Usability-Tests durchgeführt, wenn Kinder daran beteiligt sind? Welches Augenmerk sollte auf die Bedürfnisse von Kindern in solchen Tests gelegt werden? Welche Herausforderungen ergeben sich für Moderatoren von Usability-Tests mit Kindern? Das Hauptziel beinhaltet das Feedback von Kindern über das Programm und die Gesamtbeurteilung der Benutzbarkeitstauglichkeit des Systems.

**Schlüsselwörter**

Usability Engineering, Testing, Catroid, Kinder, mobile Geräte, Methodologien

## **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, October 15, 2012

---

Ferdinand Knapitsch

# Acknowledgments

My sincere thanks goes to my parents for their patience and understanding during these last life forming years. It would not have been possible for me to finish my studies without the support of my parents.

I would also like to thank Wolfgang Slany for the chance to work on the awesome Catroid project which generated the possibility of doing the research into usability testing of mobile applications for children. His enthusiasm and encouragement for this research has been an important motivation for writing this thesis.

I would particularly like to thank Barbara Weingartner, Julia Bauernfeind, Gerd Felsberger, Stefan Mooslechner and Johannes Schaflechner for many motivating discussions and their support and encouragement during the numerous difficult moments I came upon during the past few years.

Last but not least, I would like to thank Daniel Burtscher and all other people who contributed directly or indirectly to help me complete this work.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Related Work</b>	<b>3</b>
2.1	Catroid . . . . .	3
2.1.1	Catroid Software Environment . . . . .	3
2.1.2	Catroid Community Website . . . . .	4
2.2	Scratch . . . . .	5
2.3	Other Visual Programming Environments . . . . .	5
2.3.1	GameMaker . . . . .	6
2.3.2	Kodu . . . . .	6
2.4	Mobile Usage Among Children and Teenagers . . . . .	8
2.5	Theoretical Background . . . . .	10
<b>3</b>	<b>Usability Engineering</b>	<b>11</b>
3.1	Definition of Usability . . . . .	11
3.1.1	Six Usability Attributes . . . . .	12
3.1.2	Measuring Usability . . . . .	13
3.2	Discoverability vs. Usability . . . . .	16
3.2.1	The Myth of Discoverability . . . . .	16
3.2.2	Why not Everything Can Be Discoverable . . . . .	17
3.2.3	How to Decide What to Make Discoverable? . . . . .	17
3.2.4	How to Make Something Discoverable? . . . . .	18
3.3	Usability Engineering . . . . .	18
3.3.1	Context of Use . . . . .	19
3.3.2	Specify the User and Organisational Requirements . . . . .	20
3.3.3	Production of Design Solutions . . . . .	21
3.3.4	Evaluate Designs Against Requirements . . . . .	21
3.4	Usability Engineering Lifecycle . . . . .	22
3.4.1	Know the User . . . . .	23
3.4.2	Usability Benchmarking . . . . .	25
3.4.3	Goal-Oriented Interaction Design . . . . .	27

---

3.4.4	Iterative Design . . . . .	29
3.4.4.1	Prototyping . . . . .	30
3.4.4.2	Formative and Summative Usability Evaluation . . . . .	32
3.4.5	Follow-Up Studies . . . . .	33
3.5	Usability Inspection Methods . . . . .	34
3.5.1	Heuristic Evaluation . . . . .	35
3.5.2	Cognitive Walkthrough . . . . .	39
3.5.3	Guideline Reviews: Checking and Scoring . . . . .	41
3.5.4	Action Analysis . . . . .	42
3.5.5	Severity Ratings . . . . .	44
<b>4</b>	<b>Usability Testing Methods</b>	<b>46</b>
4.1	Thinking-Aloud . . . . .	51
4.2	Constructive Interaction . . . . .	53
4.3	Formal Experiments . . . . .	54
4.4	Questionnaires and Interviews . . . . .	55
4.5	Observational Study . . . . .	56
<b>5</b>	<b>Practice Guidelines</b>	<b>57</b>
5.1	Practice Guidelines for User-centred Design . . . . .	57
5.2	Practice Guidelines for Touch-screen Devices . . . . .	59
<b>6</b>	<b>Characterising Children</b>	<b>61</b>
6.1	Preschool: Ages 2 to 5 . . . . .	61
6.2	Young Childhood: Ages 6 to 9 . . . . .	63
6.3	Tweens: Ages 10 to 14 . . . . .	64
6.4	Summary . . . . .	65
<b>7</b>	<b>Usability for Children</b>	<b>66</b>
7.1	Child-centred Design . . . . .	66
7.2	Usability Testing with Children . . . . .	68
7.2.1	Preparations . . . . .	68
7.2.2	Methods . . . . .	71
7.3	Guidelines for Testing with Children . . . . .	74
<b>8</b>	<b>Usability Test</b>	<b>76</b>
8.1	Introduction . . . . .	76
8.2	Catroid and Paintroid . . . . .	77
8.2.1	Catroid . . . . .	77
8.2.2	Paintroid . . . . .	77
8.3	Method . . . . .	77
8.4	Goals . . . . .	78

---

8.5	Setting up the Test . . . . .	79
8.5.1	Environment . . . . .	79
8.5.2	Team . . . . .	80
8.5.3	Equipment . . . . .	80
8.5.4	Participants . . . . .	82
8.5.5	Procedure . . . . .	82
8.5.6	Tasks . . . . .	84
8.6	Post-test Interview . . . . .	87
<b>9</b>	<b>Results and Recommendations</b>	<b>89</b>
9.1	Findings Regarding Usability . . . . .	89
9.1.1	Main Findings . . . . .	89
9.1.2	Positive Findings . . . . .	95
9.1.3	Negative Findings . . . . .	96
9.2	Findings Regarding Testing with Children . . . . .	101
9.3	System Usability Scale . . . . .	103
<b>10</b>	<b>Conclusions and Future Work</b>	<b>105</b>
	<b>Bibliography</b>	<b>107</b>

# List of Figures

2.1	Catroid . . . . .	4
2.2	GameMaker: A project in development . . . . .	6
2.3	A very early version of Kodu . . . . .	7
2.4	The new user interface of Kodu . . . . .	7
2.5	Mobile phone usage among children in Germany . . . . .	8
2.6	Worldwide mobile phone and internet usage among children and teenagers . . . . .	9
2.7	Android: Time spent on web vs. apps . . . . .	9
2.8	Available apps across major mobile platforms . . . . .	10
3.1	Nielsen’s model of attributes of system acceptability . . . . .	12
3.2	The four stages of the ISO user-centred design model . . . . .	19
3.3	Users’ experience differ in three dimensions . . . . .	23
3.4	Learning curve of a system . . . . .	24
3.5	An example of a usability target line . . . . .	26
3.6	Example persona: Angelika . . . . .	28
3.7	Example persona: Tobias . . . . .	29
3.8	Example persona: Silvia . . . . .	30
3.9	Working prototypes . . . . .	31
3.10	Evaluation methods . . . . .	33
3.11	Average proportion of usability problems found . . . . .	38
4.1	A simple single room usability test setup . . . . .	49
4.2	A dual room usability test setup . . . . .	49
4.3	An example of a portable usability kit . . . . .	50
4.4	Sample test setting for a thinking-aloud test . . . . .	52
6.1	Children’s growth rates . . . . .	62
7.1	Roles of children in a child-centred design process . . . . .	67
7.2	Usability problems found in a thinking-aloud test . . . . .	69
8.1	Test facilitator is observing two boys working on their tasks. . . . .	78

---

8.2	Usability test laboratory . . . . .	80
8.3	Usability test performed in a constructive interaction setting . . . . .	81
8.4	Boy and girl working together on test tasks . . . . .	83
8.5	Three different Catroid projects from the usability test . . . . .	87
9.1	Catroid: three properties tabs of an object . . . . .	90
9.2	Catroid: “New” buttons . . . . .	91
9.3	Catroid: “Set costume” procedure . . . . .	92
9.4	Catroid: “Set costume” brick empty . . . . .	92
9.5	Three ways how to start Paintroid . . . . .	93
9.6	Paintroid: tools menu . . . . .	93
9.7	Paintroid: pointer and stamp tool . . . . .	94
9.8	Paintroid: colour picker . . . . .	95
9.9	Paintroid: change shape and width . . . . .	95
9.10	Catroid: sound recorder . . . . .	96
9.11	Catroid: add bricks view . . . . .	97
9.12	Catroid: bricks tab on the tablet . . . . .	98
9.13	Catroid: delete a brick . . . . .	99
9.14	Catroid: delete bricks tab on the tablet . . . . .	99
9.15	Catroid: visibility of bricks . . . . .	100
9.16	Catroid: inconsistent or faulty bricks . . . . .	101
9.17	Catroid: more inconsistent or faulty bricks . . . . .	101

# List of Tables

3.1	Example questions for measuring subjective satisfaction using a Likert scale	15
3.2	Example semantic differential scales . . . . .	16
3.3	Nine common evaluation methods, classified according to type and purpose	34
3.4	Guidelines for user interface design in six functional areas of interaction . .	42
3.5	Average times for typical keystroke-level computer interface actions . . . . .	43
3.6	Five-Point Severity Scale . . . . .	45
7.1	Example rating scale with symbolic marks . . . . .	71
8.1	Specifications of test device . . . . .	81
8.2	Number of users and settings . . . . .	82
8.3	Summary of participating evaluators . . . . .	83

# Chapter 1

## Introduction

Computers have influenced the everyday-life of people from all over the world. Computers should help to make things more convenient. In the past few years, mobile devices have grown to be one of the most common consumer devices. Nowadays, mobile phones follow in the footsteps of personal computers. In connection with modern information technologies, mobile phones influence human lifestyle more and more. The mobile phones themselves have expanded in functionality. Mobile phones changed from devices that only dial numbers to smarter devices, which make personal phone directories, appointment calendars, cameras and gaming available any time and anywhere. Decreasing costs and the availability of internet flat rates, as well as the availability of thousands of mobile apps make mobile phones into a personal digital assistant and a constant companion. This frees people from their desktops and enables them to be mobile active users. Many applications and websites offer increasingly valuable instructional or educational content and entertainment for children and adolescents. Mainstream websites offer more and more dedicated sections for children, often to build brand loyalty from an early age Nielsen [2011*b*].

The use of internet-enabled mobile phones is increasing rapidly around the globe and this trend is not limited to adults [Slany 2012]. Children are increasingly confident users of mobile technologies. In industrialised countries the number of children with their own mobile phone is rapidly increasing. For example, in Germany in 2010 more than 50% of children aged between 6 and 13 years old had their own mobile phone and about 15% of these mobile phones were connected to the internet [Medienpädagogischer Forschungsverbund Südwest 2010]. Similar findings can be seen in other countries around the world. Nearly 70% of children surveyed in Japan, India, Egypt and Paraguay used a mobile phone in 2011. Approximately 40% of these children accessed the internet from their

mobile phones [GSM Association and the Mobile Society Research Institute within NTT DOCOMO Inc. 2011].

Despite this growth in number of users and mobile phones, little is known about how children use mobile applications. Children as users of modern technologies challenge software developers and user interface designers in ways that are different from adults.

*“A central tenet for user centred design practices is that there is no design that fits all, but rather design should be driven by knowledge of the target users.”*

[Markopoulos and Bekker 2003]

Developing usable applications relies on knowledge in the field of human-computer interactions with the focus on user-centred design. The user-centred design process should take place during the whole development process and should be repeated iteratively. When designing user interfaces for children, children should be involved in the user-centred design process.

The ideas for this thesis have arisen during my collaboration at Catroid, an on-device visual programming system for Android devices. This work will concentrate on user-centred design and usability testing on mobile applications with children. Several months before the striven Google Play release date of Catroid, the app’s usability was tested with children for six days in a row. The usability test pursued two main goals. The first goal included the question, how you do usability testing with children: how to do usability testing when children are participating in a usability test? What attention should be paid to address the needs of children during such a test? What challenges come up for supervisors of usability tests with children? The answers to these questions should be summarised into some guidelines for mobile testing with children. The second goal was to get feedback about the usability and potential runtime errors of the then current development state of the user interface, and how the concrete results could improve the usability of the Catroid user interface.

*“This work should be used as a kind of an information and instruction set for usability engineering of mobile devices for children and teenagers. The content of this work should represent a set of guidelines for future usability engineering processes, which will be done by the members of the usability team of the Catroid programming community.”*



## Chapter 2

# Related Work

### 2.1 Catroid

Catroid<sup>1</sup> is a free and open source programming environment for Android<sup>2</sup> mobile devices, with an additional image manipulating program and website. The application allows casual and first-time users, starting from the age of eight, to create interactive content such as multimedia animations, games or small learning programs directly on their mobile devices without requiring previous knowledge of software development. Catroid supports and encourages childrens imagination and creativity intensely, and makes use of the fascination that computer games may cause. The aim of this software is to empower children to encounter their first programming experiences. By that, children acquire exciting and useful knowledge in a playful way. Since imagination is unlimited, the system presents a children's playground with endless possibilities which is always interesting. It is not about promoting games on the computer, but the how children learn to create computer games creatively and independently [Slany 2012].

#### 2.1.1 Catroid Software Environment

Catroid is an on-device visual programming language which runs on Android mobile devices, is inspired by the Scratch programming language, a visual programming language for children developed by the Lifelong Kindergarten Group at MIT Media Lab. Like in Scratch or Google App Inventor<sup>3</sup>, Catroid programs are written in a graphical -style. This makes it easy to create interactive content without the children needing any programming

---

<sup>1</sup><http://developer.catrobat.org>, last visited on September 28th 2012

<sup>2</sup><http://www.android.com>, last visited on March 16th 2012

<sup>3</sup><http://appinventor.mit.edu>, last visited on October 10th 2012

skills or experience. While Catroid applications can be created simply by using an Android device and therefore support available hardware sensors, developing with Scratch or App Inventor needs the use of a personal computer that does not support any hardware sensor.

The software environment consists of two parts:

- Programming environment: Catroid
- Image manipulating program: Paintroid<sup>4</sup>.

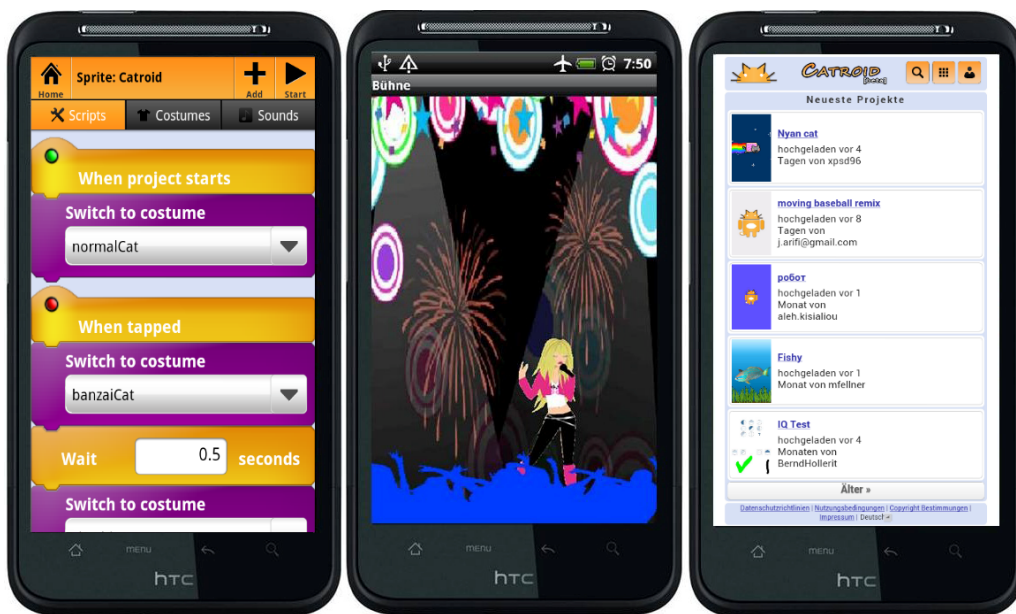


Figure 2.1: The Catroid app used for the first usability tests. This is an early version of Catroid. Left: A project in development on the phone. Center: Execution of a Hannah Montana interactive music video animation created by children (remixed from an original Scratch project<sup>5</sup>). Right: Projects on the community website. Images originally published by [Slany 2012].

### 2.1.2 Catroid Community Website

The Catroid community website<sup>6</sup> supports an online community of users of all ages. Remixing was a core idea behind the Scratch online community. So, everyone can download and edit every project from the website, add new things or change the behaviour and

<sup>4</sup><http://developer.catrobat.org>, last visited on September 28th 2012

<sup>5</sup><http://scratch.mit.edu/users/tyster>, last visited on October 14th 2012

<sup>6</sup><http://www.catroid.org>, last visited on October 14th 2012

upload the project again. Uploaded projects are open source and are published under a free software license which is by default restricted to non-commercial use. Each program published on the community website can be downloaded, viewed, modified or rebuilt and certainly uploaded again as a newly modified version of the program.

## 2.2 Scratch

Scratch<sup>7</sup> is a networked, media-rich programming environment designed to enhance the development of technological fluency at after-school centres in economically-disadvantaged communities developed by the Lifelong Kindergarden Group at MIT Media Lab. Scratch is available for desktop computers and is intended to motivate beginners, particularly children and teenagers, to become familiar with programming concepts in a playful and experimental way. Using the motto “imagine, program, share”, young users can create interactive content (projects) such as multimedia animations, games or small learning programs. “Scratch adds programmability to the media-rich and network-based activities that are most popular among young people at afterschool computer centres.” [Maloney et al. 2004].

Scratch consists of two parts: a programming environment and a community website. The website offers an example of how children can use the Web as a platform for learning, enabling children to create and remix and share personally meaningful content and not simply access information [Monroy-Hernández and Resnick 2008].

## 2.3 Other Visual Programming Environments

Beside Scratch some other programming environments exist for building interactive content and/or games with an educational context for children. GameMaker<sup>8</sup> and Kodu<sup>9</sup> are two examples which are also connected to a community. Both programming environments are used in an educational curriculum and are rated by teachers and users as great products.

---

<sup>7</sup><http://scratch.mit.edu>, last visited on October 14th 2012

<sup>8</sup><http://www.yoyogames.com/gamemaker/studio>, last visited on October 14th 2012

<sup>9</sup><http://fuse.microsoft.com/page/kodu>, last visited on October 14th 2012

### 2.3.1 GameMaker

GameMaker was created to support the founders' belief that a new generation of games development talent and devices was arising. GameMaker is a rapid-application-development tool for children and young people to use at home, in schools and in Universities worldwide. It allows the development of two-dimensional and isometric games, just by using drag-and-drop techniques. Students like GameMaker because it is easy to use and teachers state it as a great product to implement problem-based, project-based and inquiry-based learning into the educational curriculum. The community website allows users to upload, share, and play the games they have created using GameMaker. [YoYo-Games Ltd. 2012; Overmars 2004]. "There is absolutely no better learning motivation for our youth than providing them with tools that they are familiar with in their personal lives." Shanna Falgoust, Officer, Special Interest Group Games and Simulations (SIGGS), International Society for Technology in Education (ISTE)



Figure 2.2: GameMaker: A project in development [YoYo-Games Ltd. 2012].

### 2.3.2 Kodu

Kodu is a simple visual programming language for PC and Xbox, which is developed by Microsoft. As [MacLaurin 2011] writes, "Kodu was initially inspired by the relatively easy access to programming - through a ROM-based BASIC interpreter - provided with early 1980s personal computers." Kodu's developers wanted to enable an easily accessible, creatively powerful programming experience for children, hoping to provide a benefit in the cognitive development of children and to eventually increase the overall size and quality of the programmer community. Kodu lets kids create their own games on the PC and Xbox

via a simple visual programming language allowing to share these games with others in the Kodu community. Kodu can be used to teach creativity, problem solving, storytelling, as well as programming. Anyone can use Kodu to make a game, children as well as adults without any design or programming skills [Microsoft 2012b; Microsoft 2012a]. The Microsoft Research team started an after school program with the non-profit organisation Girls Inc. and the University of Santa Barbara. The main goal was to see how Kodu helps young users develop their capabilities in science, maths, logic and problem solving. As USATODAY.com [2012] writes, the feedback from the children was incredible and they loved to use it to build their own games.

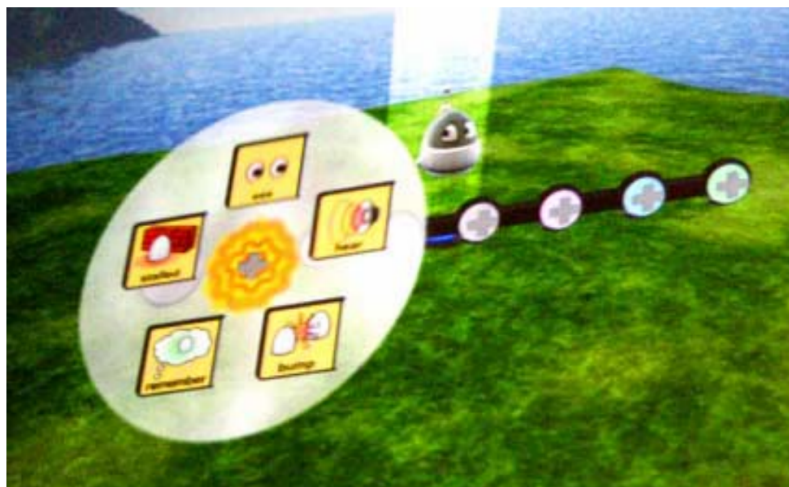


Figure 2.3: A very early version of Kodu [MacLaurin 2011].

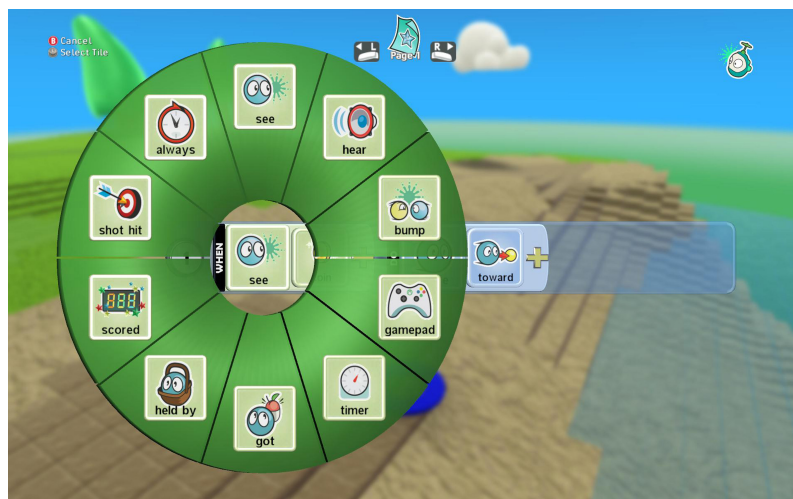


Figure 2.4: The new user interface of Kodu [Microsoft 2012a].

## 2.4 Mobile Usage Among Children and Teenagers

The use of internet-enabled mobile phones by children is spreading rapidly around the globe. Children are becoming increasingly confident and passionate users of mobile technologies everywhere in the world. In industrialised countries the number of children having their own mobile phone is increasing fast. For example, mobile phone usage among children in Germany between the ages of six and seven is more than 14% (Figure 2.5). In the years 2010 and 2011, the usage of mobile phones by children in Germany between the ages of six and thirteen was higher than 50%, in Egypt and Korea approximately 90%, while in China, the worlds fastest growing market, the rate was only about 42%. As seen in Figure 2.6, the rate of internet use on a mobile phone is 70% in Japan, 54% in Egypt, 36% in China, 15% in Germany and the average rate out of the 9 countries surveyed is 30%.

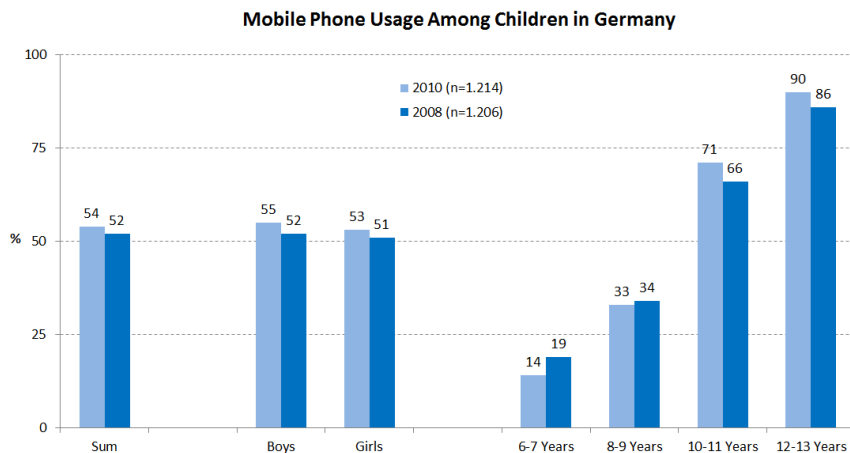


Figure 2.5: Mobile phone usage among children in Germany between the ages of 6 and 13 years. [Medienpädagogischer Forschungsverbund Südwest 2010]

On average, teenagers have about 24 apps installed on their mobile phone, while the most important apps connect the youngsters with social networks or are games. Children use their mobile phones for other popular activities, such as making phone calls, writing short messages, taking photos or recording and watching videos.

