Master’s Thesis

Smart Adaptive User Interfaces for enhancing User Experience in Enterprise Applications on Mobile Devices

Michael Geier, BSc.

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

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

Graz, March 2012

<This page intentionally left blank>

Masterarbeit

(Diese Arbeit ist in englischer Sprache verfasst)

Smarte, adaptive Benutzerinterfaces zur Verbesserung der User Experience für Business-Anwendungen auf mobilen Endgeräten

Michael Geier, BSc.

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

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

Graz, März 2012

<This page intentionally left blank>

Abstract

User interfaces have become an essential part in the development of mobile applications. However, mobile workers usually are in complex and hectic work places. The main challenge is to design the user interface in such a simple manner, that the end user can completely concentrate on his/her task - not the device; a possible solution is on smart adaptation (El-Bakry et al., 2010). Consequently, the designer needs to know the limits of context and systems and that mobile end-users interact differently (Holzinger and Errath, 2007).

Based on usability evaluations of enterprise framework-based desktop applications a novel usability checklist for mobile business applications was elaborated, specifically focusing on the properties of modern smartphone and tablet computer applications by regarding context-based adaptivity and supporting simplicity of operation. Usability consultancy activities of an enterprise framework-based mobile application are documented within this work as well.

This work also focuses on applying smart adaptivity for hiding complexity from the end user by not restraining functionality at the same time (Crosby et al., 2001). Following the hypothesis that simpler user interfaces increase the performance of end-users, an experiment for evaluating the performance of smart adaptive user interfaces on mobile touch-based devices was carried out.

Keywords

Mobile Computing, Context Aware Computing, Adaptive User Interfaces, Enterprise Applications, Usability Tests

ÖSTAT-Topics

1161	40 %	1157	40 %	1140	15 %	1138	15 %

1161 =	Human-Computer Interaction (HCI)
1157 =	Usability Research
1140 =	Software-Engineering
1138 =	Informationssysteme

ACM Classification

H.3, H.4, H.5, H.5.1

H. Information Systems,
H.5 INFORMATION INTERFACES AND PRESENTATION (I.7),
H.5.2 User Interfaces (D.2.2, H.1.2, I.3.6);

Subjects: Graphical user interfaces (GUI), Input devices and strategies (e.g., mouse, touchscreen)

<This page intentionally left blank>

Zusammenfassung

Benutzeroberflächen wurden zu einem wesentlichen Teil in der Entwicklung von mobilen Anwendungen. Allerdings befinden sich mobile Arbeiter häufig in komplexen und hektischen Arbeitsumgebungen. Die größte Herausforderung ist es, die Benutzeroberfläche so einfach zu gestalten, dass sich der Endverbraucher ganz auf seine Aufgabe konzentrieren kann – nicht auf das Gerät; eine mögliche Lösung ist intelligente Anpassung (El-Bakry et al., 2010). Folglich muss der Designer die Grenzen des Kontexts und der Systeme kennen und wissen, dass mobile Endnutzer unterschiedlich interagieren (Holzinger und Errath, 2007).

Basierend auf Usabilityevaluierungen existierender, Enterprise Framework-basierter Desktopsoftware wurde eine Usabilitycheckliste für mobile Businessanwendungen ausgearbeitet, die auf die speziellen Erfordernisse von Anwendungen von modernen Smartphones und Tablet-Computern eingeht und sowohl kontextbasierte Adaptivität berücksichtigt als auch die Einfachheit der Bedienung in den Vordergrund rückt. Außerdem werden im Rahmen dieser Arbeit Usabilityberatungstätigkeiten einer Enterprise Framework-basierten mobilen Anwendung dokumentiert.

Diese Arbeit fokussiert außerdem auf die Anwendung von intelligenter Anpassung zur Verbergung von Komplexität vor dem Endbenutzer bei gleichzeitiger Erhaltung der Funktionalität (Crosby et al., 2001). Der Hypothese folgend, dass einfachere User Interfaces die Performanz von Endbenutzern steigert, wurde ein Experiment, das die erhöhte Performanz bei Verwendung von adaptiven User Interfaces auf touch-basierten Geräten belegen soll, durchgeführt.

Schlüsselwörter

Mobile Computing, Context Aware Computing, Adaptive User Interfaces, Enterprise Applications, Usability Tests

ÖSTAT-Fachgebiete

1161	40 %	1157	40 %	1140	15 %	1138	15 %

1161	Human-Computer Interaction (HCI)
1157	Usability Research
1140	Software-Engineering
1138	Informationssysteme

ACM Klassifikation

H.3, H.4, H.5, H.5.1

H. Information Systems,
H.5 INFORMATION INTERFACES AND PRESENTATION (I.7),
H.5.2 User Interfaces (D.2.2, H.1.2, I.3.6);

Subjects: Graphical user interfaces (GUI), Input devices and strategies (e.g., mouse, touchscreen)

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, March 12th 2012 ……………………………………………

Michael Geier

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, 12. März 2012 ……………………………………………

Michael Geier

Danksagung / Acknowledgments

I would like to thank Boom Software AG for the good partnership, especially Joachim Schnedlitz, CEO of Boom Software AG, who made the cooperation possible, and Roman Bobik, our contact person during our work with Boom. Furthermore, I would like to thank the test users who spent time for the thinking aloud tests. Also, I would like to thank Peter Treitler for the good cooperation during the work at Boom Software AG. Finally, I would like to thank my supervisor, Andreas Holzinger, for his continuous support during my work.

Michael Geier
Graz, March 12th 2012

Abbreviations

API	Application Programming Interface
CLR	Common Language Runtime
CMS	Content Management System
DBMS	Database Management System
HCI	Human–Computer Interaction
HE	Heuristic Evaluation
HTML	HyperText Markup Language
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IME	Input Method Editor
ISO	International Organization for Standardization
ITU	International Telecommunication Union, UN agency for information and communication technologies
JDK	Java Development Kit
LAN	Local Area Network
MCF	Mobile Context Factor
MS	Microsoft
MVC	Model View Controller
NAN	Not A Number
NFC	Near Field Communication
OO	Object-oriented
OS	Operating System
PC	Personal Computer
RIA	Rich Internet Application
RIM	Research In Motion
SD	Secure Digital
SDK	Software Development Kit

SIM	Subscriber Identity Module
SUS	System Usability Scale
TA	Thinking Aloud
TBBP	Time Between Button Presses
UCD	User Centered Development
UCD	User Centered Design
UE	Usability Engineering
UI	User Interface
UMUX	Usability Metric for User Experience
UX	User Experience
UXD	User Experience Design
UXR	User Experience Research
W-LAN	Wireless Local Area Network
WPF	Windows Presentation Foundation
WWW	World Wide Web
XAML	Extensible Application Markup Language
XML	Extensible Markup Language
XP	Extreme Programming
XU	Extreme Usability

Table of Contents

1. Introduction and Motivation for Research 17
   - 1.1. Structure of this Thesis 22
2. Theoretical Background and Related Work 23
   - 2.1. Mobile Devices 23
     - 2.1.1. Mobile Computing Da Capo 26
     - 2.1.2. Mobile Computing Today 28
     - 2.1.3. Future of Mobile Computing 30
     - 2.1.4. Device Types 33
     - 2.1.5. Mobile Platforms 37
     - 2.1.6. Input, Sensors, and UI controls 42
     - 2.1.7. Showcase UI Creation 51
   - 2.2. Enterprise Applications 54
   - 2.3. User Experience 56
     - 2.3.1. User Experience vs. Usability vs. Accessibility 56
     - 2.3.2. Usability Engineering 59
     - 2.3.3. Evaluation Methods 61
     - 2.3.4. Heuristic Evaluation 68
     - 2.3.5. Thinking Aloud 75
   - 2.4. Adaptive User Interfaces 76
     - 2.4.1. Context-awareness 76
     - 2.4.2. Adaptive vs. adaptable UIs 78
3. Materials and Methods 80
   - 3.1. Boom Software AG 80
     - 3.1.1. The Enterprise Framework 81
     - 3.1.2. Future Development 88

3.2. Usability Consultancy ... 89
3.2.1. Mobile App ... 89
3.2.2. Usability Evaluation ... 92
3.2.3. Leseratte ... 93
3.2.4. HE ... 96
3.2.5. Leseratte 2.0 ... 98
3.2.6. TA ... 101
3.2.7. Usability Guidelines ... 104
3.3. AUIs vs. Non-AUI Experiment ... 105
3.4. HE Checklist for Mobile Devices ... 111
4. Results ... 113
4.1. Usability Consultancy ... 113
4.1.1. Mobile App ... 113
4.1.2. Results of HE ... 115
4.1.3. Results of TA ... 118
4.1.4. Implementation and Guidelines ... 120
4.2. AUIs vs. Non-AUI Experiment ... 121
4.3. HE Checklist for Mobile Devices ... 130
5. Lessons Learned ... 136
6. Conclusion ... 138
7. Future Work ... 139
8. List of Figures ... 140
9. List of Tables ... 142
10. Glossary ... 143
11. References ... 145
A. Appendix ... 154

<This page intentionally left blank>

1. Introduction and Motivation for Research

Mobile devices become more and more important nowadays. End users want to be online not only at home with their desktop computers. Mobile computers, which allow being online even outside, on the way to work, or in leisure time, gain more and more popularity. This fact was also recognized by famous web services, such as the social network Facebook. Formerly available as websites for desktop browsers only, applications or special web pages for mobile devices such as smartphones or tablet computers were created. According to the Google Play website (formerly Android Market)[1] currently more than one hundred million users have installed the mobile Facebook application. This shows a vast interest in mobile usage of this social network. Although social networks might be used for work as well as for leisure activities, there are also more serious business scenarios in which mobile applications (applications running on mobile devices, short: apps) are used in enterprises for enhancing productivity. Figure 1 shows the trend to mobile connectivity. Cellular phone subscriptions per 100 inhabitants are raising year by year as well as mobile-broadband subscriptions.

With the new class of mobile applications for devices such as smartphones and tablets, new challenges arise. Mobile devices are usually smaller and therefore more portable than common desktop computers or notebooks. Additionally, computational power and many other properties differ. Obviously, the computational power of small mobile devices is not that high than the computational power of current average desktop computers. However, it is high enough for most applications. Anyway, software engineers have to consider that they are not developing software for desktop computers and therefore should not waste resources by accessing hardware extensively.

More limitations and challenges to consider when developing software for mobile devices include relatively low battery life and the danger of unstable and often changing network connection states.

[1] Google Play / Android Market: *https://play.google.com/store* or *https://market.android.com*

Apart from the points mentioned above, mobile devices have smaller screens than desktop computers and are usually operated differently. Instead of using a mouse and a keyboard a touchscreen is often used as main input device. Also, the high portability raises the problem that input is often made in situations where end-users might be distracted by environmental factors (for example during walking, in a noisy or bright environment or even while driving, which is not recommended).

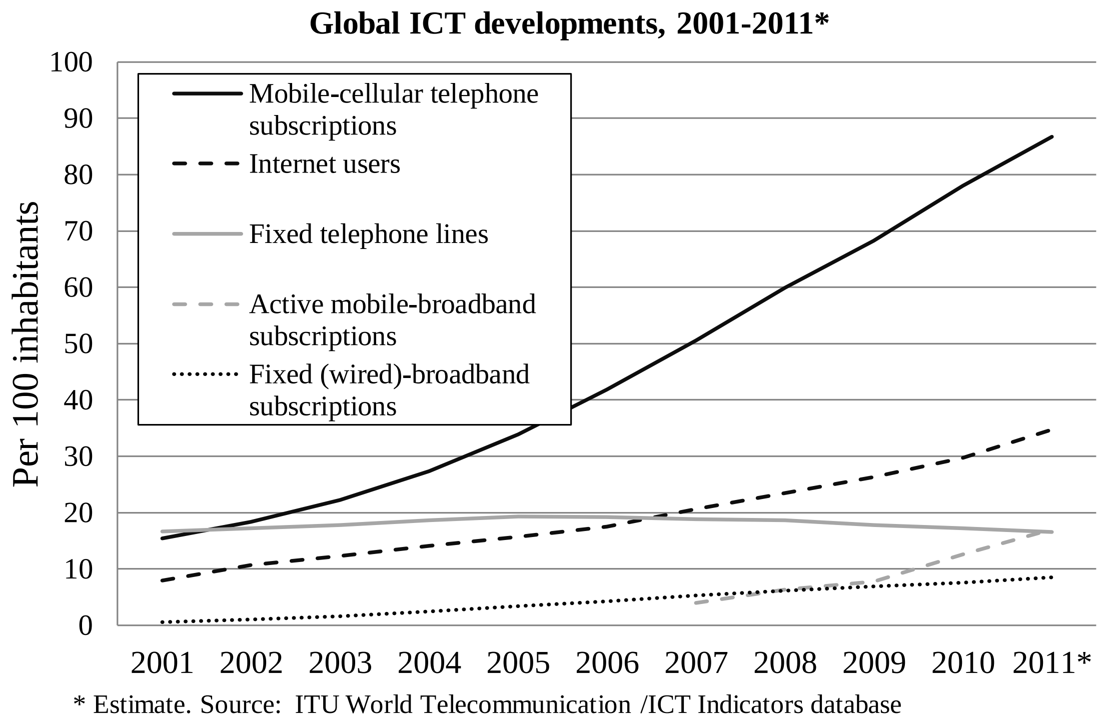


Figure 1: Global ICT developments, 2001-2011. Source: ITU[2]

One of the most important points to consider when developing software for mobile devices is that the application is usable even on devices having the properties mentioned before. Therefore, user interfaces (UIs) have become an essential part in the development of mobile applications. End users not only expect well designed but also intuitively operable and simple but at the same time powerful user interfaces. Current smartphones and tablet computers, for example, often provide a large touchscreen for

[2] ITU: International Telecommunication Union, *http://www.itu.int/ITU-D/ict/statistics/*

most of the user input. Therefore, a well-designed user interface must fulfill the expectations end users have when intuitively touching the screen.

Mobile devices are used more and more in professional environment as well. However, mobile workers are often in complex and hectic work places. For mobile workers who work outdoor, potentially under rough conditions, it is essential to have easy, quick and intuitive user interfaces. Being outdoor might also complicate receiving help for operational problems. Remote help via telephone might be difficult, especially if the mobile device running the problematic app is the telephone at the same time. Apart from issues concerning the user interface, mobile workers should not have to care about synchronizing data or similar technical aspects. Therefore, in the professional but also in the nonprofessional area the main challenge is to design the user interface and the operation of the software in such a simple manner, that the end user can completely concentrate on his/her task - not the device. However, at the same time the user interface and the application must not lose functionality.

But what makes a user interface simple? Experienced end users might feel patronized by too simple user interfaces which hide advanced features. From the inexperienced end user’s point of view a user interface which simultaneously provides all available features on the screen might be confusing or distracting. So, simple must not necessarily have the same meaning for all types of end users. Some might consider a simple UI as UI which does not hide any features; others might consider a simple UI as UI which does hide unimportant features. Figure 2 shows a satiric comparison of simple user interfaces compared to a more complex user interface.

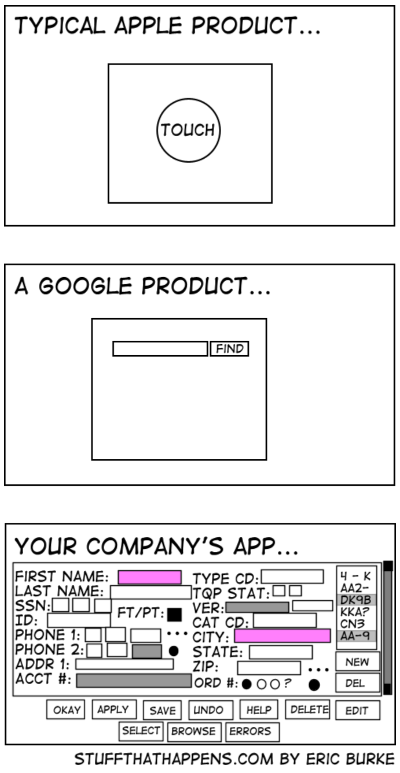


Figure 2: Simple vs. complex user interfaces (found in Lorz (2010))

One possible solution for addressing the challenge mentioned in the previous paragraph is on smart adaptation (El-Bakry et al., 2010). Adaptive user interfaces (AUIs) are user interfaces which are modified dynamically in a way that the special demands/requirements/needs of individual end users are satisfied. In other words, adaptive user interfaces can be used to present the end user a UI which is tailored to the special needs of the end user (Germanakos P., 2009). Smart adaption means that adaptive user interfaces should not arbitrarily make certain parts less accessible by hiding functions in submenus. Instead, the system should take the context into account for deciding which parts of the UI are currently needed and which are not.

The term context has many different meanings (Yuan-Kai, 2004, Schmidt, 2000) and includes the end user's experience, special abilities of the users, current needs and feelings as well as environmental factors (surrounding light, noise, location, …) and the system's or the device's current state (Schmidt (2000) called the latter "situational context"). Also, time must not be forgotten when referring to the context. According to Schmidt et al. (1999) the term context is defined as "*that which surrounds, and gives* *meaning to something else*" in The Free On-line Dictionary of Computing. All relevant contextual factors should be taken into account when designing adaptive user interfaces. For retrieving information about the situational context a large amount of sensors is available in current smartphones and tablet computers. However, one of the big challenges in this area is to determine the variables used as basis for developing different adaptions (Germanakos P., 2009). Consequently, the designer needs to know the limits of context and systems and that mobile end-users interact differently (Holzinger and Errath, 2007).

Today, location is mostly used for determining the environmental context. But location-awareness is only one part of context-awareness (Schmidt et al., 1999). Combined data from different sensors can give a more holistic understanding of the current context (Gellersen et al., 2002).

The key question about smart adaptive user interfaces is "Do simpler user interfaces which were created using smart adaption enhance the performance of end users?" Therefore we assume that smart adaption for simplifying user interfaces which meet the special requirements of the current context do enhance the performance and try to support this statement by conducting an experiment.

1.1. Structure of this Thesis

The work consists of four main sections plus three sections including the lessons learned, a conclusion, and future work. Lists of figures and tables, a glossary, a list of references can be found before the appendix.

This section not only provides an overview of the work but also explains why research on this topic is important. Section 2 describes terms and theoretical concepts around the topics mobile devices, user experience, adaptive user interfaces, and enterprise applications. Existing literature was studied to give an overview of the development and the state-of-the-art of relevant technologies.

Section 3 presents the practical work which was conducted within the context of this work. This includes the usability consultancy activities for Boom Software AG but also the development and the design of an experiment for comparing the performance of an adaptive user interface in contrast to a non-adaptive user interface as well as the documentation of the elaboration of a usability checklist for mobile devices.

In Section 4, the outcomes of the practical work are discussed.

Finally, the lessons learned during the work on this thesis and during the practical work are stated, a conclusion is made and work to be done in the future is mentioned. The work also includes a list of figures, tables and references as well as a glossary for the most used and the most important terms.

2. Theoretical Background and Related Work

This section covers concepts and foundations of the main areas this work is about: Smart Adaptive User Interfaces, User Experience, Enterprise Applications, and Mobile Devices. Starting with basics about mobile computing, mobile devices and mobile applications in Section 2.1, an explanation of enterprise applications follows in Section 2.2. The term user experience as well as usability in general and usability of mobile devices in particular are discussed in the following Section 2.3. Finally, adaptive user interfaces are covered in Section 2.4.

Before it is went on with the first subsection, we define the term user interface (UI). A user interface is the interaction point between a person and a device. The person interacting with the device (the user) operates the device via the user interface. The user interface must not necessarily be a graphical user interface (GUI). UIs can also be haptic interfaces, command line (text based) interfaces or other types of interfaces. On current mobile devices such as smartphones, tablet computers or notebooks the user interface mostly is a graphical user interface operated via touchscreen or via mouse and keyboard.

2.1. Mobile Devices

The focus of this subsection is on mobile computing and mobile devices. Before we describe current mobile device types and the future trend in mobile computing, we will give a short overview of the historical development of mobile devices. We will distinguish between notebooks, tablet computers, different types of mobile phones, such as smartphones, feature phones, classic cell phones, and other mobile devices.

The umbrella term mobile computing is strongly related to mobile devices and describes any technology that enables people to access information and which supports them in daily workflows independent of location (Holzinger and Errath, 2007).

When creating mobile applications, one has to face the fact that the capabilities of mobile devices are different than the capabilities of desktop computers or notebooks. In this work we are also talking about the challenges of porting business applications to mobile devices (i.e. smartphones and tablets). However, we do not consider games or applications which make vast use of advanced graphics. Such applications have different needs in respect to optimization of power consumption and input devices.

The applications we are talking about can be created mostly with the default controls provided by different SDKs (software development kits). This includes controls such as labels, text boxes, buttons, lists, single images, tab controls and so forth. This is what common business applications are usually composed of.

However, the domains for mobile devices are numerous. Mobile devices are nowadays not only used in the private sector and in businesses. Also in the area of health care, in the area of e-learning, in the area of sports, and in many more areas mobile devices are used. Larger mobile devices, such as notebooks or netbooks, are often not suitable for situations where high portability is required. Also, notebooks might be too expensive and overkill for certain scenarios. Holzinger et al. (2005b) proposed the use of mobile phones instead in the area of e-learning. The advantages of mobile phones are that they are not only highly portable but also relative inexpensive. Therefore, most of the students possess one – in contrast to a notebook.

Table 1 presents a comparison of the hardware capabilities and properties of a typical notebook and a typical smartphone from the year 2011. The most drastic differences can be seen in the display size, the input methods, the computational power and the power supply.

Display size. The display size is limited on most mobile devices. One challenge is to create applications which make optimal use of the limited screen dimensions. The user interface should provide high functionality and be simply usable at the same time. When porting websites or desktop applications to mobile devices it must always be asked whether full functionality is required or whether it makes sense to limit the functionality of the mobile application. If the mobile application may have limited functionality compared to the non-mobile application, which features can be left out? It might be feasible to port only certain parts of the desktop software to the mobile device. In certain cases it might also be feasible to add extra features to the mobile application which cannot be found in the desktop version. These questions have to be decided from case to case. However, concepts such as smart adaption of user interfaces could always contribute to simpler user interfaces.