Figure 2.7 represents the result of a study into the latest Android usage trends in the

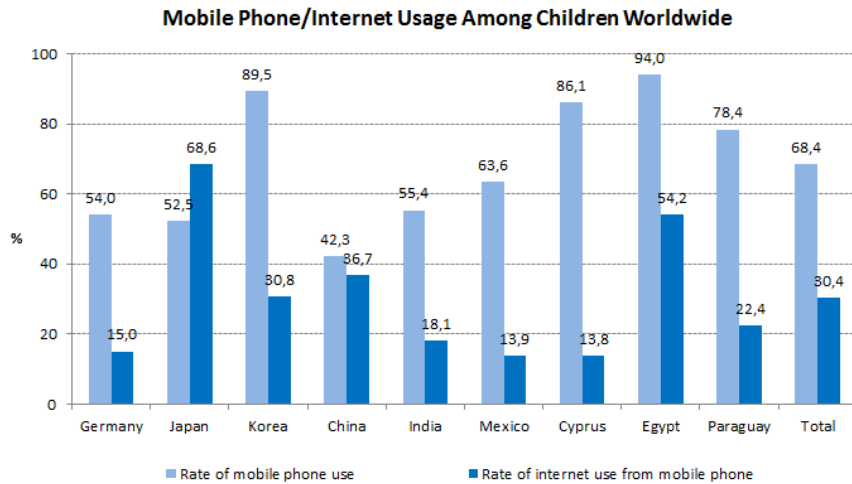


Figure 2.6: Worldwide mobile phone and internet usage among children and teenagers between the ages of 8 and 18 years. [GSM Association and Mobile Society Research Institute within NTT DOCOMO Inc. 2010, GSM Association and the Mobile Society Research Institute within NTT DOCOMO Inc. 2011, Medienpädagogischer Forschungsverbund Südwest 2010]

US. Mobile apps beat the mobile web with 66% against 33% among the US Android smartphone users.

**Proportion of Time Spent on Web vs. Apps**

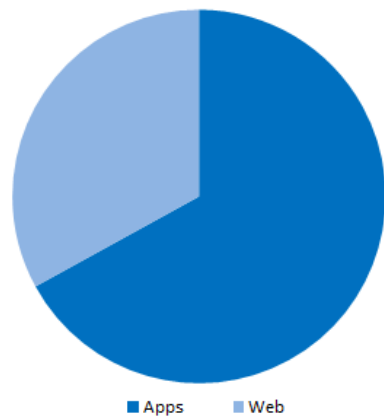


Figure 2.7: Proportion of time spent on web vs. apps, Nielsen smartphone analytics, June 2011. [Nielsen 2011a]

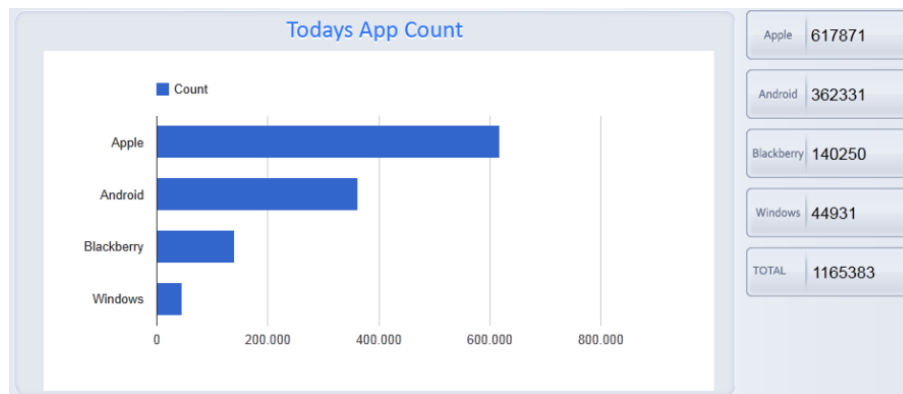


Figure 2.8: Available apps across major mobile platforms topped the million-app milestone. [Mobilewalla 2011]

The amount of mobile apps is growing rapidly and the number of available apps across all major mobile platforms has topped the milion-app milestone (see Figure 2.8).

## 2.5 Theoretical Background

Chapters 3 to 7 include a discussion of the theoretical background with regards to this thesis. The following content will be discussed in more detail in the various chapters:

- Chapter 3: Usability Engineering
- Chapter 4: Usability Testing Methods
- Chapter 5: Practice Guidelines for User-centred Design and for Finger-touch Devices
- Chapter 6: Characterizing Children between two and 14 years old
- Chapter 7: Usability for Children



## Chapter 3

# Usability Engineering

Usability engineering refers to the research and iterative design process used to improve the usability of a human-computer interface, and is located in the field of human-computer interaction.

*“Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.”* [Hewett et al. 1992].

### 3.1 Definition of Usability

The ISO defines usability as:

*“Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”* [ISO 1998].

Thus, there are three measurable usability attributes stated by ISO [1998]:

- Effectiveness: the accuracy and completeness with which a user achieves specified goals.
- Efficiency: the resources expended in relation to the accuracy and completeness with which a user achieves goals.
- Satisfaction: freedom from discomfort, and positive attitudes towards the use of the product.

Nielsen [1993] defines usability as a quality attribute, which is how easy user interfaces or systems are to use. As shown in Figure 3.1, Nielsen describes usability in the context of overall system acceptability. Given that a system is socially accepted, the system’s practical acceptability can be analysed within various categories, including traditional categories; cost, support, reliability, compatibility, etc., as well as the category of usefulness. Breaking down usefulness, as whether the system can be used to accomplish a desired goal, into the categories of utility and usability, utility stands for the design’s functionality and whether the functionality of the system can do what a user needs and usability stands for how well a user can use the system’s functionality. Usability itself can again be broken down into multiple measurable quality attributes.

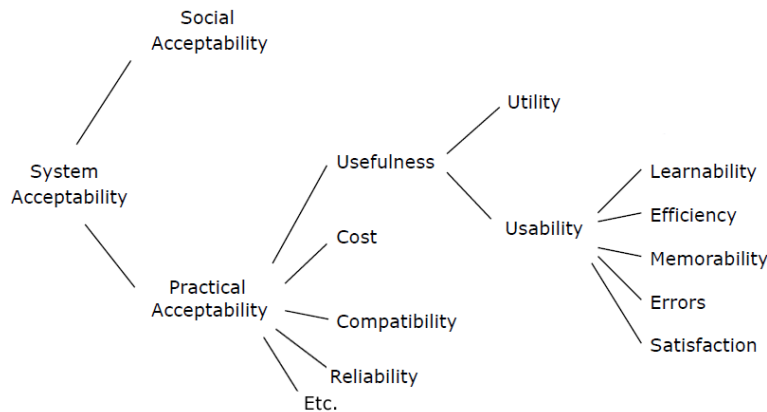


Figure 3.1: Nielsen’s model of attributes of system acceptability [Nielsen 1993].

For Nielsen [1993], both, usability and utility, are equally important because it does not matter if something is easy to use if it does not do what the user wants. Although it is also a problem if the system does what the user wants, but the user cannot interact with it because the user interface is too complicated. This model makes clear that usability has to be traded off against many other aspects in a development project. As Nielsen [2011c] says, user research methods that improve usability can also be used to study a design’s utility.

### 3.1.1 Six Usability Attributes

As seen in Figure 3.1, Nielsen defines usability by five quality parameters. Thus, by combining Nielsen’s usability attributes with the ISO usability attributes, six usability attributes are created [Andrews 2011]:

- **Effectiveness:** How precisely can a user achieve a specified goal?

- **Ease of learning:** How fast can a user start to accomplish some basic tasks from the first use of the system?
- **Efficiency of use:** Is a high level of productivity possible, once a user has learned the system?
- **Memorability:** Is a user able to return to the system after a period of not using it, without learning it all over again?
- **Errors:** How often does a user make errors while using the system, how serious are these errors and how does a user recover from them?
- **Subjective Satisfaction:** How enjoyable is the system to use and does the user like using the system?

### 3.1.2 Measuring Usability

Usability can either be measured by having a number of selected test users as representatives of the intended users of the system as possible, use the system to perform a predefined set of tasks. Or it can be measured by having real users perform tasks they normally do [Nielsen 1993].

#### **Effectiveness**

It is necessary to produce a specification for the criteria of successful goal achievement. Accuracy is measured by the extent to which the quality of the output corresponds to the specified criteria, and completeness as a proportion of the output that has been achieved.

#### **Learnability**

Learnability refers to how long it takes a novice user to complete a certain task successfully. It can be measured by collecting separate measurements from totally novice users of the system, who have no computer experience and from users with some general computer experience. When analysing learnability, users do not fully learn a whole user interface before they start using it, but start using a system as soon as they have learned part of it. Because users often jump in and start using a system, one should not just measure how long it takes to achieve mastery of a system but also how long it takes to achieve a sufficient level of proficiency to do useful work.

**Efficiency**

Efficiency refers to the expert user's level of performance at the time when the user's learning curve of a system flattens out. To measure efficiency of use in experienced users, access to experienced users is required. Experience can for example be defined in the number of hours spent using a system. This definition is often used in tests with new systems without an appropriate user pool. Therefore, test users are brought in and asked to use the system for a certain number of hours. After these users have become acquainted with the system, the time it takes the users to perform typical tasks is measured. Another way of measuring efficiency: is to decide on a definition of expertise, get representative users with that expertise (this is sometimes quite difficult) and measure the time it takes the users to perform typical tasks.

**Memorability**

Casual users represent the third major group of users beside novices and experts, and can be described as people who intermittently use a system. In contrast to novice users, casual users have used a system before and do not need to learn it from scratch. Casual users remember how to use a system from their previous experience. An interface that is easy to remember is very important for users who return after some time from vacation or some other reason for temporarily not using a system. Improvements in learnability often also make an interface easier to remember. There are two main ways of measuring memorability: one is a standard performance test with casual users, who have been away from the system for a certain time, and measure the time to perform typical tasks. Another is a memory test after users have finished a test session with the system, where one asks them to explain various commands or to remember the command's name or icon for a specific task. The performance test is most representative of the reason that modern user interfaces are built on the principle of making as much as possible visible to the users. Users of such systems do not need to remember what is available, since the system will remind them when necessary. A study of such interfaces showed that users were not able to remember names or icons from the menus after they were away from the system, but could interact with the same menus with no problems when they were back in front of the system [Nielsen 1993; quoted Mayes et al. 1988].

## Errors

Users should make as few errors as possible when using a system. An error is defined as any action that does not accomplish a desired goal. A system's error rate is measured by counting minor and catastrophic errors made by users while performing a specific task. Some errors are corrected immediately by the user, have no subsequent consequences beside to slow down the user's transaction rate somewhat, and need not to be counted, as their effect is measured in the efficiency of use in terms of the user's transaction time.

## Satisfaction

Subjective satisfaction can be measured by asking the users for their subjective opinion. Replies from any single user are subjective, but multiple users' replies combined can produce an averaged satisfaction level of the system. A subjective satisfaction usability attribute shows whether users like a system or not. It seems to be an appropriate way to measure it by simply asking the users, and this is done in an overwhelming number of usability studies. To ensure effective and consistent measurements, subjective satisfaction is normally measured by a short questionnaire that is given to users after a user test. For new systems it is important not to ask the users for their subjective opinions until they have tried to use the system for a real task. Subjective satisfaction questionnaires are typically very short and users are often asked to rate a system on 1-5 or 1-7 rating scales. The scales are normally either Likert scales or semantic differential scales [Nielsen 1993; quoted LaLomia and Sidowski 1990]. A Likert scale asks the users to rate their degree of agreement with a statement (e.g., "Using this system was a very frustrating experience."). A 1-5 rating scale normally is grouped into 1 = strongly disagree, 2 = partly disagree, 3 = neither agree nor disagree, 4 = partly agree, and 5 = strongly agree (see Table 3.1). A semantic differential scale lists opposite terms along some dimension and asks the users to place their impressions of the system on the most appropriate rating along this dimension (see Table 3.2).

*Please indicate a scale to which you agree or disagree with the following statements about Catroid:*

"It was very easy to find out how to select a sound file for a brick."

"With Catroid I can do all the things I think I would need."

"Catroid is very pleasant to work with."

Table 3.1: Example questions for measuring subjective satisfaction using a Likert scale. Users would indicate a 1-5 scale for each statement. [Nielsen 1993]

*Please mark the positions that reflect your impressions of Catroid:*

Complete	-----	Incomplete
Simple	-----	Complicated
Pleasing	-----	Irritating
Readable	-----	Unreadable
Clear	-----	Unclear

Table 3.2: Example semantic differential scales to measure subjective satisfaction. “See [Coleman et al. 1985] for a list of 17 such scales.” [Nielsen 1993]

## 3.2 Discoverability vs. Usability

Rick Osborne explains the distinction between Usability and Discoverability as follows:

*“Discoverability is about the first time a user does something, and usability is about doing that thing over and over again.”* [Osborne 2007]

One true advantage of graphical user interfaces was that commands did not have to be memorised anymore and actions in the interface design could be discovered through a systematic exploration of menus and buttons [Norman and Nielsen 2012].

Users are forced to locate features and discover functionality. Discoverability is the ability of users of a user interface to locate things they need, in order to complete several tasks. But the drawback of discoverability is that not everything can be discoverable in a user interface design [Berkun 2003].

### 3.2.1 The Myth of Discoverability

Scott Berkun explained the myth of discoverability in Berkun [2003] with two statements: the belief of the core myth is that good user interfaces make things utterly and extremely discoverable. Any design that makes a feature less than extremely discoverable, cannot be a good interface design and should be thrown away. The result that is applicable for large teams is that instead of a truly good design, people in a team often will accept a design that makes features they care about (because they work on them, they are new, etc.) discoverable, regardless of the importance compared to other features. Berkun mentioned that this result is often applied without knowing the core myth.

### 3.2.2 Why not Everything Can Be Discoverable

Every feature in an application can be made discoverable by adding a button for each function on a toolbar. This approach was used by MS Office in the versions before 2003 [Osborne 2007]. But, all things cannot be made easily discoverable because of screen real estate, the users' attention span, and the abilities to perceive and understand things are limited. All in all, when designing interfaces, tradeoffs must be made and priorities must be set to get the opportunity of a good outcome for the users. Giving everything in an interface design the same attention would lead only to a mediocre interface design, and it would force users to make choices of prioritization. People are generally not so interested in making such choices, but are happy that whoever designed a product has made basic choices in their interest.

Anyone who builds things that others will use has to pay attention to the prioritisation of which things need to be discoverable first. Of course, this proves to be difficult and experts try everything to avoid such kinds of considerations. Many mediocre design solutions arise through the avoidance of tough decisions, instead of an inability to design well [Berkun 2003].

### 3.2.3 How to Decide What to Make Discoverable?

Furthermore Scott Berkun explained "*The golden rule for what to make discoverable*" as the following:

- Things that most users do, most often, should be prioritized first.
- Things that some users do, and somewhat often, should come second.
- And, things few users do, rarely, should be prioritized last.

Depending on what will be designed, and for whom it will be designed for, these categories might have a deeper prioritisation regarding the relative importance of tasks and features, with different mixed categories and orders.

#### Exceptions to the rule

- **Start up tasks:** login, registration, application installation, etc. Even though these things are not done frequently, they may score high in the priority list.
- **Some features might be needed in case of emergencies:** No pilot wants to use the eject button, but they should not have to look for it when they need it.

- **Complex systems may require more complex considerations:** The more complex the usage pattern of an application is, the more thinking and iterations about the design may be required.
- **Diverse user groups:** sophisticated applications may have specific types of usage patterns associated with several user roles. Different things are discoverable for different types of users.

### 3.2.4 How to Make Something Discoverable?

As Scott Berkun argues, Discoverability is an important link in the chain of a design process. In order to complete certain tasks, users have to locate commands or links in question. Interface designers have several resources for emphasizing things, and trying to draw attention towards to them:

- **Real estate:** There is only a certain amount of pixels that can be used. Big targets are easier targets and are mostly located first and are easier to find again.
- **Order:** Things are placed in specific orders, that might form patterns that users can learn to follow.
- **Form:** Colour, font, shape, shadow, composition, and several other graphic design tools can help to make effective use of the given real estate.
- **Expectation and Flow:** Things can be put into forms or patterns that are somehow familiar to users. It is possible to emulate the design of another product, or from something in the real world.
- **Consistency:** Using the screen in consistent ways can teach people to find certain commands or items in certain places. Being predictable can often be assisted by using proven conventions, and making use of knowledge that users already have.

## 3.3 Usability Engineering

As explained at the beginning of the chapter, usability engineering is a method of improving the usability of a human-computer interface. Traditional approaches of the user-centred design process propose to evaluate the functionality of a product at the end of the development process. But, modern approaches disagree and say that usability engineering is not just a single action that solves usability problems before the release of a product.



The ISO standard 13407 [ISO 1999] provides a guideline for the modern user-centred design process for computer-based interactive systems. User-centred design is a multidisciplinary activity with a knowledge of the human factors, ergonomic knowledge and techniques included. The use of this knowledge when designing interactive systems can increase effectiveness and efficiency of systems. User-oriented systems facilitate users and motivate them to learn. In iterative methods for the design process, the feedback from users is a critical source of information. Iterations allow stepwise validation of preliminary design solutions in the “real-world”. The standard presents four essential user-centred design activities that should take place during a system development project. The user-centred design approach should start at the beginning of a project, and be repeated iteratively during the whole development process. The four activities of the ISO 13407 user-centred design model are illustrated in Figure 3.2.

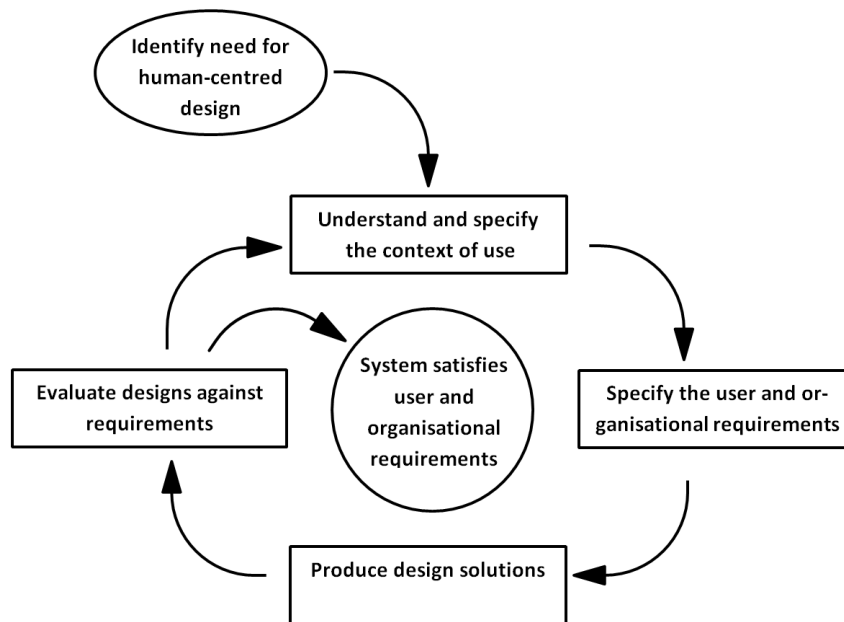


Figure 3.2: The four stages of the ISO standard 13407 user-centred design model are repeated iteratively until the product satisfies the specified requirements. [ISO 1999]

The description of the four stages of the ISO user-centred design model in the following sections are a summary of the ISO standard 13407.

### 3.3.1 Context of Use

The first stage of this design model is to understand and specify the context of use. A detailed understanding of the context allows early design decisions and provides a basis

for evaluation. The context of use is defined by three main factors:

#### **Characteristics of the intended users**

Important characteristics of users are knowledge, skills, experience, training, exercises, physical characteristics, preferences and abilities.

#### **Tasks to be performed by users**

Tasks should not only be described in terms of the functions or features provided by a system. The description of the tasks should contain the following items: overall goals of the tasks, characteristics that influence usability, impacts on health and safety, and the distribution of activities and tasks between human and technical resources.

#### **The environment in which users use the system**

The environment includes the hardware, software and materials used. Relevant characteristics of the physical and social environment should also be described.

### **3.3.2 Specify the User and Organisational Requirements**

Most design processes have major activities, which specify functional and other requirements. In the user-oriented design, this activity should be extended to provide explicit user requirements and organisational requirements. The ISO standard recommends the following aspects:

- The required performance of a new system in terms of functional and financial goals.
- Legal requirements, including health and safety.
- Cooperation and information exchange between users and other stakeholders involved.
- The user tasks, including allocation of tasks, the user satisfaction and motivation.
- The performance of the tasks.
- Work design and organization.
- Change management, training and persons involved.
- Feasibility of operation and maintenance.

- Human-computer interface and workplace.

The requirements are used to derive user and organisational requirements and to set objectives with appropriate trade-offs between different requirements. The requirements should be formulated in a form that enables subsequent testing to be possible.

### **3.3.3 Production of Design Solutions**

Design solutions should be produced with state of the art technologies. Scientific knowledge and theory from ergonomics, psychology, cognitive science, product design and other relevant disciplines can be used as input for potential design solutions. Organisational internal user interface style guides, product knowledge and marketing information can also provide useful information.

Simulations, models, mock-ups and other forms of prototypes are good tools for an effective communication with users and inside the project team. Users can be involved very early in the design process. Prototypes can make design decisions more explicit and allow the exploration of several design concepts before decisions are made. User feedback can influence the design early in the development process, and evaluation of designs over several iterations can enhance the quality and completeness of the functional design specifications. Simple prototypes can be produced very early in the design process and can then be very valuable for exploring alternative solutions. It is important to make design solutions as realistic as possible but it is also important not to invest too much time or money. If it is impractical to get feedback from users early in the design process, evaluations can also be conducted by experts. However, expert reviews should not replace user testing. User comments and reviews of prototypes can offer important information for design changes which can improve system usability, or can help to refine the scope and purpose of an interactive system.

### **3.3.4 Evaluate Designs Against Requirements**

The evaluation stage is a major step in user-centred design. Evaluations deliver formative feedback to improve a design, or summative feedback to assess whether user and organisational requirements have been achieved. Early in the design process, formative feedback guides design decisions, while summative feedback is only possible if a realistic prototype is available. Changes are relatively inexpensive in early stages but become more expensive the more a system is defined. Therefore it is important to start evaluation as early as

possible. The evaluation techniques used depend on the environment in which the evaluation is conducted, financial and time constraints as well as the stage of the development cycle and the nature of the system itself. Expert evaluation is possibly faster and cheaper and will identify major problems but must not replace user-based evaluations. User-based evaluation can provide feedback at any stage of design. While user-based evaluations in the early stages are restricted to scenarios, simple paper mock-ups or partial prototypes, evaluations on more developed design solutions are based on progressively more complete and concrete versions of a system.

Field validation means testing the final system for conformance with the requirements of the users, the tasks and the environment. The main techniques suggested are help desk data, field reports, real user feedback, performance data, reports of health impacts, design improvements, and requests for changes.

Long-term monitoring is the collection of different user inputs over a period of time. Some effects of working are not recognisable until the system has been in use for a period of time. Long-term monitoring is also useful for seeing if performance requirements have met.

### 3.4 Usability Engineering Lifecycle

Traditional approaches propose to evaluate the functionality of a product at the end of the development process. But, usability engineering is not just a single action that solves usability problems before the release of a product. It should ideally be part of the whole development phase and start as early as possible in the stages before design of the user interface has begun. The earlier usability actions are performed the less expensive is it to detect and correct usability problems.

In the following, an overview of a summarized usability engineering lifecycle is illustrated. This usability lifecycle is taken from Andrews [2011] and based on the usability engineering lifecycle published in Nielsen [1993]. It will be discussed in more detail in the next sections.

- Know the User
- Usability Benchmarking
- Goal-Oriented Interaction Design
- Iterative Design:

- Prototyping
- Formative Usability Evaluation
- Summative Usability Evaluation
- Follow-Up Studies

### 3.4.1 Know the User

Know the user, which is the first stage of the usability engineering lifecycle, is the process that identifies the characteristics of future users and use of the product. The future user group should include everybody whose work may be affected by the product, not only the people who sit at the keyboard. This user group should also include the support staff as well as the users of the system's end product or output even if they never see the system itself. Although "know the user" is the basic of all usability guidelines, it is very difficult for developers to gain access to users.

Users can be classified in several ways: work experience, educational level, age, previous computer experience, and more. As described by Nielsen [1993], there are three main dimensions on which users' experience differ: experience of computers in general, understanding of the task domain, and expertise in using the specific system. See Figure 3.3 for an illustration of the three dimensions.