	Notebook	Smartphone
Screen size	15’’ to 17’’	4’’
Screen resolution	1280 x 768	480 x 800
Input devices	Physical keyboard, mouse, sound recording (e.g. used for speech recognition)	Touchscreen, virtual keyboard, sound recording, camera, positioning and GPS sensors, accelerometers, other sensors
Computational power[3]	About 30 GFLOPS/s	About 50 MFLOPS/sec
Typical power supply	Battery and AC	Mostly on battery
Battery life	8 hours on battery	One to two days (in stand-by mode). Several hours in use.
Usage	Mostly stationary	Mobile only

Table 1: Comparison of Notebook and Smartphone

Another challenge in the area of screen sizes of mobile devices is to create applications which run on several different device types with different screen sizes. The popular Android operating system, for example, is available for many different device types, such as smartphones and tablet computers. When developing applications for Android it must either be ensured that the application works with all imaginable screen sizes, or the availability of the application must be limited to devices with supported screen sizes. In general, this challenge is also present in the area of desktop systems; however, the screen size divergency between screens for desktop computers is usually relatively low. Also, due to the in general relatively large screen sizes, divergences do not have such a high impact compared to mobile devices.

Input. Although pointing and clicking is somehow similar to touching, there are several differences to consider. Using fingers and a touchscreen, it is not possible with current

[3] Measured by Linpack for Android on Nexus S and a Linpack based benchmark for Intel Core i5.

technologies for the mass market to simply point somewhere (without clicking). Popups triggered by "MouseOver"-events are therefore useless in the mobile touchscreen world. Also, using a mouse it is much easier to click on a precise point on the screen than using clumsy fingers. Also the so called "fat finger" problem has to be considered when touchscreens are used for input (Siek et al., 2005). The problem addresses the issue that fingers may hit the wrong target if the targets are small and close to each other. Also the fact that fingers hide the part of the screen under the finger is a problem especially on small touch-based devices (Baudisch and Chu, 2009). The positive aspects of using touch for input are new intuitive ways of operating such devices. Multi-touch screens make it possible to use more than one finger at the same time and therefore allow new ways of interaction.

Computational power and power supply. When creating mobile applications it must be taken into account that the computational power of mobile devices is limited. Although smartphones are powerful enough for running even complex applications nowadays, their computational power still cannot be compared to desktop computers. Complex and long calculations should be avoided on mobile devices as they also reduce battery life drastically. Although the stand-by time of modern smartphones is acceptable (about two days), the charging level of the battery goes down rapidly when the device's hardware is under high load.

2.1.1. Mobile Computing Da Capo

Already in 1908, a patent was issued to Nathan B. Stubblefield (Stubblefield, 1908) for a mobile phone device. However, the first cellular network as we know it today was established 76 years later in Japan[4]. In September 1983 the first commercial mobile phone was released by Motorola: The DynaTAC (Dynamic Adaptive Total Area Coverage) 8000X by Motorola[5]. The first PDA (Tandy ZOOMER, short for

[4] Source: *http://scienceray.com/physics/the-history-of-mobile-phones/*, last visited 07/02/2012
[5] Source: *http://www.focus.de/digital/handy/mobilfunkgeschichte/tid-10733/der-urahn-der-handys-motorola-dynatac-8000x_aid_310544.html*, last visited 07/02/2012

"consumer") was introduced by Palm in 1992. It had 1 MB RAM, 4 MB ROM, a monochrome LCD display (resolution: 320 x 256 pixels)[6].

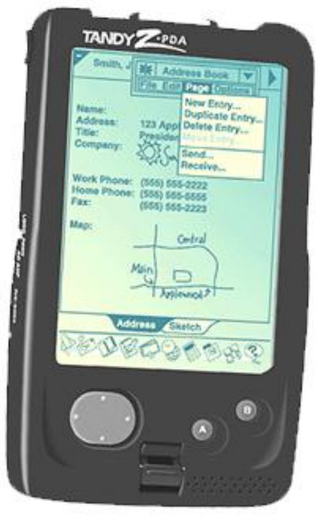


Figure 3: Tandy ZOOMER, source: *8bit-micro.com*

Apple's Newton Message Pad (1993) was a tablet computer having 640 kilobytes of RAM and an ARM CPU with 20 MHz. It was supplied with power using four AAA batteries which lasted six to eight hours. The display resolution was 240x336 (Forman and Zahorjan, 1994).

The first smartphone (Figure 4) was designed by IBM in 1992 and included several applications, such as calendar, address book, world clock, calculator, note pad, email, and games (Lobo et al., 2011). It was operated via touchscreen.

Most of the early devices did not sell well due to the high price, the low performance and the large size. But already in 1994 Forman and Zahorjan stated that wireless internet connections are essential in mobile computing (Forman and Zahorjan, 1994). Security issues and low bandwidths are still challenges to be dealt with. Providing location-dependent information is one obvious feature mobile devices can provide. Low power capacity has to be regarded when designing mobile applications.

[6] Source: *http://www.8bit-micro.com/tandy-zoomer-z-pda.htm*, last visited 07/02/2012

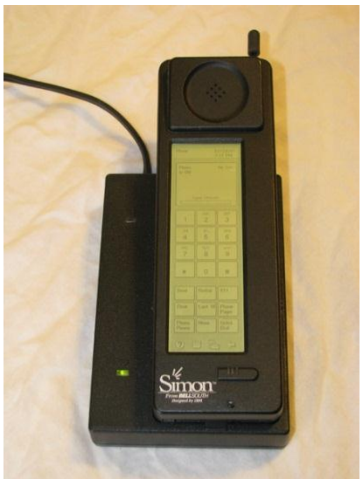


Figure 4: The first Smartphone "Simon" by IBM 1994
(Source: Wikimedia Commons, own work by user Bcos47)

Since 1999 RIM produces BlackBerry phones[7] which have relatively large screens and offer e-mail and organizer functionalities. RIM's operating system BlackBerry OS is extensible with third-party applications. BlackBerry phones are typically operated with a built-in hardware keyboard and a track pad, later versions also via touchscreen.
Up to 2007 so called feature phones became more and more popular. Feature phones are mobile phones which provide additional functions such as a calendar, a music player or simple games, but are usually small sizes and do not provide touchscreens.

2.1.2. Mobile Computing Today

Since Apple introduced the iPhone in 2007[8], smartphones as we know them today (large touchscreens dominating the device) became more and more popular. Two years later, when the Open Handset Alliance (founded by Google in 2007)[9] introduced their smartphone operating system Android[10] the market was opened for other smartphone producers. The Linux-based open source operating system Android can be run on any capable device while iOS, Apple's operating system for mobile devices, is only allowed to be run on Apple devices - the iPhone and later the iPod Touch and the iPad.

[7] http://us.blackberry.com/company.jsp
[8] *https://www.apple.com/*
[9] *http://www.openhandsetalliance.com/*
[10] *http://www.android.com/*

Currently, the most popular operating systems for smartphones and tablets are iOS by Apple and Android by Google. As shown in studies by Gartner and Nielsen (Table 2 and Figure 5) these operating systems have the highest market share. Windows Phone as young but promising operating system, currently has a very low market share and is contained within "other" operating systems in the market share statistics. It is predicted by Gartner that the Android operating system stays the most popular mobile operating system for the next few years.

OS	**2010**	**2011**	**2012**	**2015**
Symbian	111,577	89,930	32,666	661
Market Share (%)	37.6	19.2	5.2	0.1
Android	67,225	179,873	310,088	539,318
Market Share (%)	22.7	38.5	49.2	48.8
Research In Motion	47,452	62,600	79,335	122,864
Market Share (%)	16.0	13.4	12.6	11.1
iOS	46,598	90,560	118,848	189,924
Market Share (%)	15.7	19.4	18.9	17.2
Microsoft	12,378	26,346	68,156	215,998
Market Share (%)	4.2	5.6	10.8	19.5
Other Operating Systems	11,417.4	18,392.3	21,383.7	36,133.9
Market Share (%)	3.8	3.9	3.4	3.3
Total Market	**296,647**	**467,701**	**630,476**	**1,104,898**

Table 2: Worldwide Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units). Source: Gartner (April 2011)

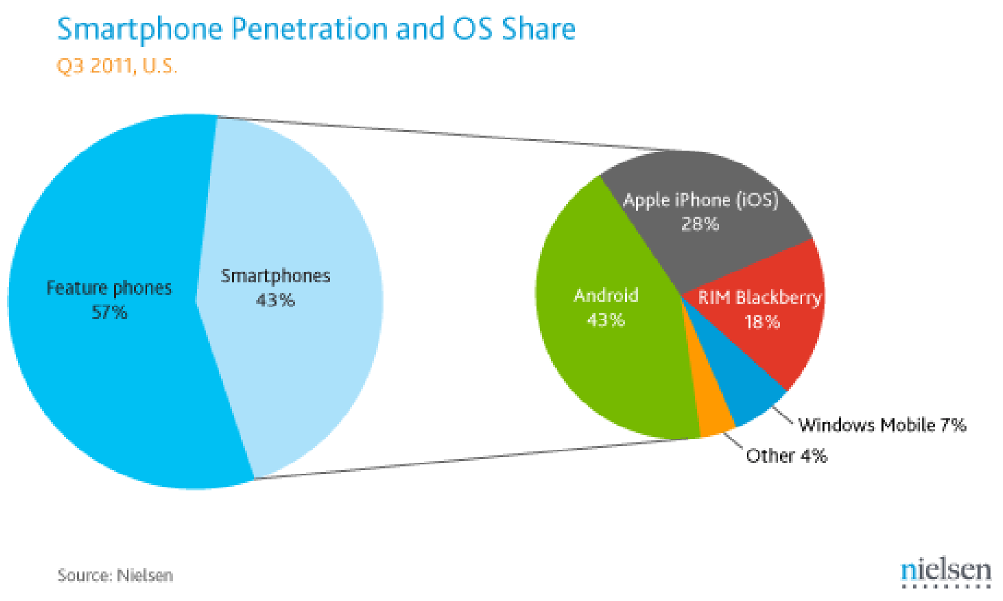


Figure 5: Smartphone Penetration and OS Share, third quarter 2011, USA. Source: Nielsen

2.1.3. Future of Mobile Computing

Currently clear trends concerning the future of mobile computing and of mobile devices can be observed. According to a new Gartner study[11] Mobile devices – especially smartphones and tablets – do more and more replace desktop computers. Devices are getting more compact and at the same time more powerful. Sensors are integrated in more and more mobile devices. Location-aware computing becomes more popular and the integration of NFC (Near Field Communication) readers proceeds. Portable devices with large screens (tablet computers) are also gaining popularity. Also the so-called Internet of things or Internet of devices grows – even faster than the desktop Internet (Schilit, 2011).

Forrester research has published the expected technology trends for the upcoming two years (Figure 6). Business intelligence is expected to be a field which will underlie the highest changes and will have the highest impact in the near future. Business intelligence refers to collecting and analysis of (big) business-related data in order to optimize the company’s strategy. The relevance of mobile apps and application

[11] Found via *http://www.telekom-presse.at/Gartner_Tablets_und_Smartphones_bedrohen_den_PC-Markt.id.18689.htm*, last visited 05/02/2012

platforms is estimated to play an important role in the near future as well. Mobile platforms are expected to change to a high degree within the next two years.

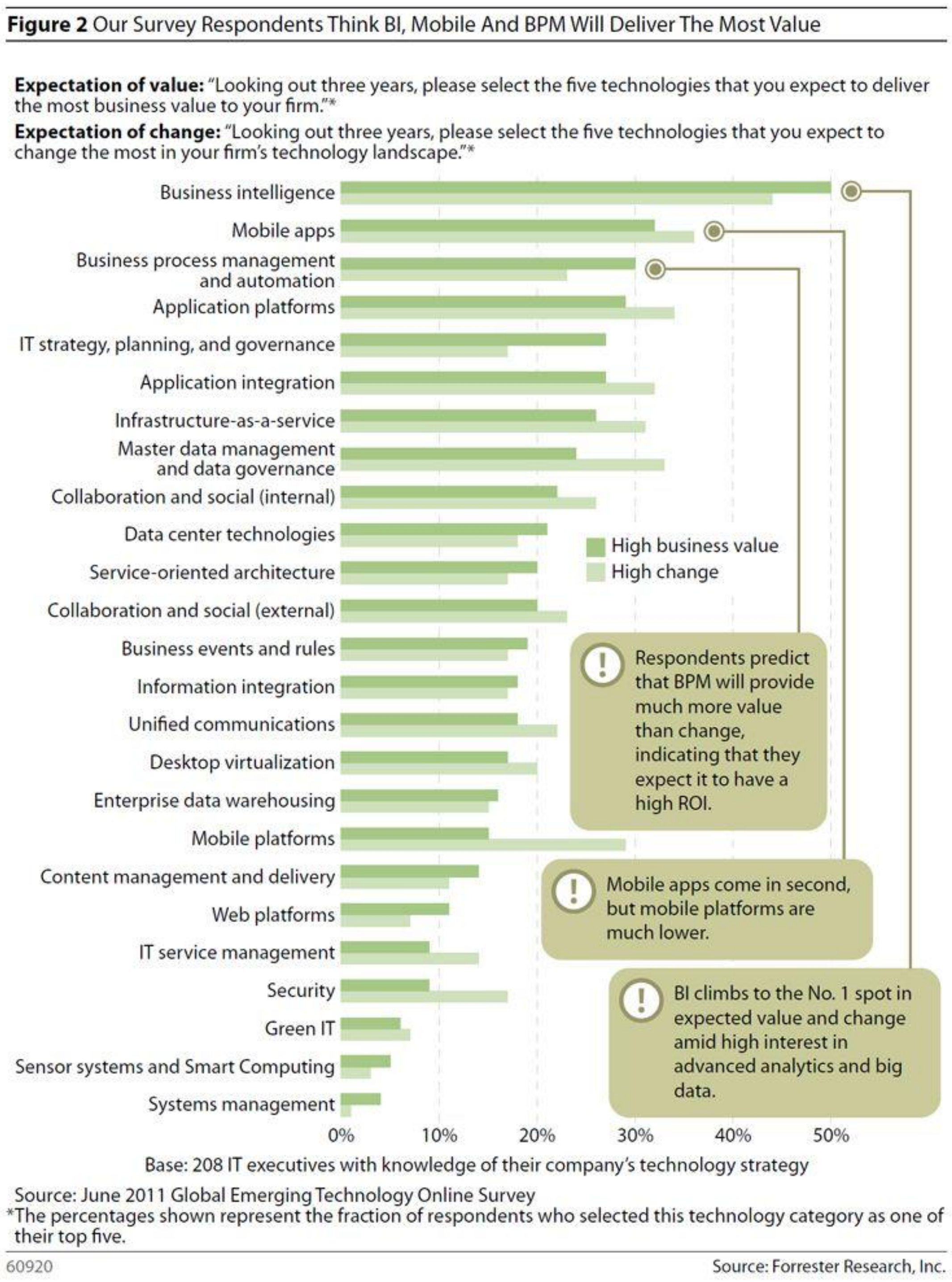


Figure 6: Ranking of technology trends 2012-2014. Source: Forrester Research, Inc. via *http://www.cio.de/knowledgecenter/bi/2292300/*

According to Gartner the top 10 strategic technologies for 2012 include (cited (shortened) from Gartner's Weblog[12]):

- *Media Tablets and Beyond: Users can choose between many different devices and form factors.*
- *Mobile-Centric Applications and Interfaces: UIs with windows, icons, menus, and pointers will be replaced by mobile-centric interfaces emphasizing touch, gesture, search, voice and video. Applications themselves are likely to shift to more focused and simple apps that can be assembled into more complex solutions. These changes will drive the need for new user interface design skills. Application user interfaces that span a variety of device types are needed.*
- *Contextual and Social User Experience: Context-aware computing uses information about an end-user or objects environment, activities, connections and preferences to improve the quality of interaction with that end-user or object. A contextually aware system anticipates the user's needs and proactively serves up the most appropriate and customized content, product or service.*
- *Internet of Things: The Internet of Things (IoT) is a concept that describes how the Internet will expand as sensors and intelligence are added to physical items such as consumer devices or physical assets and these objects are connected to the Internet.*
- *App Stores and Marketplaces: Application stores by Apple and Android provide marketplaces where hundreds of thousands of applications are available to mobile users. Gartner forecasts that by 2014, there will be more than 70 billion mobile application downloads from app stores every year.*
- *Next-Generation Analytics: Analytics is growing along three key dimensions: From traditional offline analytics to in-line embedded analytics. This has been the focus for many efforts in the past and will continue to be an important focus for analytics. From analyzing historical data to explain what happened to analyzing historical and real-time data from multiple systems to simulate and predict the future. Over the next three years, analytics will mature along a third*

[12] *http://www.gartner.com/it/page.jsp?id=1826214*, article from October 18, 2011, "Gartner Identifies the Top 10 Strategic Technologies for 2012", last visited 30/01/2012.

dimension, from structured and simple data analyzed by individuals to analysis of complex information of many types (text, video, etc…) from many systems supporting a collaborative decision process that brings multiple people together to analyze, brainstorm and make decisions. Analytics is also beginning to shift to the cloud and exploit cloud resources for high performance and grid computing.

- *Big Data: The size, complexity of formats and speed of delivery exceeds the capabilities of traditional data management technologies; it requires the use of new or exotic technologies simply to manage the volume alone.*
- *In-Memory Computing: Gartner sees huge use of flash memory in consumer devices, entertainment equipment and other embedded IT systems.*
- *Extreme Low-Energy Servers.*
- *Cloud Computing.*

Gartner points out that the variety of devices will grow and that touching will stay an important way for interaction with mobile devices. Also contextual information will be used for customizing applications to the need of the end-user. Also (historical) data analysis for prediction purposes is an important field in the future as well as collaborative decision processes. More and more big and complex data is to be handled and analyzed.

Location-based services will be more widely used in the future and GPS receivers will be integrated in more and more mobile devices. However, security and privacy concerns slow this development down (Vaughan-Nichols, 2009). Solving the privacy issues related to location-based services will therefore be a big challenge in the future.

2.1.4. Device Types

In the next few paragraphs we will gain an overview of the large variety of different mobile device types. Therefore definitions from different sources were collected and compared. We will see that in some cases a clear distinction between the device types is not easy to make (for example for the terms notebook and laptop). In everyday speech certain technically different device types are used as synonyms (for example tablet PC, tablet computer). In general, devices which are more or equally portable than notebooks and have independent power supply (i.e. can run on battery) are usually included in the class of mobile devices. However, it has to be clarified that within this work, when referring to mobile devices, in most cases smartphones or tablet computers are meant as the focus of this work is on mobile devices with touchscreens.

Notebooks and Laptops. According to Oxford Dictionaries (OD) a notebook is *"a laptop computer, especially a small, slim one",* while a laptop is defined as *"a computer that is portable and suitable for use while travelling."*
According to the Cambridge Advanced Learner's Dictionary & Thesaurus (CALDT) a laptop is "*a computer which is small enough to be carried around easily and is designed for use outside an office*". CALDT does not provide a definition for notebooks in the sense of portable computers.

Subnotebook. According to the pcmag.com encyclopedia a subnotebook is "*a laptop computer that weighs less than four pounds. In order to reduce weight, subnotebooks, also called "ultraportables" or "ultralights," often eliminate built-in CD/DVD drives.*"

Netbook. According to Oxford dictionaries a netbook is "*a small laptop computer designed primarily for accessing Internet-based applications.*" The pcmag.com encyclopedia defines a netbook as "*a subnotebook computer in the $200 to $400 U.S. dollar range (as of 2010). [...] netbooks have screens in the 8"-10" [...]. The term was coined by Intel in 2008 for machines that used its Atom microprocessor.*"

Tablet PC. According to pcmag.com encyclopedia a tablet PC is "*a tablet computer that runs Windows*". According to Oxford Dictionaries a tablet PC is "*a small portable computer that accepts input directly on to its screen rather via than a keyboard or mouse*". According to the Cambridge Advanced Learner's Dictionary & Thesaurus tablet PC is "*a small computer with a screen that you can write on using a special pen or that you can connect a keyboard to*". A tablet PC can also be seen as special notebook which is mostly used with touch input.

Ultra-mobile PC. An ultra-mobile PC (UMPC) is, according to the pcmag.com encyclopedia, "*a lightweight, small tablet PC with an on-screen keyboard introduced in 2006. Weighing two pounds or less and using a hard drive for all content, the Ultra-Mobile PC (UMPC) runs under the Windows Tablet PC operating system with Touch Pack software.*"

Tablet computer. According to pcmag.com encyclopedia a tablet computer is "*a general-purpose computer contained in a single panel. Its distinguishing characteristic is the use of a touch screen as the input device. Modern tablets are operated by fingers, whereas earlier tablets required a stylus*". Tablet computers are also called tablets. Tablets are a larger version of smartphones, but basically provide similar functionality. Due to the larger (screen) size of tablets they target other usage scenarios than smartphones.

Smartphone. According to Oxford Dictionaries a Smartphone is "*a mobile phone that is able to perform many of the functions of a computer, typically having a relatively large screen and an operating system capable of running general-purpose applications*". According to the Cambridge Advanced Learner's Dictionary & Thesaurus a Smartphone is "*a mobile phone that can be used as a small computer*". When we talk about smartphones in every-day life we typically mean a mobile telephone having a large touchscreen as main input device which runs a feature-rich operating system providing an API for running apps from third-party developers.

Feature phone. According to Oxford Dictionaries a feature phone is "*a mobile phone that incorporates features such as the ability to access the Internet and store and play music but lacks the advanced functionality of a smartphone*".

Mobile Phone and Cell phone. According to Oxford Dictionaries and to the Cambridge Advanced Learner's Dictionary & Thesaurus a cell phone simply is "*a mobile phone*". According to Oxford Dictionaries a mobile phone is "*a telephone with access to a cellular radio system so it can be used over a wide area, without a physical connection to a network*".

Other device types. Devices such as the iPod Touch are very similar to smartphones; however, the capability for connecting to cellular networks is missing. However, there are many more device types than the types listed above. Portable music players, navigation devices, e-book readers, and portable gaming consoles are only four examples. Also very small devices which can be integrated into clothes or even can be implanted into the body may be considered as mobile devices. Another example for mobile devices are digital pens. There are aims to make paper interactive by using digital pens which transmit their location on the paper to the connected device (Signer et al., 2007) .

As stated before, this work focuses on mobile devices which can be classified as smartphones or tablet computers which are mainly operated using a touchscreen, without stylus. Therefore we will now take a closer look on these devices and we will ask how especially smartphones influence our lives.

Smartphones are very powerful devices – even more powerful than average desktop computers around the year 2000. Therefore smartphones can also be considered as portable computers having a telephone function. More and more people exchange their cell or feature phones with smartphones although the price of smartphones is significantly higher than the price of feature phones. Table 3 lists some positive and some negative aspects of modern smartphones compared to classic (cell) phones. However, based on the trend mentioned above that more and more people are getting smartphones, it can be concluded that the positive aspects of smartphones seem to dominate the negative effects.

	Positive aspects	Negative aspects
Social	Allows to be always connected with the workplace and friends via social networks.	Owner is always contactable - even at mistime.
	Allows to look for information everywhere and anytime.	Privacy issues.
	Supports private reporting in areas where free press is constrained.	Web surf behavior cannot be supervised and controlled easily by parents.
	Status symbol (for those who have one).	Missing status symbol might pressure children.
Technical	Powerful hardware in one relatively small device (sensors, large screen).	Relatively expensive.
	One device replaces several devices (camera, music player, satellite navigation system)	Shorter battery life.
	Extendable software via apps.	Relatively large compared to small feature phones.

Table 3: Positive and negative aspects of smartphones compared to classic (cell) phones.

2.1.5. Mobile Platforms

There are several different mobile operating systems available nowadays. The most popular systems are currently iOS by Apple and Android by the Open Handset Alliance (see Section 2.1.2). More current operating systems for mobile devices include Windows Phone (Microsoft, 2010), Bada OS (Samsung, 2009), BlackBerry OS (2002), Symbian (Nokia/Accenture), and webOS (Palm, 2009).

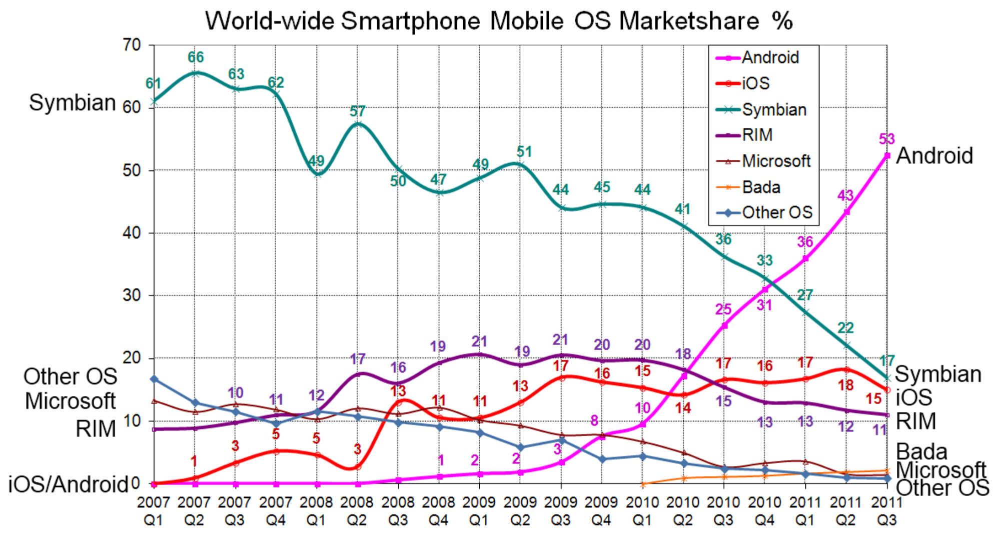


Figure 7: World-wide smartphone mobile OS market share. Image source: Wikimedia Commons, own work by user milominderbinder2; data originally from Gartner.

According to Net Applications' service *netmarketshare.com* in end of 2011 iOS and Android are the most popular platforms in the mobile segment. On desktop computers Microsoft's Windows has a market share of more than 90 percent, while Apple's operating systems have a market share of about 6 percent. Since the beginning of 2011, when Nokia and Microsoft announced a partnership[13], Windows Phone 7 is another candidate for a future popular smartphone and tablet operating system.

Mobile platforms usually include the operating system, a documented API which makes operating system functions available for third-party developers, and web services which are provided by the OS vendor. For unified publishing of apps most OS vendors offer a platform for publishing native apps. Most of the current operating systems for smartphones and tablets are capable of running both so called native apps and HTML5 apps. A so-called HTML5 app (also RIA, Rich Internet Application) basically is a website which often makes vast use of HTML5 and JavaScript features[14]. The website can be displayed in any browser and is usually designed and programmed in a way that the site resembles the design and the behavior of a native smartphone app. The main properties of a HTML5 app are: Platform independency, instantly updateable from server side, inability to access all device features (such as certain sensors), lack of

[13] *http://www.bbc.co.uk/news/business-12427680*, post from 11/02/2011, last visited 11/02/2012.
[14] Differences to earlier HTML versions: *http://www.w3.org/TR/html5-diff/*, last visited 04/02/2012

smooth integration in the system, not monetizable via the operating system provider's billing infrastructure (Android Market (Google Play) / Apple App Store).

Native apps do not run in inside a browser. They are – as the browser itself – programmed for the native mobile operating system's API. The main properties of native apps are: Platform dependency, seamless integration in the OS-environment, higher performance, possibility of using all available hardware and software capabilities. From now on the term app refers to native apps if not stated otherwise.

Native apps, in contrast, are executed directly by the operating system's execution environment. The following enumeration lists the names of the execution environments of the operating systems Android, iOS, and Windows Phone.

- Android: DVM (Dalvik Virtual Machine), Linux and Java-based
- iOS: Objective C Sandbox
- Windows Phone 7: Silverlight, .NET-based, therefore C# and Visual Basic .NET are, amongst others, supported programming languages.

Table 4 lists the main differences between native applications and web applications. Although native apps might run faster than HTML5 apps most apps don't really need the speed of native execution. But on the other hand on mobile devices where performance is expensive the app development should keep this fact in mind. Developing software using platform-specific tools might be easier and certain tasks are easier to perform in native apps. In contrast, apps have to be developed for every operating system to be supported. HTML apps run on any operating system where a HTML5 capable browser is available (usually built-in).

Native App	*HTML5 app*
Runs natively on the OS' platform and therefore usually faster.	Runs in a browser.
Different technological frameworks	More or less[15] same technological framework.
Possibility of using all intended hardware features.	Restricted possibilities for accessing hardware.
Usage of platform's billing system.	Third-party billing system must be used.
Look-and-feel corresponds to the OS.	UI can be designed to approximate the desired look-and feel.
Deployment of apps unified by app stores / marketplaces.	No unified app distribution framework.

Table 4: Differences between native apps and HTML5 apps.

When developing and publishing native apps for the different platforms there are several differences to regard between the platforms of Apple, Google, and Microsoft:

- The hardware capabilities of phones running the mentioned operating systems are similar. However, Apple's iOS is bound to very few device types, the iPod, the iPhone, and the iPad, while for Android there are no such strict restrictions.
- The programming languages used for developing native apps differ, however. Apps for iOS are written in Objective C, apps for Android are written in Java, apps for Windows Phone 7 are usually written in C# or Visual Basic .NET. Also the language for defining user interfaces differs amongst the different operating systems. The development of iOS applications is bound to Apple computers, the development of Windows Phone applications is bound to Windows PCs. Applications for Android can be developed on Windows PCs, Apple computers and on machines running Linux.

[15] Different browsers implement different implementations for interpreting HTML/CSS and for JavaScript execution.

- Apps are checked for certain criteria before being published on the operating system’s app shop. The restrictiveness, however, differs greatly. Apple apps have to pass a quality check before being approved; Microsoft also requires certain criteria to be met. Google, in contrast, has very few mandatory quality criteria.

App distribution services. Apps can be published by developers by using an app distribution service – a service provided by the mobile operating system’s vendors for unifying the app distribution for their systems. The different OS vendors use different names for their service: Apple (for iOS) calls it App Store[16], until March 2012 Google (for Android) called it Market[17] (now Google Play), and Microsoft (Windows Phone) calls it Marketplace[18]. From now on, when referring to a generic app distribution service, the terms app shop or app distribution service will be used. For developers it is possible to publish apps for free as well as for a certain fee. When selling apps via one of the app distribution services currently all three companies keep back 30 percent of the fee. The amount of the fee can be chosen freely by the app’s publisher (however, there is a lower and an upper limit). In December 2011 the Android Market exceeded 10 billion app downloads[19]. There are also app shops for RIM’s BlackBerry (App World) and for Nokia phones (Nokia Store).

According to *http://androidistic.com*[20] more than 37 percent of the apps hosted in the Android app store are categorized as game or entertainment App. More than ten percent are categorized as social or communication. The other half of the apps are from many different categories, such as system tools, travel guides, music players and many more.

[16] *http://itunes.apple.com/us/genre/mobile-software-applications*
[17] *https://market.android.com/*
[18] *http://www.windowsphone.com/marketplace*
[19] *http://android-developers.blogspot.com/2011/12/10-billion-android-market-downloads-and.html*
[20] *http://androidistic.com/961/android-market-growth-chart-2012/*

2.1.6. Input, Sensors, and UI controls

An input method (or input method editor, IME) is an application which creates or translates input for other applications. Examples for input methods are soft keyboards (on-screen software keyboards), hand-writing recognizers, and hard keyboard translators[21]. Input devices, in contrast, are physical devices such as physical keyboards for creating input for applications. Touch input is very common and popular nowadays and provides a very intuitive way of interaction. Clicking by tapping and scrolling by moving the finger are just two examples for natural interaction with touch screens (Rieman et al., 2008).

Although much input on mobile devices with touch screens is made by touching and selecting predefined values, text input via on-screen keyboard is still widely used. However, applications that require text input often annoy users (Longoria, 2001). Therefore it is important to reduce text input to a minimum (Holzinger et al., 2006). An alternative for text input via on-screen keyboard is handwriting recognition. Already before the era of the smartphones, there were aims to bring handwritten text to computers (Tappert et al., 1990). On tablet PCs - often used as (paper) notebook replacement - handwriting recognition aims not only to preserve the classic paper notebook feeling, but also to speed up text entry (Pittman, 2007). However, studies also show that users prefer soft qwerty/qwertz keyboards over handwriting for text input (Holzinger et al., 2010). On small devices users still prefer tapping over handwriting recognition.

Some on-screen keyboards use techniques for speeding up text input, such as prediction of the word to be typed. However, it has been shown that input via predictive keyboards is not more effective than input via standard soft keyboard (Lewis, 1999). SpreadKey, a predictive soft-keyboard system (Merlin and Raynal, 2010) dynamically makes the most probable characters more accessible. It was shown by Merlin and Raynal that SpreadKey turns the input easier and less tiring physically. KeyGlasses (Raynal and Vigouroux, 2005) also uses character prediction to display the most probable next characters in accessible positions on the soft keyboard. However, it could not be shown

[21] *http://developer.android.com/resources/articles/on-screen-inputs.html*

that KeyGlasses is a better solution than a simple classic non-predictive qwerty soft keyboard.

One advantage of soft keyboards compared to physical (hardware) keyboards is that buttons can appear or disappear according to the interaction context. However, according to the “engadget” weblog (Melanson, 2010) Microsoft created an adaptive hardware keyboard where the button labels can be exchanged similar to software keyboards on current smartphones. The key captioning is realized with LCD panels. One disadvantage of software keyboards is the lack of tactile feedback. It reduces input performance (Lee and Zhai, 2009).

In order to exploit the feature of soft keyboards to dynamically exchange of buttons depending on the interaction context, text fields should always specify the expected input type using metadata. This enables soft keyboards to adapt the key configuration in a way that keys which are more likely to be used for certain input types, appear in exchange for less likely used keys. For text boxes which expect an e-mail address, for example, an “@” key appears while the question mark key “?” disappears.

Several enhancements such as an offset zoom decrease the user’s anxiety in using the device. Offset zoom means to display the currently pressed soft key character next to (or above) the pressed key a second time. This addresses the problem that fingers hide the part of the display which is currently below the finger. However, no performance gain can be achieved by using offset zoom (Martin et al., 2009).

Another way for (text) input is speech recognition. Speech recognition (White, 1976) in the area of mobile devices becomes more and more popular nowadays. In 2011 Apple introduced a speech recognition system for the iPhone 4S called Siri. But also Microsoft and Google make improvements in speech recognition (Tsimhoni et al., 2004, Baker et al., 2009).

Visions of even more natural interaction with computers are 15 years old (Ishii and Ullmer, 1997): Tangible User Interfaces where users interact with devices by moving and touching objects in their environment. Microsoft’s Surface is a similar concept as the ClearBoard or metaDESK devices described in the paper mentioned above.

Exploiting context (Rodden et al., 1998) is an important concept for creating adaptive user interfaces and also for reducing the need for manual (text) input. For perceiving the environment and for collecting contextual information sensors of modern smartphones and tablets can be used. Modern smartphones and tablets are supplied with a vast amount of different sensors and input devices:

- Touch-related: Soft keyboard, Gestures
  Software (on-screen) keyboards are used for text input. The term gesture refers to moving one or more fingers on the touch surface. Gestures based on more than one finger are called multi-touch gestures. Gestures are mostly used for navigation or manipulation of UI objects, but also gesture-based text input is possible.
- Haptic: Accelerometer
  Accelerometers are sensors for registering accelerations or movement of the device. This can be used, for example, for registering shaking of the device.
- GPS sensor, Gyroscope, Magnetic sensor, Height sensor
  Sensors for determining the device's location and orientation can be used for adapting the user interface to the device's current locational context. Determining the desired screen orientation or prefilling text boxes in forms based on the device's location are currently widely used features in several apps.
- Audio: Microphone
  Recording audio can be used for speech recognition as well as for song recognizing or for using the device as audio recorder.
- Optical sensors: Cameras (front, back), Brightness sensor
  Many smartphones have two built-in cameras. One is located on the same side as the display, which is mostly used for video conferences; one is located on the back side of the device, mostly used for taking photographs. However, the cameras can also be used for scanning barcodes, QR codes or other objects for speeding up (text) input. Brightness sensors are sensors which register the brightness of the environment, mostly for adapting the brightness of the screen automatically in order to make reading more comfortable. However, the environment's brightness can also be used for contextual inferences (inside/outside, day/night, cloudy/sunny).

- Other sensors: Proximity sensor
  Proximity sensors are used for determining whether the user holds the phone to his/her ear or not. If the phone is held to the ear the proximity sensor reports that an object is near to the device's screen and disables it to save energy, as the user cannot see the screen anyway.
- Physical devices: Physical keyboard, External classic input devices (mouse, keyboard): Also with smartphones and tablets it is possible to use physical keyboards and mice.
- NFC reader
  NFC can be used to easily transmit small amounts of data between two devices without the need for pairing (as in the case of Bluetooth).

For giving feedback there are also several different output devices available in current smartphones and tablets:

- Optical: 2D high-resolution color screens, in some devices even 3D-screens or secondary screens, and (flash)lights.
- Audio: Speaker
  Used for audio feedback and text-to-speech functions.
- Haptic: Vibrator
  Used for giving haptic feedback

Touch input seems to be very well accepted and easy to learn, even amongst non-computer literate people and elderly people (Holzinger, 2002, Holzinger, 2003). Devices with touchscreens and well-designed user interfaces are especially useful in hospitals, where patients can, for example fill out questionnaires while they are waiting for their examination, for the reception by the doctor, or during other spare times (Holzinger et al., 2006) Direct input of questionnaire answers by the patients makes error prone and time consuming copying of filled out paper sheets unnecessary. This saves time which can be used for direct contact with the patient and improves the overall quality of the interaction between doctors and their patients.

Touching with fingers is a very intuitive way for interaction, however, it was shown that in a professional medical context styluses are preferred over finger-based input (Holzinger et al., 2008a). So not in all cases touching with fingers necessarily is the best method for interacting with touch screens.

Input via stylus hast the advantage that input can be made more precisely and writing similar to writing on sheets of paper (for example for handwriting recognition) is made possible. For addressing the problem of imprecise touching using fingers Vogel and Baudisch (2007) developed a system called Shift which makes it possible to make more precise selections by finger. Shift shows a copy of the touched screen location and shows a pointer representing the selection point of the finger if the finger is placed over a small target. However, for further improving the precision of touch input via finger it must be better understood how people touch touchscreens (Holz and Baudisch, 2011).

Another problem with touch input using fingers is that the user's "fat fingers" also cover the areas the user intends to touch. To circumvent this problem Wigdor et al. (2007) developed a mobile device which can be operated from the back side (Figure 8). In addition by using back-of-device interaction it is possible to create very small touch devices (Baudisch and Chu, 2009).

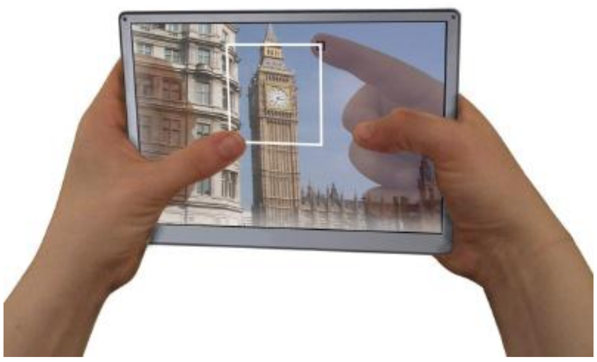


Figure 8: Concept sketch of LucidTouch: a pseudotransparent device (Wigdor et al., 2007)

There are also aims to make mouse and keyboard based user interfaces easier usable with touchscreens (Foulk et al., 2009).

For blind people input via touchscreen is problematic. Tinwala and MacKenzie (2009) presented a method for eyes-free text entry on touchscreen phones. Auditory and tactile stimuli are used to give feedback to the visually impaired user.

In the next few paragraphs we want to ask “What are the most frequent user interface controls and hardware components used in current smartphone applications?” in order to get an overview of current typical mobile apps. To answer this question first 237 randomly selected apps were investigated. In another investigation Austria’s 27 top-ranked apps in Google’s Android Market (now called Google Play) were investigated.

First investigation: Permissions required by 237 apps. Looking at the permissions Android applications request, we can see that on a phone with 237 different, randomly selected apps for different purposes (including games) 69 (29,1 percent) require permissions for accessing fine or coarse location data such as GPS data or location data from the cellular network, 29 (12,2 percent) access the camera for recording images (including barcode scans) or videos, 13 (5,5 percent) access the Bluetooth module, 12 (5,1 percent) record audio, four (1.7 percent) require access to NFC technologies (RFID reader) and 203 (85,7 percent) require full access to the internet. However, we have to keep in mind that some apps need the internet access permission just for downloading advertisement. This means that the number of applications actually needing internet access for their main functionality is slightly lower.
The fact that more than 29 percent of the investigated apps access location data supports the statement by Schmidt (2000) that location is a concept that is well understood. Apps for displaying the public transport schedule, for example, make use of the location data for create an ordered list of nearest bus stops. Within the study NFC was mostly used for scanning RFID tags which inform the app about the situational context. Using fixed tags in different rooms, for example, informs the app in which room the device currently is. This can be used to mute the phone when entering the conference room. (Krishnamurthy et al., 2006) show the possibilities of the NFC technology in combination with mobile phones. The camera and audio recording devices are often used for replacing manual text input. An Internet connection in combination with sensors can be used to download more information about the current context, such as information about the current location, and may improve the adaption of the user interface. Table 5 summarizes the mentioned numbers.

Permission	Needed by	
Location (fine or coarse)	69	29,1%
NFC	4	1,7%
Internet	203	85,7%
Bluetooth	13	5,5%
Camera	29	12,2%
Record audio	12	5,1%

Table 5: Analysis of permissions needed by 237 randomly selected apps

Second investigation: Austria's top 27 apps. A similar analysis of the usage of sensors was made by investigating Austria's 27 top-ranked Android applications (excluding games). The popularity ranking is determined by Google using a secret ranking algorithm. Certainly the ranking is influenced by the user's current location. Therefore Austria-related apps could be found in the ranking, such as the ÖAMTC (Austrian automobile club) app or the Krone.at app (the app of an Austrian newspaper). Other investigated apps include YouTube, Google Maps, Facebook, Skype, WhatsApp Messenger, Barcode Scanner, Shazam, wetter.com, TuneIn Radio and the IMDb app.

User account and internet access: One third of the investigated apps require or allow using a user account in order to receive personalized information. More than 75 percent of the investigated apps use an internet connection for their main functionality. This means that apps which try to access the internet just for downloading advertisement did not count. More than 60 percent of the apps using the internet are not working at all without network access. The remaining apps using the internet are usable without internet connection, but often only with limited functionality or cached data.

Sensor usage: 22 percent of the investigated apps use the geolocation hardware capabilities either for their main functionality (map applications) or for prefilling input controls so that the end user does not have to type or select his/her current location by himself/herself. 18.5 percent of the apps use the built-in camera to scan barcodes or QR tags in order to speed up user input. One third of the applications uses other applications to perform a task. For dialing a telephone number from inside an application the dialer application is started by simply tapping the phone number, for example.

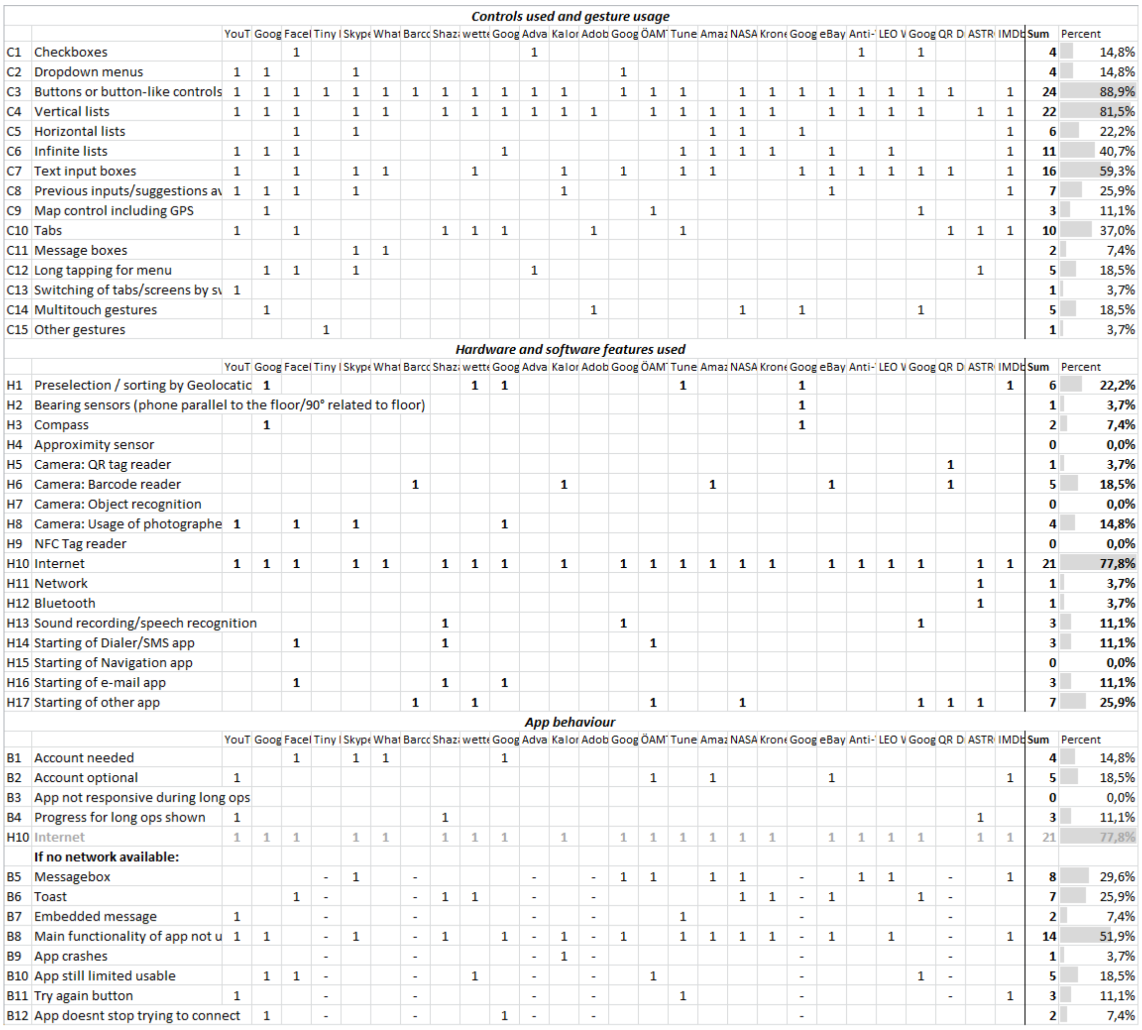
UI Controls: Buttons, Lists, Text input controls are the most frequently used controls in current smartphone applications. Tabs are often used to structure screen contents. Context menus and gestures are less frequently used. One reason might be that they are often overlooked by the end-users as there is no visual clue that they are available.