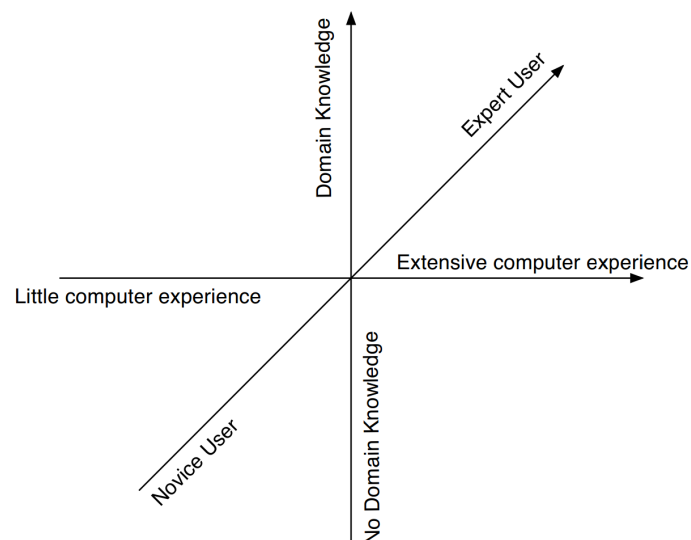


Figure 3.3: Users' experience differ in three dimensions: computer experience, system experience, and understanding of the task domain. [Nielsen 1993]

Systems need to be easy to learn. According to the learning curves of Nielsen [1993] some systems are designed to focus on learnability. Others focus on efficiency for highly-skilled users (see Figure 3.4). A system designed to be easy to learn, can slow down expert users. Easy to learn systems have a steep learning curve at the beginning and may allow users to reach a higher level of usage skills within a short time. Most user interfaces have learning curves that start out with zero efficiency of the user at time zero. Systems also exist that need to have literally zero learning time. Museum information systems or ticket machines are such systems, that are intended to be used once and therefore must allow users to use a system from the first contact. Some systems support both user groups, and offer an easy to learn system with an “expert mode” included. Useful accelerators such as more complicated menus which include extended functionality for faster access of system functions or a command line functionality can increase efficiency for proficient users. Such a system could hide expert-features at the beginning for novice users. If usability trade-off seems necessary, Nielsen recommends trying to find a win-win solution for both requirements. If this is not possible, the project’s usability goals, which should define the most important usability attributes of the specific circumstances of the project, should be resolved.

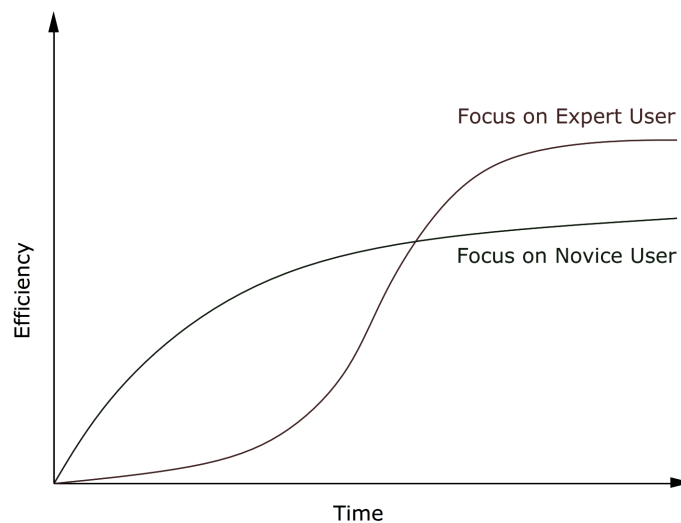


Figure 3.4: Learning curve: a system that is easy to learn may be less efficient for novice users to use. A system that is highly efficient for experts to use may be hard to learn. A well-designed system rides the best parts of both learning curves. [Nielsen 1993]

Information needed to characterize individual user characteristics and the domain of usage can be taken from market analysis or from observational studies which can be part of the task analysis. Such information can also come directly from questionnaires or inter-

views. Nielsen recommends to not rely totally on written information, but also to obtain new insights by observing and interviewing actual users in their working environment.

The next essential step for getting an early input into system design is a task analysis to find out the user's overall goals, how users currently approach a task, the users' information needs, and how they deal with unexpected circumstances. The users' task model should be studied to identify metaphors for the user interface. Weaknesses of the current situation, such as points where users actually fail to achieve goals, spent too much time, or are made uncomfortable, are all opportunities for improvements in a new product. The outcome of a task analysis will provide information about the users' goals, the steps that need to be performed, the dependencies between these steps, the outcome and reports needed to be produced, the criteria used to define the quality and acceptability of the outcome and the communication needs of the users as they exchange information.

The underlying functional reason for a task should be analysed. Understanding the functional reasons for the users' goal can help to improve the ways doing things. The analysis' result identifies what really needs to be changed to reveal a better way of achieving the users' goal. When users use the system they will not stay the same. As users change they will use the system in new ways. A very typical change is that users want to use interaction shortcuts when they become more proficient users after some time.

Nielsen [1993] presents various ways of getting to "know the user", but no optimal solution exists for this problem. Good ways of getting insights and a greater understanding of the users' tasks are, interviewing or observing the future users, the future users' domain, and the users' clients.

### 3.4.2 Usability Benchmarking

Usability benchmarking deals with the heuristic and empiric analysis of competing state of the art products or interfaces, measurable usability targets, and the return on investment of usability activities [Andrews 2011; Nielsen 1993].

*"Some usability evaluation is always better than none."* [Nielsen 1993]

Current state of the art competitor products are often the best prototypes for new systems. Nielsen notes, that competitive analysing of existing products does not imply stealing other copyrighted user interface designs. Existing products can be analysed heuristically against established usability guidelines with usability inspection methods and empirically by performing user tests. As all usability aspects cannot be equally weighted in a project,

they have to be prioritised. These priorities have to be made clear on the basis of the user and task analysis. It is important to specify the usability targets, which then allows the definition and specification of how much better the new product should be in terms of measured usability.

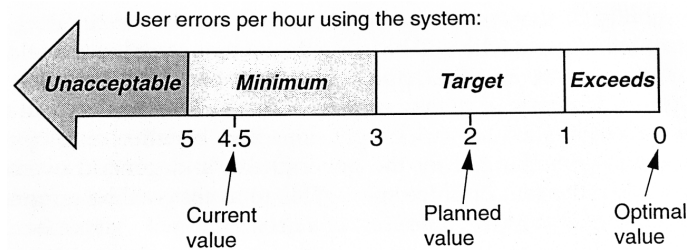


Figure 3.5: An example of an usability goal line [Nielsen 1993].

Figure 3.5 shows an example of an usability goal line. The number of user errors per hour is counted. The current system performs at an average error rate of 4.5 errors per hour, and the planned number of errors is 2.0 per hour. If the new system is measured at a error rate between 1.0 and 3.0, it will be treated as a target in terms of the desired usability goal. A performance of less than 3.0 would be an alarm signal that the target was not met.

The ease of use of a system does not only come from thinking about it. Furthermore it comes from the performance of systematic usability engineering activities throughout all project phases. As Aaron [2004] writes, usability may increase customer satisfaction and productivity, as well as possibly leading to customer trust and loyalty, and may have cost savings and profitability effects. Key design decisions made in the first 10% of the design process, can determine 90% of the product's cost and project performance. Aaron [2004] lists a number of key usability benefits and appropriate value propositions. One statistic presents the cost-benefit ratio for usability. This is \$1:\$10-\$100 and means that once a system is in development, the costs for correcting problems are 10 times as much if the same problem had been fixed in the design phase, and 100 times as much relative to fixing it in the early design stages. Nielsen [2003] recommends spending about 10% of a project's budget on usability activities and to double this value to 20% and more for optimized return of investment results.



### 3.4.3 Goal-Oriented Interaction Design

Designing of software should be based on an understanding of the user's goals. Let's start this section with a quote from Alan Cooper:

*“Logic is a powerful and effective programming tool, it is a pathetically weak and inappropriate interaction design tool.”* [Cooper 2003]

When designing a product, logic will make it as broad in its functionality as possible to meet the requirements of most people. Alan Cooper [Cooper 1999] recommends that to have far greater success, a product should be designed for a single person.

Cooper developed a method called “Goal-Directed” design, which consists of novel ways of looking at problems. A way of doing this is to make up pretend users and design for them. Cooper calls these pretend users *personas*, and since this concept was published in Cooper's book [Cooper 1999], personas have become very popular. Personas are *hypothetical archetypes of actual users* which have become an essential base of good interaction design. They are not real, but represent real users and accompany the whole design process. A persona is used to check if the design meets the requirements of such a user.

Personas are specific and have to be concrete. The more specific personas are made, the more effective they are as design tools. The most important part of designing personas is to give each persona a name. Without a name, personas will never be concrete in anyone's mind. To make a persona more real to the project members, each persona get a face (images), an age, and subtle, believable details.

Interaction design is only good if someone is using it for a purpose. Purposes cannot exist without a user. A goal is related to a user's purpose, and a purpose is related to a persona. Completing this circle, personas are defined by the users' purposes, and are used to define specific goals. A goal is a desired result, an aim, or an end condition which can be reached by performing particular tasks. There are many ways to accomplish a single goal. For example [Andrews 2011]:


- Goal: *get to work*
- Task: *go by bus*
- Task: *go by taxi*
- Task: *go by car*

Each persona gets its own usage scenario. This is a precise description of a persona using an interface to achieve a goal. Three types of scenarios are described: daily-use, necessary use and edge-case scenarios. As seen in the following, edge-case scenarios are less important during the design process than daily-use scenarios.

- Daily-use scenarios: primary actions that users perform which need a very robust design.
- Necessary use scenarios: infrequent actions used from time to time.
- Edge-case scenarios: can be largely ignored during design.

In the following, Figure 3.6, Figure 3.7 and Figure 3.8 present concrete examples of personas which are used for the goal-oriented interaction design in the Catroid (see Section 2.1) project.

**Angelika Bacher**



**16.10.1975 , weiblich, Maierhof (AT)**

**„Ich möchte, dass meine Kinder kreativ sind und logisch denken lernen.“**

**Beschreibung** Angelika Bacher ist Mutter von drei Kindern zwischen 8 – 14. Durch die Geburt ihrer Tochter Anna hat Angelika ihr JUS-Studium abgebrochen um sich voll und ganz um ihr Kind zu kümmern. Seit auch ihr Jüngster, Felix, zur Schule geht arbeitet Angelika halbtags als Anwaltsheferin.

**Sozialer Hintergrund** Angelika Bacher lebt mit Ihrem Mann Bernhard und ihren drei Kindern in einem renovierten Bauernhaus. Bernhard ist als Bauingenieur tätig und arbeitet oft bei internationalen Projekten mit. Daher ist er manchmal über längere Zeiträume nicht zu Hause und Angelika kümmert sich alleine um die Kinder. Bernhard ist in einer Großstadt aufgewachsen und daher der Meinung, dass seine Kinder das Landleben genießen und nicht den ganzen Tag im Haus verbringen sollen. Die Kinder Anna 14, Fredrik, 10 und Felix, 8 gehen alle zur Schule und sind gute Schüler. Anna und Fredrick sind oft im Internet um sich mit ihren Freunden auszutauschen. Felix hat keinen eigenen Computer und spielt lieber mit seiner Konsole. Da Anna öfters mit ihren Freunden unterwegs ist, besitzt Anna seit einem Jahr ein Handy, allerdings ein altes Gerät ihrer

Eltern. Angelika würde es gerne sehen, wenn ihre Kinder viel Zeit im Freien verbringen. Da zu ihrem Haus ein sehr großer Grund gehört, ist es möglich, dass die Kinder alleine draußen spielen.

Wenn die Kinder im Haus sind hätte Angelika gerne, dass sie gemeinsam Dinge machen anstatt fernzusehen oder Computerspiele mit fragwürdigen Inhalten spielen. Angelika ist der Meinung, dass Kinder bei bestimmten Fernsehserien und Computerspielen zwar was lernen, aber hier die soziale Komponente fehlt. Sie hätte gerne ein Computerprogramm oder Spiel bei dem die Kinder einerseits jedes für sich kreativ sein kann aber gleichzeitig sollen sie auch zusammenarbeiten können. Vor allem die gegenseitige Unterstützung ist ihr wichtig, da ihr dies in der Beziehung zu ihren eigenen Geschwistern immer gefehlt hat.

**Finanzielle Situation** Die Familie Bacher ist durch den Beruf von Bernhard finanziell gut situiert. Daher reicht es, dass Angelika nur einen Halbtagsjob hat und sie kann somit die restliche Zeit für ihre Kinder da sein. Die beiden älteren Kinder haben jeweils einen eigenen Rechner, wobei das Internet von Angelika oder Ihrem Mann Bernhard freigegeben werden muss.

**Web** Angelika nutzt das Web privat nur um E-Mails lesen und versenden. In der Arbeit benötigt sie es für den Zugriff auf diverse Rechtseiten, ansonsten verwendet sie es kaum. Anna und Fredrick benutzen das Internet um sich mit ihren Freunden auszutauschen, vor allem Anna nutzt diese Methode um auch von zu Hause aus mit ihren Freundinnen zu „plaudern“.

**Negative Erfahrungen** Durch ihre Tätigkeit als Anwaltsheferin bekommt Angelika auch viele Probleme mit Onlineapplikationen, Betrugsmaschinen, Phishing, usw. mit ihre Tochter Anna hat über eine Online-Community dubiose Angebote bekommen, dies möchte Angelika nach Möglichkeit verhindern.

**Wunschliste** Angelika möchte, dass ihre Kinder gemeinsam an etwas arbeiten können - dass sie einander unterstützen und einander helfen. Außerdem möchte sie Kontrolle über die Inhalte haben und gewalttätige und pornographische Darstellungen von ihren Kindern fernhalten. Hierfür würde Angelika allerdings eine Benutzeroberfläche benötigen, die einfach zu handhaben ist und sie unterstützt.

**Zukunftsvorstellung** Angelika und Bernhard möchte, dass ihre Kinder sich mit Technik auskennen und diese verstehen. Sie glauben, dass ihnen dieses Wissen später helfen wird. Vor allem Bernhard glaubt, dass technisches Verständnis und logisches Denkvermögen seinen Kindern viele Türen öffnen würde.

**Zusätzliche Informationen** Dadurch dass Bernhard des Öfteren über längere Zeiträume nicht zu Hause ist, hat er das Gefühl etwas zu verpassen. Angelika hält ihn zwar über E-Mails auf dem Laufenden, aber durch ihre geringen technischen Kenntnisse beschreibt sie die Ereignisse meist nur verbal und ohne Bilder. Bernhard würde es aber schön finden auch visuell mehr am Leben seiner Kinder teilhaben zu können

Figure 3.6: Example persona named Angelika: she is the mother of three children between the ages of 8-14 years old. Image used with kind permission from Cure ©2010



Figure 3.7: Example persona named Tobias: he is 15 years old and will start high school soon. He is a tech-geek and likes it to find out how things work. Image used with kind permission from Cure ©2010

### 3.4.4 Iterative Design

Evaluating and testing a design early in the design cycle is essential for getting usability feedback as early as possible. Designing, evaluating and redesigning of an interface is a cycle of continuous improvement.

These are typical steps of iterative design:

- Complete an initial design (prototype)
- Present it to several test users for evaluation
- Note any problems the users had
- Refine interface and fix usability problems
- Repeat steps two to four until the problems are resolved

A design prototype that should be evaluated can be either a verbal prototype, a paper prototype, a working prototype, or an implemented final design.

**Silvia Eder**



23.05.1999, weiblich, Linz (AT)

„Ich möchte mit meinen Freunden reden und meine Bilder tauschen.“

<b>Beschreibung</b>	Silvia hat im Herbst mit dem Gymnasium begonnen. In der Schule hat sie neue Leute kennengelernt, mit denen sie auch im Internet kommuniziert. Neben der Schule ist sie mit ihrer besten Freundin zusammen in einem Handball-Team. Außerdem malt und bastelt Silvia sehr gerne. Silvia darf nur eine Stunde pro Tag fernsehen, wobei sie „Hannah Montana“ und „Die Zauberer vom Waverly Place“ bevorzugt, da es auch ihre Freundinnen schauen und am nächsten Tag in der Schule darüber reden.	<b>Web</b>	Bei komplizierten Oberflächen hat sie immer Angst zu scheitern. Wichtig für Silvia wäre ein Programm, das sich ihr auf einfache Art und Weise erschließt und wo sie nicht viel falsch machen kann. Allerdings möchte sie keine langen Anleitungen lesen, sondern eher spielend lernen. Da nur der Rechner im Wohnzimmer Internetanbindung hat, ist es für Silvia wichtig, dass ihre Lernprogramme auch Offline gut funktionieren. Wenn Silvia das Internet nutzt, dann ist sie vor allem auf Social Network Sites, wo sie in erster Linie mit ihren Freundinnen Nachrichten austauscht. Ein Foto von ihr hat sie nicht online, da dies ihre Eltern nicht erlauben. Silvia schreibt auch E-Mails mit einer Freundin, die vor kurzem in eine andere Stadt gezogen ist. Die beiden Mädchen schicken sich regelmäßig Bilder um sich auf dem Laufenden zu halten. Silvia spielt alleine eher wenig am Computer, da nützt sie lieber ihre Lernprogramme. Wenn ihre Freundinnen zu Besuch sind macht ihr Computerspielen mehr Spaß.
<b>Sozialer Hintergrund</b>	Silvia lebt mit ihren Eltern und Brüdern in einer Doppelwohnhälfte. Silvias Vater arbeitet die meiste Zeit, während ihre Mutter derzeit zu Hause ist. Silvia hat zwei Brüder, Andreas und Matthias, 5-jährige Zwillinge.	<b>Negative Erfahrungen</b>	Silvia bekommt hin und wieder Spam auf ihre E-Mail Adresse, was sie gar nicht mag. Außerdem hat Silvia einmal eine Hausübung in einem Programm geschrieben. Die Daten waren nach einem Systemabsturz gelöscht. Außerdem mag Silvia es nicht, wenn sie in einem Programm herum klickt und sie bekommt kein Feedback was passiert oder was sie gemacht hat, da sie bereits einmal den Schwierigkeitsgrad in ihrem Lieblingsspiel geändert hat.
<b>Gesundheit</b>	Durch einen Unfall in ihrer Kindheit hört Silvia auf einem Ohr etwas schlechter, aus diesem Grund mag sie Musik nicht so gerne.	<b>Wunschliste</b>	Silvia würde sich gerne mehr mit dem Computer beschäftigen. Wichtig ist es ihr, dass Dinge einfach zu machen sind und sie sich nicht langweilt. Am liebsten wäre es ihr, wenn sie ein Programm hätte, mit dem sie sich am Computer grafisch kreativ betätigen könnte, dass jedoch auch für Anfänger leicht verständlich ist. Sie möchte außerdem, dass sie in Programmen ohne wildes herum klicken ihre Arbeit erledigen kann. Wenn sie etwas alleine macht, würde sie dies gerne ihren Freundinnen zeigen und auch mit ihnen tauschen. Lustiger findet sie es allerdings, wenn sie sich mit ihren Freundinnen gemeinsam mit Spielen usw. beschäftigt, da sie dann weniger Angst vor Fehlern hat. Silvia würde gerne kurze Filme oder kleine Spiele machen, die sie einerseits ihren Freundinnen zeigen aber auch ihren kleinen Brüdern spielen lassen kann. Auf vielen ihrer Zeichnungen sind Pferde und andere Tiere zu sehen, die sie gerne „zum Leben erwecken“ würde.
<b>Technisches Wissen</b>	Silvia hat kein fixes Handy. Wenn sie allerdings unterwegs ist, z. B. im Handballtraining bekommt sie ein Handy. Ihre Freundinnen besitzen eigene Geräte, mit denen sie in der Pause spielen oder Fotosessions machen. Silvia hat bisher nur wenig mit dem Computer zu tun gehabt und sie hat das Gefühl, sich nicht gut damit auszukennen. Nur mit Social Network Sites ist sie einigermaßen gut vertraut, da sie hier Kontakt mit ihren Freundinnen hat. Vor kurzem hat sie ein älteres Notebook bekommen, allerdings hat sie Angst etwas kaputt zu machen. Wenn Silvia am Computer etwas macht, dann mag sie Programme mit einfachen, überschaubaren und hübschen	<b>Zukunftsvorstellung</b>	Silvia möchte mal Künstlerin oder Designerin werden. Wichtig ist ihr, dass sie kreativ sein kann.
		<b>Zusätzliche Informationen</b>	Silvia ist in Mathematik durchschnittlich, da sie sich gewisse Dinge nicht vorstellen kann. Dafür ist Silvia ein sehr kreatives Kind, das sehr gerne zeichnet und gut Geschichten schreiben kann.

  
center for usability research & engineering

Figure 3.8: Example persona named Silvia: she is new to secondary school and is using the internet to communicate with her new classmates. Image used with kind permission from Cure ©2010

### 3.4.4.1 Prototyping

As discussed before, the design of a product should be evaluated as early as possible in the development process to assess the usability and get feedback for further changes. Prototypes increase in complexity during the iterative design process [Andrews 2011].

#### Verbal prototypes

These are simple textual descriptions of the interface, the elements and dialogues and all kinds of choices and results.

#### Paper prototypes

Screen and dialogue elements are sketched and simulated on paper. Each interface screen including all screen elements can be sketched on a separate paper. These first *throwaway* hand-made designs provide a good feedback for a minimum of effort. These early prototypes can be replaced later by more detailed coloured printouts, which look more like

a finished design. Hand-made sketches are known as *low-fidelity* paper prototypes, while ones made on computer are known as *high-fidelity* paper prototypes.

### Interactive prototypes

Hand-drawn interface sketches are scanned and put together interactively using computer software. These animated interactive prototypes are linked with clickable elements. A casual look at the interface encourages for criticism and discussion.

### Working prototypes

Working prototypes are simpler algorithms which ignore special cases and can vary in complexity. They can include fake data or screen-shots instead of videos, and are cut down on either the number of features, or the depth of functionality. As shown in Figure 3.9, horizontal prototypes keep the features and elements of the interface but without in-depth functionality. Vertical prototypes give full functionality for a few, selected set of features. Scenario based prototypes have features and full functionality for a specific scenario or path through the interface that is to be evaluated.

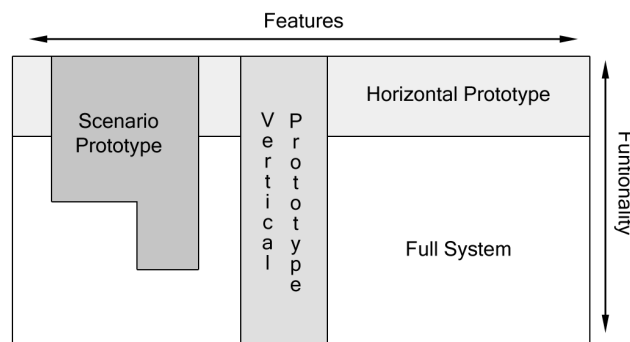


Figure 3.9: Working prototypes: the two dimensions vary according to the width and depth of implemented features [Nielsen 1993].

### Implementation

Implementing a final design and performing a competitive analysis of software components: using existing interface frameworks saves a lot of work. Strive to use existing components and applications rather than reinventing the wheel.

### 3.4.4.2 Formative and Summative Usability Evaluation

Usability feedback assessed from the iterative design process is used for refining an interface design. There are a variety of usability evaluation methods. While some evaluation methods use data from users, others use usability information from usability experts. The term *evaluation* means different things to different people. Andrews [2008] has grouped common evaluation methods according to their purpose and type (see Figure 3.10). The methods themselves are described in Table 3.3.

Formative and summative usability evaluation methods are classified into two major categories, according to who performs the usability evaluation.

- *Usability Inspection Methods*: inspection of interface design is done by specialist evaluators using their experience and judgment (no test users). Usability inspection methods are summarized in Section 3.5.
- *Usability Testing Methods*: real representative test users are empirically tested when using a user interface. Usability testing methods are discussed in more detail in Chapter 4.

Andrews [2008] classified four types of evaluation, according to their purpose. He extended Robert Stake's [Stake, Robert E. and Organisation for Economic Cooperation and Development, Paris (France). Centre for Educational Research and Innovation. 1976; Lockee et al. 2002] *soup analogy* for a better understanding.

- *Exploratory*: how is an interface used and what is it used for.

“When the cook tastes other cooks' soups, that's exploratory.”

- *Predictive*: how good is the performance based on an interface design.

“When the cook predicts the quality of a soup from a recipe, that's predictive.”

- *Formative*: how can a user interface be made better based on the feedback about problems and recommended solutions.

“When the cook tastes his own soup while making it, that's formative.”

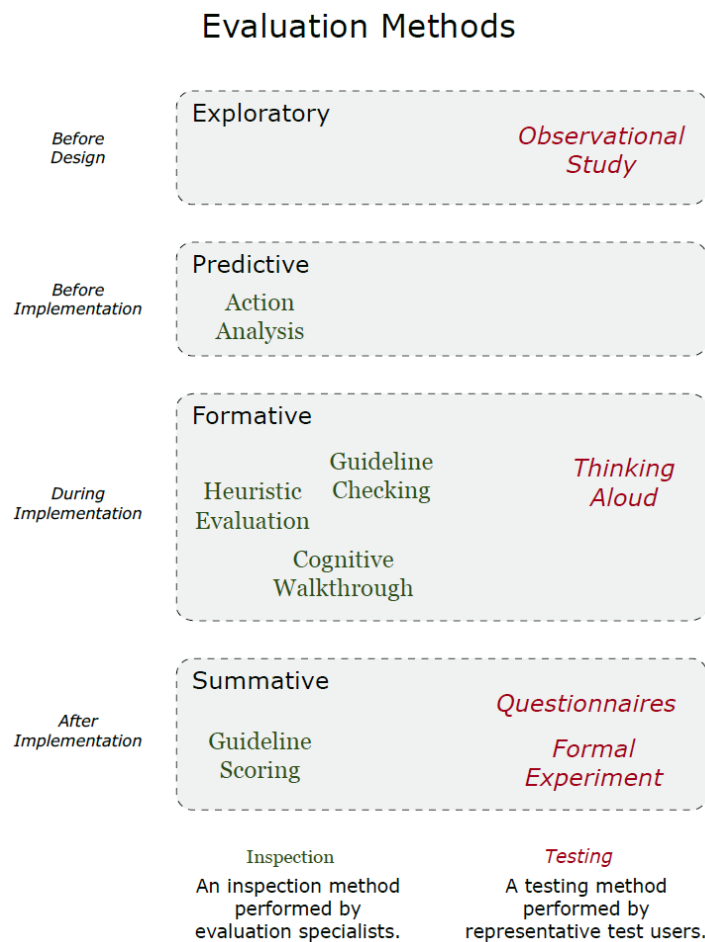


Figure 3.10: Nine common evaluation methods grouped by purpose and who performs them. Image used with kind permission from [Andrews 2008].

- *Summative*: how good is the interface based on a numerical data which is statistically analysed.

“When the guests (or food critics) taste the soup, that’s summative.”

### 3.4.5 Follow-Up Studies

Important information for future versions can be obtained by studying users’ working with the released product. The released product is then considered as a prototype for future releases or new products. Feedback about the new release can be conducted by special field studies such as interviews, questionnaires, or observation. Standard marketing studies about what people are thinking about the product are a very cheap when making use of

<i>Method</i>	<i>Type</i>	<i>Purpose</i>	<i>Description</i>
Observational Study	Testing	Exploratory	A longer term study following a small sample of users as they use an interface for their own tasks. Observations and anecdotal evidence are collected and assessed.
Action Analysis	Inspection	Predictive	An evaluator produces an estimate of the time an expert user will take to complete a given task, by breaking the task down into ever smaller steps and then summing up the atomic action times.
Heuristic Evaluation	Inspection	Formative	A small team of evaluators inspects an interface using a small check-list of general principles and produces an aggregate list of potential problems.
Guideline Checking	Inspection	Formative	An evaluator checks an interface against a detailed list of specific guidelines and produces a list of deviations from the guidelines.
Cognitive Walkthrough	Inspection	Formative	A small team walks through a typical task in the mind set of a novice user and produces a success or failure story at each step along the correct path.
Thinking Aloud	Testing	Formative	Representative test users are asked to think out loud while performing a set of typical tasks. The insight gained into why problems arise is used to produce a list of recommendations.
Guideline Scoring	Inspection	Summative	An evaluator scores an interface against a detailed list of specific guidelines and produces a total score representing the degree to which an interface follows the guidelines.
Questionnaires	Testing	Summative	After using one or more interfaces for some typical tasks, test users are asked to rate the interface(s) on a series of scales.
Formal Experiment	Testing	Summative	A larger sample of users performs a set of tasks on one or more interfaces. Objective measurement data is collected and statistically analysed.

Table 3.3: Nine common evaluation methods, classified according to their type and purpose [Andrews 2008].

newsgroups, mailing lists, product reviews or tests in magazines. Logging and protocoling the product's activities can also provide very useful information, but should only be used with permission of the users. The analysis of user complaints from support hotlines, modification requests or bug reports can also be a very important data source.

### 3.5 Usability Inspection Methods

Usability inspection as a term that was introduced by Nielsen [1993] and is a synonym for a set of methods which involve the inspection of an interface design using heuristic methods. Although usability testing is probably the most commonly used empirical method for evaluating user interfaces, real end-users are often expensive or difficult to recruit. Informal



methods such as inspection are highly cost-effective and provide an effective way of reducing the number of users required and saving time [Nielsen 1994c]. A usability inspection is conducted by a number of experts, or sometimes also by representative users. In the following, five usability inspection methods, which are referenced in Subsubsection 3.4.4.2, will be discussed in more detail.

### 3.5.1 Heuristic Evaluation

Heuristic evaluation is the most popular usability inspection method, was first described in Nielsen and Molich [1990] and is explicitly intended as a “discount usability engineering” method [Nielsen 1993]. A set of evaluators (mainly usability specialists) systematically inspect the user interface and records good and bad points about the interface against a set of requirements. These requirements consist of a small set of recognised usability principles, which are referred to as the “heuristics”.

The heuristic evaluation method is very easy to use and does not need usability expertise, but usability specialists with a knowledge of the specific kind of interface being tested may be better at finding usability problems [Nielsen 1992].

#### Usability Heuristics

Usability heuristics are a small set of fairly broad usability principles for heuristic evaluation of the design of the user interface. Various sets of heuristics are available and the variety of heuristics is not limited to existing lists. The following list was originally developed by Nielsen and Molich [1990]. It has been refined from a factor analysis of 249 usability problems to derive a revised set of heuristics. The categories are exact quotations from the “10 Usability Heuristics” from Nielsen [2005b].

1. **Visibility of System Status**

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

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

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

3. **User control and freedom**

Users often choose system functions by mistake and will need a clearly marked

“emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

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

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

#### 5. **Error prevention**

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

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

Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

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

Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

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

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

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

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

#### 10. **Help and documentation**

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

### Response Time

Basic advice regarding response times has been the same for some thirty years. The following list is taken from Section 5.5 of Nielsen [1993].

- **0.1 second** is the limit for users to feel that the system is reacting instantaneously.
- **1.0 second** is the limit for the users' flow of thought to stay uninterrupted. Between 0.1 and 1.0 seconds no special feedback is necessary. For delays between 1.0 and 10 seconds a "busy" cursor is a common solution.
- **10 seconds** is the limit for keeping the users' attention focused on the task. For delays longer than 10 seconds a common solution is to display a progress indicator.

### Usability Problems Found

A single evaluator can perform a heuristic evaluation of an interface. [Nielsen 1993] indicated that poor results are achieved when heuristic evaluation relies on a single evaluator. The average result of an experiment over six projects was that only 35 percent of the usability problems were found. Experience from many different projects has shown that different evaluators mainly find different usability problems. Furthermore, heuristic evaluations will have much better results when multiple evaluators are involved, independently of the others. As seen in Figure 3.11 the number of problems found grows rapidly in the interval from one to five evaluators but reaches the point of diminishing number of problems found around at the point of ten evaluators. Nielsen and Molich [1990] recommend at least a number between three and five evaluators for a heuristic evaluation.

### Heuristic Evaluation: How-to

Heuristic evaluation is done by each individual evaluator inspecting the user interface independently. To ensure independent and unbiased evaluations, evaluators are only allowed to communicate and aggregate their findings once all of them have completed the evaluations. A typical evaluation session takes between one and two hours. Very complex user interfaces with large numbers of dialogue boxes might need longer evaluation sessions, but should be inspected in multiple sessions, each concentrating on a smaller part of the user interface.

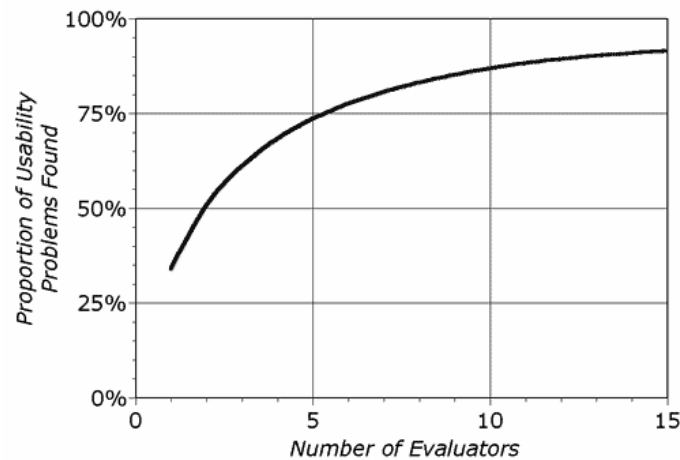


Figure 3.11: Average proportion of usability problems found by various number of novice evaluators [Nielsen 1993].

During the session, the interface is usually examined in several passes. The evaluators should go through the interface at least twice. In the first pass, the evaluator has to become acquainted with the interface and its dialogues, and during the second pass, all various dialogue elements are inspected and compared against a checklist of general heuristics (see Section 3.5.1). The evaluators should explain as specifically as possible why they do not like something and list each problem separately. In addition, the evaluator is obviously allowed to note all additional usability principles or results that may be relevant for every specific element.

In the next step, the individual problems found by each evaluator are discussed and put together into one list of problems that have been found. This can be done by way of a discussion session with all evaluators or by an evaluation manager. A heuristic evaluation does not provide any systematic schemes to create fixes to problems, but provides explanations for any observed usability problem with reference to the usability principle. The output of the second step is a list of usability problems, annotated with references to the usability problems that the interface design produced.

It is probably desirable to increase the number of serious problems found by a heuristic evaluation. As a large proportion of problems found by heuristic evaluation tend to be minor problems, it is still possible to focus on the serious problems by using a severity rating (see Subsection 3.5.5) method to prioritise the list of usability problems [Nielsen 1994a]. This list can be used to discuss fixes and the redesign of important problems that have been found.

### 3.5.2 Cognitive Walkthrough

The cognitive walkthrough was first described in Lewis et al. [1990]. Clayton Lewis, Peter Polson, Cathleen Wharton, and John Rieman developed the cognitive walkthrough method as a task-oriented walkthrough of an interface. It is a usability inspection method that focuses on evaluating the “ease of learning” of a user interface [Wharton et al. 1994]. Many users prefer to learn a software and its functionality by exploration, and without having to invest much of time in any formal instruction. Instead of formal training, users prefer to learn about a user interface while working on their usual tasks and acquiring knowledge on how to use new features when they need to use them. A cognitive walkthrough has analogies to other design walkthroughs, including requirement or code walkthroughs.

The design of the user interface does not have to be implemented in detail, but may be either a mock-up or a working prototype. It is also possible to evaluate a detailed description of a user interface design. The purpose of evaluating a design using the cognitive walkthrough method is to evaluate the ease with which users can perform a task with little or no formal or informal instruction. Reviewers evaluate a proposed interface in the context of one or more specific tasks. The input to a walkthrough session takes account of the interface’s detailed design description, a task scenario, explicit assumptions about the target user population and its context of use, and a sequence of actions which a user needs to successfully perform to complete a determined task.

#### **Cognitive Walkthrough: How-to**

The cognitive walkthrough analysis consists of two phases:

- a preparatory phase for the required materials and
- an analysis phase.

In the first phase, the reviewers prepare the input conditions for the walkthrough: the tasks, action sequences for each task, the user population, and the user interface. In the analysis phase, the main analytical work is done by working through the actions of every task.

The walkthrough can be done by a single evaluator or a group of experts. For a group evaluation, the members may include other designers, software engineers, usability specialists, and representatives of other disciplines such as marketing and training. Each member of the team gets a specific expert role: a subscriber or recorder, a facilitator, and various kinds of other experts with knowledge according to the member’s domain.

### Defining the Input Conditions

Before the walkthrough is performed, four questions have to be answered:

- *Who will be the users of the system?* Identifying the users' population and add specific background knowledge to the users' descriptions.
- *What task (or tasks) will be analysed?* Defining a detailed description of the tasks which should be representative for the core functionality of the system. The critical question for the selection of these representative tasks can be supported by results of marketing studies, needs and requirements analysis, and concept testing.
- *What is the correct action sequence for each task and how is it described?* Specify a detailed description of the concrete actions to accomplish each task. These actions should be as simple as possible, such as, "press Return" or "Move cursor to 'Edit' menu", or a sequence of simple actions that an average user could perform: "login to the system".
- *How is the interface defined?* Description or implementation of the interface or a prototype. What will the users see before and after each action and how to perform it.

### Walk-through

The analysis of the sequence of actions for the correct action path consists of examining each action. For every action it is assumed that users act according to the problem-solving process of Polson and Lewis [Polson and Lewis 1990].

The problem-solving process is described by the CE+ theory of exploratory learning of Polson and Lewis, and assumes that users often prefer to learn an unknown system by trial and error, and therefore guess the correct action. In brief, the process holds the following:

1. Start with a rough description of task to be accomplished.
2. Explore the interface and select the most appropriate actions that will accomplish the task.
3. Observe the interface's reactions to see if their actions had the desired effect.
4. Determine which action to take next.

For each action in the correct sequence of actions, the walkthrough schedules the construction of a credible success or failure story by answering the following questions:

- a) Will the user try to achieve the right action?
- b) Will the user notice that the correct action is available?
- c) Will the user notice that the correct action will achieve the desired effect?
- d) If the correct action is taken, will the user see that progress is being made toward solution of their task?

### 3.5.3 Guideline Reviews: Checking and Scoring

Guidelines are specific pieces of advice about the usability principles of a user interface. A guideline review is a usability inspection method where an interface is evaluated against a detailed set of specific usability guidelines. Various guidelines are available. As explained in Subsection 3.5.1, heuristic evaluation employs about ten principles, but guideline checking can involve dozens or even hundreds of very specific individual principles on a single checklist [Andrews 2011].

Designing user interface software often involves significant investment of time and effort. A set of guidelines can help ensure the value of such an investment. One of the very first popular and comprehensive sets of guidelines was developed for the United States Air Force by Smith and Mosier [1986]. The proposed set of 944 guidelines for designing user interfaces in computer-based information systems is divided into six functional areas: data entry, data display, sequence control, user guidance, data transmission, and data protection (see Table 3.4). This set was recommended as a basic reference for designing user interfaces, but as it is quite old the primary focus is not on state-of-the-art interface designs.

Another more recent set of guidelines is available from the [ISO 1998] standard and represents a collection that focuses on the “Guidance on usability”. The ISO 9241 standard, with the general title “Ergonomic requirements for office work with visual display terminals (VDTs)”, consists of a number of areas including: requirements for non-keyboard input devices (part 9), dialogue principles (part 10), menu dialogues (part 14), command dialogues (part 15), direct manipulation dialogues (part 16), form-filling dialogues (part 17).

There exist so-called “User Experience Interaction Guidelines”, which are more user interface guidelines and styleguides than usability guidelines; and are available from dif-

<i>Section</i>	<i>Functional Area</i>	<i>Number of Guidelines</i>
1	Data Entry	199
2	Data Display	298
3	Sequence Control	184
4	User Guidance	110
5	Data Transmission	83
6	Data Protection	70

Table 3.4: Guidelines for user interface design in six functional areas: data entry, data display, sequence control, user guidance, data transmission, and data protection. This collection revises and extends previous compilations of design guidelines [Smith and Mosier 1986]

ferent companies, such as Apple Computers, Microsoft, Eclipse, Sun Microsystems and Microsoft to support interface designers developing vendor-specific applications. A list of various guidelines is available on Wikipedia, the free encyclopedia [2012].

### **Guideline Checking**

Guideline checking means that an evaluator checks an interface design against the list of specific guidelines and produces a list of changes for the interface [Andrews 2011].

### **Guideline Scoring**

Guideline scoring means that the interface is rated in regard to its conformance against a weighted list of specific guidelines. The result is a total score that represents the degree to which an interface follows the specific guidelines [Andrews 2011].

#### **3.5.4 Action Analysis**

Action analysis is a quantitative analysis of the sequence of actions a user has to perform to complete a task with an interface [Lewis and Rieman 1993]. Action analysis focuses on evaluating the “efficiency” of using an interface. The aim is to predict the time an experienced user requires to complete tasks.

The method is divided into two levels of detail:

1. Formal or “Keystroke-Level”
2. Informal or “Back-of-the-Envelope”

Action analysis, whether formal or informal, has two phases:



- a) Decide what physical and mental actions a user will perform to complete the task and then list the actions.
- b) Think about the actions, analyse them and look for problems.

### Keystroke-Level

The formal approach to action analysis, also called “keystroke-level analysis” [Card et al. 1983], leads to close inspection of the action sequence that a user performs when completing a task. The formal approach involves breaking the task into minute detail, like “Use mouse to point at object X on screen” or “Move hand to pointing device or function key”, that the analysis of these actions makes it possible to calculate the time needed to perform the action within a 20 percent margin of error. But, as Lewis and Rieman [1993] states, formal analysis is not easy to do.

The formal approach has been used to make precise predictions of the time it takes skilled users to complete tasks. Predictions of times for the small steps are found by testing hundreds of different users, thousands of individual actions, and calculating the average values of each action at the end.

Developing a list of individual actions is done as follows: Divide a basic task into a few subtasks. Break each of the subtasks into smaller subtasks, and so on until the description of the action reaches the level of “fraction-of-a-second” operations. See Table 3.5 for some common actions. The result is a hierarchical description of the task and the sequence of actions needed to perform it.

	<i>Action</i>	<i>Time</i>
Physical Movements	One keystroke on a standard keyboard	0.28
	Use mouse to point at object	1.5
	Move hand to pointing device or function key	0.3
Visual Perception	Respond to a brief light	0.1
	Recognize a 6-letter word	0.34
	Move eyes to new location on screen	0.23
Mental Actions	Retrieve a simple item from long-term memory	1.2
	Learn a single “step” in a procedure	25
	Execute a mental “step”	0.075
	Choose among methods	1.2

Table 3.5: Average times for typical keystroke-level computer interface actions, in seconds. Many values are averaged and rounded. [Lewis and Rieman 1993; quoted Olson and Olson 1990]

### Back-of-the-Envelope

The informal method is called the “back of the envelope” approach. This kind of evaluation is less detailed and will not provide detailed predictions of task times, but it can reveal large-scale problems that might get lost in the multitude of details that a designer is faced with when designing interfaces. However, the-back-of-the-envelope approach is easy to do and can be performed much faster.

The difference to the formal method is that there is no development of a detailed hierarchical description of tasks. The detail of the complete task description is at the level of explaining the actions to a typical user. Actions will probably be something like: “Select Open from the File menu”, “Confirm by pressing Ok”, or a brief description of mental actions, such as “Remember your password”. At this level, a couple of seconds is needed for any of these actions on a typical good day, three to four or even more on a bad day. The back-of-the-envelope approach allows the comparison of the system’s expected performance for a particular task. This kind of analysis can be very useful when deciding whether or not to add features to a system.

#### 3.5.5 Severity Ratings

As explained by [Nielsen 2005a], severity ratings can be used to prioritize the fixing of usability problems and can also provide a rough estimate of the need for further improvements of the usability.

The severity of a usability problem consists of three factors: the **frequency** with which a problem appears: Is it common or rare? The **impact** if it appears: Will it be easy or difficult for users to overcome? The **persistence** of the problem: Is it a one-time problem or will it bother users repeatedly?

The rating process is performed as follows: a complete list of usability problems is given to each evaluator. The evaluators then assign severity ratings (see Table 3.6) to each problem independently. Typically, the evaluators need about 30 minutes to provide their ratings. Since the rating of a single evaluator is unreliable, a mean of 3-5 evaluators is satisfactory [Andrews 2011; quoted Nielsen and Mack 1994].

---

<i>Score</i>	<i>Severity</i>	<i>Priority</i>
4	catastrophic problem	imperative
3	major problem	high
2	minor problem	low
1	cosmetic problem	
0	not a problem	

Table 3.6: Five-Point Severity Scale [Andrews 2011].

## Chapter 4

# Usability Testing Methods

In general, people think they understand how others behave. This opinion is based on our own experiences and can only be disproved by testing a presumption. Intuition is often wrong. Interface designers find it very easy to use their intuition, although new experiences can change one's point of view [Andrews 2011].

Usability testing refers to a process involving representative users of the target audience. It evaluates a user interface and obtains specific information about a design. Evaluating of an interface design with real users is the foremost usability method. It provides direct information about how users interact with an interface and the exact problems that exist with the tested interface. Originally, usability tests came from experimental psychology and were primarily used to obtain statistical analysis. Usability testing places more emphases on the interpretation of the results and things found out during the test than generating statistical data [Hom 1998, Nielsen 1993, Rubin and Chisnell 2008].

Rubin and Chisnell [2008] describe that the intent of usability testing is to ensure that products:

- are useful, and appreciated by the target users,
- are easy to learn,
- help users to be more effective and efficient, and
- are satisfying to use.

Hom describes the overall process of testing in one simple sentence:

*“Get some users and find out how they work with the products.”* [Hom 1998].

Usability tests are usually performed in a usability test lab, or by using a mobile equipment which can be set up in any environment when needed. Individual users are observed whilst performing specific tasks with the interface. One or more observers collect information about what users did when performing the tasks. For example, how long did it take a user to perform the task? Or, what kind of errors did the users make? Finally, the data collected from all the experiments is analyzed in a search for new strategies.

According to Rubin and Chisnell, usability testing includes the following basic characteristics:

- Specific questions and goals rather than hypotheses have to be defined when planning tests.
- The test is performed with a number of representative end users.
- The users should work on real tasks.
- The evaluation team observes the users and records what they do and say when performing the tasks.
- Interviewing and probing of the test participants by the test moderator.
- Analysis of the collected data, and subsequent determining of problems and recommendation of improvements to the user interface.

The following list presents five important usability testing methods according to Figure 3.10. These methods are described in more detail in the following sections.

- *Thinking-Aloud*: A thinking-aloud test is a method, where test users are asked to verbalise their thoughts whilst moving through the user interface (see Section 4.1).
- *Constructive Interaction*: Co-discovery is a variation of the thinking-aloud method and involves two test users, exploring an interface together and having a conversation while performing test tasks (see Section 4.2).
- *Formal Experiment*: Some usability tests are aimed at identifying hard, quantitative data in controlled experiments with test users. Formal experiments are used for statistical analysis of objective measurements (see Section 4.3).
- *Questionnaires*: Many usability aspects of a product can be studied by simply asking representative users. This is also very useful for finding out how users use a system and what they like and dislike about the product. (see Section 4.4).

- *Observational Study*: Is the simplest of all usability evaluation methods. An observer visits one or more users and collects usage data (see Section 4.5).

Before looking at the testing methods mentioned in more detail, an overview of the preparation phase of a usability test is presented in the following section.

### The Preparations for a Test

When conducting a usability test, it is recommended that attention is paid to a number of things. The items of the following list are always part of a test domain and represent a very useful checklist. These ideas are taken inter alia from Andrews [2011].

- *Test environment*: It is recommended that a dedicated usability test lab is used (see Figure 4.1 and Figure 4.2), if one is available. Otherwise, to ensure a comfortable environment for the test, a quiet room is required. A prepared sign, such as “User testing in progress, do not disturb”, switching off phones, or closing the windows may help to avoid disturbing the events taking place. Keep in mind the light settings and provide refreshments for the participants.
- *Test equipment*: Digital video recording equipment (see Figure 4.3) is needed to record the users performing their tasks. As the moderator has to deal with observation of the users activities, recordings are useful as a backup for missed incidents. The equipment should include one or more video cameras, tripods, a good quality microphone and headphones.
- *Roles*: If a usability test team consists of more than one member, several roles will be assigned to these members. The *test facilitator* is responsible for the administration, management, and moderation of the test, and for all other interactions which include the test user, such as the introduction and the final debriefing. The *video operator* is responsible for everything that involves the recording of the test process, including checking the camera’s viewing angle, focus and picture overlays, the audio settings and collecting and organizing the recordings. The *data logger* is responsible for taking notes of activities and events of interest, including their timestamps. A *computer operator*’s responsibility is to reset the interface, clear caches, histories and favourites after a session, and to restart the system when it fails.
- *Test users*: Persons participating in the test tasks are called test user or test participants. The facilitator has to make clear that the test users are evaluating the

product and the participants are not being tested at all.

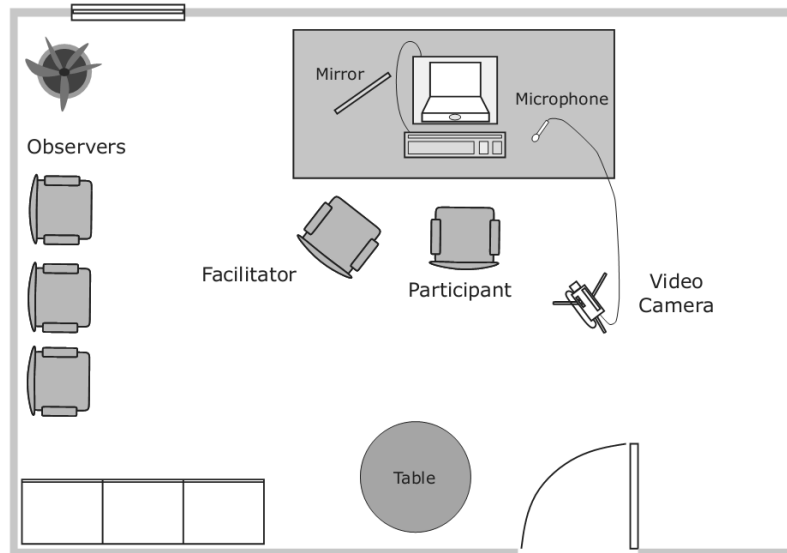


Figure 4.1: A simple single room usability test setup with one camera. Image used with kind permission from Keith Andrews Andrews [2011]. This figure is inspired by Rubin and Chisnell [2008].