The most widely used control is, as expected, the simple button, which starts an activity or an operation by tapping it. Nearly 90 percent of the investigated apps include it in their user interface. The second widely used control is the vertically scrollable list. More than 80 percent of the apps use lists mostly containing text items. 50 percent of the lists are "infinite" lists, i.e. lists which are dynamically filled with content from remote servers when scrolling down to the bottom. 22 percent use horizontal lists, mostly for displaying images.

Text input boxes are used by about 60 percent of the investigated apps. 43 percent of the apps using text input boxes assist the end user when typing by displaying input suggestions. 37 percent of the investigated apps use tabs for navigating between several screens and to structure content. Ten percent of these apps support switching the current tab by swiping to the left or to the right. 18.5 percent of the apps provide context menus when tapping and holding a screen item (mostly list items) for about one second.

About 20 percent of the apps support gestures, most of them multi-touch gestures for two fingers. The most used gesture is to spread two fingers for using the zoom function. The number of 20 percent can be explained by looking at the apps using Google maps, which supports zooming and rotating with gestures: About 10 percent of the apps use a Google maps control.

Table 6 summarizes the mentioned observations. Table cells containing a “1” mean that the app does contain/use/support the corresponding feature, control, or hardware.



	Controls used and gesture usage	YouT	Goog	Facel	Tiny I	Skype	What	Barcc	Shaz	wette	Goog	Adva	Kalor	Adob	Goog	ÖAMT	Tune	Amaz	NASA	Krone	Goog	eBay	Anti-'	LEO V	Goog	QR D	ASTR	IMDb	Sum	Percent
C1	Checkboxes			1								1											1		1				4	14,8%
C2	Dropdown menus	1	1			1									1														4	14,8%
C3	Buttons or button-like controls	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1		1	1	1	1	1	1	1	1		1	24	88,9%
C4	Vertical lists	1	1	1		1	1		1	1	1	1	1	1		1	1	1	1	1		1	1	1	1		1	1	22	81,5%
C5	Horizontal lists			1		1												1	1		1							1	6	22,2%
C6	Infinite lists	1	1	1							1						1	1	1	1		1		1				1	11	40,7%
C7	Text input boxes	1		1		1	1			1			1		1		1	1			1	1	1	1	1	1		1	16	59,3%
C8	Previous inputs/suggestions av	1	1	1		1							1									1						1	7	25,9%
C9	Map control including GPS		1													1									1				3	11,1%
C10	Tabs	1		1					1	1	1			1			1									1	1	1	10	37,0%
C11	Message boxes					1	1																						2	7,4%
C12	Long tapping for menu		1	1		1						1															1		5	18,5%
C13	Switching of tabs/screens by sv	1																											1	3,7%
C14	Multitouch gestures		1											1					1		1				1				5	18,5%
C15	Other gestures				1																								1	3,7%
	Hardware and software features used																													
		YouT	Goog	Facel	Tiny I	Skype	What	Barcc	Shaz	wette	Goog	Adva	Kalor	Adob	Goog	ÖAMT	Tune	Amaz	NASA	Krone	Goog	eBay	Anti-'	LEO V	Goog	QR D	ASTR	IMDb	Sum	Percent
H1	Preselection / sorting by Geolocatic		1							1	1						1				1							1	6	22,2%
H2	Bearing sensors (phone parallel to the floor/90° related to floor)																				1								1	3,7%
H3	Compass		1																		1								2	7,4%
H4	Approximity sensor																												0	0,0%
H5	Camera: QR tag reader																									1			1	3,7%
H6	Camera: Barcode reader							1					1					1				1				1			5	18,5%
H7	Camera: Object recognition																												0	0,0%
H8	Camera: Usage of photographe	1		1		1					1																		4	14,8%
H9	NFC Tag reader																												0	0,0%
H10	Internet	1	1	1		1	1		1	1	1		1		1	1	1	1	1	1		1	1	1	1		1	1	21	77,8%
H11	Network																										1		1	3,7%
H12	Bluetooth																										1		1	3,7%
H13	Sound recording/speech recognition								1						1										1				3	11,1%
H14	Starting of Dialer/SMS app			1					1							1													3	11,1%
H15	Starting of Navigation app																												0	0,0%
H16	Starting of e-mail app			1					1		1																		3	11,1%
H17	Starting of other app							1		1						1			1						1	1	1		7	25,9%
	App behaviour																													
		YouT	Goog	Facel	Tiny I	Skype	What	Barcc	Shaz	wette	Goog	Adva	Kalor	Adob	Goog	ÖAMT	Tune	Amaz	NASA	Krone	Goog	eBay	Anti-'	LEO V	Goog	QR D	ASTR	IMDb	Sum	Percent
B1	Account needed			1		1	1				1																		4	14,8%
B2	Account optional	1														1		1				1						1	5	18,5%
B3	App not responsive during long ops																												0	0,0%
B4	Progress for long ops shown	1							1																		1		3	11,1%
H10	Internet	1	1	1		1	1		1	1	1		1		1	1	1	1	1	1		1	1	1	1		1	1	21	77,8%
	If no network available:																													
B5	Messagebox				-	1		-				-		-	1	1		1	1		-		1	1		-		1	8	29,6%
B6	Toast			1	-			-	1	1		-		-					1	1	-	1			1	-			7	25,9%
B7	Embedded message	1			-			-				-		-			1				-					-			2	7,4%
B8	Main functionality of app not u	1	1		-	1		-	1		1	-	1	-	1		1	1	1	1	-	1		1		-		1	14	51,9%
B9	App crashes				-			-				-	1	-							-					-			1	3,7%
B10	App still limited usable		1	1	-			-		1		-		-		1					-				1	-			5	18,5%
B11	Try again button	1			-			-				-		-			1				-					-		1	3	11,1%
B12	App doesnt stop trying to connect		1		-			-			1	-		-							-								2	7,4%

Table 6: Summary of the usage of phone features of current apps.

We could see that data exchange with internet servers is widely used in current smartphone applications. Other hardware components used, in descending commonness, are: Geolocation, Camera, Audio, Bluetooth, and NFC. Modern mobile applications could make even more use of the new capabilities of modern smartphones in order to simplify operation. Some ideas for using smartphone hardware and sensors include:

- Automatic settings or adaptions depending on the current location.
- Integration of QT tag scanning or NFC Tag reading capabilities to speed up text input
- Use of multi-touch-gestures for speeding up interaction and for making it more intuitive

The disadvantage of heavy use of the smartphone's hardware is that the currently very limited battery life is reduced even more. Therefore, software designers have to carefully trade off the usage of such technologies for battery life.

Previously, the screens of current smartphones were described as relatively large. In relation to the device this statement is true, but for most applications the screen size could be even larger. This leads to new requirements to user interfaces running on such devices. While on desktop computers with large 19" screens much information can be displayed at the same time, this is not possible on smartphone screens. This fact makes it necessary to present only the most important subset of the currently needed information. Such context dependent presentation is crucial for not overwhelming the end-user with too much information.

In order to create clearly arranged applications on small screens often tabs are used for structuring screen contents. Websites which were ported to apps for smartphones may display more or less the same information as the app does. However, the app may display the different regions of one web page in different tabs. The YouTube app for Android, for example, displays the comments, the related videos, and the general information in three separated tabs, while the YouTube website displays this information in different regions of the website. For switching between tabs more and more current apps allow switching between the single tabs by supporting the swipe gesture.

For larger devices, such as tablets, concepts such as Android's Fragments[22] are used, where the screen is separated into two or more independent, but connected regions.

2.1.7. Showcase UI Creation

Creating and designing user interfaces is an essential part when creating a software product. Therefore, we exemplarily demonstrate the creation of user interfaces for the Android operating system and the Windows Phone operating system in this subsection. At first, however, some basic terms are defined.

[22] Android Fragments: *http://developer.android.com/guide/topics/fundamentals/fragments.html*, last visited 07/02/2012

Control or View. A control or a view is a single UI element such as a text box or a button. Usually several different controls are displayed at the same time on the screen inside a Window or Layout. More examples for controls are: ListView (a container for several views which are displayed as scrollable list), TextView (a text label), Spinner (dropdown box), RadioButton (option for multiple choice selections). The mentioned names are the names of the views for the Android operating system. There are similar controls for Windows Phone, however, named differently.

Activity. According to the Android documentation[23] **"***an activity is a single, focused thing that the user can do*"**.** Usually an activity creates and represents a "window" the user can interact with.

On Android as well as on Windows it is possible to define the user interface in an XML-based markup language or by creating the corresponding objects from Java (Android) or C#/Visual Basic .NET (Windows) program code. It is also possible to dynamically load XML UI definitions at runtime.
The following XML code defines the left user interface shown in Figure 9 on Android.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello world!"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="OK" />
</LinearLayout>
```

[23] Android Documentation – Activity: *http://developer.android.com/reference/android/app/Activity.html*, last visited 07/02/2012

Windows Phone and WPF for .NET Windows (desktop) applications use the same XML-based language for defining user interfaces: XAML (Microsoft, 2009). The program code for Windows Phone apps is usually written in a .NET language such as C# or VisualBasic.NET. The following XML code defines the right user interface shown in Figure 6 on Windows Phone.

```
<phone:PhoneApplicationPage
    x:Class="WindowsPhoneApplication1.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    [...]
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="PageTitle" Text="Hello World" Margin="9,-7,0,0" />
            <Button x:Name="Button" Content="OK" />
        </StackPanel>
    </Grid>
</phone:PhoneApplicationPage>
```

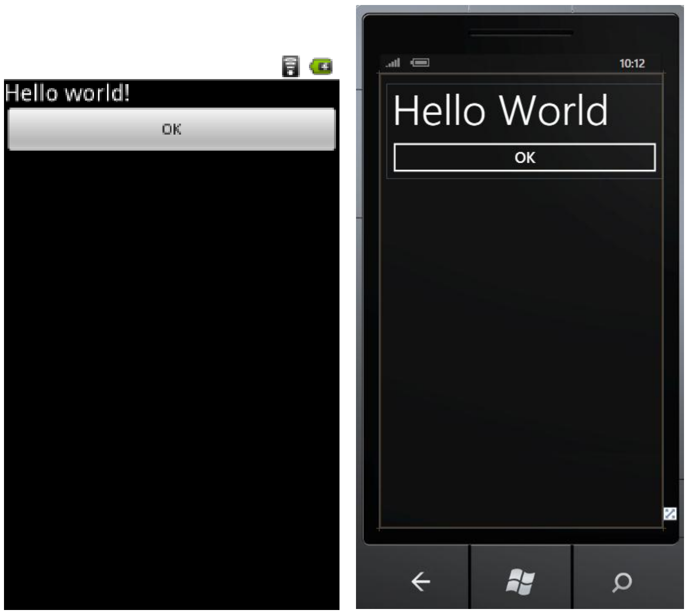


Figure 9: Basic layouts in Android (left) and Windows Phone (right).

2.2. Enterprise Applications

As the name suggests, enterprise software typically is software which is used by enterprises. However, there are more detailed definitions about the term enterprise software. Stephen Adkins defined Enterprise Systems[24] with systems which have the following attributes (quoted from officevision.com):

- *Availability - the assurance that a service/resource is always accessible*
- *Scalability - the ability to support the required quality of service as the load increases*
- *Reliability - the assurance of the integrity and consistency of the application and all of its transactions. The ability to provide a required reliability service level depends on the close coordination of the hardware, networking, operating system, storage subsystem, application framework, and application software.*
- *Security - the ability to allow access to application functions and data to some users and deny them to others*
- *Interoperability - the ability of the system to share data with external systems and interface to external systems.*
- *Leveragability - the ability that stored data, programmed logic, and other system resources available anywhere in the enterprise should be accessible from everywhere in the enterprise*
- *Maintainability - the ability to correct flaws in the existing functionality without impacting other components/systems*
- *Extensibility - the ability to add/modify functionality without impacting existing functionality*
- *Manageability - the ability to manage the system in order to ensure the continued health of a system with respect to scalability, reliability, availability, performance, and security.*

[24] *http://www.officevision.com/pub/p5ee/definitions.html*, originally posted at *http://www.nntp.perl.org/group/perl.p5ee/2001/10/msg6.html*

- *Portability - the ability of the software to run on a variety of hardware and operating system configurations*
- *Accessibility - the ability to access system functions through different user agents and in different human languages*

Adkins further states that, when systems have these attributes, they are such that large enterprises can invest substantial corporate resources in them.

Another definition for enterprise applications is provided by Gartner. According to Gartner's glossary enterprise applications are "*Software products designed to integrate computer systems that run all phases of an enterprise's operations to facilitate cooperation and coordination of work across the enterprise. The intent is to integrate core business processes (e.g., sales, accounting, finance, human resources, inventory and manufacturing). The ideal enterprise system could control all major business processes in real time via a single software architecture on a client/server platform. Enterprise software is expanding its scope to link the enterprise with suppliers, business partners and customers.*"

In computer science the term enterprise framework relates to the development environment and the base technology for creating enterprise applications. The pcmag.com encyclopedia defines an enterprise framework as "*a complete environment for developing and implementing a comprehensive information system. Enterprise frameworks provide pre-built applications, development tools for customizing and integrating those applications to existing ones as well as developing new applications. They may also provide a workflow component.*" Also processes, methodologies, and architectural practices are often called frameworks (Sessions, 2007). During the work with Boom Software AG (see Sections 3 and 4) we worked with Boom's BORA framework.

When porting desktop (enterprise) applications to mobile devices the special constraints but also the special abilities of mobile devices must be considered. In different locations in this work the specialties of mobile devices are mentioned. Often it is feasible to implement only parts of the desktop software for mobile devices due to the different domains compared to desktop computers. Lumsden (2008) states that there is a risk of losing "positive" experience when porting desktop software to mobile devices.

Therefore, in the professional domain, there is often room for improvement concerning the user experience of mobile applications.

2.3. User Experience

In this section we will discuss the term user experience (UX) and related terms such as usability and accessibility. According to Forlizzi and Battarbee (2004) the term user experience is associated with a wide range of meanings. It is not always easy to distinct between the terms user experience and usability. Sometimes, these two terms are even used as synonyms. However, as we will see later, user experience covers a wider field than usability. According to Hassenzahl and Tractinsky (2006) UX is about technology that fulfills more than just instrumental needs, however, there is a lack of empirical research on this topic. According to Bevan (2009) there is a definition of the term user experience in the ISO FDIS 9241-210 standard: User Experience is "*a person's perceptions and responses that result from the use and/or anticipated use of a product, system or service."* Further it is stated that "*user experience includes all the users' emotions, beliefs, preferences, perceptions, physical and psychological responses, behaviors and accomplishments that occur before, during and after use*".

Lumsden (2008) states that user experience is a term used to describe cognitive, affective, and social responses that are induced by the use of a product or service. Further, Lumsden (2008) points out that due to the dynamic use context of mobile devices and due constraints of mobile devices user experience evaluations are of special importance for mobile applications.

2.3.1. User Experience vs. Usability vs. Accessibility

According to Bevan (2009) the term **usability** is defined in ISO FDIS 9241-210 as follows: Usability is "*the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*", where the term context of use refers to environmental and personal characteristics. According to Nielsen (2003) usability is defined by the following five quality components:

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

Further, Nielsen provides the following definitions:

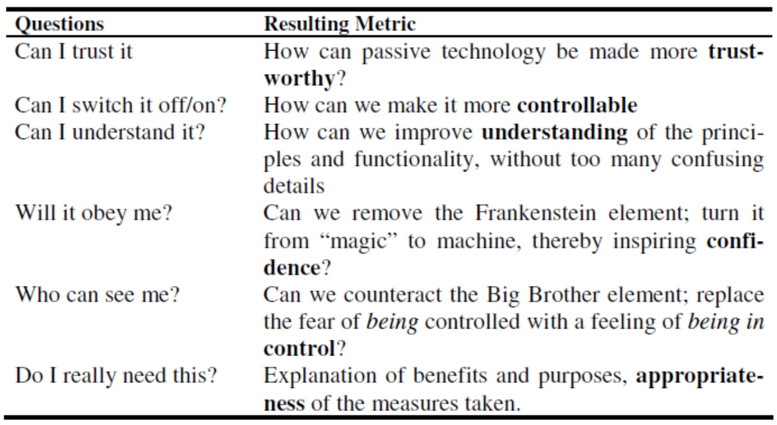
- *Utility = whether it provides the features you need.*
- *Usability = how easy & pleasant these features are to use.*
- *Useful = usability + utility.*

There is a large number of international standards concerning with usability of software (Bevan, 2001). According to Bevan these standards can be classified into the following categories:

- *The use of the product (effectiveness, efficiency and satisfaction in a particular context of use).*
- *The user interface and interaction.*
- *The process used to develop the product.*
- *The capability of an organization to apply user-centered design.*

A survey showed that **user experience** is interpreted in a similar way as usability, but with the addition of anticipation and hedonic responses (Bevan, 2009). Based on the ISO definitions for usability and UX cited above, Bevan (2009) reasons that if user experience includes all behavior, it presumably includes the user's effectiveness and efficiency. Bevan further concludes that there are there are two distinct objectives of user experience: Optimizing human performance and optimizing user satisfaction with achieving both pragmatic and hedonic goals. Laugwitz et al. (2008) call user experience criteria soft criteria while usability criteria are called hard criteria. They further state that both are of similar relevance for the end user.

For elderly or inexperienced people it is important to consider several questions in addition to common usability questions. Not only elderly users often have anxieties related to the usage of computers. However, the elderly are often of the opinion that the benefits associated with computer use fail to outweigh the necessary effort (Holzinger et al., 2008b). Holzinger et al. were able to quantify the questions of elderly users in the form of usability metrics (Table 7).



Questions	Resulting Metric
Can I trust it	How can passive technology be made more **trust-worthy**?
Can I switch it off/on?	How can we make it more **controllable**
Can I understand it?	How can we improve **understanding** of the principles and functionality, without too many confusing details
Will it obey me?	Can we remove the Frankenstein element; turn it from "magic" to machine, thereby inspiring **confidence**?
Who can see me?	Can we counteract the Big Brother element; replace the fear of *being* controlled with a feeling of *being in* **control**?
Do I really need this?	Explanation of benefits and purposes, **appropriateness** of the measures taken.

Table 7: Analogies between user anxiety and metrics (Holzinger et al., 2008b)

Another term related to user experience and usability is **accessibility**. In common language, the term accessibility means that media, buildings or any other objects are reachable - or accessible – for every human being without any restrictions, even if the individual is handicapped. In the area of software engineering accessibility refers to the operation of computers and computer programs. Accessible software uses, amongst others, sufficiently high-contrast colors, easily accessible user interface elements (for example buttons which are large enough and therefore easy to touch or to click), metadata which can be accessed by text-to-speech technologies for describing screen contents for visually impaired people, and other technologies for simplifying interaction with the application.

According to Billi et al. (2010) accessibility id defined differently in different ISO standards. On the one hand accessibility is brought in relation to people with performance limitations (ISO/IEC Guide 71), on the other hand - according to ISO 9241 - "accessibility addresses the full range of user capabilities and is not limited to users who are formally recognized as having a disability." Microsoft[25] explains accessibility with "*ensuring that programs and functionality are easily available to the widest range of users, including those who have disabilities and impairments*."

2.3.2. Usability Engineering

The goal of usability engineering (UE) is to create user interfaces with good usability. Therefore, the task of usability engineers is to create and to improve but also to test user interfaces. Usability engineers must not only have knowledge about computer science issues, but also about psychology, cognitive science, and fields related to human perceiving and processing of visual information.

Usability Engineering often follows a spiral four phase procedure model (analysis, draft, development, and test) and a three step production model: paper mock-up, prototype, and final product (Holzinger et al., 2005a). According to Holzinger and Brown (2008a) good usability engineering combines complex back-end functionalities with a well operated, attractive, effective and efficient user interface, with full regard to efficiency. The STAR model (Stage, Types, Aims, Resources) can help to bring the theory of user centered software development into practice. The model suggests considering four relevant points when developing a prototyping/test strategy: the design stage including the initial concept, different types of prototypes, the aims of the evaluation, and the available resources. For prototyping different models are possible, depending on the aims: vertical prototypes, where certain features are implemented in-depth, horizontal prototypes, where the top-level UI is working, but no in-depth functionality is available, and scenario prototype, where only functionality of parts of the UI for certain scenarios is available (Holzinger and Brown, 2008a).

[25] Microsoft User Experience Guidelines – Accessibility: *http://msdn.microsoft.com/en-us/library/windows/desktop/bb545462.aspx*, last visited 07/02/2012

An important point in user interface design is that user interfaces should be designed in a way that the users do not have to focus on handling the software or the device. Instead user interfaces should be designed user-centered which means that the focus lies on reaching the users goal in a most simple intuitive and efficient way (El-Bakry et al., 2010).

One problem in testing and evaluating UIs is that the UI to test has to be built before. If it turns out that changes have to be made, adapting the UI might be expensive in terms of both time and money. To circumvent this problem rapid prototyping can be applied or paper mockups can be created prior to implementing the user interface. UI prototypes usually only provide mock functionality or only parts of the logic behind the user interface are implemented. Studies show that it is essential to integrate usability engineering at a very early stage of development to ensure cost-effectiveness and to even have the possibility to integrate most findings into the final product (Holzinger et al., 2011a).

A new approach in usability engineering combines the aspects of Extreme Programming (XP) and Usability Engineering (UE) to a concept called Extreme Usability (XU). (Holzinger et al., 2005a). As in XP, concepts of UE are brought to an extreme in XU. The concept of XP includes the support of communication in small teams, focus on simplicity, frequent releases and small development iterations, and constant customer feedback. This results in a high ability to react to changing demands. In XU these concepts are combined with the best practices of UE. This includes making the customer continually aware of usability aspects and performing tests continuously (Holzinger et al., 2005a).

In usability engineering different user groups are to be distinguished. Everyone who uses software is a user, regardless the reason for using the software. Test users are users who use software in order to test software, either on one's own initiative or based on the instructions of someone else, such as a usability evaluation expert. End-users are the target users of a software product, the users who are using the product in order to get benefit from it. It is also possible that end users are used as test users at the same time, for example during field evaluations.

Figure 10 illustrates the correlation between single user types.

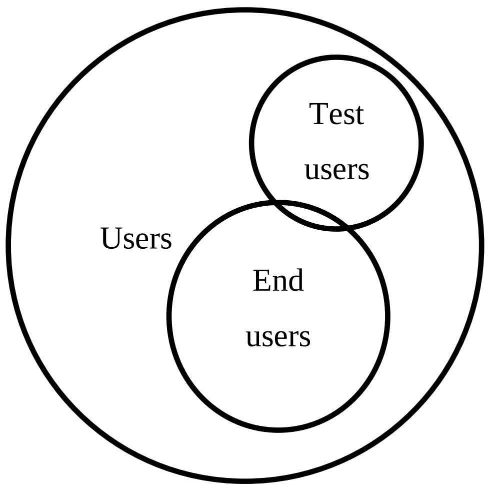


Figure 10: Users, end-users and test users.

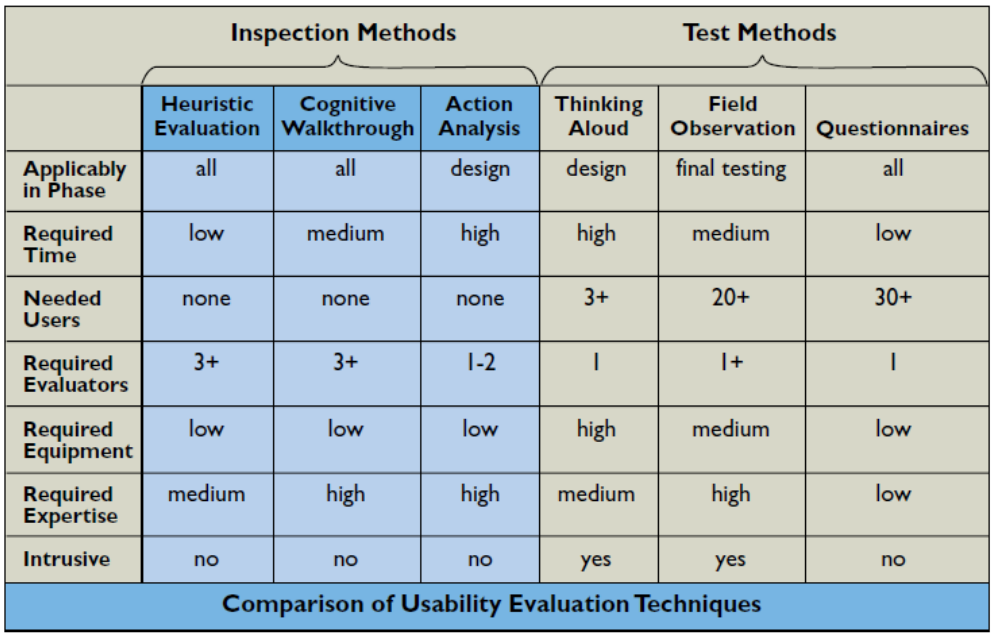
Although UE has such great practical relevance usability engineering seems not to be of great relevance in current curricula at schools or universities. UE is integrated very little and far too late in software engineering education (Holzinger et al., 2005a).

2.3.3. Evaluation Methods

For evaluating usability many different methods exist. There are methods which are performed by usability experts only, but also methods in which non-expert users work with the software to be tested, often under surveillance of one or more usability experts. For measuring the user experience or the usability of a system questionnaires are widely used, but also more sophisticated methods such as eye tracking are common techniques in certain areas. **Eye Tracking**, for example, is a helpful analysis tool for usability issues (Crosby et al., 2001). Eye tracking can reveal other information than questionnaires or methods where users are asked for providing information, such as thinking aloud tests. According to Cooke (2006) eye tracking can tell where but not why users look for certain information. However, the information about the “where” can be analyzed more deeply when using eye tracking than with user interviews or questionnaires.

In general, usability evaluation methods can be classified into inspection methods and test methods. Inspection methods are mainly performed by usability professionals while test methods are performed by test users under supervision of usability professionals.

The different evaluation methods are best applicable in different states of the development of software. Also the required time and the amount of needed test users differ from method to method. This makes not every method applicable in any situation. The usability evaluator has do decide which method suits best in the current situation or in the current development stage of the software. Table 8 provides an overview of six common usability evaluation methods and lists important properties.



	Inspection Methods			Test Methods		
	Heuristic Evaluation	Cognitive Walkthrough	Action Analysis	Thinking Aloud	Field Observation	Questionnaires
Applicably in Phase	all	all	design	design	final testing	all
Required Time	low	medium	high	high	medium	low
Needed Users	none	none	none	3+	20+	30+
Required Evaluators	3+	3+	1-2	1	1+	1
Required Equipment	low	low	low	high	medium	low
Required Expertise	medium	high	high	medium	high	low
Intrusive	no	no	no	yes	yes	no

Comparison of Usability Evaluation Techniques

Table 8: Comparison of usability evaluation techniques (Holzinger, 2005)

Test methods involve test users who use software to be tested. The users' behavior is recorded and evaluated by usability experts. For the **field observation** method the users are observed at their workplaces during normal operation of the software. The goal is to keep the influence (intrusion) by the observing person(s) to a minimum. **Questionnaires** are often combined with other methods and are usually simple and quick to perform. The thinking aloud test method is described in more detail later in this work.

Inspection methods are methods where usability experts inspect the software to be evaluated. During a **cognitive walkthrough** the end user's behavior is simulated for given tasks. **Action analysis** focuses on the actions users performed while using the software under test. Keystrokes and mouse movements are recorded in order to get detailed information about the user's actions. However, no information about the reasons for the user’s actions is available. The heuristic evaluation is described later in this work. The information about the described methods is taken from Holzinger (2005) and also available in Holzinger (2010).

The mentioned test methods are “local” methods, i.e. the usability professional and the test user are at the same spatial and temporal location. **Remote usability testing** allows testing software for usability issues even when the tester or the test monitor (Andreasen et al., 2007) is located in different locations. There are two types of remote usability test methods to distinguish: synchronous and asynchronous methods. Synchronous testing is conducted in real time but the test monitor is separated spatially from the test subjects, while for asynchronous methods the test monitor and the test subjects are separated both spatially and temporally (Andreasen et al., 2007). The following list mentions several remote usability methods (Bruun et al., 2009, Andreasen et al., 2007):

- Audio/video/text conferences or reporting
- Screen capturing/screenshots
- Questionnaires
- Interviews
- Auto logging
- User-reported critical incident reporting
- Unstructured problem reporting
- Forum/Diary

All mentioned methods have advantages and disadvantages. User-reported critical incident reporting method, for example, might be problematic because users might not be able to report their critical incidents for different reasons. Castillo et al. (1998) have investigated, whether users are able to report their own critical incidents. It has been shown that users actually are able to do so and that the method is working well. Automatic logging of mouse tracks, click locations and key presses lacks the personal feedback of the user if not combined with other methods such as interviews.

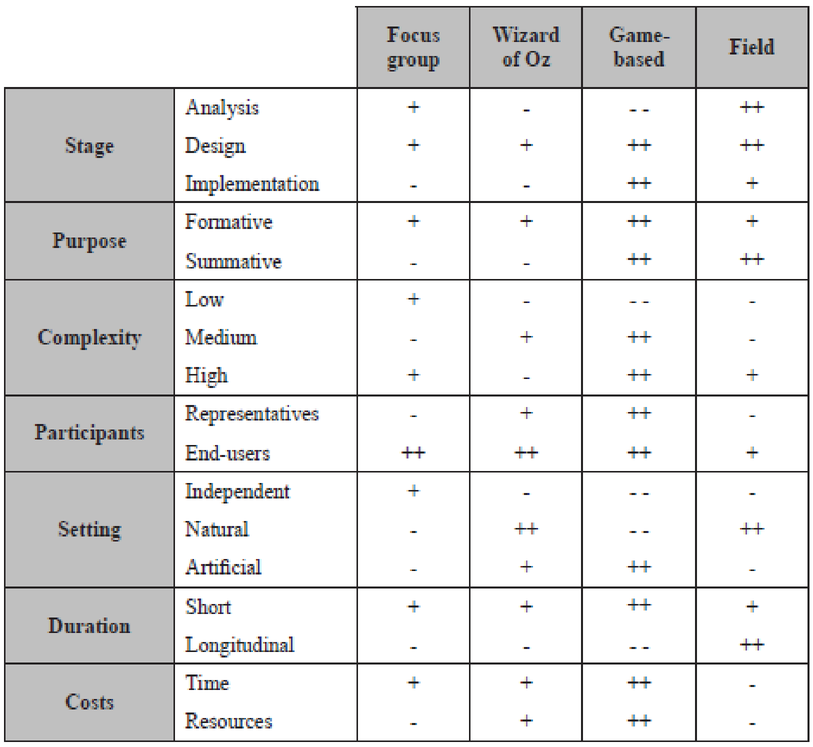
Hilbert and Redmiles (1999) describe how user interface events can be used to gain usability-related information. User interface events, such as clicks or menu selections

are recorded for later analysis. Methods such as sequence detection, where sequences of user actions are recorded and compared with a predefined action sequence can help to detect usability issues. But also generating and analyzing visualizations, for example of mouse click locations, are useful in certain situations.
Evaluation methods such as task analysis require not only expertise in software engineering. For a holistic understanding also competence in at least philosophy, psychology, sociology, and ergonomics is necessary (Diaper and Stanton, 2004).

Evaluation of Mobile Applications. Typical usability evaluation methods and heuristics (see heuristic evaluation, Section 2.3.4) which are applicable to desktop software might not effectively be applied to a mobile phone (Heo et al., 2009). Therefore Heo et al. (2009) developed a checklist-based framework for usability evaluation tailored to feature phones. The evaluation is separated into four parts: Task-based evaluation, Logical User Interface (LUI) based evaluation, Physical User Interface (PUI) based evaluation, and Graphical User Interface (GUI) based evaluation. For each of the parts a checklist was created. The Task-based Checklist focuses, amongst others, on the areas efficiency of procedure and stability of use, the LUI based checklist focuses for example on information architecture and menu label wording, the PUI based checklist deals for example with ergonomic considerations of buttons etc., and the GUI based checklist contains items for font type and size, display style and color, and more.
Lumsden (2008) presents and compares techniques for evaluating context-aware mobile user interfaces: Focus group, Wizard of Oz, Game-based, and Field evaluation: **Focus group** sessions are meetings of a selected group of people discussing a specific topic. By evaluating the discussion insight into user needs and opinions can be gained. The focus group method can be used for evaluating concepts or high-level user requirements.
When applying the **Wizard of Oz** evaluation method test users interact with a partly implemented user interface. The responses of the unimplemented parts are generated at test time dynamically by the usability researcher.
Game-based evaluation means that a realistic task environment is artificially created for the evaluation session(s). In contrast to Game-based evaluation field evaluation is

conducted in natural environments. However, field evaluation needs a stable and reliable system (Lumsden, 2008).

Table 9 compares the mentioned methods. One or more plus signs (+) indicate the appropriateness of the corresponding method, while one ore minus signs (-) means that the corresponding method is less appropriate.



		Focus group	Wizard of Oz	Game-based	Field
Stage	Analysis	+	-	- -	++
	Design	+	+	++	++
	Implementation	-	-	++	+
Purpose	Formative	+	+	++	+
	Summative	-	-	++	++
Complexity	Low	+	-	- -	-
	Medium	-	+	++	-
	High	+	-	++	+
Participants	Representatives	-	+	++	-
	End-users	++	++	++	+
Setting	Independent	+	-	- -	-
	Natural	-	++	- -	++
	Artificial	-	+	++	-
Duration	Short	+	+	++	+
	Longitudinal	-	-	- -	++
Costs	Time	+	+	++	-
	Resources	-	+	++	-

Table 9: Guidelines for the appropriateness of different evaluation techniques for context-aware mobile user interfaces (Lumsden, 2008)

Billi et al. (2010) present a unified two-step methodology for evaluating mobile applications taking into account both accessibility and usability – accessibility evaluation in the first step, followed by a usability evaluation in a second step.

Objectivity and usability evaluations. According to Bevan and Curson (1997) there are no objective criteria for usability. However, in order to make the usability of different systems comparable, usability scales were developed. Finstad (2010) mentions some scales for measuring usability via questionnaires:

- **Software Usability Measurement Inventory** (SUMI, 50 items) (Kirakowski and Corbett, 1993). SUMI is a questionnaire for measuring the quality of use of software. Quality of use is defined as the extent to which a product satisfies stated and implied needs when used under stated conditions (Bevan, 1995). According to the SUMI website the questionnaire is mentioned in the ISO 9241 standard as method of testing user satisfaction. The current version (4.0) can be viewed at *http://sumi.ucc.ie/en/*. Questions include "*The instructions and prompts are helpful*" or "*The software hasn't always done what I was expecting*". The possible choices are *Agree*, *Disagree*, or *Undecided*.
- **Questionnaire for User Interface Satisfaction** (QUIS) (Chin et al., 1988). The questions of QUIS, such as "*Use of terms throughout system: inconsistent/ consistent*"or "*Error messages: unhelpful/helpful*" must be answered on a Likert scale from 0 to 9. A online version can be found at *http://hcibib.org/perlman/question.cgi?form=QUIS*.
- **Computer System Usability Questionnaire** can be used to measure the satisfaction with the usability of computer systems (Lewis, 1995). Questions include "*I feel comfortable using this system*" or "*The organization of information on the system screens is clear*". An online version is provided at *http://hcibib.org/perlman/question.cgi*.
- **System usability scale** (SUS, 10 items) (Brooke, 1996).

The **system usability scale** probably is the most well-known usability scale. It subjectively measures the usability of a system based on a Likert scale. Finstad (2010) mentions that SUS was standardized by Intel because of its good performance characteristics, in addition to being free and relatively compact. The SUS includes 10 questions (Brooke, 1996):

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use

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

All questions (items) are answered based on a Likert scale with five grades for each item. The final usability score is calculated based on the schema describes by Brooke (1996). The result of the calculation is a value between 0 and 100, where a larger value stands for better usability. But what exactly does a certain value mean and which values are good values? In order to answer this question Bangor et al. (2008) evaluated the SUS (Figure 11). Based on an empirical investigation they found that a score of 50 does not mean that the tested product is half as good as a product scored with 100.

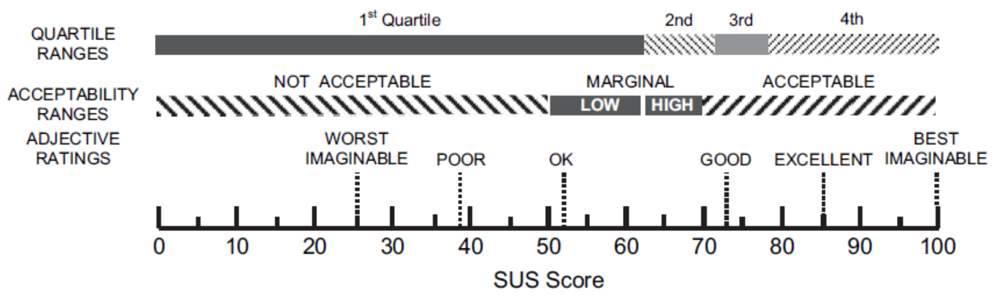


Figure 11: Meaning of SUS values (Bangor et al., 2008).

Finstad (2010) has proposed a new metric for user experience (Usability Metric for User Experience - UMUX). Although the metric’s name refers to user experience, UMUX is a usability evaluation questionnaire similar to SUS. However, UMUX only consists of four questions by having a similar reliability, validity, and sensitivity as SUS. It was designed to be an alternative to SUS and correlates with the SUS at about 80 percent (Finstad, 2010). The four questions are the following:

1. [This system’s] capabilities meet my requirements.
2. Using [this system] is a frustrating experience.
3. [This system] is easy to use.
4. I have to spend too much time correcting things with [this system].

In contrast to SUS the Likert scale has seven graduations (SUS: five).

Ghahramani (1998) claims the invention of a method for objectively measuring usability. The method measures the performance of users, measured by the time needed for performing a task, on the one hand and the user satisfaction on the other hand.
Evaluating complex adaptive systems in the use context increases the appropriateness of the final design. Also actual end-users must be involved as participants because of their intimate knowledge of mobile use context and domain-specific tasks (Lumsden, 2008).

2.3.4. Heuristic Evaluation

Heuristic evaluation is a usability inspection method which is performed by usability experts. Supported by set of usability rules the user interface under test is inspected systematically. These rules are also called heuristics. According to Oxford Dictionaries heuristic means “*enabling a person to discover or learn something for themselves*”. Another meaning from computer science is “*proceeding to a solution by trial and error or by rules that are only loosely defined*”. During heuristic evaluations rules are used to support the decision process and for supporting the usability expert’s inspection procedure.
Heuristic evaluations should usually be done by several usability experts in parallel. The results of each expert should then be consolidated. The reason is that more usability experts usually find more usability issues. By four experiments conducted by Nielsen and Molich (1990) it was empirically shown that a heuristic evaluation should be performed by at least three to five usability experts (Figure 12) in order to find many usability problems. However, a more common rule, similar to the rule for the required number of thinking aloud test users (Holzinger, 2006), might be appropriate: Depending on the complexity to the inspected software, the number of usability experts should be increased as long as the number of new issues found is larger than a certain threshold.

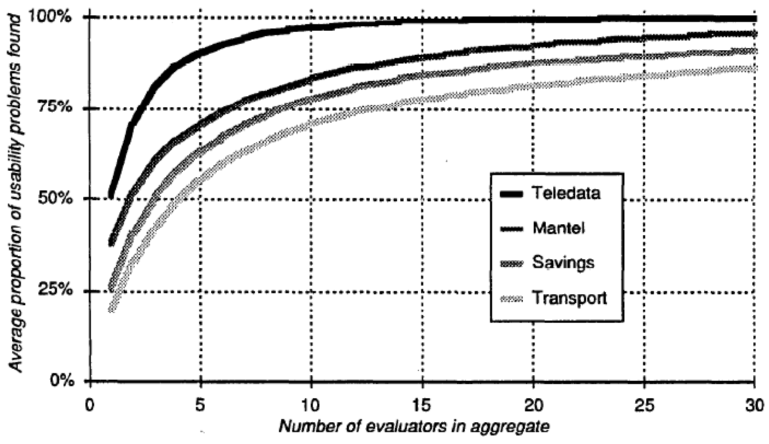


Figure 12: Proportion of usability problems found by aggregates of size 1 to 30 (Nielsen and Molich, 1990)

The advantages of heuristic evaluation include a low need both time and equipment. At the same time, as the evaluation is done by usability experts, different usability issues might be revealed than by performing usability test methods where test users are involved.

Usability heuristics were developed by different authors. Nielsen (1994c) published ten usability heuristics, Shneiderman and Plaisant (2004) published “Eight Golden Rules of Interface Design”, and Raymond and Landley (2004) also published several rules of usability. There are more sources available, but already when comparing the heuristics by the mentioned authors certain overlaps can be discovered.