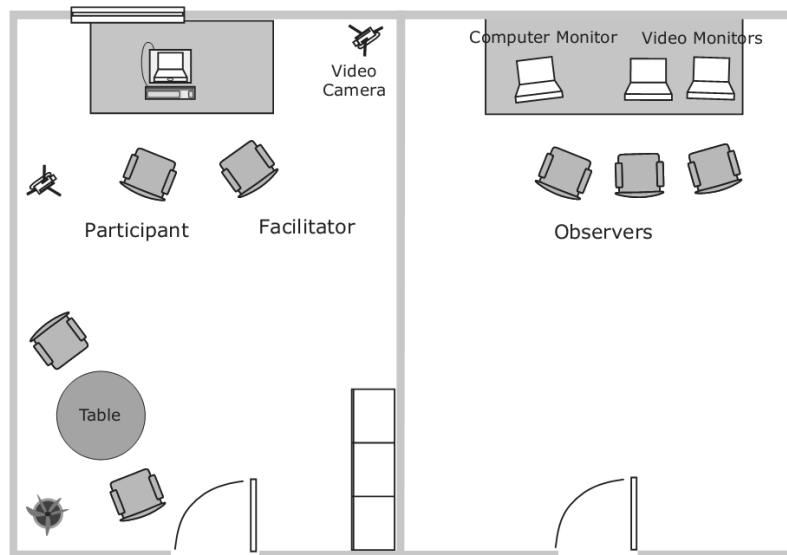


Figure 4.2: A usability test setup with its own observation room and real-time monitoring. Image used with kind permission from Keith Andrews Andrews [2011]. This figure is inspired by Rubin and Chisnell [2008].

## Usability Kit Inventory List

1. Tripod Hama Profil 74
2. Rucksack LowePro
3. Headphones
4. Headphones extension cable
5. Microphone Philips SBC ME570
6. Microphone extension cable
7. Headphone adapter
8. Usability kit inventory list
9. Video camera manual
10. Video camera power supply
11. Video camera mains cable
12. Microphone adapter cable
13. Video camera Sanyo Xacti HD1010
14. Video camera bag
15. Video camera remote control
16. Transcend SD HC Card 16gb



Figure 4.3: An example of a portable usability kit. It is used at IICM, Graz University of Technology. The inventory shows all of the components of the kit. Image used with kind permission from Andrews [2011].

## Stages of Conducting a Test

Rubin and Chisnell [2008] describe the process for conducting a test as the following. The list is extended by recommendations of Lewis and Rieman [1993] and Andrews [2011]:

1. Decide what data to collect.
2. Develop the test plan and set up a test environment.
3. Select and acquire test participants.
4. Prepare test materials.
5. Always run a pilot test, prepare refreshments.
6. Conduct the test.
7. Analyse the data obtained.
8. Present final report of findings and recommendations.



## 4.1 Thinking-Aloud

The thinking-aloud method involves having test users using the interface whilst verbalising what they are doing [Nielsen 1993]. By thinking out loud, the test users enable an understanding of how they view the system by verbalising their thoughts. This method shows how users interpret each of the interface items, which makes it easy to identify major misconceptions of the users, and makes it possible to get a very direct understanding of what parts cause the most problems.

Originally, this method was used in the psychological research field, but is now one of the best practices to collect qualitative data about the usability of a system from a small number of users. The strength of thinking out loud is that it shows what users are doing and why they are doing it while they are doing it, without rationally thinking about it.

As Lewis and Rieman [1993] describe, thinking-aloud enables the detection of vocabulary problems in interfaces, when texts are read out loud. Furthermore, asking participants to think out loud during a usability session can offer many insights about how they are thinking about the product and if the way they use it matches up with how it was designed. Such things are hard to find out, if a user is not thinking out loud in usability tests. However, thinking-aloud slows down the user's performance significantly and makes it impossible to collect performance data during such a test.

### Instructions

The test users should tell the facilitator what they are trying to do, the things they are reading, questions that come to their mind, things that confuse them and the decisions they make [Andrews 2011]. To demonstrate the thinking-aloud method to the user, the facilitator performs an unrelated task and can also show a short video clip of another thinking-aloud session. Then, the user performs a short task on a different interface for practice. At least, if questions from the test user arise, they can be asked when they occur but will only be answered after the test. Before the test starts, the facilitator must make clear that the participants are going to be evaluating the product, and that they themselves are not being tested at all.

### Role of the Facilitator

Since the thinking-aloud approach seems to be very unnatural to most users, some users will have problems in keeping it up. Even though spontaneous comments from the test

user are the best sort of data to collect, the facilitator will often need to prompt users to keep verbalising their thoughts. Questions such as, “*What are you thinking now?*”, “*Can you say more?*”, or “*Please tell what you are doing now*” and “*What do you think this message means?*” are very helpful phrases. Questions from a user, like “*Can I do that?*” should not be answered, but can be replied by a counter-question like, “*What do you think will happen if you do that?*”. A test user should not be confronted with “Why”-questions like, “*Why did you do this?*” or “*Why did you not click this?*” [Andrews 2011, Nielsen 1993].

### Recording the Test

Lewis and Rieman [1993] recommend either to record the information obtained by taking notes on paper or by making a video recording of what happens on the screen and also the users’ facial expression and an audio record of the user’s comment. See Figure 4.4 for an example setup. Writing down the observations in the order that the users say and do it is done in an abbreviated form. It takes some practice to keep this up in real time, because it needs a general idea of where interface items are going. The best way to do is to combine a digital record with written notes.



Figure 4.4: Sample test setting for a thinking-aloud test. Image used with kind permission from Keith Andrews [Andrews 2011]

### Summarizing Tests and Using the Results

The aim of usability testing is to get useful data that can help to improve a design. A list of all difficulties that occurred in the test, including a reference back to original data if questions arise, and an assessment of why each problem appeared is required.

Considerations on what changes to the interface are needed are based on the results of the test. Lewis and Rieman proposes that results are looked at from two points of view. First, deal with the question:

*“What do the data tell you about how you THOUGHT the interface would work?”* Lewis and Rieman [1993]

This includes finding out if users took the same approaches that were expected, or different ones. The next step is to analyse the tasks and what the system should perform for each of the tasks, based on what the analysis shows. These improved results are a basis for a discussion about how to make the design better support what users are doing.

Second follows the analysis of errors and problems that have been found. Each issue will be judged by its importance, and how difficult it is to fix. The factors taken into account when judging the importance of an issue are the cost to the user and how many users it will affect. The difficulty level of the fix depends on how comprehensive the changes to fix it will be. All the issues flagged as important will be fixed as well as those that have been deemed easy to fix.

## 4.2 Constructive Interaction

Constructive interaction is known as a variation of thinking-aloud testing [Nielsen 1993, Rubin and Chisnell 2008]. This technique is sometimes also called “Co-discovery”, and involves two participants testing simultaneously during a usability test session. This test situation is more natural and the participants are encouraged to communicate with each other when trying to solve a problem together. The dialogue between them becomes an important factor for understanding how users work through problems when using a product. Talking to each other offers an alternative to a moderator prompting one test user to think out loud. Practitioners believe this to be more natural and therefore reliable than a typical moderator-participant session. The method does have a disadvantage: if two test participants are not willing to work together or one person is dominating the session, the moderator has to intervene in a session. However, it does not matter why

participants are not compatible, any intervention inevitably slows down the participants in any case.

Constructive interaction is especially effective for the testing of user interface designs with children since it is hard to get them to speak and to follow the instructions as in a normal thinking-aloud session. It does not depend on whether the children know each other. They may be more willing to work together than to try something on their own or with an unknown adult.

### 4.3 Formal Experiments

Some usability tests are aimed at identifying hard, quantitative data in controlled experiments with test users. Formal experiments are more or less thorough statistical analysis of objective measurements. Such measurement studies are important for assessing whether the usability goals of an implemented design have been met, or for comparisons of two or more competing products.

Formal experiments provide elementary statistical data by testing the absolute performance of an interface. Such user performance is usually measured by a large number of users performing several predefined test tasks and collecting time and error information. Typical quantifiable performance measurements collect objective and quantitative data, such as:

- Time users take to complete a specific task.
- Number of tasks completed within a given time limit.
- Number of errors.
- Ratio between successful actions and errors.
- Number of extra clicks from the optimal sequence of sub-tasks.
- Time spent recovering from errors.
- Number of commands/features used by the user.
- Number of commands/features the user can remember after the test.
- ... see Nielsen [1993] for an extended list.

It is important to ensure the validity of the obtained results, and that they are relevant to the system's usability in real world conditions. Validity problems often occur when *testing with the wrong kind of users*, when *testing the wrong tasks*, or when *not including time constraints or environmental influences*.

The absolute performance of one interface can be tested by running an experiment to objectively determine if an interface has met specific requirements. For example, one can measure how long it takes for a couple of expert users to perform a specific task. The result for this example would be an average time with a deviation of some seconds to perform the specific task.

Comparing two (or several more) interfaces can be done by running an experiment to objectively determine which of the compared interfaces is better, according to several specific criteria. When comparing interfaces, there are two ways of designing the formal experiment. A *between-groups* experiment uses independent, "equally-sized groups" of randomly assigned test users, where each group tests one system by performing identical tasks. A *within-groups* (or *repeated measures*) designed experiment uses one group of test users. The users are assigned randomly to two "equally-sized pools", where each pool tests all systems by performing equivalent tasks [Andrews 2006; 2011, Nielsen 1993].

## 4.4 Questionnaires and Interviews

Subjective data about the users' view of a system can be obtained by simply asking the test users after they have used a system to perform a couple of representative tasks. There are two useful query techniques for studying various aspects of usability, and how users use the system or what users particularly like or dislike of a system:

- Interview
- Questionnaire

Nielsen [1993] mentions questionnaires and interviews as indirect usability methods, since they are used to collect information about the users' opinions and not about the interface itself. However, they are direct methods when measuring the user satisfaction of the product. Both methods are very similar since they involve asking a set of questions and recording their answers.

A questionnaire is a structured form, printed on paper or presented interactively on a computer, that is filled out by the test user, and does not need to have any other

people present when answering the questions. Questionnaires normally are better for getting quantitative and qualitative data, can usually be used to reach an entire user population, and are easy to repeat. An interview is more flexible, since an interviewer can ask direct questions and probe interesting issues. He can explain difficult questions and rephrase questions if they are misunderstood by the respondent. Interviews are more time-consuming and are often harder to analyze quantitatively [Andrews 2011, Nielsen 1993].

## 4.5 Observational Study

Visiting users in their natural environment to observe them working is very important for task analysis and getting information about the usability of systems that have already been installed. An observational study is the simplest of all usability evaluation methods. An observer visits one or more users working on their own tasks and collects usage data. When conducting an observation, the observer does as little as possible so as not to interfere with their work and should stay quiet most of the time. Of course, it is possible to take notes or video recordings, as long as it does not bother any users. An advantage of observing users working in natural environment is that an interface sometimes is used in unexpected ways that a test in a planned experiment would not have covered [Nielsen 1993].

## Chapter 5

# Practice Guidelines

This chapter presents some practice guidelines that can assist designers in the iterative development of user interfaces for mobile touch devices.

### 5.1 Practice Guidelines for User-centred Design

One goal of the user-centred design process is to ensure that the final product fulfils the user's needs. UsabilityNet [2006] published 10 best-practice principles for designing usable systems, which can be tailored to fit specific needs. The following principles are taken directly from the source and can be accepted as generally sound.

#### **Design for the users and their tasks**

Interactive systems always exist to support users performing their tasks. A successful computer system is always a conglomerate of being user centred and task-oriented. Always bear in mind the characteristics of the user group, their real-world tasks, and their real-world working environment.

#### **Be consistent**

Make the behaviour of common interface elements and dialogue boxes as consistent as possible. This also means that it should be consistent with other existing components of the computer system. New styles of interaction, which are inconsistent with the rest of a system, can take users' time and effort to learn and get used to.

**Use simple and natural dialogue**

Interaction involving dialogue between the user and the system should always follow the natural sequence given by the task. Only information that is necessary to complete the current task should be displayed, to avoid unnecessary complexity of a dialogue.

**Reduce unnecessary mental effort by the user**

Users should be able to concentrate on their task. The more complicated the interaction with the system, the more frustrated users become. If users have to invest too much mental effort in working out how to operate, for example the computer, they will be inefficient and the users' error-rate will increase.

Simplify common tasks as much as possible. There should not be the need to remember information from one part of a system to another. Instructions on how to use the system should be visible and easily accessible.

**Provide adequate feedback**

Users should be informed at several levels of interaction. Users need to be confident that actions have been completed and whether they have been successful or not. If completion takes more than a second, a progress indicator should be displayed to give confidence that the system is still operating. A button should indicate immediately when it has been operated, and users should be informed when an operation - a longer sequence or just a short single operation - has been completed.

**Provide adequate navigation mechanisms**

Show the users where they are. This can be achieved by applying a meaningful and consistent mechanism for assigning titles to windows or pages and using location indicators such as cursor positions, scroll bars or page indicators.

Clear and easy routes between different windows that users need to access for particular tasks should be provided in an appropriate form for each stage of a task. Users often use system functions by mistake. There should always be a clearly marked note and an "emergency exit", such as a Cancel- or an Undo-button, to leave such an unwanted state without having to go through an extended window.



**Let the user drive**

Users should be able to select the information they need in an easy way to support each individual task. The system should provide as few constraints as possible when performing a task, and frequently performed tasks should be performed in a as simple way as possible.

**Present information clearly**

On-screen information should be arranged in a way that enables the user to differentiate between several items or groups of data easily. The system should not provide more information than necessary to perform a task. It should display information, e.g., feedback or error messages, consistently across different windows to enhance learnability.

**Be helpful**

Systems should be as self-explanatory as possible, so that the system can be used by its target users with a minimum of help. Information should be displayed in terms that the target user can easily understand. Help pages should relate to the specific context of the actual interaction.

**Reduce errors**

The users should be guided through the system to accomplish their goals and the system should validate user data as often as necessary. Error messages should be displayed in the target user's terms, precisely indicating the problem and offering a possible solution.

## 5.2 Practice Guidelines for Touch-screen Devices

[Haywood and Boguslawski 2009] did research into the users' experience whilst interacting with mobile devices, particularly with the iPhone. Based on their research, they offered a range of key best practice guidelines to help design and evaluate finger-operated touchscreen solutions. The guidelines aim to optimise the user experience by bringing qualities such as simplicity and ease of use, as well as consistency and responsiveness to the fore. The following summarizes some of the relevant key factors that need to be considered when evaluating or designing touchscreen user interfaces for small screen devices.

### Screen and icon design

- Do not overload a screen, so that everything remains legible.
- Icons should be suitably sized and spaced to avoid selection of nearby icons and screen elements.
- Also, ensure sufficient space between entries in a vertical list, to minimize mis-selection.
- Keep labels and instructions short and simple, and avoid abbreviations if possible.
- Make use of familiar icons and colour conventions so users can associate with them.
- To enhance visibility, force a high contrast between discrete touch elements, text, and background colours.
- Furthermore, controls and text should not be placed over an image or patterned background.
- Consider adopting a sans serif font for all text and labels to aid legibility on small screens.
- Carefully consider the design and placement of visual feedback so as to avoid fingers covering important information by placing feedback above the selected item.

### Navigation

- Present a clear and direct navigation to the Main Menu or “Home” area.
- Minimise steps to access or perform core functions.
- Ensure consistency throughout the interface (“learning curve”).
- Ensure that navigation and selection are easily discernible, to avoid accidentally making selections while scrolling.
- Allow actions to be readily reversible.
- Aim for short response times, as delays will frustrate and confuse users. Otherwise display information to show that the system is operating.

## Chapter 6

# Characterising Children

Children at different ages have extensively different physical, cognitive, and psycho-social characteristics. According to their age their likes, dislikes, interests, and their fears also vary.

In this chapter we will discuss children's congenital developmental capabilities and proclivities. The discussion below distinguishes three common age ranges, or stages of development for children: preschool children (two to five years), young childhood (six to nine years) and the tweens (usually ten to twelve years, in certain contexts from nine up to fourteen years). They have been adopted from Baumgarten [2003], who has based these stages on research in the areas of child development, psychology, education, and technology. Within these stages we discuss the development skills, needs and knowledge of children and supplement the summarized research of Baumgarten by useful content from Liebal and Exner [2011], Markopoulos and Bekker [2003] and Hanna et al. [1997].

### 6.1 Preschool: Ages 2 to 5

From ages two to five, children's growth rates slow down to about less than one quarter of what they were between birth and two (see Figure 6.1). Normally, boys are taller than girls and preschool children gradually lose their characteristic toddler look and begin to have a disposition to a lean and more athletic appearance of young childhood. The brain matures in this development phase and attains approximately 90% of its adult weight at the age of five. Children up to the age of six are slightly farsighted and show a right- or left-hand preference.

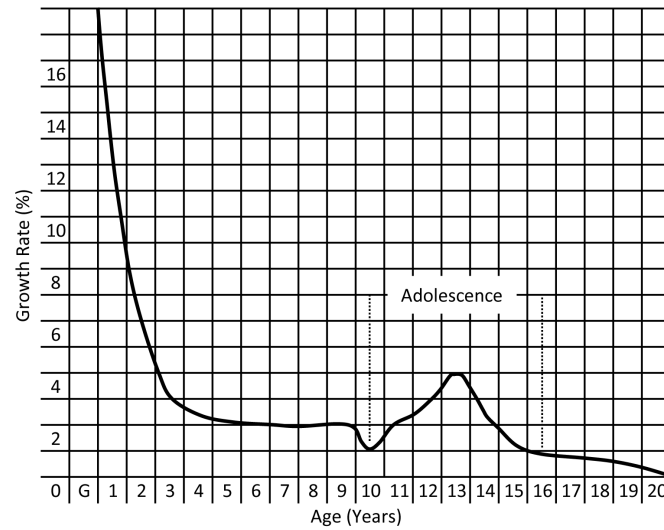


Figure 6.1: Children's growth rates from birth to adulthood [Liebal and Exner 2011].

Development of the brain in the preschool years leads to a major shift in cognitive skills. Children at this age develop their knowledge of symbols, words and language quickly, and their attention span and memory increases. But there are cognitive difficulties as well. Baumgarten [2003] describes a number of thinking problems (defined by Piaget), including:

- problem of concentration: inability to see more aspects of an object;
- egocentrism: difficulty in understanding another's perspective;
- animism: giving inanimate objects a personality;
- fantasy is equal to reality;

At this age, children learn about their social environment and social rules, have their first friendships, and develop a sense of who they are. Children begin to obtain an understanding of their gender and the characteristics of their gender group. It is a time of discovery, frustration and struggle. They test their limits and come face to face with the reality of their daily incapacity. Being close to home is mostly important and children at this age enjoy stories and myths.

Preschoolers are excited by computer activities that are interesting for them and that offer learning opportunities, fun and a sense of accomplishment. Computer games for this age group are often placed in environments with familiar and attractive characters, where they have to search for items that enable them to reach a goal.

Despite this, preschoolers are limited by their physical capabilities (their manual skills are not fine-tuned) in computer activities, have minimal tolerance for frustration or technical difficulties and their attention span is short. Any activities must be simple to access and adequately limited so that children in this age range can reach their objective in a relatively short time. The visual appearance should be clear and bright and feedback should be immediate and readily available at any time within the program. Simple directions, if necessary, should be verbal. The use of words should be avoided and the meaning of on-screen buttons should be depicted with symbols.

## 6.2 Young Childhood: Ages 6 to 9

As seen in Figure 6.1, the child's growth rate at this age is slower than in the preschool years or adolescence. Though, strength, physical ability, and performance in sports are increasing, and girls are less physically skilled than boys, often are heavier and have more fat tissue.

According to cognitive theory, interactions within the competitive schoolroom environment during the first years of young childhood lead to a significant improvement in thinking skills [Baumgarten 2003]. Between the ages of five and nine, children make a step-by-step shift toward and develop

- a sense of logic and reasoning;
- the ability to understand another's perspective;
- an understanding of the concepts of conversation and mathematics;

Children start developing a sense for simple abstract concepts and become very similar to an adult in logical thinking and memory capacity, primarily with the differences in knowledge and experience. Language skills increase and children develop an increasingly large vocabulary and understanding of words - "children shift from learning to read, to reading to learn" [Markopoulos and Bekker 2003].

School experiences during these years are important in order to reaffirm a sense of self worth outside of the home. There is a need for acceptance and success and the influence of the parents is shifted to a bigger influence from friends and (school-based) social groups. Children of this age group are developing their "child's play", from parallel playing in the preschool age to associative playing in the young childhood [Liebal and Exner 2011]. While

younger members of this age-group favour fun, simplicity and familiarity, older members of this group are inclined to prefer challenge and competition and are more ambitious.

Young children like the same computer activities as preschoolers do and they continue to prefer computer activities from which they can learn, have fun, and encourage self-confidence through challenge and reinforcement, although at a different and higher level of difficulty. The increasing physical and cognitive capabilities of this age group offer the possibility of using more complex interfaces, such as more functions under one button and nested menu-structures as well as the use of more complex and abstract language. A growing ability for logical thinking, a growing mathematical understanding, and the enhanced memory capacity, allows young children the ability to play memory- or strategy-based games and contests, and more complex number-based activities.

### 6.3 Tweens: Ages 10 to 14

The tween years are the years when children become increasingly independent of the adult world. Children at this age almost reach their full physical size. Girls are usually taller and look more mature than boys. This is the time of physical and emotional processes related to adolescence. Baumgarten [2003] describes primary and secondary physical aspects of adolescence:

- primary: maturation of the body with the development of secondary sex characteristics and a very strong growth spurt (see Figure 6.1);
- primary: sexual reproduction is now possible (primary sex characteristics);
- secondary: alterations in skin, hair and nails, and changes in fat deposits;

Cognitive skills are growing and according to Piaget, adolescents develop their abstract thinking and logical skills. Youngsters can handle complexity, abstract problems, and are able to reason morally. The appearance of adolescent egocentrism begins; adolescents believe that they are in the centre of attention above all others.

In the tween years, peers become more important as information sources and peer pressure imposes standards of behaviour, dress, and sexual identity to gain acceptance. A major change during adolescence is that tweens want to play a larger role in the decision-making process, and parenting becomes more of an interaction between parents and tweens. Fears of social or sexual unacceptability and rejection determine the adolescent years.

Tweens want to fit into the group or their peers, and the more they have in common with the group the better it is; they gravitate toward the same music, film and clothes.

Tweens like computer activities they can learn, grow and have fun with, whereas learning includes social learning, rather than academic. They like activities that appeal to sports and social activities. While male tweens favour games and activities involving sports and competition as well as danger and violence, female tweens still favour romance and relationship, beauty, clothes and trends. Tweens favour content that is “out-of-the-mainstream” [Markopoulos and Bekker 2003]. Games with a higher level of difficulty are preferred, and tweens demand games which require the use of logic, strategy and abstract thinking.

## 6.4 Summary

Finally we can say that children like things that

- are new
- they can learn from
- are easy to use and technically well executed
- that are forbidden or “out-of-the-mainstream”
- and are particularly fun.

## Chapter 7

# Usability for Children

The following chapter presents a summary of the “Handbook for ergonomic design of software and websites for kids”, written by Janine Liebal and Markus Exner [Liebal and Exner 2011].

For the development of software for children, the children’s preferences, aversions and needs have to be respected. Usability principles for adults are not necessarily ideal for children and have to be evaluated for suitability for young users. Many user interface designers believe that they have good ideas about expectations and perception of children. A closer look at these ideas makes it clear that these ideas are constructed on the basis of their own childhood memories, personal experiences with children and a social view of children.

Developing for children needs founded and extensive knowledge about the main target group, which can only be obtained by directly including the target users in the development process. In the following, a discussion about the user-centred design process with children is presented.

### 7.1 Child-centred Design

There are different approaches to how users should be included into the development process of a product. Traditional approaches propose to evaluate the functionality of a product at the end of the development process, but modern user-centred development approaches recommend that users are included as early as possible and throughout the whole process. As explained in Section 3.3, a modern user-centred design process should start at the beginning of a project, and be repeated iteratively during the whole development



process.

Children can play several roles during a development process: *users*, *testers*, *informers* and *design partners*. Each role is associated with a certain degree of influence in the development of new technologies.

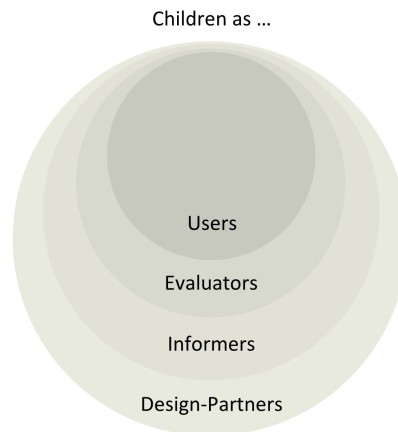


Figure 7.1: Roles of children in a development process [Liebal and Exner 2011].

### **Users**

When children are acting as users, they are observed and analysed by various methods by adults while they are spending time with existing technologies. This approach implies two main targets for developers: obtain data on how software effects the children's learning process, and on the other hand gain ideas for future technologies.

### **Testers**

As testers, children use prototypes of newly developed technologies. Goal-oriented questions may provide informative feedback about the usability of a new product for children.

### **Informers**

In a conversation with developers, children who act as informers may contribute important statements regarding the usability of existing technologies or any type of prototype from a children's view.

### Design partners

Children in the role of a design partner are broadly similar to that of informers. The key difference is that design partners are involved as equal partners directly throughout the whole development process.

## 7.2 Usability Testing with Children

The majority of testing methods have been developed for adults. There is a longer tradition of involving children as evaluators in usability tests for interactive systems for children. But, practices with children that have been reported in literature are manifold and there is no systematic account of which of these testing methods work on children. Druin [2002] argues that children can easily slip into the role of a tester, because they are asked to simply act as users, which is something they do in any case when they use the product [Markopoulos and Bekker 2003].