Nielsen’s ten usability heuristics:

- N1 Visibility of system status
- N2 Match between system and the real world
- N3 User control and freedom
- N4 Consistency and standards
- N5 Error prevention
- N6 Recognition rather than recall
- N7 Flexibility and efficiency of use
- N8 Aesthetic and minimalist design
- N9 Help users recognize, diagnose, and recover from errors
- N10 Help and documentation

Shneiderman’s Eight Golden Rules of Interface Design:

- S1 Strive for consistency: The same design, terminology should be used throughout the system. Also operation should be consistent.
- S2 Enable frequent users to use shortcuts: Experienced users should have the possibility to use shortcuts and advanced features such as macros.
- S3 Offer informative feedback: There should always be feedback for actions. For infrequent and/or major actions the feedback should be more intense.
- S4 Design dialog to yield closure: Dialogs or dialog sequences should make clear when the task is completed by providing feedback.
- S5 Offer simple error handling: If an error cannot be prevented beforehand (which should always be prioritized) a simple way for handling the error should be provided.
- S6 Permit easy reversal of actions: A simple undo function should be provided.
- S7 Support internal locus of control: Users should feel that they started an action (not the system) and have full control over the system.
- S8 Reduce short-term memory load: As little information as possible should be necessary to be remembered by the user (for example between single screens).

Raymond’s Rules of Usability:

- R1 Rule of Bliss: Allow your users the luxury of ignorance.
- R2 Rule of Distractions: Allow your users the luxury of inattention.
- R3 Rule of Flow: Allow your users the luxury of attention.
- R4 Rule of Documentation: Documentation is an admission of failure.
- R5 Rule of Least Surprise: In interface design, always do the least surprising thing.
- R6 Rule of Transparency: Every bit of program state that the user has to reason about should be manifest in the interface.
- R7 Rule of Modelessness: The interface's response to user actions should be consistent and never depend on hidden state.
- R8 Rule of Seven: Users can hold at most 7±2 things at once in working storage. Based on new studies Farrington (2011) lowers this number to three to four.

- R9 Rule of Reversibility: Every operation without an undo is a horror story waiting to happen.
- R10 Rule of Confirmation: Every confirmation prompt should be a surprise.
- R11 Rule of Failure: All failures should be lessons in how not to fail.
- R12 Rule of Silence: When a program has nothing surprising to say, it should say nothing.
- R13 Rule of Automation: Never ask the user for any information that you can automatically detect, copy, or deduce.
- R14 Rule of Defaults: Choose safe defaults, apply them unobtrusively, and let them be overridden if necessary.
- R15 Rule of Respect: Never mistake keeping things simple for dumbing them down, or vice-versa.
- R16 Rule of Predictability: Predictability is more important than prettiness.
- R17 Rule of Reality: The interface isn't finished until the end-user testing is done.

Pierotti (2000) supplemented Nielsen's heuristics with more heuristics and also presents a detailed checklist implementing Nielsen's and the other heuristics. The following list only lists the heuristics relevant for usability evaluation which are not already part of Nielsen's rule set.

Pierotti's heuristics:

- P1 Skills: The system should support, extend, supplement, or enhance the user's skills, background knowledge, and expertise - not replace them.
- P2 Pleasurable and Respectful Interaction with the User: The user's interactions with the system should enhance the quality of her or his work-life. The user should be treated with respect. The design should be aesthetically pleasing- with artistic as well as functional value.

As mentioned above, overlaps can be observed when comparing the single heuristics listed before. Table 10 presents the aim for consolidating heuristics covering similar areas to categories C1 to C11. Before, the single heuristics were numbered with unique IDs, such as N1 or R9. Also the identified categories were assigned a unique ID ranging from C1 to C11.

Category	**Nielsen**	**Shneiderman**	**Raymond & Landley**	**Pierotti**
C1 Feedback	N1 VisSysStat	S3 InfFeedback, S4 DlgYieldClsr	R1 Transparency, R2 Silence	
C2 Input			R5 Defaults	
C3 Match	N2 MatchSysWorld			
C4 User control, Flexibility	N3 UC&Freedom, N7 Flexibility	S2 Shortcuts, S7 SuppIntLocus	R3 Modelessness, R9 Automation, R16 Predictability	P1 Skills
C5 Undo		S6 Reversal	R4 Reversibility	
C6 Consistency	N4 Consistency	S1 Consistency	R6 Least Surprise, R3 Modelessness	
C7 Errors	N5 ErrorPrevention, N9 RecovrFromErrors	S5 SimpleErrHandl	R14 Failure, R10 Confirmation	
C8 Short-term mem.	N6 RecognRTRecall	S8 RedShrtTrmMemLd	R7 Distractions, R8 Seven	
C9 Design	N8 AestheticMinDesgn		R11 Bliss, R12 Flow, R13 Respect	P2 RespInteract
C10 Documentation	N10 Help&Doc		R15 Documentation	
C11 UI Testing			R17 Reality	

Table 10: Assignment of different heuristics to categories

According to the Measuring Usability blog[26] Weinschenk and Barker (2000) created a list of 20 heuristics based on different sources using the card sort method which was also used by Lowry (2008):

- *User Control: The interface will allow the user to perceive that they are in control and will allow appropriate control.*
- *Human Limitations: The interface will not overload the user's cognitive, visual, auditory, tactile, or motor limits.*
- *Modal Integrity: The interface will fit individual tasks within whatever modality is being used: auditory, visual, or motor/kinesthetic.*

[26] *http://www.measuringusability.com/blog/he-cw.php*, last visited 07/02/2012.

- *Accommodation: The interface will fit the way each user group works and thinks.*
- *Linguistic Clarity: The interface will communicate as efficiently as possible.*
- *Aesthetic Integrity: The interface will have an attractive and appropriate design.*
- *Simplicity: The interface will present elements simply.*
- *Predictability: The interface will behave in a manner such that users can accurately predict what will happen next.*
- *Interpretation: The interface will make reasonable guesses about what the user is trying to do.*
- *Accuracy: The interface will be free from errors*
- *Technical Clarity: The interface will have the highest possible fidelity.*
- *Flexibility: The interface will allow the user to adjust the design for custom use.*
- *Fulfillment: The interface will provide a satisfying user experience.*
- *Cultural Propriety: The interface will match the user's social customs and expectations.*
- *Suitable Tempo: The interface will operate at a tempo suitable to the user.*
- *Consistency: The interface will be consistent.*
- *User Support: The interface will provide additional assistance as needed or requested.*
- *Precision: The interface will allow the users to perform a task exactly.*
- *Forgiveness: The interface will make actions recoverable.*
- *Responsiveness: The interface will inform users about the results of their actions and the interface's status.*

It cannot be avoided that heuristic evaluation is to a certain degree subjective due to different interpretations of the rules by the different usability evaluators. Additionally, as Figure 12 suggests, the results of heuristic evaluations are usually not complete. There might be more issues which could not be found during the heuristic evaluation.

The mentioned heuristics originally were designed for desktop systems. But there also exist usability heuristics for other types of software than desktop software. There are heuristics for games (Korhonen and Koivisto, 2006) and for mobile devices mobile devices (Lowry, 2008). However, heuristics applicable for modern devices such as smartphones with multi-gesture capabilities are still rare. Especially detailed checklists, such as the checklist by Pierotti (2000), are missing. However, heuristic evaluation for mobile devices might be problematic as contextual influences are only poorly represented (Po et al., 2004). Varsaluoma (2009) states that heuristic evaluation and other existing usability evaluation methods must be redesigned in order to create more awareness of the mobile context. Po et al. (2004) experimented with a method called **Contextual Walkthrough** which includes the situational context into a Heuristic Walkthrough. A Heuristic Walkthrough is the combination of heuristic evaluation with scenarios of use.

Ji et al. (2006) developed a usability checklist for evaluating user interfaces of mobile phones, however, the checklist targets feature phones and not smartphones. Also Bertini et al. (2006) worked on appropriating heuristics for mobile devices.

Lobo et al. (2011) collected rules for designing websites for smartphones: Mobile websites should comply with several simple common rules: Keep it Simple, Simplify User Input, Scroll Vertically Only, Multiple Versions of the Website. The latter rule means that on the server side it should be detected whether a mobile device or a desktop client accesses the website. Depending on the browser type a mobile or a desktop version of the same website should be sent to the client (Holzinger and Errath, 2007).

Lowry (2008) described a methodology for creating heuristics for mobile devices, however, currently there is only a little amount of work dealing with heuristics for modern mobile devices such as current smartphones and tablets with large multi-touch screens. (Varsaluoma, 2009) stated that the context of use is also important when heuristically evaluating mobile applications. So, in order to get reliable usability measures it is important to understand and to consider the context of use (Bevan and Macleod, 1994).

During the work with Boom Software AG Boom software was heuristically evaluated based on adapted checklists by Pierotti (2000), Nielsen (1994a), Raymond and Landley (2004), and the fluid project[27]. More information can be found in Sections 3.2.4 and 4.1.2. Additionally, within this work a checklist for heuristically evaluating mobile devices was created. The list is based on checklists and heuristics by Pierotti (2000), Nielsen (1994a), Raymond and Landley (2004), Ji et al. (2006), the fluid project, Microsoft, Google, and Apple. The list certainly is not complete, but it covers many potential usability issues and is a good starting point for further elaboration. For more information about the usability checklist see Sections 3.4 and 4.3. The full checklist can be found in Appendix A.

2.3.5. Thinking Aloud

The thinking aloud (TA) test method originates from the problem solving sciences from the year 1932 (Holzinger, 2006) and was later also brought to computer science. It is a method of good practical applicability (Holzinger, 2006). For thinking aloud tests test users are asked to perform predefined tasks while being supervised by the usability experts. For later analysis of the user's behavior during the tasks the sessions are recorded in audio and video.

According to Nielsen (1994b) tree to five test users are sufficient. However, it might be necessary to increase the amount of test users. In practice it is often reasonable to stop the thinking aloud test when no more information can be gained (Holzinger, 2006).

In general, thinking aloud tests should be applied in early development stage of the user interface because later it might be difficult to implement the suggested changes. With thinking aloud tests insight into the thinking processes of the test users can be gained. This information might be very helpful for further design decisions. However, the results of thinking aloud tests might be biased because usually such tests are performed in artificial environments under artificial conditions. Also test users often behave differently when being under supervision. This effect is called Hawthorne Effect

[27] fluidproject.org - An open, collaborative project to improve the user experience of community source software (*http://wiki.fluidproject.org/display/fluid/Usability+Evaluation+Questions*), last visited on 05/02/2012.

(McCarney et al., 2007). Additionally, verbalizing information may influence cognitive processes (Ericsson and Simon, 1980).

Another problem is that some test users feel uncomfortable about expressing their thoughts verbally or simply forget to do so. Therefore it is necessary to constantly remind the test users to speak out what they are thinking.

Time and equipment requirements of TA are relatively high compared to other usability evaluation methods. TA can, however, be applied in very early development stages using paper mockups (Holzinger and Brown, 2008b). Using paper mockups in early development stages has the advantage that they can be easily changed based on the feedback of the test users.

Thinking aloud tests are usually accompanied by questionnaires about the tested software and about the background knowledge of the test users. Later in this thesis, in Section 3.2.6, the methodology of the thinking aloud tests applied during this work are described.

2.4. Adaptive User Interfaces

Smart adaptive user interfaces (AUIs) are one way to satisfy the special needs of end users (El-Bakry et al., 2010). Adaptive user interfaces can be used to present the end user a UI which is tailored to the special needs of the end user (Germanakos P., 2009). Therefore, AUIs can be modified even at runtime in a way that the needs of certain end users are satisfied. However, one of the big challenges in this area is to determine the variables used as basis for developing different adaptions (Germanakos P., 2009). Criteria might include the background knowledge of the end user, the environment in which the application is used, technical criteria or combinations of those criteria. Context-awareness is the keyword in this context.

2.4.1. Context-awareness

According to Oxford Dictionaries the term context is defined as follows: Context is "*the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood*". Being aware of the current context makes it possible to adapt oneself to the context.

The idea of context-awareness is not new. Schilit et al. (1994) introduced context-awareness for ubiquitous computing in 1994. However, the capabilities of mobile devices have changed drastically since 1994.

Nowadays, smartphones with powerful processors and large, bright touch screens are in the pockets of many professionals, students and even children. Software providing different modes for different user groups is available, such as ArcheoApp (Holzinger et al., 2011b), which provides modes for students, tourists and children. In this case the professional context is taken into account for an adaption of the software. The topic context awareness and adaptation in mobile learning is also discussed in (Yuan-Kai, 2004).

Schmidt (2000) describe an application for PalmPilot called *Context NotePad*. They also define the term implicit human-computer interaction which describes the concept of interaction based on situational context rather than on explicit GUI manipulation.

Comprehensive user profiles where also visual, cognitive and emotional-processing parameters are included may improve the performance of adapted Web-based content. Evaluation results demonstrate the effectiveness of incorporating human factors in Web-based personalized environments (Germanakos et al., 2009). Also Bevan and Azuma (1997) emphasize the importance of considering human factors in usability design.

According to Lumsden (2008) context-aware mobile user interfaces are developed to improve the user experience by adapting the system behavior, based on a model of relevant use context factors. Also Biel et al. (2010) recognize the importance of considering the mobile usage context when designing mobile applications and evaluating usability. Therefore so called mobile context factors (MCFs) were defined: properties of the environment, user, device, task, and application.

One challenge is to give the bare sensor data a meaning (Schmidt et al., 1999). What does it mean if a brightness sensor reports 20 percent lightness and an acoustic noise sensor reports a sound pressure level of 20 dB? These values must be translated to a higher-level contextual meaning such as “indoors/outdoors”, “engaged in conversation”, “in a meeting” so that the application can react accordingly. Korpipaa et al. (2003) present a framework for the Symbian platform which implements such a mapping from low-level sensor date to a high-level representation of context. Instead of just using snapshots of the current sensor data analyzing time series of sensor data can be useful for forming higher-level contexts (Himberg et al., 2001).

But also displaying different content based on the current context falls in the area of context-awareness. Lemlouma and Layaida (2004) present a platform independent system based on a web service which is able to display different content based on the current user settings, the user profile, and other criteria such as network connection speed.

Context sensitivity for text input is widely used by mobile applications and software keyboards. Soft keyboards can change the keyboard layout in a way that symbols which are not likely to be used are replaced by other symbols or characters. For enabling this feature, software developers have to define an expected input type for text boxes (e.g. “text”, “email”, or “number”). This not only works for native mobile applications, but also for web pages using HTML5.

2.4.2. Adaptive vs. adaptable UIs

It has to be differentiated between adaptive and adaptable user interfaces. Adaptable UIs are adapted by software developers at design or implementation time manually, while adaptive software adapts itself at runtime automatically based on data gained from the end-user or the environment.

Customization is a concept in enterprise software where highly adaptable software can be developed for customers. The software usually builds on a framework which allows creating software which is tailored to the needs of the customer by combining existing software components to a whole system.

Mash-ups allow end-users to customize a user interface by (re)combining existing UI components (widgets) to a personalized UI. Such a customized user interface allows the end user to focus on his/her special needs. *Taptu* is a popular social network aggregation app which makes use of UI mash-ups.

What are the benefits of adaption? Adaptive UIs make it possible to provide the simplest user interface suitable for the current context. As stated before, the hypothesis is that simpler user interfaces enhance performance. But what does performance exactly mean and how can performance be measured on mobile devices?

Better performance not only means that less time is needed to execute a certain task, but also that the task is done with a lower error rate. On mobile devices as well as on non-mobile devices, the performance can be measured by measuring the time needed for a certain task and to what extent the task was accomplished. The challenge when measuring performance on mobile devices is that screen capturing and capturing of input may technically be more difficult because of a lack of tools, computational power and storage space. Also touch input is harder to track and to interpret than keyboard input.

What are the benefits of adaptability? Easily adaptable UIs make it possible for enterprises to customize the UI of an app without much effort. This leads to less expensive and to less time-consuming adaption processes. Usually adaptable software is designed in a modular way where single independent components can be combined to a new product. Later, Boom Software AG's framework for Total Customizing will be presented where more advantages of adaptable and customizable software will be pointed out.

3. Materials and Methods

In this section the methods and materials for the practical part of this work are described. First, in Section 3.1 the company we worked with – Boom Software AG– is introduced. The technology Boom Software AG uses for developing software is presented and the relation between the usability team (Peter Treitler and Michael Geier) and Boom Software AG (from now on referred as Boom) is described. In Section 3.2 the work with Boom and the goal of our work with Boom is explained. Finally, in Sections 3.3 and 3.4, the focus is on usability on mobile devices presenting an experiment about adaptive user interfaces and usability heuristics tailored to mobile devices.

3.1. Boom Software AG

Boom Software AG (*www.boomsoftware.com*) mainly creates easily and highly adaptable software for maintenance management and production control. Founded in 1995 by Joachim Schnedlitz, Boom currently has about 50 employees. Boom's headquarter is situated in Leibnitz, Styria, Austria.
Boom's concept of highly adaptable software is called Total Customizing. This means that the software can be customized to the needs of their customers to a high degree.

"*Boom Software is the first software developer in the area of maintenance management software to offer total customization and, as a result, the highest possible benefit for the customer.*" – Boom Software AG.

Boom has developed a framework which is the base technology for the Total Customizing concept and supports easy creation and adaptation of enterprise software. Boom also offers complete, adaptable products based on their framework. Maintenance Manager and Production Manager are two examples. Maintenance Manager is a system for management, standardization and optimization of maintenance processes for facilities and infrastructure. Production Manager is software for optimization and documentation of production processes as well as for supporting quality assurance. *ÖBB*

TS GmbH, *IKB AG* (*Innsbrucker Kommunalbetriebe AG*), *austriamicrosystems AG*, *Steyr-Daimler-Puch GmbH*, and *MAV – Hungarian Railways PLC* are some of Boom's customers using Maintenance Manager or Production Manager.

The information provided about Boom Software AG is either available at *www.boomsoftware.com* or was taken from documents provided to us during our work with Boom. Also information about discussions with our contact person, Roman Bobik, is used for this work. However, all information about Boom or Boom technology presented in this document was authorized by Boom for publication.

3.1.1. The Enterprise Framework

BORA (Business Oriented Rapid Adaption) is the name of Boom's framework which supports the Total Customizing concept. It is important to point out that customizing and Total Customizing describe different concepts. While the term customizing refers to the generic customization of software, Total Customizing is a concept where the base framework is built in a way that the application which is created based on the framework is highly customizable. Additionally, according to our contact person Roman Bobik, the BORA framework "allows reducing the technical efforts from about 80% to approximately 20% of the project volume by minimizing redundant and hard technical tasks and provides the developer with a clear software development process – from requirements analysis, rapid prototyping to deployment and maintenance."

All applications developed by Boom (for example Maintenance Manager and Production Manager) are based on their Model-View-Controller (MVC) based framework. Figure 13 illustrates the concept of Total Customizing the BORA framework suggests.

ADAPT TO INDUSTRIES NEEDS
ADAPT TO BUSINESS PROCESSES
ADAPT TO YOUR COM-PETITIVE EDGE
ADAPT TO NEW TECH-NOLOGIES

Figure 13: Total Customizing with Boom's BORA framework (source: Boom)

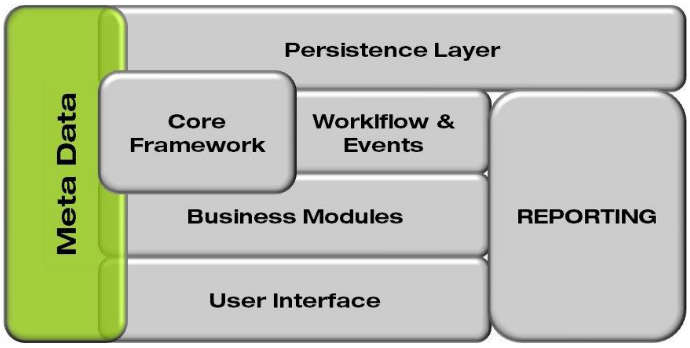


Figure 14: BORA framework - main components (source: Boom)

BORA is not only an internal design and development method as well as a set of unified software development tools based on Microsoft's .NET Framework, but also a technical basis, an application framework and a standard. All Boom products are based on the methodologies and tools specified by BORA. According to Bora Man, head of the BORA team, the BORA Framework allows to focus on the development goals by reducing the need for solving technical issues. All software products created with BORA consist of modules, which can be combined to a whole product.

The reader shall remind that it must be differentiated between adaptive and adaptable software. The aim of Total Customizing is to create adaptable software – adaptivity is not the primary goal. Adaptable (or customizable) software is adapted by the software developers usually at design and implementation time while adaptive software adapts "itself" automatically at runtime based on the application's context.

Figure 14 shows a schema of the BORA framework. BORA consists of several components: The persistence layer, the workflow and events engine, the business modules, the user interface, and the reporting module. The core framework is the central component of BORA. Metadata is used for modeling applications based on the framework. The framework creates the application based on the model description. The following paragraphs give a more detailed description of the single modules.

Persistence Layer. The persistence layer translates the OO-model to a relational database. The O/R-mapper which is part of the BORA framework is able to translate complex OQL queries so that developers do not have to write any SQL.

Core Framework. The central element of the core framework is the object broker which manages the business objects including caching and transactions. Also, the core framework provides functionality for defining and checking rules, security, sessions, scripting and more.

Workflows & Events. Event-driven workflows are started by changing the states of objects. Workflows can be simple operations such as the calculation of a value as well as complex sequences of commands.

Business Modules. This part combines the components mentioned above (model, rules, events, ...) to single modules. Modules can be core services (e.g. user management) as well as subsystems (e.g. warehouse management) or whole applications (e.g. Boom Maintenance Manager). All modules can be combined at any layer.

User Interface. BORA's UI framework automatically considers the parts mentioned above when creating the user interface. This makes it possible to create working user interfaces without writing any line of program code.

Reporting. Apart from mainly OLTP (Online-Transaction-Processing) based core functions most of the business modules need to create reports. BORA provides a component to define and to export reports which also considers the parts mentioned above. Reusing definitions for labels, headings and other parts both in the user interface and in reports makes the output consistent.

When generating the application, the BORA framework does not explicitly generate source code from the model, but instead abstract classes are pre-generated. The actual concretion is derived at run-time from the model description which makes it possible to make changes even at the customer's place without the need for recompiling the application. A more detailed view on BORA's core components (Figure 15) clarifies the strict separation between abstract specifications (right part) and the implementation by the system (left part).
The common generative approach of similar systems, where source code is generated from a model, for example defined in UML, has several disadvantages compared to the BORA approach: First, the common generative approach is more error prone than the BORA model because in the common generative approach the generated code must be merged with changes to previously generated code. Secondly, the common generative approach is more time-consuming because in every trip post-processing in necessary.

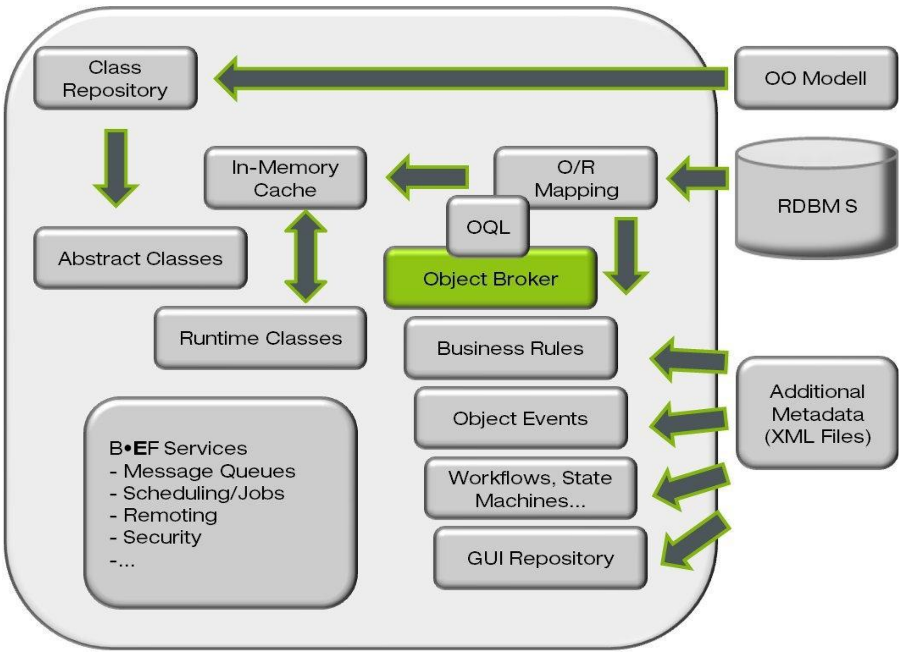


Figure 15: Detailed view on BORA's core components (source: Boom)

Advantages of Boom’s BORA approach include: Modular concept and therefore high reusability of single modules, good scalability, simple interfaces and separation of business logic and user interface. As basic forms and user interfaces can be created very quickly, rapid prototyping can be realized to get feedback in an early stage of development. Minimization of time-to-market, focusing on the problem – not on technical issues, flexible adaptation, a unified development process, upgrading of applications without the need for changing customer-specific settings, minimization of the risk for cost overrun and easy internationalization are more properties which result from the usage of the BORA framework.

During the work with Boom we experienced that the enterprise framework allows quick creation of basic applications using a database in the background when creating the tutorial application *Leseratte* (see Section 3.2.3). Using a data model designer (*Boom BORA Designer*) the data model and the corresponding database schema is created without the need for writing any C# code. *Boom BORA Designer* also creates a Visual Studio solution containing the resulting application which implements default behavior. The default behavior can be customized in following steps. Using Visual Studio (or any text editor) XML files defining the UI and other parts of the system such as business rules can be edited in order to create a customized application. Changes to the XML files can even be made during execution of the application - the changes can be loaded at runtime and immediately influence the look and/or the behavior of the application.
For implementing special features or for overriding default functionality C# can be used as programming language. As BORA-based applications require a database in the background, a DBMS such as Microsoft SQL Server, SQL CE, Oracle, firebird, or DB2 must be available.
The following XML snippets, taken from Boom’s BORA presentation slides, give examples of the used XML language for defining the data model, storage mapping, business rules, and the graphical user interface.

Example class repository:

```
<Package Name="TAnf">
  <Entity Name="TAnf" CID="-389585192">
    <InterfaceRef RefCID="2109958077" />
    <Property Name="PkgCount" EntityRef="10"
      MultiplicityMin="1" MultiplicityMax="1" />
    <Property Name="PkgWeight" EntityRef="13"
      MultiplicityMin="1" MultiplicityMax="1" />
    <!-- ... -->
```

Example storage mapping:

```
<entity name="TAnf" dataSource="default">
  <table name="tabTAnf">
    <oid column="TAnfID" />
    <field name="PkgCount" column="PaketAnzahl" />
    <field name="PkgWeight" column="Gewicht" />
    <field name="PkgVolume" column="Volumen" />
    <!-- ... -->
    <assocEnd name="Requestor" column="AnfUserID" />
    <assocEnd name="RequPlant" column="AnfWerkID" />
    <!-- ... -->
```

Example Business Rules:

```
<rulesRepository>
  <entity name="TAnf" type="save">
    <range min="1" max="99">PkgCount</range>
    <rule>
      <expression>DelvrDate >= FetchDate</expression>
      <textArgs>DelvrDate, FetchDate</textArgs>
      <text>[0] darf nicht vor [1] liegen.</text>
    </rule>
    <!-- ... -->
```

Example GUI de:

```
<entity name="TAnf">
  <feature name="PkgCount">
    <style><title lang="de-AT">Anzahl VPE</title></style>
  </feature>
  <!-- ... -->
  <form>
    <section><style>
      <title>Anforderung bearbeiten/hinzufügen</title>
    </style>
    <feature edit="none">OID</feature>
    <feature>Name</feature>
    <feature gui="dropdown">TransportType</feature>
    <feature>CCtr</feature>
    <!-- ... -->
```

The GUI definition language for Windows Applications is based on XAML, while the generic UI definitions are written in another XML dialect.

Boom’s three development areas include core development, domain development, and application development. The core development relates to the development of the base technology BORA. Few experts are part of the core development, i.e. the development of the base framework BORA, while the majority of the software engineers focus on the application development. Application developers use the BORA framework for creating adapted applications for Boom’s customers. Domain development relates to the development of base products such as BMM or BPM which can be further customized. Figure 16 illustrates the mentioned development areas.

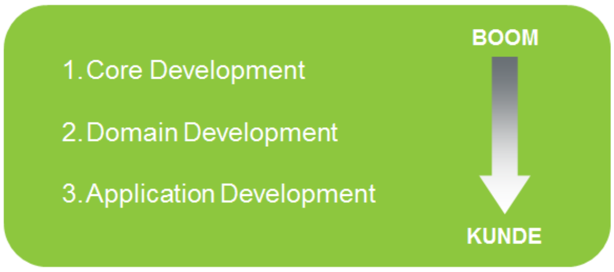


Figure 16: Boom’s three development areas (source: Boom)

A typical development process at Boom starts with defining the customer requirements. From the requirements, an application model is created and then a default application is created from the model. Finally, the created default application is adapted to the special customer requirements.

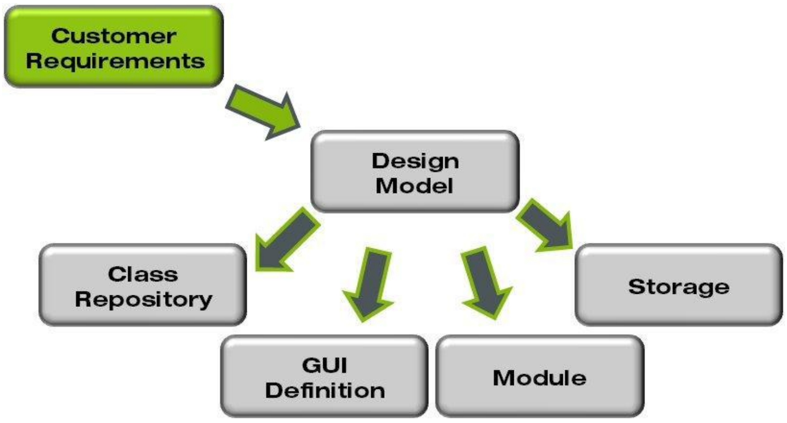


Figure 17: Boom’s simplified process model for software development (source: Boom)

According to Bobik, Boom’s software solutions tend to be very consistent when speaking of processes, navigation and user interface design because many parts of the user experience composed from well-tested user interface components and navigation concepts which are part of the base framework BORA.

“*Since all of our and our partner’s software solutions build on these UI concepts we are very concerned to improve the usability of these components. By improving them we make more than 20.000 users more productive every day*.”

– Roman Bobik, Boom Software AG.

3.1.2. Future Development

During our work with Boom Software AG we identified two directions the company wants to establish or improve. First, Boom will respect the need for mobile applications. While Boom’s main focus has historically been on Desktop-Software for Microsoft Windows, over the years it was heavily invested in creating a HTML-based UI-layer for BORA-applications in order to bring specific parts of Boom applications to the world of occasionally-connected mobile devices such as smartphones and tablets. Therefore, Boom will further expand their technology to mobile devices such as Android-based devices or Windows-based Tablet PCs in the near future. The future goal is to supplement the existing desktop system with mobile applications. The goal is not to port the whole desktop application to a mobile device, but only parts which are needed outside the company's headquarters. The decision whether to go HTML5 or to create native mobile applications hasn’t been taken yet.

Secondly, it is strived for further improvement of the usability of Boom software. As the reader will see later, the overall usability already is on a high level; however, there are some minor issues to be resolved. Additionally, the awareness of usability should be raised amongst the developers at Boom Software AG and company-wide usability guidelines, providing BORA-developers with a simple checklist of dos and don’ts, should establish a unified understanding and implementation of usability.

3.2. Usability Consultancy

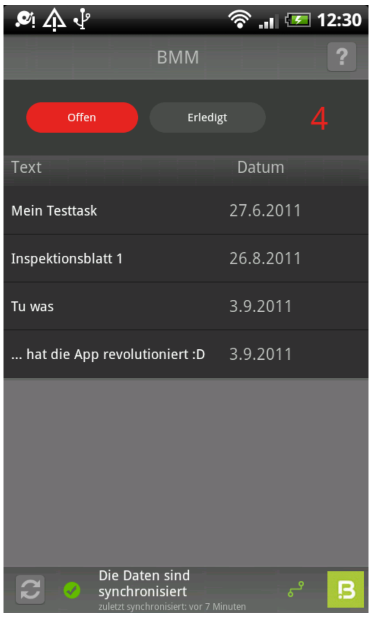
During the work with Boom Software AG we acted as consultants in usability questions. We accompanied the development of a prototype for a mobile Android client and at the same time we performed usability evaluations of existing desktop software. Apart from the following Section 3.2.1, the order of this section's subsections corresponds to the chronological order of the work at Boom.

3.2.1. Mobile App

During the work with Boom we had the opportunity to follow the development of a prototype for a mobile Android application which is based on the BORA framework. The purpose of the mobile application is to implement parts of the desktop user interface module for mobile devices. The prototype of the app is to be used as digital task sheet. The goal is to make printed task sheets redundant and to speed up the task reporting for the customer.

Figure 18 shows a screenshot of the latest version of the Android app. The screenshot shows the user's open task list. It also shows that there are four open tasks and that the data is currently synchronized (bottom). The synchronization feature is important for countryside workers because it is not guaranteed that there is always an Internet connection available. As stated in the introduction of this work, outside workers often have to work under rough conditions. Also outside workers do not have access to immediate help from colleagues or a help center in case of technical troubles. Therefore good usability and error prevention is particularly crucial.

The mobile application only implements the presentation layer which creates the UI based on an XML UI description received from the application server. It is up to the client software how to interpret the UI description and how to finally render the UI. Input on the mobile device is sent back to the server which then processes the input data.



Text	Datum
Mein Testtask	27.6.2011
Inspektionsblatt 1	26.8.2011
Tu was	3.9.2011
... hat die App revolutioniert :D	3.9.2011

Die Daten sind synchronisiert
zuletzt synchronisiert: vor 7 Minuten

Figure 18: Mobile Android app for task sheets

The platform independent XML description generally describes the UI layout consisting of sections, text input boxes, labels and other controls. When the client application receives the UI description the client interprets it in a way it looks and behaves best for the device type the software runs on. Therefore the final UI could theoretically look completely different on different devices. However, there are standards which require the UIs to be rendered ant to behave similarly in order to have a certain recognition value. The concept is similar to SaaS (Software as a Service) with the difference that the client is not necessarily a web browser but can also be a specialized client application.

Languages which describe user interfaces are called user interface markup languages. UIML (User Interface Markup Language), XAML (Extensible Markup Language), XUL (XML User interface Language) are popular examples for UI markup languages. Most user interface markup languages are based on XML (El-Bakry et al., 2010). There can be different levels of detail in describing a UI using such a markup language. There are languages which allow to define UIs very detailed including exact measures and positions, but there are also languages which intentionally only allow specifying relatively abstract UI descriptions (e.g. only relative positions of controls, no concrete measures or positions). The latter type of UI description language allows the client which finally renders the UI to adapt the UI to the screen dimensions of the device.

The advantages of such a distributed approach where the client application is only responsible for rendering and for forwarding user input include:

- Only the frontend app has to be published to the customers.
- Changes in the backend which do not affect the frontend can be made without the need for updating the frontend application.
- Even changes in the frontend can be made without updating the frontend app by changing the UI definition on server side.
- Changes in the UI have only to be made once on the server side. All the clients on different operating systems automatically provide the new UI without the need for updating the client applications.

A disadvantage of systems as described above is the lack of flexibility in UI design. Also, retaining platform independency by supporting multiple different device types (such as desktop and mobile systems) at the same time might be difficult in some cases. If, for example, multi-touch gestures should be supported by mobile applications the desktop application must properly handle the gesture instructions as well.

In general, a UI description can be written in any suitable, machine readable language. For reasons of standardization, simplicity, and (human) readability XML might be a good choice. More compact representations (some binary format, for example) might be necessary in case of very low bandwidth. However, in practice the size of the UI description data should not be relevant as, once submitted, this data is usually cached on the client side.

Of more importance is the format in which data is exchanged between the client application and the server. Again, XML could be used for the reasons mentioned above. The big disadvantage of XML is that XML is large in terms of bandwidth requirements. This might not be a problem when using W-LAN or HSDPA connections or when the amount of data to be transmitted is low anyway. In case of slow connections (e.g. UMTS) which is still used on the countryside, and in case of large amounts of data the proper choice of the data description language might be relevant for stability and reliability of the mobile software and for acceptable user experience.

In a meeting with the app developers we tried to give advice and we discussed certain aspects of the mobile application for task sheets. The discussion identified certain potential issues and led to some improvements of the application. The results of the meeting are discussed in Section 4.1.1.

3.2.2. Usability Evaluation

Another part of the work with Boom Software AG was the evaluation of existing user interfaces as well as new user interfaces components for Boom's desktop applications. We applied the heuristic evaluation (HE) inspection method as well as the thinking aloud (TA) test method.
We heuristically evaluated three existing products - two productive systems and one application for demonstration and teaching purposes. For optimal results we got the opportunity to evaluate the two productive systems (Boom Maintenance Manager and Target Manager) under realistic conditions, i.e. with real data.
In a further step the application for demonstration and teaching purposes ("*Leseratte*") was extended with newly developed, partly experimental and yet unused controls. These controls were then tested for usability issues. Design and layout of the application were similar to the design and the layout of software used at clients due to the fact that all Boom applications build on the same framework. This allowed us to draw conclusions from the evaluated software to other Boom software in certain cases.
For the heuristic evaluation checklists of several sources were combined to one large set of checklist items. Some not applicable checklist items had to be removed; others had to be added in order to comply with the special needs of the tested business software.
The thinking aloud tests were conducted with nine test users. The common number of test users (according to Nielsen (1994b) about five) was intentionally exceeded in order to get most accurate results. Also, the rule of thumb that about five test users are sufficient is disputable. A better rule of thumb is to stop the thinking aloud tests whenever no more new information is gained (Holzinger, 2006). One goal for the

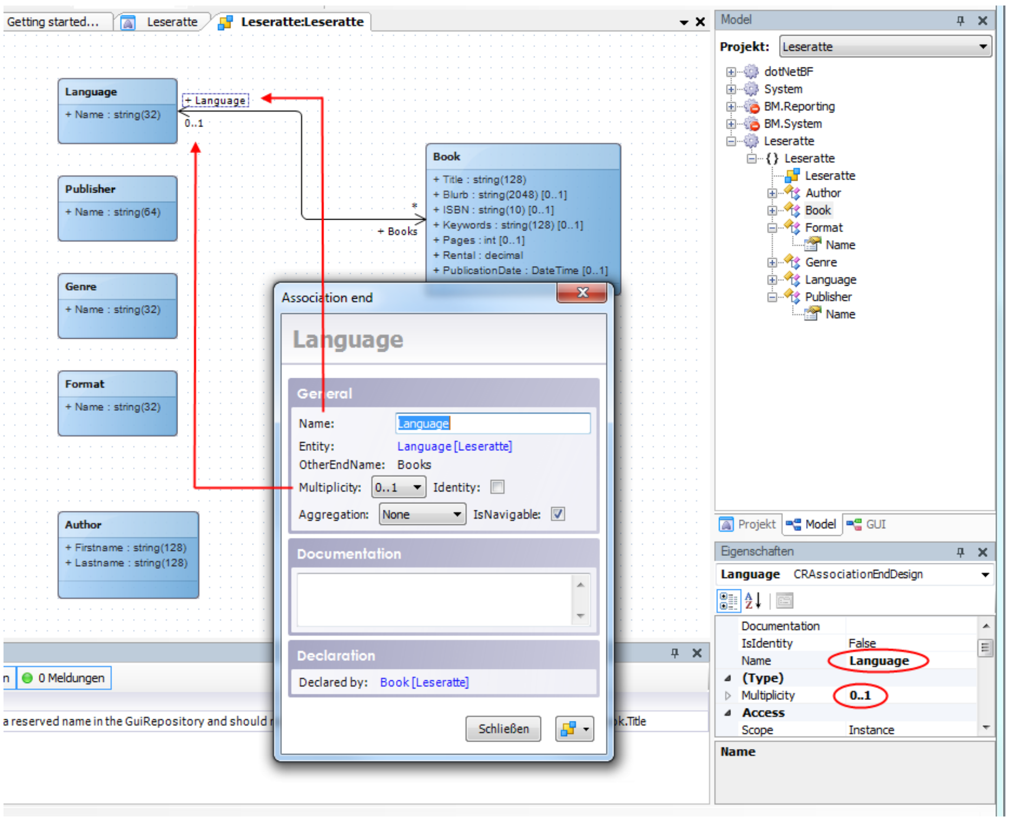
thinking aloud tests was to cover as many different user types as possible (still targeting the potential end users of the tested system).
Summing up, there were three different goals to reach by performing the usability tests. First, specific and applications currently used by Boom's clients should be tested for usability issues. Secondly, the overall structure of boom applications should be revised. Third, newly developed controls/components should be tested for usability flaws.
The documentation of the progress of our work and the communication was done via Boom's Support Manager – a web-based bug tracking, communication and support platform.

3.2.3. Leseratte

The work on the usability evaluation of Boom software started with getting an introduction to Boom's BORA technology. For developers who want or need to get familiar with the BORA framework Boom provides a tutorial in which a simple application is built. The application is named *Leseratte* (German for bookworm) and implements a simple book management software for libraries.
As first step we created the application based on the tutorial. After that, when we gained more insight into Boom's framework, we started with the first usability evaluations of BORA-based software. First, we heuristically evaluated the tutorial application *Leseratte*, and then we evaluated two applications used by Boom in the real world. For a detailed description of the methodology of the heuristic evaluation refer to Section 3.2.4. Based on the results of the heuristic evaluation of *Leseratte* we proposed changes in order to improve the usability of the tutorial application. As next step the usability of new UI controls, developed by Boom, were to be tested. In order to test these controls we integrated them into the improved version of the *Leseratte* application. This new version of the *Leseratte* ("*Leseratte* 2.0") was used as object to be tested in a thinking aloud test with nine test users. In Section 3.2.6 we will describe the methodology of the thinking aloud tests.

In the remaining lines of this subsection the development of the basic tutorial version of *Leseratte* is described, the design and the development of *Leseratte* 2.0 follows in Section 3.2.5.

As mentioned above *Leseratte* is the name of the application built during a tutorial Boom provides for software developers who want to get familiar with the BORA framework. The first steps in the tutorial explain how to install *Boom BORA Designer*. During the tutorial *Boom BORA Designer* is used to define the data model and to generate the application and the database schema from the data model (Figure 19). The tutorial further explains how to adapt the user interface by modifying the XML UI definition files and introduces the topic event handling. It is also demonstrated how to implement a search feature and how extend the UI with custom features, such as a copy function for single datasets. Figure 20 shows a part of the tutorial web page which led to our first version of the library software (Figure 21).



Association end

Language

General

Name: Language
Entity: Language [Leseratte]
OtherEndName: Books
Multiplicity: 0..1 Identity: ☐
Aggregation: None IsNavigable: ☑

Documentation

Declaration

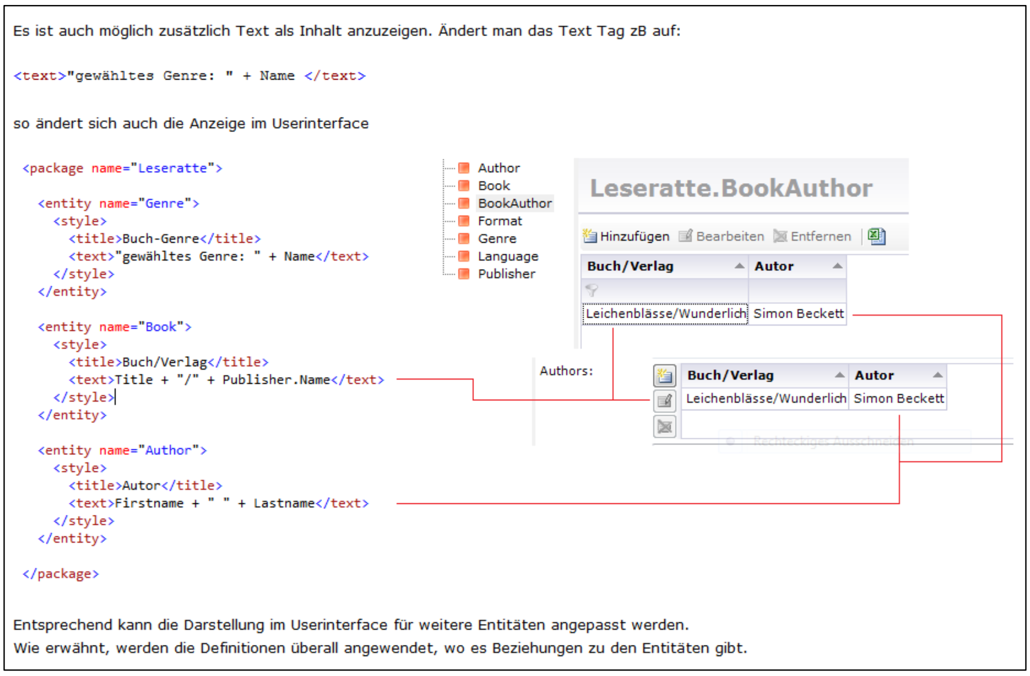
Declared by: Book [Leseratte]

Language CRAssociationEndDesign

Documentation	
IsIdentity	False
Name	Language
(Type)	
Multiplicity	0..1
Access	
Scope	Instance

Name

Figure 19: Leseratte tutorial – screenshot of Boom BORA Designer



Es ist auch möglich zusätzlich Text als Inhalt anzuzeigen. Ändert man das Text Tag zB auf:

```
<text>"gewähltes Genre: " + Name </text>
```

so ändert sich auch die Anzeige im Userinterface

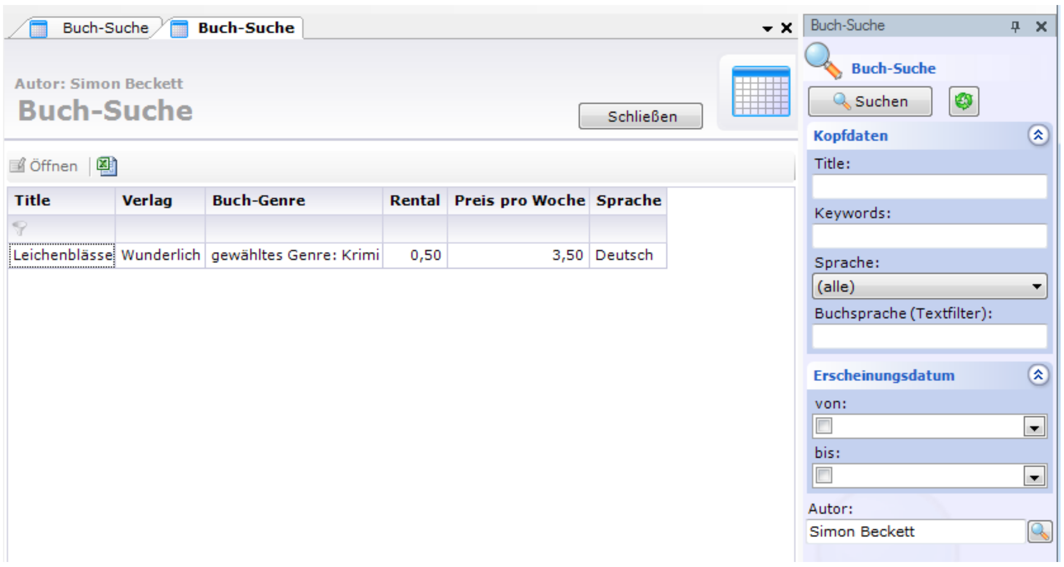
```
<package name="Leseratte">