However, in practice when children are involved in usability testing, they have to deal with much more than using the system under evaluation: *becoming acquainted with the test environment, interacting with the test facilitator, and following instructions* to report their experiences. Therefore, an appropriate communication strategy, a careful task formulation and the social skills of the test facilitator are much more significant than when testing with adults.

Studies showed that it is useful to use several methods at once. As children are very talkative and eager at one hand, some are shy and quiet on the other hand, the test facilitator has to attune to the different test users and needs to intervene more sometimes and less at other times.

### 7.2.1 Preparations

#### Selecting the evaluators

As Nielsen explains in his book “Usability Engineering” [Nielsen 1993], it is possible to find about 75% of the usability problems by testing only four or five participants (see Figure 7.2). If children are the testers of the system, the number of subjects should be at least five, or if possible be increased up to eight participants. However, if it is hard to find the given number of participants, one child is always better than no children at all [Barendregt and Bekker 2005].

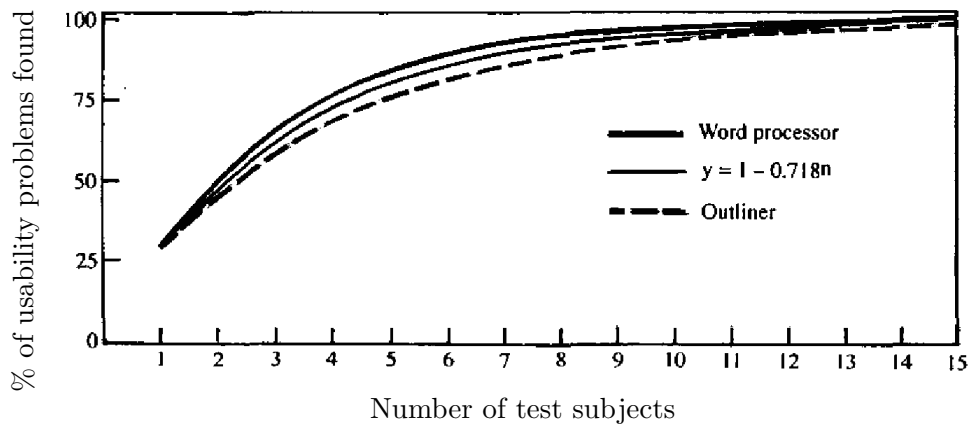


Figure 7.2: Usability problems found with various numbers of test users in a thinking-aloud test [Nielsen 1994b].

It is important to get much information out of the child. Thus, when selecting the children to be included in the usability test, a precondition is that children have some experience in interacting with the input device, because the test is usually carried out within a very short time-frame. On the other hand, children with too much computer experience, such as the children whose parents are working in IT occupations or children of members of the project team can be unrepresentative, unless they are part of the target user group.

Children in the age range of six to ten can be included in a usability test. They are able to concentrate on a specific problem or a specific task. But, it is hard for this age group to put their troubles, impressions and feelings into words. As many authors found out, children at the age of eleven are very easy to include in a usability test and are only slightly different from adult.

### Environment

As explained in Chapter 4, the test environment can be a usability laboratory or a quiet room that is prepared for conducting a test session. A familiar environment makes children feel more comfortable and gets them going much faster as well as letting them feeling much safer. It is important that the tests do not affect the children's daily routine. The disadvantage of a familiar environment is that the mobile recording equipment has to be pitched and disassembled and is not as unobtrusive as it would be in a laboratory.

A laboratory has the main advantage that the environment is the same for all participants. This improves the comparability of the quantitative data. Furthermore the

recording equipment is already installed and ready to use. To minimize negative impacts and indisposition of children, the laboratory should be suitably equipped for children. Small furniture or a colourful poster breaks up the atmosphere of a laboratory. But, the equipment should be selected carefully, so as not to distract the test users too much from their core task.

The presence of the parents in the test room is not necessary, but there must be a signed consent from the parents or a person in charge.

### **Consent**

It is necessary to always obtain a consent when testing with children. A declaration of the consent includes information and motivation for the test, information about recording the test and the option to allow the usage of the data. If necessary, a non disclosure agreement is also included in the declaration. The signed consent of the parents or a person in charge must be collected by the test facilitator before the test begins.






### **Questionnaires and interviews for children**

When preparing a questionnaire for children, Libal and Exner recommend that one consciously switches between open and closed questions. Open questions allow an unlimited number of answers, which can deliver extensive, but difficult to evaluate results. Closed questions, which include alternative answers, may hinder new and interesting ideas, but the quantitative analysis and acquisition of the gained data is easier. A mix of open and closed questions in a questionnaire produces a result including targeted information and new ideas.

Rating scales are a special form of closed questions, and are for direct self-classification. As shown in Subsection 3.1.2, questionnaires about the satisfaction of a product are typically very short and users are often asked to rate them with a 1-5 or 1-7 rating scales. If rating scales are used in questionnaires for children, they have to be specially prepared. Symbolic or graphical markers are particularly suitable for young children. Hanna et al. [1997] mentions that symbolic scales need to have meaningful start and end marks. Table 7.1 shows an example rating scale with symbolic marks in the form of smilies.

It can be observed that younger children tend to lean towards positive ratings in order to please the test facilitator and to receive praise. It looks as though children often do not pay attention to the meaning of the symbols but choose the one they like best. For young children three unique symbols with clear meanings may be enough. If applied to

---

				
5	4	3	2	1

---

Table 7.1: Example rating scale for older children with symbolic marks and values from 1 to 5. A rating scale from 1 to 3 with the symbols one, three and five are best to use for younger children, to ensure unique answer options. [Liebal and Exner 2011]

Table 7.1, number one, number three and number five are best to use for this target group. When completing such questionnaires with young children, the test facilitator should be present to explain misunderstood questions and provide helpful support.

Typically, an interview has the aim of finding out personal information or facts. Similar to questionnaires, an interview makes the distinction between open and closed questions. The fact that questions in a closed interview are prepared and equal for all respondents makes the quantitative analysis of an interview easier. An interview with a single child can easily be intimidating and feel like an examination situation. Such a stress situation often goes along with an inhibition of children, which means that children may not give open answers or their opinion. In many cases they cannot remember their favourite games or hobbies. An extended period of acclimatisation for the child is recommended when conducting an interview with children, so that the child can get to know the interview-partner.

### 7.2.2 Methods

Statements, comments as well as the mimic and gesture of an evaluator during a test are important tools and key sources of information about the usability of a product. The thoughts of the evaluators clearly deliver background information about interaction problems. There are several compatible techniques that are applicable for usability testing with children.

#### Thinking-Aloud

This method is used to encourage children to verbalise their thoughts and perceptions loud when performing the evaluation of the user interface (see Section 4.1). It is very useful if this method is demonstrated by the facilitator using an example. If the product is very complex and requires a high level of concentration from the children, it is very difficult for

children to verbalise their thoughts at the same time. Young children particularly have difficulty doing several things at once. As a result, either verbalising their thoughts or performing the task will become of subordinate importance. As Barendregt and Bekker [2005] wrote, a further fact is present: speaking without another person answering is very unnatural for children.

### **Active Intervention**

The child is questioned about his approaches and actions by the facilitator while performing the tasks. The questions are selected in the run-up to the test and are task-specific. Every child should be told about the questions before the test. As children do not have a broad, systematic perspective on the product, the number of statements depends on the facilitator and the number of predefined questions.

### **Retrospection**

Within the retrospection, the test is recorded with a video camera so that it can be analysed after the session. For this, the facilitator and the child watch the recording after the test session and the child is asked about his interaction and any problems he encountered with the product. Children often cannot remember specific thoughts from the test, afterwards. As watching the recording takes as much time as the test itself, the limited attention span of children influences the memory capacity too.

### **Co-discovery**

As discussed in Section 4.2, constructive interaction involves two participants simultaneously performing specific tasks with the product during a usability test session. The test situation is more natural if young participants communicate with other children than speaking to adults. The dialogue between children when performing specific tasks becomes an important factor for understanding how young users work through problems when using a product. In order to ensure the children are working together, the facilitator must tell children that the test is not a competition between them, but that it is very important that they find the solution together. For this reason it is necessary to give each child a specific task which should be performed together. Children of this age group are developing their “child’s play”, from parallel playing in the preschool age to associative playing in young childhood (see Section 6.2). Thus, children from the age group six and above should be able to work on tasks in cooperation with others. But, experience has

shown that children that are observed in a test situation often work alone instead of in cooperation.

### **Peer Tutoring**

This method implies that children work on their tasks in two phases. In the first phase, the child gains experience of the product by performing the tasks in the usual way. In the second phase, the child is encouraged to explain to another child the functionality of the product, and how to use it. The main benefit lies in the fact that children have to slip into both roles: the role of a tutor and of a tutee.

It is the only method that is especially designed for the implementation of a usability test with children. It has some very important benefits:

- a more natural conversation between tutor and tutee than by thinking-aloud
- cognitive load is divided between two children
- the tutee can fully concentrate on the task, while the tutor focusses on the communication
- the tutor teaches the tutee and answers the facilitator's questions simultaneously
- the facilitator can thereby see the test situation through the other child's eyes

Children in the role of the tutee are often quiet and listen to the instructions of the tutor. They do not ask questions at all. On the other hand, a tutee that switches into the role of the tutor afterwards, often becomes lively and talkative. But experiences show that children who are acting in the role of the tutor only explain things they like or understood or of which they believe that they are important.

### **Observation**

As children generally tend to make positive statements rather than negative ones, observations with camera recordings should be preferred. Negative emotions can be clearly identified by body language, mimic and gesture. So on one hand, a child that seems bored, or turns away from screen shows a very distinctly recognisable indication for this. On the other hand, smiling or leaning towards the screen are clear signs of a positive rating of the product.

### Post-task interview

Baauw and Markopoulous [2004] and Barendregt and Bekker [2005] recommend conducting an interview in combination with a questionnaire after the test. The answers from the questioned children do not have to correlate with the observed body language, as children usually tend to give positive answers and ratings about a product.

## 7.3 Practice Guidelines for Usability Testing with Children

Several guidelines exist that should be of interest when conducting a usability test with children. The following practical guidelines are a summary of Hanna et al. [1997], Barendregt and Bekker [2005], Hadj-Karim-Kharrazi [2005], Zaman [2006] and Liebal and Exner [2011].

- **Build up a relationship with the children**, when you first meet them. Try to find out more about them and let them talk about their favourite (computer) games, films, or their favourite subjects or sports at school. This helps them to relax and breaks up the test situation. The dialogue between the test facilitator and the children should be on an equal footing.
- **Introduce children to the test situation**. It should be explained to every child that it is not them being tested but the product. A small script can help to introduce all children to the test situation equally, but too formal a proceeding should be avoided. An example of a possible script is published by Hanna et al. [1997]: Even though we call this a test, we are not going to test you at all. We are asking you to help us to test and rate our software. We want to see what is too hard or too easy for children your age, so that we can make it better. We will ask you to try on your own most of the time, but are here to help you if you get stuck.
- **Motivate older children** by underlining their role as a tester. Tell children that you have forgotten what children like and you need help to figure that out to make a good product. Point out that they can strongly influence the development of the product design.
- **Do not disseminate false expectations**. Explain in an appropriate way what children have to expect and why it is so important to know their opinion.



- **Allow children time to practice with the product** before they perform the test tasks. Many usability problems occur when working on real tasks. Such problems can be found more easily if children are a bit familiar with the structure and functionality and they can work more effectively on the tasks.
- **Children often want to have their parents present in the room.** It is recommended that the test facilitator is inside the test room to support children mentally. Children under five years may need their parents in the room. Siblings should be split so that they are not distracted too much.
- **Calculate a suitable failure rate.** Deficits in the development of children, or a lack of motivation can cause a high level of failure.
- **Give a helping hand.** Children tend to ask for help if they get stuck, or are not sure what to do any more. Encourage them to try again whilst trying to answer with counter-questions.
- **Do not ask “yes-no” questions.** For example, when they are asked if they want to do a task, they can say no. Use phrases like: “You need to do...”, or “Let’s do...”.
- **A test session should not exceed one hour.** Children easily tire and may need a break.
- **Prepare children for the end of the test session.** For children it can be difficult to finish a test, especially when testing games. Make it clear to them that it’s time to stop the test after the specified limit or goal.

## Chapter 8

# Usability Test

This chapter presents information and data on the implementation of the usability test of the fully implemented prototypes of Catroid and Paintroid (see Section 2.1).

### 8.1 Introduction

One important part of the practical work accompanying this thesis was done by conducting a usability test of the prototypes of Catroid and Paintroid at their then current state of development. The development of these products is done using agile software development methods. Both apps are developed with state of the art technologies, are under heavy development and things change rapidly and often. Thus, the usability test was conducted to obtain feedback on the usability at their current stage of development on the one hand and find parts of the software that were prone to bugs on the other hand. It was also useful to gather feedback on how satisfied users were with Catroid. All in all, the expectation of the test was to find out whether the development of the user interface for younger children was heading in the right direction or not, and what one in the Catroid usability team should pay attention to when testing Catroid with young children. From this perspective, the test did not include a performance measurement of the user interface at all.

The test was performed with a total of eleven children and conducted using thinking-aloud in combination with active intervention in a constructive interaction setting (see Section 8.3). As seen in Table 8.3, one pilot test with a single child and five co-discovery test sessions with two children participating were performed. The pilot test was finished twice as fast as the other tests. The thirteen year old boy in this test worked through the predefined tasks in about half of the estimated time. The duration for a single test was

estimated to be around 50 to 60 minutes, with a break of 15 minutes halfway through.

In the following, the software and the usability evaluation method is briefly described. Then, a closer look at the goals and the test setup is provided. The test setup includes the discussion of the environment, the team, the equipment, the participants, the procedure and the tasks. The closing will be a short discussion about the post-test interview and questionnaire during and after the test.

## 8.2 Catroid and Paintroid

### 8.2.1 Catroid

As explained in Section 2.1, Catroid is a visual programming system that allows the creation of different types of interactive content. Such interactive content was called a *Catroid program* and could be for example a game, a music video, an multimedia animation, etc. When a new Catroid program was created, a look for the *background* had to be defined. A background appearance was set by assigning an image to the background object. This was done with the “*Set background*”-command. Thus, it was possible to define several other *objects*, which then were the actors of the particular Catroid project. Every object could have several scripts that were responsible for the object’s behaviour, look and sounds.

The program version of the evaluated prototype for this test was Catroid 0.5.a.

### 8.2.2 Paintroid

Paintroid was the image manipulating program that was developed for the use with Catroid. It involved several functions for creating or altering images. The main goal of Paintroid was to provide a small common tool that allowed the manipulation of images. The program version of the evaluated prototype for this test was Paintroid 0.5.1.

## 8.3 Method

The practical usability test was done using a combination of the thinking-aloud method with active intervention, and children working on their tasks in a constructive interaction setting (see Section 7.2.2). Constructive interaction seemed to be the most natural setting for children at this age and was applied with two children working simultaneously on the tasks during the usability test. The participants were encouraged by the facilitator to communicate with each other. The dialogue between them, and the observation of their

inputs on the touch screen became the most important factors for understanding how they worked through problems. The facilitator only prompted the participants when they stopped verbalising any communication or thoughts. Active intervention came into use when children got lost in the user interface, or when they tended to use only a small proportion of the product's user interface.

Figure 8.1 shows two boys working together on their tasks. The facilitator is observing the evaluation close to the action in order to be ready to prompt the evaluators anytime it was needed.



Figure 8.1: Test facilitator is observing two boys working on their tasks. He is ready to prompt the users if needed.

## 8.4 Goals

Two main goals were recognised by this usability test: how to do usability testing when children are included into the evaluation process, including what attention should be paid to particularly address the needs of children during a test and what should be avoided. The second main goal was to get feedback about the usability, possible runtime errors, and what should be done to improve the usability of the interface over all.

The following list represents several questions that it should, at least partially, be possible to answer once the results have been received.

- How well is the programming environment understood generally?
- Which parts of the interface include hard problems for children of this age?
- Which functions are used by the young users, which are not?
- What do they like?
- What do they dislike?
- What feedback and rating do they give for the Catroid software environment in all?
- How should usability testing be performed when children are the participants?
- What attention should be paid to address the needs of children during such a test?
- What challenges arise for supervisors of usability tests with children?

Therefore it was necessary to find a suitable environment, assemble a test team, set up recording equipment, and select the participants and suitable tasks for the test.

## 8.5 Setting up the Test

In the following the preparations for the test are discussed.

### 8.5.1 Environment

The series of tests were conducted at the premises of Graz University of Technology. As a fixed usability laboratory did not exist, it was a challenge to find a quiet room that was suitable. A classic test setup is seen in Figure 4.4. The co-discovery setting in this test involved two users working together on one device. Two main challenges had to be managed: first, this setting did not allow the use of a mirror for visualizing the mimic and gesture of both users, and since the test was conducted on a mobile device, it was difficult to record the screen and the inputs on the touch screen for later analysis.

It was necessary to use two cameras at once to capture the screen and the inputs on the touch screen and the faces of the users (including their mimic and gestures). For the recording of their faces, one camera was mounted in front of the test users. The first idea for a second camera for capturing the touch inputs on the screen of all test users was to use a helmet video camera. But, this is not a good approach when testing more than one user simultaneously, since it is not ensured that both users always keep their view on the

screen. The final decision was to mount the camera on a tripod and to record the screen from a higher point of view (Figure 8.2).



Figure 8.2: Usability laboratory especially built up for the test sessions. A web cam was mounted in front of the test users. A second camera was mounted on a tripod next to the users' seats. To ensure a consistently good view of the screen, the test device was fixed in a holder on the table.

### 8.5.2 Team

Besides the test facilitator, the test team consisted of several members. Two video and computer operators were responsible for the recording equipment, particularly at the beginning and the end of the test. During the test they were also acting as observers and data loggers. One additional observer and data logger was present during the test period.

### 8.5.3 Equipment

The test device for the usability test was the Samsung Galaxy Tab 10.1 Android device and had the following specifications:

<i>Type</i>	Samsung Galaxy Tab 10.1 P7500
<i>Display</i>	10.1" WXGA TFT-LCD
<i>Resolution</i>	1.280 x 800 Pixel
<i>CPU</i>	1 GHz Dual-Core Processor
<i>Operating System</i>	Android 3.1 (Honeycomb)

Table 8.1: Specifications of the test device, a Samsung Galaxy Tab 10.1.

The device was mounted on the table (see Figure 8.2). A starter kit that contained the basic equipment for a usability test was provided by the Institute for Information Systems and Computer Media at Graz University of Technology. This usability kit (Figure 4.3) included most of the equipment that was used for the test. Two video sources were used for streaming. The camcorder from the usability kit streamed the screen including the touch inputs of the test users. As seen in Figure 8.4, an additional Logitech USB webcam was mounted in front of the device to stream the capture of the faces of the users. The video streams were the video sources for the screen recording software, running on a desktop computer. An external microphone streamed the users' voices. Morae by TechSmith<sup>1</sup> was used to capture and record the combined audio-video stream, including the screen, the faces and the voices (see Figure 8.3).

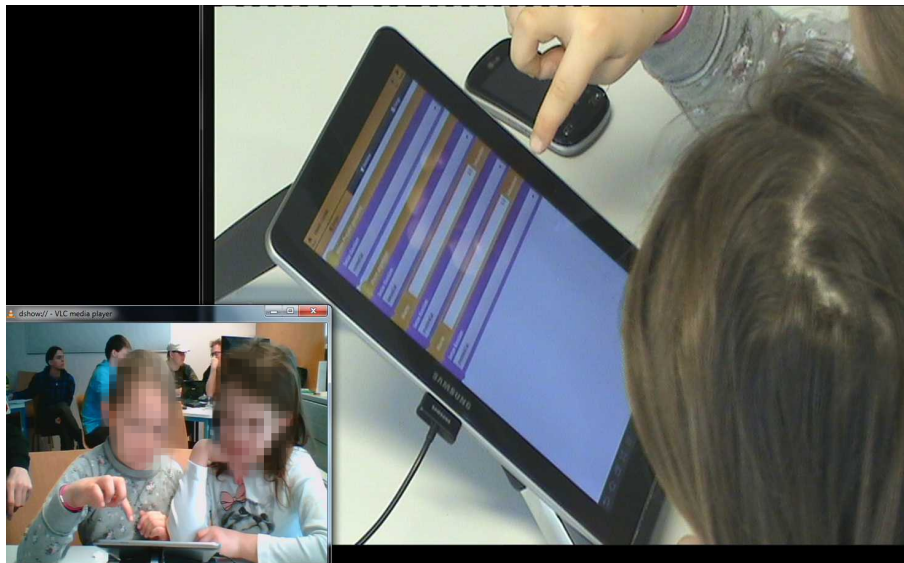


Figure 8.3: Two video cameras were used to record the usability tests. A webcam and a video camera acted as sources for the Morae screen capturing software.

The first plan included the recording of a screen stream, which should have come directly from the test device. But, this idea was rejected after many issues arose: the

<sup>1</sup><http://www.techsmith.com/morae.html>

most important issue involved massive performance problems of the recording during the test run. As these performance issues would not have been resolved in due course, the idea was discarded.

#### 8.5.4 Participants

Eleven children, aged from eight to thirteen years, participated in the usability test. The children came from different schools in the wider area of Graz, Austria. One pilot test with a thirteen years old boy was performed. The distribution of girls and boys participating in the rest of the tests was half and half. Table 8.2 is a short overview of the number of participants in each test and Table 8.3 contains further information about the participating test users. The data was collected with a background questionnaire at the beginning of each test session.

Test	Age	Gender	Setting
Pilot Test	13	male	single
Test 1	8 / 9	male / male	pair
Test 2	9 / 8	female / female	pair
Test 3	10 / 10	male / male	pair
Test 4	9 / 10	male / female	pair
Test 5	8 / 8	female / female	pair

Table 8.2: The table presents the number of users and the test settings.

As seen in Table 8.3, the children were mostly between eight and ten years old. More than 90% owned a mobile phone or were able to use one at home, and about 45% of them used their mobile phone either for playing games or for watching YouTube videos. Both activities require an Internet connection. In comparison to desktop computers, 100% of the children that were asked had experience using a mouse and keyboard, and 27% of them had some programming experience too.

#### 8.5.5 Procedure

The test procedure for each single test session was as follows:

1. Check settings of test device, cameras, microphone, and screen capture software.
2. Pan the camera to the welcome area and start recording software.
3. Greeting of children and person in charge in front of the door.
4. Enter the test lab together.



Test name	Pilot 1	Test 1	Test 1	Test 2	Test 2	Test 3	Test 3	Test 4	Test 4	Test 5	Test 5
Test number	1	2	2	3	3	4	4	5	5	6	6
Test setting	Single	Pair	Pair	Pair	Pair	Pair	Pair	Pair	Pair	Pair	Pair
Date	23.02.2012	27.02.2012	27.02.2012	28.02.2012	28.02.2012	29.02.2012	29.02.2012	01.03.2012	01.03.2012	02.03.2012	02.03.2012
Time	12:22	09:30	09:30	09:35	09:35	09:45	09:45	09:30	09:30	09:13	09:13
Name *	Andrew	Bart	Charlie	Denise	Elisa	Felix	Gabriel	Harry	Iris	Jaimie	Karen
Gender	male	male	male	female	female	male	male	male	female	female	female
Age	13	8	9	9	8	10	10	9	10	8	8
School	Secondary	Primary	Primary	Primary	Primary	Primary	Primary	Primary	Primary	Primary	Primary
Mobile phone	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	yes
Internet/Games	yes	yes	yes	no	no	no	yes	no	no	no	yes
<b>Experience</b>											
Computer	3 years +	1 year	1 year	2 years	1 year	3 years	3 years	2 years	3 years	1 year	2 years
Programming	yes (scripts)	no	no	no	no	yes (Lego)	yes (Lego)	no	no	no	no
Usability test	no	no	no	no	no	no	no	no	no	no	no

*\* Names have been changed to ensure anonymity*

Table 8.3: Summary of the background information of the participating evaluators received by interviews and background questionnaire. As seen in the table, the children were mostly in the age range of eight to ten years old.



Figure 8.4: A boy and a girl are working together on test tasks. A USB webcam that was mounted in front of the table, was used to capture the faces.

5. Introduce each member of the test team by their first name.
6. Obtain the parents signed consent from the children or their person in charge.
7. Motivate children and make it clear that it's not them being tested but that they can test and rate the program.
8. Tell the users what they should do during the test.
9. Do the background questionnaire together with the children and fill out the template.
10. Pan the camera to the test area.
11. Explain the test area and the test device.
12. Show the participants to a seat.
13. Do the short training exercise with the users.
14. Guide users through the program and show all parts of it, without comments.
15. User(s) start the test tasks.
16. Conduct an interview.
17. Fill out feedback form together with the user(s).
18. Thank the users and hand out a Catroid sticker.
19. Stop recording.

### 8.5.6 Tasks

Catroid is a visual programming language that allows the creation of interactive content such as multimedia animations, games or other interactive content. Depending on the user's creativity, there are a variety of ways to develop programs with a minimum of the provided functionality. The test users should be motivated to use a wide range of the given functions and to create a simple but fully adequate project step by step. For that, the test team decided to work out a simple task list. This list involved several predefined tasks to guide the test users through most of the main functions and features of the application, but leaving them the choice of how to attain a goal in some cases too.

As seen in Table 8.3, the participating users were both, girls and boys. Therefore, the tasks were developed to be used with two different topics: one dealt with a simple

rocket launch and the other was about a mystic fairy (see Figure 8.5). The template that is presented in the following was a first idea for a proper topic and is about an animal story. The subject was exchanged either with the rocket launch or the mystic fairy in the individual test session. The first task was an introductory task which was done together with the test facilitator.

**1. Short training with the gallery app (1-2 minutes)**

Explain “co-discovery” and “thinking-aloud” by demonstrating with the gallery app, and then let the participants practice for some time.

**2. Get to know the program and gather some first impressions (10 minutes)**

You can try Catroid for a few minutes. Try everything and anything.

**3. Create a new project and start it (2 minutes)**

Create a new project

Name the project

Start the project

**4. Create an animal (10-15 minutes)**

Find out how to create an animal. The users have 2 ways of programming it:

**(a) Create a new one (15 minutes)**

Find out how to insert an animal, give it a name and an appearance.

*If child can't continue, ask : “How would you insert a new costume?”*

*if child can't continue: Give more hints, else help.*

Find out how to set the appearance.

*If child can't continue: 1. You need a script.*

*If child can't continue: 2. Which script could it be?*

**The child might insert the script at a wrong location.**

Find out how to move a brick.

*If child can't continue: What would you do on your cell phone?*

*If child can't continue: Give more hints, else help.*

Find out how to delete the object “Cat”. If you found out, delete the object “Cat”.

*If child can't continue: How would you delete something?*

*If child can't continue: Give more hints, else help.*

**(b) Replace an existing item (10 min.)**

Find out how to change the appearance of an object. Make a new animal out of the existing one.

*If child can't continue, ask: How would you insert a new costume?*

*If child can't continue: Give more hints, else help. If child can't continue: 1.*

*You need a script to assign the new appearance. 2. Which script could it be?*

**5. The animal moves (10 minutes)**

The animal should move if you tap it. Find out how the animal can move.

*If child can't continue: 1. You need a script 2. Which script could it be? 3. What exactly do you want to do?*

**The child might insert the script at a wrong location.**

Find out how to move a brick

*If child can't continue: What would you do on your cell phone?*

*If child can't continue: Give more hints, else help.*

**6. The sound of the animal (10 minutes)**

The animal should make a sound if it is tapped. Find out how the animal can make a sound.

*If child can't continue: 1. You need a sound. 2. How would you insert a sound?*

Find out how the animal can make a sound if it is tapped

*If the child can't continue: 1. You need a script. 2. Which script could it be?*

**The child might insert the script at the wrong location.**

Find out how to move a brick

*If child can't continue: What would you do on your cell phone?*

*If child can't continue: Give more hints, else help.*

**7. Change Background (5 minutes)**

Find out how you can insert a new background.

*If child can't continue: How would you insert a new image?*

Children in this age range are not able to concentrate for a very long time span. Thus, appropriate time limits were assigned to each task. These predefined limits gave the team a feeling for the estimated time span for the whole test. However, as the time limits were not established for performance measurements, they were not set in stone. Children that were motivated to work through the problem on their own got extra time. When children got bored or lost attention while solving a problem, they were helped to complete the task.

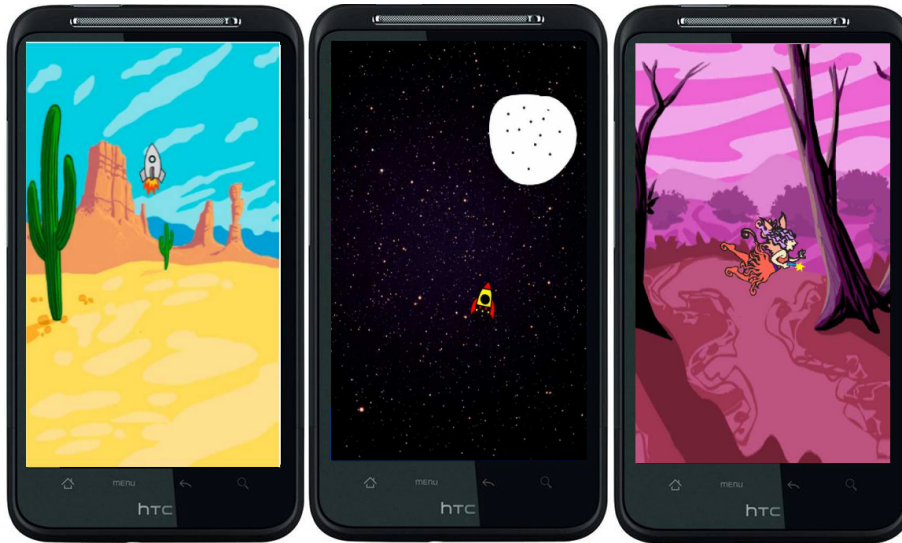


Figure 8.5: Three different Catroid projects, which were a result of the usability tests. The results varied by what children worked on during tasks. The rocket launch project on the left was made together by two boys. The project in the middle was developed by a thirteen year old boy. He developed an additional second scene to the rocket launch story, where the rocket landed on the moon. The wonderland project on the right was made by two girls working together.

## 8.6 Post-test Interview

### Interview

The plan was to do a short interview after the test to deliver feedback about the test and the program. Several questions were prepared for the post-test interview.

- How was the test?
- What parts of the program did you like best?
- What did you dislike most?
- What was hard for you to do?
- What was easy for you to do?

The plan for the interview was to let the children talk as freely as possible about their experiences. The objective was to collect information about the users' opinion and therefore to receive as much subjective information as possible.

### System Usability Scale (SUS)

A modified version of the so-called *System Usability Scale* was used to create a global assessment of the system's usability. This simple usability scale is a ten-item *Likert*-scale that gives a global view of subjective assessments of usability [Brooke 1986]. The scale consists of ten selectable items that cover a variety of aspects of system usability. It is generally used directly after an evaluation, before any debriefing or discussion took place. Respondents should immediately answer each item, rather than thinking about items for a long time. All items should be checked. If a respondent cannot respond to a particular item, the centre point of the scale should be marked.

The result of the usability scale is a single number that represents the measure of the overall usability. The calculation is done as follows:

- Calculate the score contributions from each item. Each score contribution will range from 0 to 4.
- The score contribution for items 1,3,5,7, and 9 is the scale position minus 1.
- The score contribution for items 2,4,6,8, and 10 is 5 minus the scale position.
- Multiply the sum of the scores by 2.5 to obtain the overall value of the system usability scale.
- SUS scores between a range of 0 to 100.

The original items for the usability scale were developed in English language. For the use within the evaluation of Catroid, the items were translated into German language, and were adapted in cooperation with a German teacher of the desired age range. The main focus of the translation was to keep the original meaning of the items unchanged.

## Chapter 9

# Results and Recommendations

This chapter outlines the findings of the usability test. The findings regarding the usability of Catroid and Paintroid are discussed first, and are followed by findings about usability testing with children in general. The negative findings presented are valid for all screen sizes. Most results are explained using screenshots of the respective parts of the software, whereby some were taken from a mobile phone and, if necessary, some were taken from a tablet computer.

### 9.1 Findings Regarding Usability

The observation data from the thinking-aloud tests that relates to usability events was marked with time stamps during the test, and thoroughly analysed afterwards by the test facilitator. The facilitator examined all events and extended them by a type (positive, negative or annotation) and comments. Finally, similar positive and negative impressions were grouped together towards a review phase. The findings of this analysis will be discussed below, and are presented with recommendations for the discussed findings and problems. The most important main findings are discussed first, and are followed by a presentation of several positive and negative findings.

#### 9.1.1 Main Findings

The Catroid programming environment was very well received overall by the test users, and all of them marked Catroid as a great program that is fun to use and one they would like to use again. However, one of the main problems for the children who tested Catroid was being able to understand the logical structure of a Catroid project, including it's

corresponding objects and their properties. Thus, the most difficult thing for evaluators was to find out how programming is done with the interface. The participants mentioned this relatively often by using the keyword “*difficult*”. The tasks for the tests were designed so that each part of the user interface was considered step by step. During the tests, the observers were under the impression that the participants often found it hard to understand where they were located in the user interface. Associations of several objects, for example the background object and a simple cat object, were not clearly recognized, and the belonging of scripts, costumes and sounds to an object was not really spotted.

### Catroid

Some of the tasks seemed to be a big challenge for users. Furthermore, it sometimes seemed that the participants had no idea what to do. For example, when users navigated to the properties of an object for the first time, they were overstrained by the barrage of information they saw. Figure 9.1 shows the properties of an object which were presented by three different tabs: scripts for the behavior, costumes for the appearance, and sounds.



Figure 9.1: Catroid: three properties tabs of the object that is named “Catroid”. The image on the left shows the scripts tab containing several scripts. The image in the middle shows the costume tab with several costumes inserted. The image on the right demonstrates the sound tab.

Every tab-view contained a lot of functionality. Thus, these views were very complex. The first thing that a user saw in an object was the scripts-tab view, and this view was



usually very confusing for the test users. The users mainly tended to switch to another tab immediately, after they noticed that there were more tabs to pick.

About half of the test users did not use the *New*-button (Figure 9.2) without any help or prompt. It was not recognized as a useful button at all. Instead of using the New-button to create a new costume that was needed to perform the tasks (see Subsection 8.5.6), the test users tried to insert a new image for a costume with the *Import*-tool of Paintroid (Figure 9.6), the image manipulating program for Catroid. It did not seem to be clear that using Paintroid was the slowest, most complex, and in fact also not the usual way to insert an image into Catroid. Paintroid was more or less an extended possibility to insert or simply to alter an image. One of the main strengths of Paintroid was that it was very easy to create images with transparent areas. However, the functionality behind the new-button was very important when using Catroid. It was needed to insert either a new script, a new costume, a new sound or a new object.



Figure 9.2: The “New” buttons which are used to insert either a new brick (A), costume (B), background (D), sound (C), or an object (E).

Almost all users that attended at the test had problems with the concept of inserting and assigning either a costume, or a sound to an object. The concept was the same for both types of media: a *set*-brick and a costume (or sound) had to be inserted. This could be done by inserting a costume first, followed by inserting the brick, or vice versa. Finally, the desired costume was assigned to the object by selecting it from the select-list of the brick. Figure 9.3 demonstrates the three views.

When users wanted to do this the first time, they did not know that both items were needed. Some tried to insert a brick, and were confused about the given string “Nothing...”, as seen in Figure 9.4. On the other hand, some test users managed it to insert a costume, but did not know why it was not shown on the stage. There was no sign, no instruction or note about what to do next. Thus, none of the test users did it right without the help of the facilitator.

The recommendation here is, that the connection of these, of course, very important elements of an object should be made more visible. One could be supported with this procedure by the help of a short, automated program sequence. For example: if a new costume was inserted, the program could start an automated play-back, which shows the



Figure 9.3: The “Set costume”-procedure. Left: “Set costumes” bricks exist. Middle: various costumes exist. Right: several entries in the list selectable costumes.

next few steps which are needed to finalise this procedure. It may switch independently to the scripts view, insert a *Set costume*-brick and open the select-list without a user’s intervention, or on the other hand, it could switch to the costume view and may open the insert-a-new-custom dialogue by itself, depending on what was inserted first. Furthermore it is recommended that the New-buttons are redesigned to ensure there is consistency across all screens.

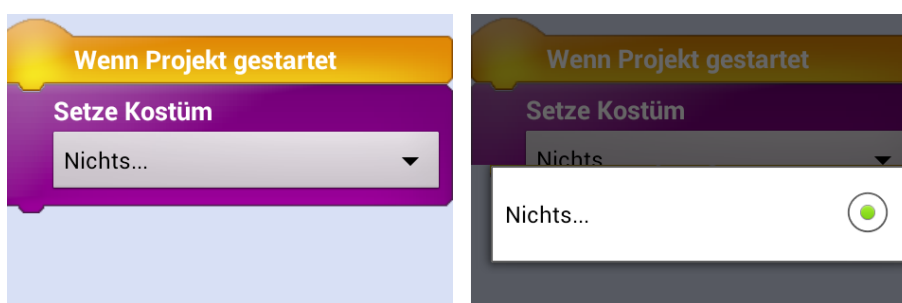


Figure 9.4: The “Set costume”-brick. Left: if there is no costume assigned to this brick, the text in the select box shows “Nothing...”. Right: the select box reveals a list with no entry.

### Paintroid

There was one thing that all users did the same way: they took the line of the least resistance through the program, which guided them directly into Paintroid. Paintroid is a standalone drawing program that is used to manipulate images in Catroid. The test users tended to spend much of the test time in the drawing program and the test facilitator had to prompt them very often to switch back to Catroid. There are three ways how to start Paintroid directly from Catroid. As seen in Figure 9.5, in the evaluated version of the

Catroid program, it was very easy to jump into the drawing program by clicking either the New-button (A), the thumbnail (B), or the Paintroid-button (C).

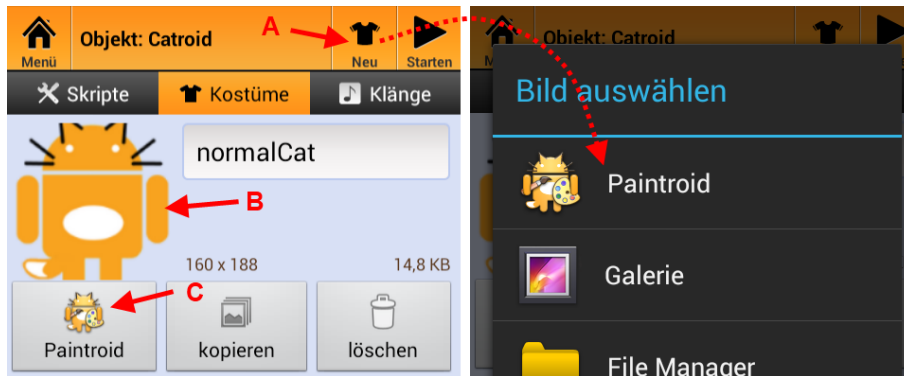


Figure 9.5: Three ways how to start Paintroid in Catroid.

There is much functionality integrated into Paintroid. First of all, it was observed that the usage of some of these tools were not clearly understood and the visibility of how to use something was not thought out very well. And, they were not self-explanatory for some users, but very mystifying when they tried to use them. For example, users did not understand what the *Stamp*-tool was, and that they must double-click on the screen to switch to the execution mode. When they drew with the stamp by accident, they were confused. Furthermore, they often thought that it was the drawing tool, and as a consequence they did not try to use the brush for a while. The same problem regarding the visibility of how to use a tool occurred with the *Pointer*-tool (Figure 9.7).

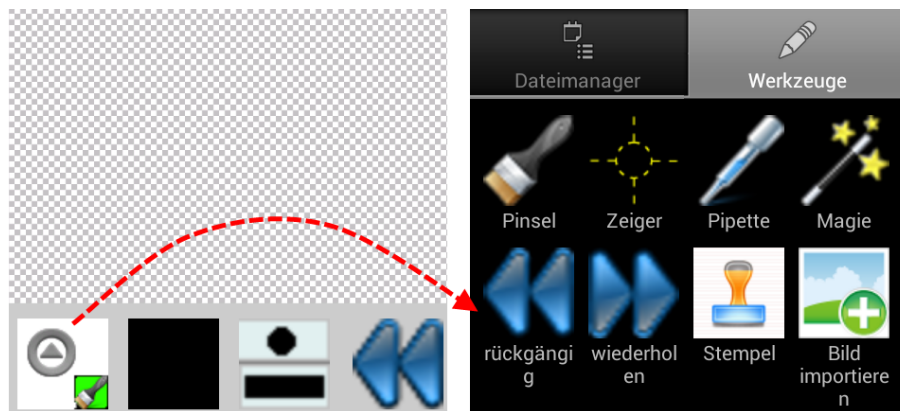


Figure 9.6: Paintroid: tools menu.

The *colour picker* dialogue opened when the coloured square in the Paintroid-menu (see Figure 9.6) was clicked. As seen in Figure 9.8, the tool opened with the most complex

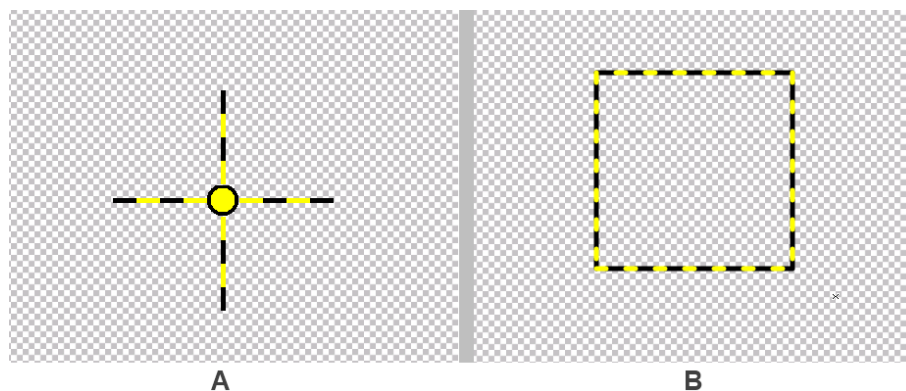


Figure 9.7: Paintroid: pointer (A) and stamp tool (B).

version of the three different colour pickers. This picker has confused several users so that they were somewhat scared and closed the dialogue immediately. The picker worked as follows: a new colour had to be confirmed by clicking the *New colour*-button, which indicated the chosen colour. The confirmation closed the picker. It is recommended to re-order the colour tabs and use the easiest colour picker with predefined colours first. Furthermore it is recommended that the colour is selected immediately every time a colour in the picker is changed without clicking the confirmation button. The two colour buttons, *Old colour* and *New colour*, may only have the functionality of switching between these colours in the picker, and the picker may close when touching out of the picker area or clicking the Android's back button.

Figure 9.9 shows the screen for changing the shape and width of the drawing stroke in Paintroid. It was not possible to select both, the shape and the width at once. Whenever either the desired stroke shape or width was clicked, the dialogue closed immediately. Thus, the dialogue had to be opened twice when a user wanted to change both attributes. It is recommended that the behaviour of this dialogue box is changed so that both attributes can be selected without closing the dialogue after each action.

### Software Errors

Several software failures and bugs occurred during the test sessions. Script bricks were invisible suddenly without apparent reason or were not visible after they were inserted into the scripts list. Another issue was that bricks jumped to the end of the list after they were picked up. Such behaviour made the test users confused so that they thought they were doing things incorrectly. A couple of full crashes arose while test users were performing their task. Most of the issues and errors that occurred were not repeatable at



Figure 9.8: Paintroid: colour picker dialogue. Left: complex HSV colour space picker dialogue. Right: dialogue with predefined colour entries.

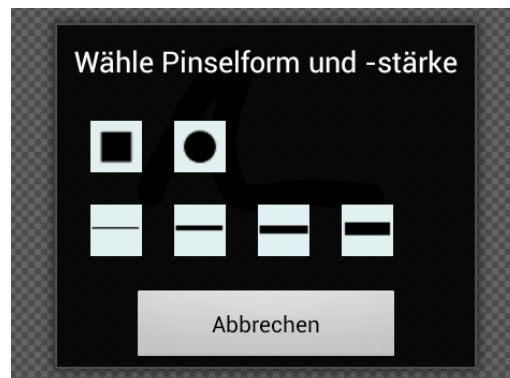


Figure 9.9: Paintroid: change shape and width.

all.

### 9.1.2 Positive Findings

Catroid includes a nice tool for users to record sounds on their own without the need of an external sound recorder. The recorder is very simple and was used without any problems. The recorder was started when the “Recording”-button was clicked and stopped when it was clicked again. The round indicator showed the status of the recorder: gray signified “Standby” and red was for “Recording now”. All in all, the recorder was easy

to understand and easy to use. One recommendation can be stated here: some of the children tried to start and stop the recorder by clicking the indicator sign. So it would be nice if the indicator also had the functionality to start and stop the recorder.

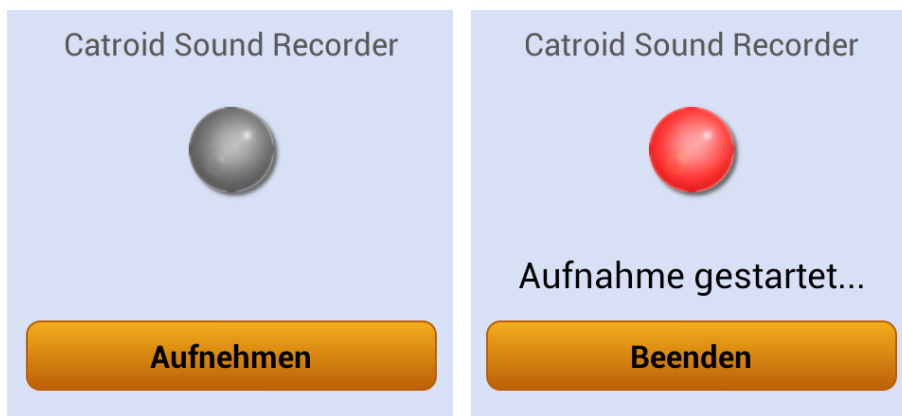


Figure 9.10: Catroid: sound recorder.

The “Undo”-tool in Paintroid was frequently tried but not understood correctly. One comment underlined this misunderstanding: *“This is used for deleting something.”* The “Pipette”-tool that allows to pick up a colour from a part of an image was understood correctly, but not often used.

### 9.1.3 Negative Findings

Users had to click on the “New”-button as shown in Figure 9.2, to **insert a new brick** for a script. Then they had to choose one of the different categories in the brick dialogue. For example, when a test user wanted to move an object or to set its position, in order to perform one of the test tasks, they had to select a brick in the “Motion”-category. The image on the right in Figure 9.11 shows several selectable bricks that were part of this category. Users had to touch the desired brick to select it. This worked anywhere inside the coloured areas of a brick, but not within the bright parts of the input elements, like select boxes, text boxes or buttons. The functional behaviour of these elements was deactivated in this dialogue for this version of the program. Thus, they were clickable but did not respond to a user action. However, these bright input elements attracted attention like magnets, so that users tried to click or touch them all the time and wondered why nothing happened. Furthermore, they became confused and thought it simply did not work at all. Some users needed several attempts until they managed to get a brick inserted by accident. It is recommended to change the bricks in a way, that the whole area of a brick

will react immediately when it is touched.



Figure 9.11: Catroid: add a new brick. Left: different categories of bricks. Right: bricks that are part of the “Motion” category.

A similar problem occurred with bricks in the scripts tab. It was possible to **move or delete a brick** by executing a *long-press*-gesture on the desired item. However, it was hard to discover that a brick was touchable by a long-press. So, the test users did not find out on their own that they could do this or how to do it. As seen in Figure 9.12, bricks on a tablet computer were presented even larger and took up more space, but they had also even larger input elements. In general, the functionality of input elements is activated and they react immediately when they were clicked or touched with a long-press gesture. For example, when a text field is clicked, it gets prepared for inserting or modifying text, and when select box is clicked, it shows up a list of selectable elements. Unlike the insert dialogue for a brick, the input elements in the script list in the scripts tab view were activated. Figure 9.4 shows an example for what happens when a “Set costume” select box was clicked in Catroid. As explained before, the bright input elements attracted attention and users wanted to *pick up* a brick by touching them on these bright areas as shown in Figure 9.12. This, of course, did not work. Finally, some users recognized that it was easier to pick up a brick in the area on the left of the input elements. A recommendation for this problem is that every brick should get a defined visual area to shift the attention towards touchable parts.

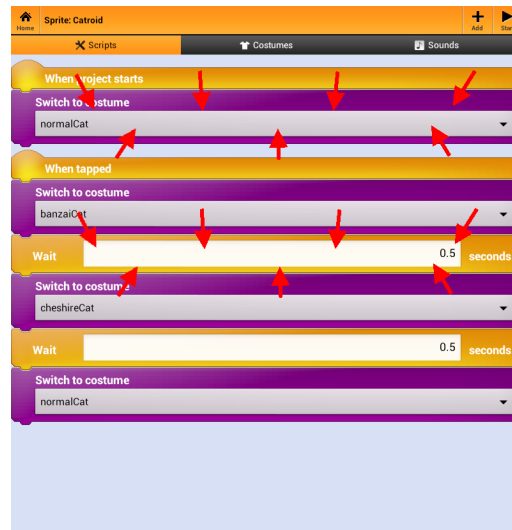


Figure 9.12: Catroid: bricks tab on the tablet

When users tried to delete an item for the first time, and therefore made a long-press gesture by accident or by prompt, they then did not know what to do. If a brick was picked up, it was detached from the list and the recycle bin came up on the right edge (see Figure 9.13). It was not possible to detach a brick from the left edge, and furthermore several bricks changed their appearance too (see Figure 9.16 and Figure 9.17). There was no clear signal to indicate that it just required the finger to be moved towards the recycle bin to delete a brick. As shown in Figure 9.14, users tried everything but did not figure how to only slide the finger. One candidate mentioned during the test: *It can only be moved up and down*. Users tried to slide the brick up and down, or to the right, or down and to the right and finally to touch the recycle bin with the second hand simultaneously. One recommendation at this point is that a brick may detach from the edge, so that users can imagine that a brick can be moved into the bin. Other advice includes the implementation of a context menu for several actions.



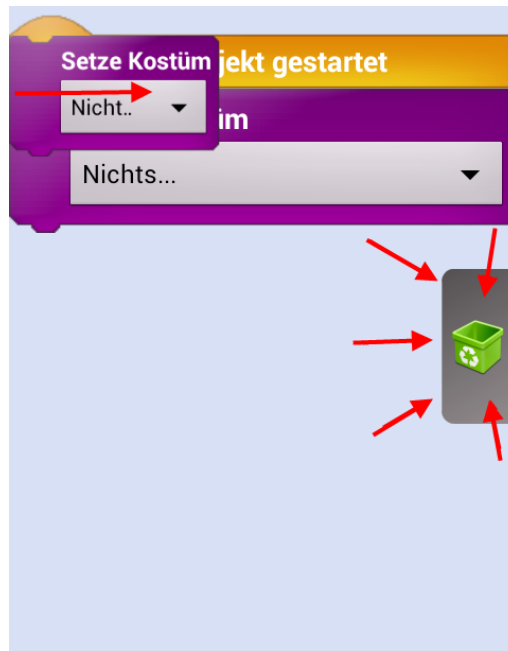


Figure 9.13: Catroid: delete a brick

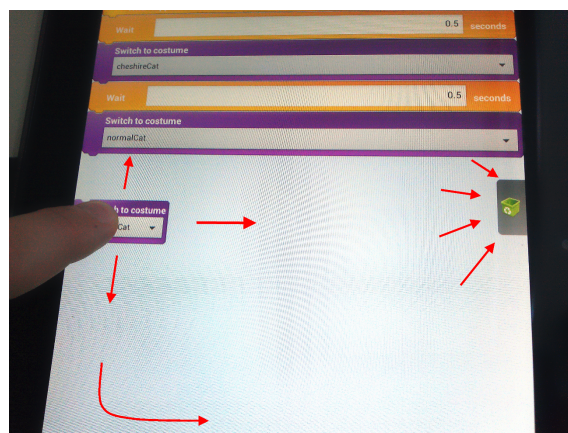


Figure 9.14: Catroid: delete bricks tab on the Samsung Galaxy Tab

As explained before, many issues regarding the visibility of touchable parts of bricks existed. A further issue related to the general visibility of bricks, especially when they were inserted or picked up. Figure 9.15 shows two examples where the **visibility of bricks** in the list view was not good. The figures show bricks that are not clearly visible after they were inserted.

A recommendation to increase the visibility of bricks is as follows: bricks should be rendered with an illuminating outer edge on one hand and a layer that darkens the back-

ground on the other hand. This may highlight a brick more strongly and make it stand out against all other elements on the screen when it is inserted or picked-up.



Figure 9.15: Catroid: visibility of bricks

In addition to the variety of findings already discussed, users noticed that some bricks were defective in several ways. Some bricks showed a **defective behaviour or inconsistent look** when they were inserted or picked up. As seen in Figure 9.16 and Figure 9.17, several bricks were shorter or bigger than others, or text was displayed incorrectly. It is recommended that all bricks are checked for inconsistency and furthermore to develop several fixed sizes and forms for the various bricks.

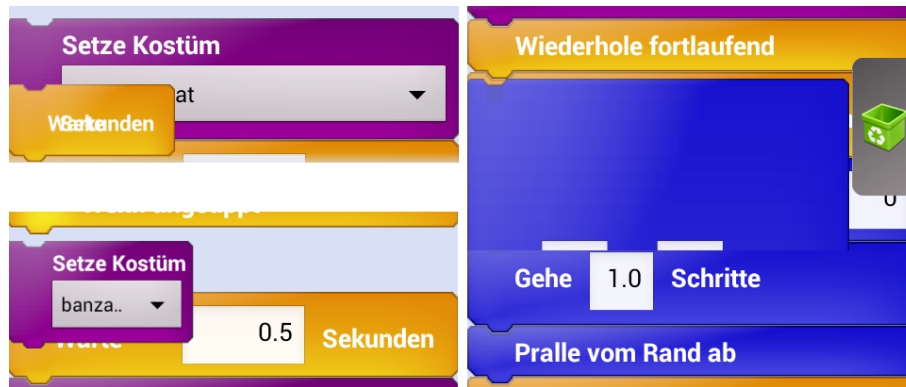


Figure 9.16: Catroid: inconsistent or faulty bricks. Top left: the “Wait ... seconds” brick has a compressed shape and displays text incorrectly. Bottom left: the “Set costume” brick is shorter than others. Right: “Place at” brick has a broken shape and does not show any text or input elements.

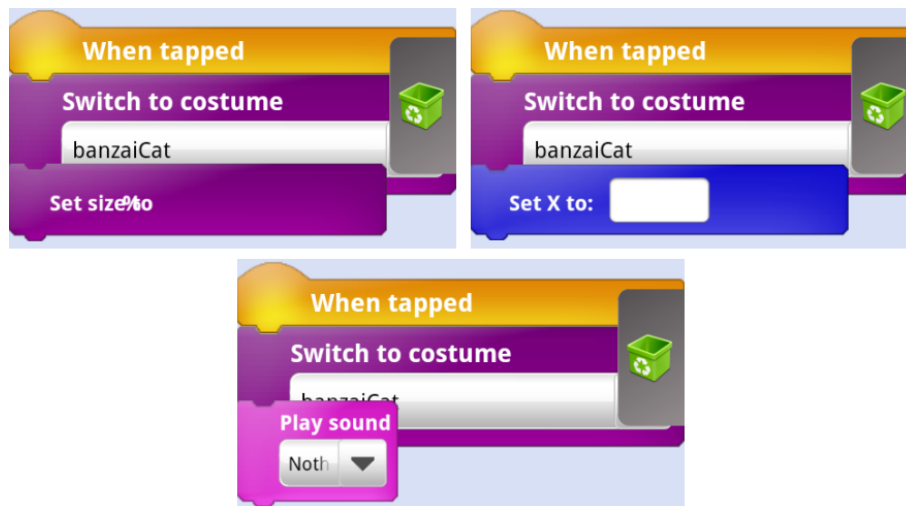


Figure 9.17: Catroid: more inconsistent or faulty bricks.

## 9.2 Findings Regarding Testing with Children

The findings regarding usability testing with children were indeed very positive. The summary of the outcome of the tests generated a basis for some guidelines. These guidelines include already proven general approaches that were verified by the obtained data after the tests. The following guidelines are formed with the aim of helping the Catroid usability team in the future when testing with children or the younger generation.

### **Introduction**

- Background questionnaire must be in a child-orientated language. Try to prepare the questions with a teacher of the expected age range.
- Be sure you have a parents-signed consent form (or from a person in charge) for the recording of children.
- Show children around the test lab. This gives them a better sense of control and trust in you. Explain cameras and the microphone, and tell them that the recording is very important for the other members who cannot be at the test. It is necessary that they see what they have said about the software.
- Explain why it is important to get their feedback about the software. Tell them that its not them being tested, but rather they can test and rate the software. Motivate children by emphasising the importance of their role in testing your software.
- Introduce test team by their first names, to increase confidence of the participants.

### **During the test**

- Offer generic feedback to encourage them.
- Be sure you have a various number of observers, that help you to note events during the test.
- Do not forget to take breaks during a test.
- Do an interview before every break. Children in this age range easily forget events that happened during the test. The longer events are ago, and the more complex they were, the higher the probability that they do not remember the details of an event.

### **Finishing**

- Be sure to slow children down before the end of a session. They need a soft landing, because it is hard for them to stop and quit a test immediately.
- Do an interview and ask them how the test was, what they liked or disliked, and what was easy or very difficult for them. Let them talk until they stop of their own accord.

### Insights

- A combination of thinking-aloud with active intervention in a constructive interaction setting is recommended as a test method for the age group from eight to ten.
- Some children are distracted and slow down if their parents are in the testing room during a test. It may be better to place parents out of the room while testing. Persons in charge inside the room may not be such a “slow down”-factor.
- Avoid the presence of too many adults in the testing room. If there is a hustle or huge imbalance of adults and children, children tend to feel uncomfortable and are quieter.
- Do not test for longer than about half an hour at once. Children in this age range have a limited attention span and are not able to concentrate for a very long time.
- Children often do not know how they did something. For example, if children solved an average difficulty task by chance and you ask them to repeat how they did it, they often do not remember and cannot repeat.
- Do short feedback questionnaires or interviews after each part of a test. Children in this age range do not remember special events that happened long ago. Thus, such an interview at the end of the test may falsify answers.
- Children tend to rate you and your software as positive over all.
- Avoid yes-no questions. Children tend to answer these questions with ‘yes’ rather than with ‘no’.

## 9.3 System Usability Scale

The main goal of using this method was to obtain subjective data, and furthermore then to receive a global assessment of the system’s usability. Since the items of the SUS were developed in a very abstract form, it was observed that children had huge problems in understanding the abstract content and information with which they were confronted. The observers detected responses that did not correlate with the impressions that were obtained during the tests. In addition to the misunderstanding of several statements, the common answers were also falsified due to wrong conclusions about the test. As explained

before, children did not remember specific events that happened long ago. Children had a feeling of success after the test. The reason for this was that they performed all tasks, of course with the help of the facilitator, and were rewarded with a working output at the end. Furthermore, the closer the end of the test came, the more children tended to forget the massive problems that occurred before while performing certain tasks.

Due to the fact that the System Usability Scale did not produce meaningful results during the first several days, the method was cancelled for the remaining test days.

## Chapter 10

# Conclusions and Future Work

Research has shown that issues exist regarding the usability of the user interface design. Negative findings that arose from the evaluation of Catroid and Paintroid were caused by a lack of discoverability of several functions, interface items, and essential user interactions. It was observed that the lack of useful hints was one of the most negative impacts for discovering features and important user interactions in the interface design.

Thus, the provision of a small and simple tutorial for important tools and functions that are often needed should definitely be considered. Such a tutorial must not be animated, but may include some simple graphical representations that involve several clearly understandable hints and short descriptions. Another solution could be that simple animated wizards assist the first steps in several views. These wizards might be enabled or disabled in the global settings. Furthermore a simple explanation of the logical structure and the connection of the different parts of the Catroid software environment should be developed. It is essential that such explanations should be kept short and easy to understand. These approaches might help to improve the discoverability and furthermore the learnability of the evaluated tools and can increase a common understanding of the procedure of how to develop programs with Catroid.

Another essential aspect that can improve discoverability as well as the usability of the product is the lack of visual feedback. Missing dialogues, information messages and graphical indicators everywhere in the program decreased the visibility of things, and furthermore the visibility of events that occurred. Children need to see if events occur or something is happening. If events occur very quickly, they are often not perceived by some users. Observations showed that children do not take notice of events that occur too fast, or without a suitable feedback or note.

The System Usability Scale was used as an attempt to receive a global assessment of the system's usability and to get a general subjective feedback about the programming environment for this age group. But, the results showed that this method is not a good tool for the usage when testing with children. The results of the SUS were not usable for any usability assessments and were discarded.

As this was the first usability test in the whole development lifecycle that included more than one test session in a row, the results provide a useful basis for regular future usability tests. Many of the findings found represent major weaknesses regarding the usability of the user interface design for young users on the one hand, and bugs and error-findings of the program on the other hand. In future usability evaluations of the Catroid environment, it is recommended that the program is better prepared for a usability test and therefore to do a heuristic evaluation and an expert review prior to a test. It is very important to eliminate as many errors as possible, and to provide a mostly error-free program version to the test users.

While this thesis has addressed many usability evaluation aspects that provide useful experiences for the Catroid development community, it would be necessary to use these results as the basis for other usability evaluation processes in the future. The expectations that further evaluations of the Catroid programming environment will be done are high and should be done regularly. Only periodical evaluations of the software environment will show whether these improvements of the user interface design are a valuable approach, and take effect or not. As already discussed in this thesis, the work with children is characterised by difficulties in receiving subjective information. Thus, the test team is required to address the needs of children during a test, so as to obtain as much information as possible.



## Bibliography

- Aaron, M. [2004], ‘Return on investment for usable user-interface design: Examples and statistics’, White Paper. Available online at [http://www.amanda.com/joomla\\_uploads/whitepapers/AM+A\\_ROIWhitePaper\\_20Apr0%201.pdf](http://www.amanda.com/joomla_uploads/whitepapers/AM+A_ROIWhitePaper_20Apr0%201.pdf); visited on July 25th 2012.
- Andrews, K. [2006], Evaluating information visualisations, BELIV ’06, ACM, New York, NY, USA, pp. 1–5.
- Andrews, K. [2008], ‘Evaluation comes in many guises’, *Information Visualization* pp. 8–10. <http://www.dis.uniroma1.it/~beliv08/pospap/andrews.pdf>.
- Andrews, K. [2011], ‘Human-computer interaction’, Lecture Notes. Available online at <http://courses.iicm.tugraz.at/hci/hci.pdf>.
- Baauw, E. and Markopoulous, P. [2004], A comparison of think-aloud and post-task interview for usability testing with children, in ‘Proceedings of the 2004 conference on Interaction design and children: building a community’, IDC ’04, ACM, New York, NY, USA, pp. 115–116.
- Barendregt, W. and Bekker, M. M. [2005], ‘Guidelines for user testing with children’. Available online at [http://w3.id.tue.nl/fileadmin/id/objects/doc/Guidelines\\_for\\_user\\_testing\\_with\\_children.pdf](http://w3.id.tue.nl/fileadmin/id/objects/doc/Guidelines_for_user_testing_with_children.pdf); visited on August 30th 2012.
- Baumgarten, M. [2003], ‘Kids and the internet: a developmental summary’, *Comput. Entertain.* 1.
- Berkun, S. [2003], ‘26 the myth of discoverability’, Blog. Available online at <http://www.scottberkun.com/essays/26-the-myth-of-discoverability/>; visited on September 21th 2012.
- Brooke, J. [1986], Sus - a quick and dirty usability scale, Technical report, Electronic Systems Division, Air Force Systems Command, United States Air Force, Hanscom Air Force Base, Massachusetts, Bedford, Massachusetts, USA.
- Card, S. K., Newell, A. and Moran, T. P. [1983], *The Psychology of Human-Computer Interaction*, L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

- Coleman, W. D., Williges, R. C. and Wixon, D. R. [1985], 'Collecting detailed user evaluations of software interfaces', *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* **29**(3), 240–244.
- Cooper, A. [1999], *The Inmates Are Running the Asylum*, Macmillan Publishing Co., Inc., Indianapolis, IN, USA.
- Cooper, A. [2003], 'The origin of personas', Website. Available online at [http://www.cooper.com/journal/2003/08/the\\_origin\\_of\\_personas.html](http://www.cooper.com/journal/2003/08/the_origin_of_personas.html); visited on July 27th 2012.
- Druin, A. [2002], 'The role of children in the design of new technology', *Behaviour and Information Technology* **21**, 1–25.
- GSM Association and Mobile Society Research Institute within NTT DO-COMO Inc. [2010], 'Childrens Use of Mobile Phones and Personal Relationships - An International Comparison 2010'. Available online at <http://www.gsma.com/publicpolicy/childrens-use-of-mobile-phones-and-personal-relationships-an-international-comparison-june-2010-japan-korea-cyprus-china-india-and-mexico>; visited on July 10th 2012.
- GSM Association and the Mobile Society Research Institute within NTT DOCOMO Inc. [2011], 'Childrens Use of Mobile Phones - An International Comparison 2011'. Available online at <http://www.gsma.com/latinamerica/childrens-use-of-mobile-phones-2011>; visited on July 10th 2012.
- Hadj-Karim-Kharrazi, H. [2005], 'Usable guidelines for usability testing with children'. Available online at [http://web.cs.dal.ca/~kharrazi/courses/HF/Midterm\\_revised\\_final.pdf](http://web.cs.dal.ca/~kharrazi/courses/HF/Midterm_revised_final.pdf); visited on March 18th 2012.
- Hanna, L., Risdien, K. and Alexander, K. [1997], 'Guidelines for usability testing with children', *Interactions* **4**, 9–14.
- Haywood, A. and Boguslawski, G. [2009], I Love My iPhone ... But There Are Certain Things That Niggle Me, in J. Jacko, ed., 'Human-Computer Interaction. New Trends', Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp. 421–430.
- Hewett, T. T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G. and Verplank, W. [1992], ACM SIGCHI curricula for human-computer in-

- teraction, Technical report, New York, NY, USA. Available online at <http://old.sigchi.org/cdg>; visited on July 26th 2012.
- Hom, J. [1998], ‘The usability methods toolbox’, Website. Available online at <http://usability.jameshom.com>; visited on August 13th 2012.
- ISO [1998], ‘Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability (ISO 9241-11:1998)’. Available online at [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=16883](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16883); visited on March 15th 2012.
- ISO [1999], ‘Human-centred design processes for interactive systems (ISO 13407:1999); German version EN ISO 13407:1999’. English version available online at [http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=21197](http://www.iso.org/iso/catalogue_detail.htm?csnumber=21197); visited on March 16th 2012.
- LaLomia, M. J. and Sidowski, J. B. [1990], ‘Measurements of computer satisfaction, literacy, and aptitudes: A review’, *International Journal of Human-Computer Interaction* **2**(3), 231–253.
- Lewis, C., Polson, P. G., Wharton, C. and Rieman, J. [1990], Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces, in ‘Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people’, CHI ’90, ACM, New York, NY, USA, pp. 235–242.
- Lewis, C. and Rieman, J. [1993], *Task-Centered User Interface Design: A Practical Introduction*.
- Liebal, J. and Exner, M. [2011], *Usability für Kids: ein Handbuch zur ergonomischen Gestaltung von Software und Websites für Kinder*, Schriften Zur Medienproduktion, Vieweg+teubner Verlag.
- Lockee, B., Moore, M. and Burton, J. [2002], ‘Measuring Success: Evaluation Strategies for Distance Education’, *Educause* .
- MacLaurin, M. [2011], ‘The design of kodu: a tiny visual programming language for children on the Xbox 360’, *SIGPLAN Not.* **46**(1), 241–246.
- Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B. and Resnick, M. [2004], Scratch: A sneak preview, in ‘Proceedings of the Second International Conference on Creating,

- Connecting and Collaborating through Computing', IEEE Computer Society, Washington, DC, USA, pp. 104–109.
- Markopoulos, P. and Bekker, M. [2003], 'Interaction design and children', *Interacting with Computers* **15**(2), 141–149.
- Mayes, J. T., Draper, S. W., McGregor, A. M. and Oatley, K. [1988], Information flow in a user interface: the effect of experience and context on the recall of macwrite screens, in 'Proceedings of the Fourth Conference of the British Computer Society on People and computers IV', Cambridge University Press, New York, NY, USA, pp. 275–289.
- Medienpädagogischer Forschungsverbund Südwest [2010], 'KIM-Studie 2010 Kinder + Medien + Computer + Internet'. Available online at <http://www.mpfs.de/fileadmin/KIM-pdf10/KIM2010.pdf>; visited on April 10th 2012.
- Microsoft, R. [2012a], 'Kodu - Microsoft Research', Website. Available online at <http://research.microsoft.com/en-us/projects/kodu>; visited on July 10th 2012.
- Microsoft, R. [2012b], 'Microsoft Research FUSE Labs - Kodu Game Lab', Website. Available online at <http://fuse.microsoft.com/page/kodu>; visited on July 10th 2012.
- Mobilewalla [2011], 'App Intelligence - App Count'. Available online at [http://www.mobilewalla.com/Desktop/AppIntel\\_AppCount.htm?filterDevicePlatform=101](http://www.mobilewalla.com/Desktop/AppIntel_AppCount.htm?filterDevicePlatform=101); visited on July 11th 2012.
- Monroy-Hernández, A. and Resnick, M. [2008], 'Empowering kids to create and share programmable media.', *interactions* **15** pp. 50–53.
- Nielsen, J. [1992], Finding usability problems through heuristic evaluation, in 'Proceedings of the SIGCHI conference on Human factors in computing systems', CHI '92, ACM, New York, NY, USA, pp. 373–380.
- Nielsen, J. [1993], *Usability Engineering*, Academic Press, San Diego, CA, USA.
- Nielsen, J. [1994a], Enhancing the explanatory power of usability heuristics, in 'Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence', CHI '94, ACM, New York, NY, USA, pp. 152–158.
- Nielsen, J. [1994b], 'Estimating the number of subjects needed for a thinking aloud test', *Int. J. Hum.-Comput. Stud.* **41**(3), 385–397.

- Nielsen, J. [1994c], Usability inspection methods, in ‘Conference companion on Human factors in computing systems’, CHI ’94, ACM, New York, NY, USA, pp. 413–414.
- Nielsen, J. [2003], ‘Return on Investment for Usability’, Website. Available online at <http://www.useit.com/alertbox/roi-first-study.html>; visited on July 26th 2012.
- Nielsen, J. [2005a], ‘Severity ratings for usability problems’, Website. Available online at <http://www.useit.com/papers/heuristic/severityrating.html>; visited on September 30th 2012.
- Nielsen, J. [2005b], ‘Ten usability heuristics’, Website. Available online at [http://www.useit.com/papers/heuristic/heuristic\\_list.html](http://www.useit.com/papers/heuristic/heuristic_list.html); visited on July 27th 2012.
- Nielsen, J. [2011a], ‘All About Android: Insights From Nielsen’s Smartphone Meters’, Report. Available online at <http://www.nielsen.com/us/en/insights/events-webinars/2011/all-about-android-insights-from-nielsens-smartphone-meters.html>; visited on July 16th 2012.
- Nielsen, J. [2011b], ‘Children’s websites: Usability issues in designing for kids’, Website. Available online at <http://www.useit.com/alertbox/children.html>; visited on September 30th 2011.
- Nielsen, J. [2011c], ‘Usability 101: Introduction to usability’, Website. Available online at <http://www.useit.com/alertbox/20030825.html>; visited on November 16th 2011.
- Nielsen, J. and Mack, R. L. [1994], *Usability Inspection Methods*, Wiley.
- Nielsen, J. and Molich, R. [1990], Heuristic evaluation of user interfaces, in ‘Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people’, CHI ’90, ACM, New York, NY, USA, pp. 249–256.
- Norman, D. A. and Nielsen, J. [2012], ‘Gestural interfaces: A step backwards in usability’, Website. Available online at [http://www.jnd.org/dn.mss/gestural\\_interfaces\\_a\\_step\\_backwards\\_in\\_usability\\_6.html](http://www.jnd.org/dn.mss/gestural_interfaces_a_step_backwards_in_usability_6.html); visited on September 24th 2012.
- Olson, J. R. and Olson, G. M. [1990], ‘The growth of cognitive modeling in human-computer interaction since goms’, *Hum.-Comput. Interact.* **5**(2), 221–265.
- Osborne, R. [2007], ‘Usability vs discoverability’, Blog. Available online at <http://rickosborne.org/blog/2007/04/usability-vs-discoverability/>; visited on September 21th 2012.

- Overmars, M. [2004], 'Teaching computer science through game design', *Computer* **37**(4), 81–83.
- Polson, P. G. and Lewis, C. H. [1990], 'Theory-based design for easily learned interfaces', *Hum.-Comput. Interact.* **5**(2), 191–220.
- Rubin, J. and Chisnell, D. [2008], *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, 2 edn, Wiley Publishing.
- Slany, W. [2012], Catroid: a mobile visual programming system for children, in 'Proceedings of the 11th International Conference on Interaction Design and Children', IDC '12, ACM, New York, NY, USA, pp. 300–303.
- Smith, S. L. and Mosier, J. N. [1986], Guidelines for designing user interface software, Technical report, Redhatch Consulting Ltd., United Kingdom, Earley, READING RG6 2UX, UK.
- Stake, Robert E. and Organisation for Economic Cooperation and Development, Paris (France). Centre for Educational Research and Innovation. [1976], *Evaluating Educational Programmes [microform] : The Need and the Response / Robert E. Stake*, Distributed by ERIC Clearinghouse, [Washington, D.C.].
- UsabilityNet [2006], 'Key principles of user centred design', Website. Available online at [http://www.usabilitynet.org/management/b\\_design.htm](http://www.usabilitynet.org/management/b_design.htm); visited on March 16th 2012.
- USATODAY.com [2012], 'New program allows everyone to design video games', Website. Available online at [http://www.usatoday.com/tech/gaming/2009-01-07-microsoft-kodu\\_N.htm](http://www.usatoday.com/tech/gaming/2009-01-07-microsoft-kodu_N.htm); visited on July 10th 2012.
- Wharton, C., Rieman, J., Lewis, C. and Polson, P. [1994], Usability inspection methods, John Wiley & Sons, Inc., New York, NY, USA, chapter The cognitive walkthrough method: a practitioner's guide, pp. 105–140.
- Wikipedia, the free encyclopedia [2012], 'Human interface guidelines'. Available online at [http://en.wikipedia.org/wiki/Human\\_interface\\_guidelines](http://en.wikipedia.org/wiki/Human_interface_guidelines); visited on August 10th 2012.
- YoYo-Games Ltd. [2012], 'Gamemaker:studio yoyo games', Website. Available online at <http://www.yoyogames.com/gamemaker/studio>; visited on July 13th 2012.

---

Zaman, B. [2006], 'Evaluating games with children', Report. Available online at [http://soc.kuleuven.be/com/mediac/cuo/admin/upload/Zaman\\_Evaluating%20games%20with%20children.PDF](http://soc.kuleuven.be/com/mediac/cuo/admin/upload/Zaman_Evaluating%20games%20with%20children.PDF); visited on August 16th 2012.