  <entity name="Genre">
    <style>
      <title>Buch-Genre</title>
      <text>"gewähltes Genre: " + Name</text>
    </style>
  </entity>

  <entity name="Book">
    <style>
      <title>Buch/Verlag</title>
      <text>Title + "/" + Publisher.Name</text>
    </style>
  </entity>

  <entity name="Author">
    <style>
      <title>Autor</title>
      <text>Firstname + " " + Lastname</text>
    </style>
  </entity>

</package>
```

Buch/Verlag	Autor
Leichenblässe/Wunderlich	Simon Beckett

Buch/Verlag	Autor
Leichenblässe/Wunderlich	Simon Beckett

Entsprechend kann die Darstellung im Userinterface für weitere Entitäten angepasst werden.
Wie erwähnt, werden die Definitionen überall angewendet, wo es Beziehungen zu den Entitäten gibt.

Figure 20: Leseratte tutorial – UI definition



Title	Verlag	Buch-Genre	Rental	Preis pro Woche	Sprache
Leichenblässe	Wunderlich	gewähltes Genre: Krimi	0,50	3,50	Deutsch

Figure 21: Screenshot of the tutorial version of Leseratte

When the tutorial-based development of the application was finished, some – preferably real – data was needed for populating the database. Therefore a C# program was written which was used to automatically import real book data using Google's Books API.

3.2.4. HE

The first step was to select the applications to be tested. Together with Boom Software we agreed on inspecting three different applications. First, the tutorial application *Leseratte* was evaluated during a pilot inspection. Secondly, Boom Maintenance Manager (BMM, native Windows client and web client) and Target Manager (TM, native Windows client) were evaluated. TM is a system for customer tie and customer acquisition, supports communication with customers, efficient fundraising as well as for internal administration. BMM is a system for management, standardization and optimization of maintenance processes for facilities and infrastructure.

The second step was to collect and to develop a set of heuristics suitable for Boom software. Rule sets from different sources were reviewed and merged to one rule set applicable for the software to be tested. The single rules (for example "Can users cancel out of operations in progress?" by Pierotti's rule set (Pierotti, 2000)) were assigned to categories similar to the categories C1 to C12 as described in Section 2.3.4 in order to get a better overview and to avoid duplicates. All in all 315 rules were collected and applied during the heuristic evaluation. When collecting the rules most of the rules which obviously were not applicable for the software to be tested were excluded. However, there was still one or the other rule which was not well applicable in certain situations and therefore ignored. Other rules had to be re-interpreted for the current situation.

Example: The rule which origins from times where text terminals were used "Is reverse video or color used to indicate that an item has been selected?" (source: Pierotti) must be reinterpreted in a way that it fits into the current GUI world. That is, the rule could be interpreted as "Are list items highlighted in an easily recognizable way to indicate that an item has been selected?" However, we saw that, even though some of the found rules originate from relatively old times, many rules still have validity. Some rules might even be timeless, such as "Is vocabulary familiar to the intended user, avoiding system-oriented terms?" (source: fluid). Table 11 exemplarily shows some rules used for the heuristic evaluation of Boom Software.

C1 Feedback (N1 *Visibility of system status* and also S3, S4, R1, R2):	
Pierotti	*Do menu instructions, prompts, and error messages appear in the same place(s)?*
Pierotti	*In multipage data entry screens, is each page labeled to show its relation to others (page numbers, ...)?*
fluid	Does a page's title accurately describe its purpose?

C5 User control, Flexibility (N3 *User control and freedom* and also S2, S7, R3, R9, R10, P1):	
Pierotti	*Can users cancel out of operations in progress?*
Pierotti	*Can users reduce data entry time by copying and modifying existing data?*
Pierotti	*If the system uses a pointing device, do users have the option of either clicking on menu items or using a keyboard shortcut?*

C8 Errors (N9 *Help users recognize, diagnose, and recover from errors* and also N5, S5, R14):	
Pierotti	*Do prompts imply that the user is in control?*
fluid	*Can users easily recover from errors, unintended actions, or actions that did not lead to desired results (eg undo, back)?*
fluid	*Is confirmation required when an action is difficult or impossible to undo?*

Table 11: Examples of checklist items from different categories.

Finally, the evaluation was performed by the usability test team (Peter Treitler and Michael Geier). First, all three applications (four different user interfaces) were inspected independently by the two usability experts. The software was not only explored based on common use cases defined by Boom but also freely without any restrictions. The use cases helped us to gain knowledge about the common usage of the software. Then, the independently collected results were merged and discussed by the usability team. Finally, a detailed report was written for each application. The results were presented and discussed with Boom. Within this work the results of the heuristic evaluation are presented and discussed in Section 4.1.2. The proposed changes for

BMM and TM are subject to be implemented by Boom Software AG. The proposed changes on *Leseratte*, however, were implemented by the usability team. The resulting application *Leseratte* 2.0 was supplemented with more features and test object for the thinking aloud tests.

3.2.5. Leseratte 2.0

After the heuristic evaluation of the *Leseratte* application, BMM, and TM, an improved and extended version of *Leseratte* was designed and implemented (*Leseratte* 2.0). The goal was to implement the changes we proposed during the heuristic evaluation as well as the integration of Boom's newly developed UI controls in order to evaluate their usability during a thinking aloud test.

For the design of *Leseratte* 2.0 UI mockups were created and a concept for an extended data model was elaborated. Figure 22 and Figure 23 show the UI mockups created using the open source software Pencil[28]. Figure 24 shows a diagram representing the extended data model. Figure 25, Figure 26, and Figure 27 show screenshots of the resulting application. The pie chart, the tiled list view, and the calendar view are three of the new controls to be tested for usability issues.

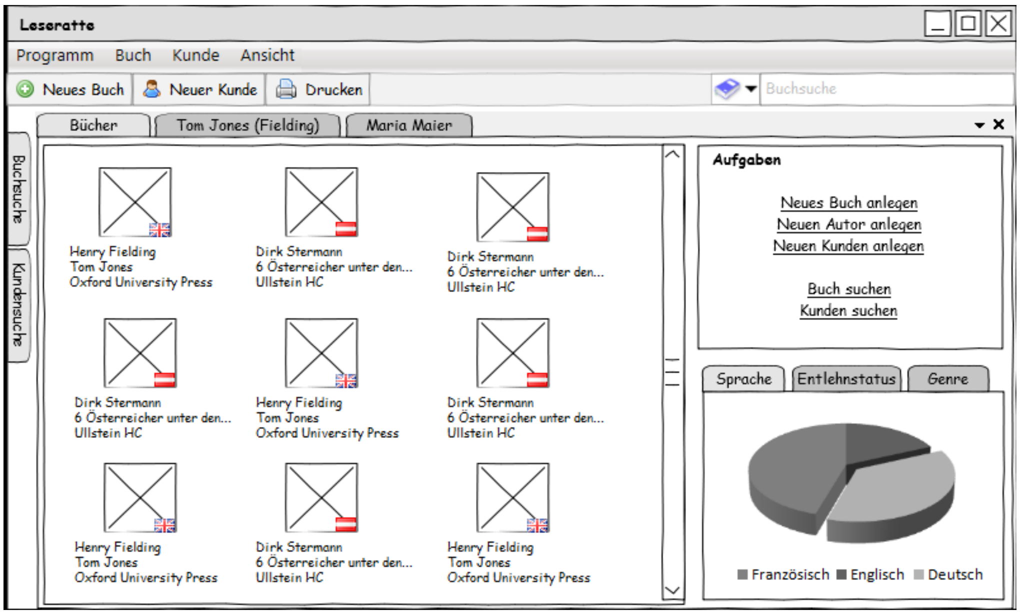
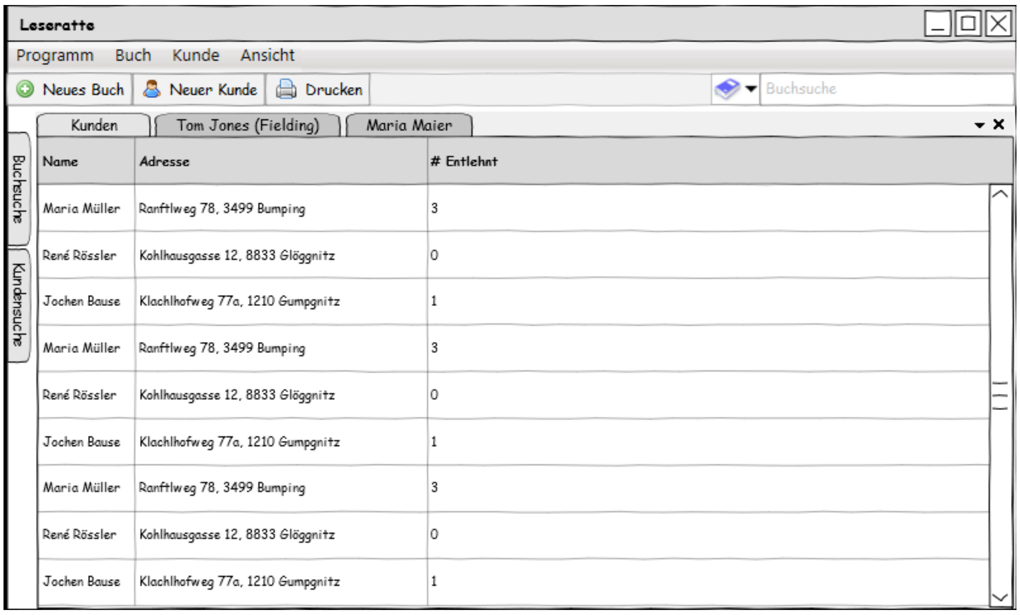


Figure 22: Mockup for Leseratte 2.0 – Start page and book overview.

[28] Pencil: *http://pencil.evolus.vn*, last visited 02/02/2012



Name	Adresse	# Entlehnt
Maria Müller	Ranftlweg 78, 3499 Bumping	3
René Rössler	Kohlhausgasse 12, 8833 Glöggnitz	0
Jochen Bause	Klachlhofweg 77a, 1210 Gumpgnitz	1
Maria Müller	Ranftlweg 78, 3499 Bumping	3
René Rössler	Kohlhausgasse 12, 8833 Glöggnitz	0
Jochen Bause	Klachlhofweg 77a, 1210 Gumpgnitz	1
Maria Müller	Ranftlweg 78, 3499 Bumping	3
René Rössler	Kohlhausgasse 12, 8833 Glöggnitz	0
Jochen Bause	Klachlhofweg 77a, 1210 Gumpgnitz	1

Figure 23: Mockup for Leseratte 2.0 – customer list.

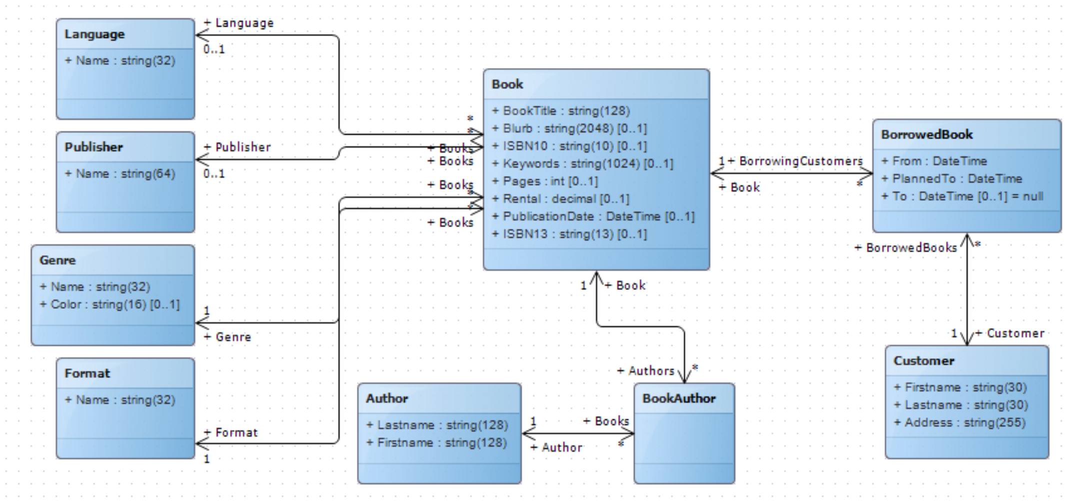
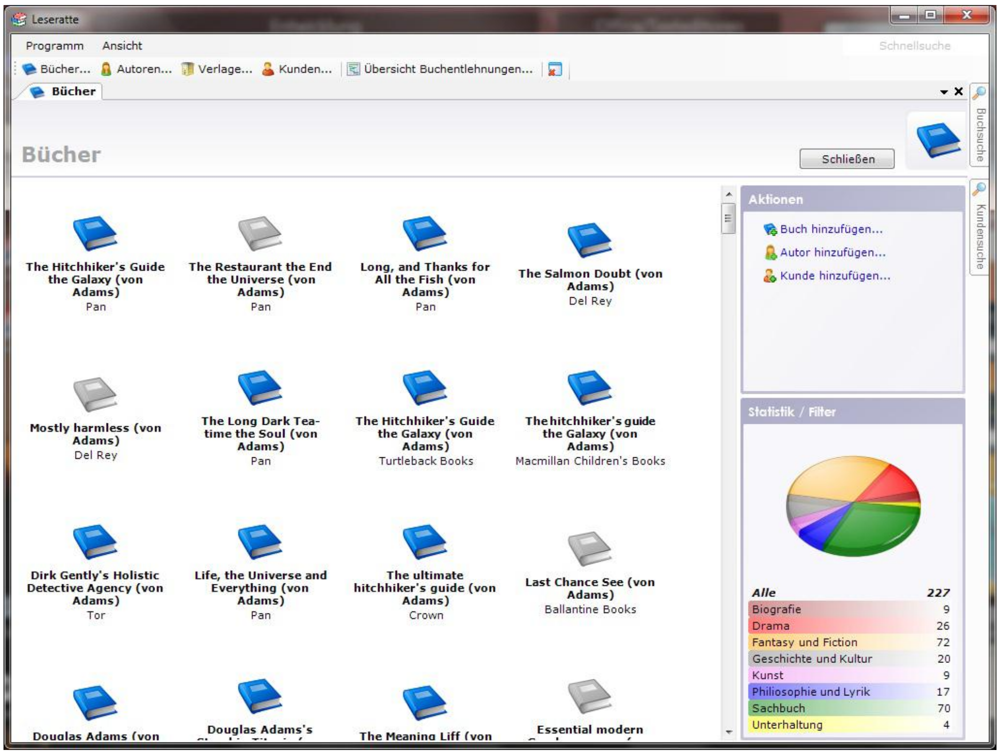


Figure 24: Data model of Leseratte 2.0



Bücher

The Hitchhiker's Guide the Galaxy (von Adams)
Pan

The Restaurant the End the Universe (von Adams)
Pan

Long, and Thanks for All the Fish (von Adams)
Pan

The Salmon Doubt (von Adams)
Del Rey

Mostly harmless (von Adams)
Del Rey

The Long Dark Tea-time the Soul (von Adams)
Pan

The Hitchhiker's Guide the Galaxy (von Adams)
Turtleback Books

The hitchhiker's guide the Galaxy (von Adams)
Macmillan Children's Books

Dirk Gently's Holistic Detective Agency (von Adams)
Tor

Life, the Universe and Everything (von Adams)
Pan

The ultimate hitchhiker's guide (von Adams)
Crown

Last Chance See (von Adams)
Ballantine Books

Douglas Adams (von

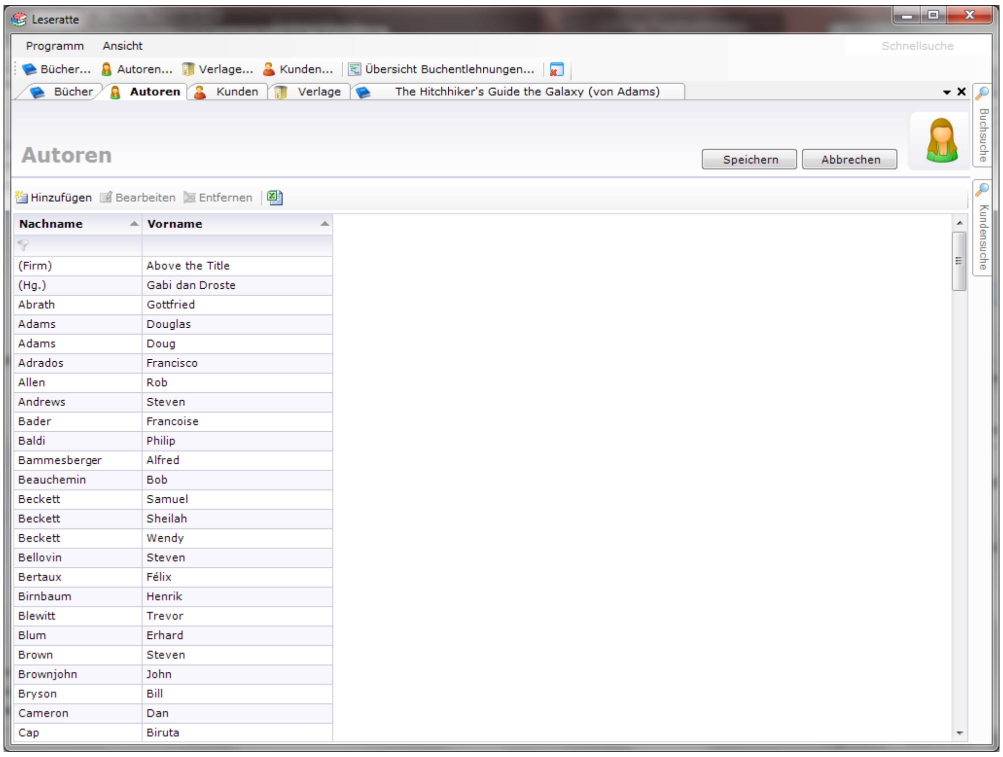
Douglas Adams's

The Meaning Liff (von

Essential modern

Alle	*227*
Biografie	9
Drama	26
Fantasy und Fiction	72
Geschichte und Kultur	20
Kunst	9
Philiosophie und Lyrik	17
Sachbuch	70
Unterhaltung	4

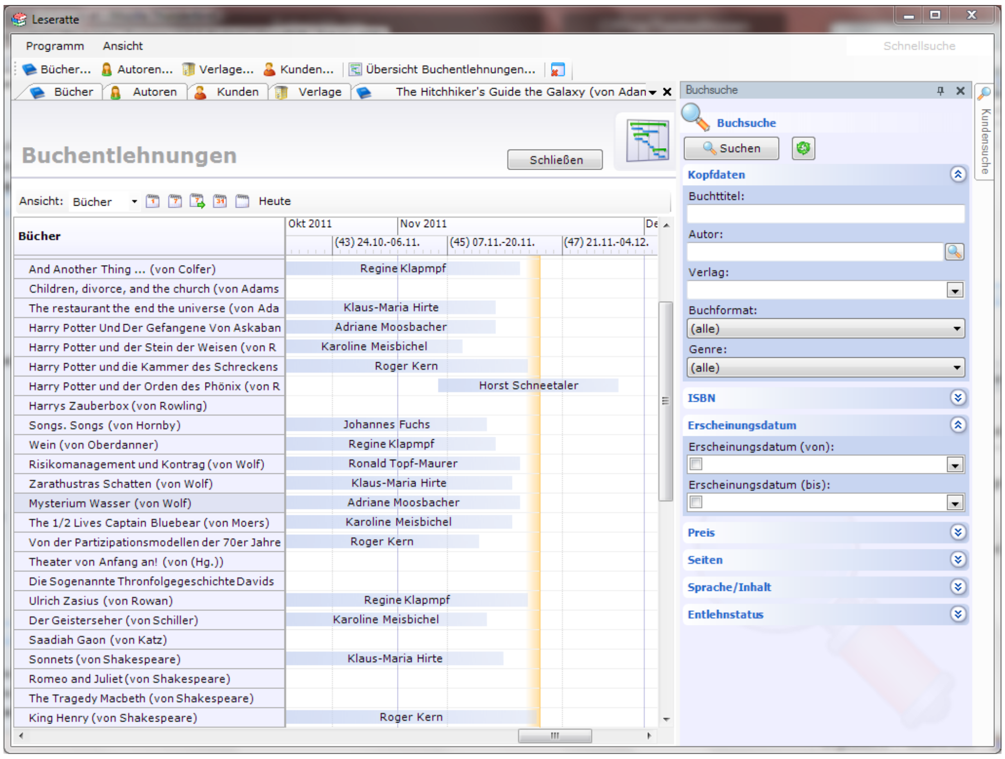
Figure 25: Screenshot of final Leseratte 2.0 – Start page and book overview.



Autoren

Nachname	Vorname
(Firm)	Above the Title
(Hg.)	Gabi dan Droste
Abrath	Gottfried
Adams	Douglas
Adams	Doug
Adrados	Francisco
Allen	Rob
Andrews	Steven
Bader	Francoise
Baldi	Philip
Bammesberger	Alfred
Beauchemin	Bob
Beckett	Samuel
Beckett	Sheilah
Beckett	Wendy
Bellovin	Steven
Bertaux	Félix
Birnbaum	Henrik
Blewitt	Trevor
Blum	Erhard
Brown	Steven
Brownjohn	John
Bryson	Bill
Cameron	Dan
Cap	Biruta

Figure 26: Screenshot of final Leseratte 2.0 - List of authors.



Buchentlehnungen

Ansicht: Bücher

Buchsuche

Buchtitel:

Autor:

Verlag:

Buchformat: (alle)

Genre: (alle)

Erscheinungsdatum (von):

Erscheinungsdatum (bis):

Bücher	Okt 2011 – Nov 2011 – De: (43) 24.10.-06.11. / (45) 07.11.-20.11. / (47) 21.11.-04.12.
And Another Thing ... (von Colfer)	Regine Klapmpf
Children, divorce, and the church (von Adams	
The restaurant the end the universe (von Ada	Klaus-Maria Hirte
Harry Potter Und Der Gefangene Von Askaban	Adriane Moosbacher
Harry Potter und der Stein der Weisen (von R	Karoline Meisbichel
Harry Potter und die Kammer des Schreckens	Roger Kern
Harry Potter und der Orden des Phönix (von R	Horst Schneetaler
Harrys Zauberbox (von Rowling)	
Songs. Songs (von Hornby)	Johannes Fuchs
Wein (von Oberdanner)	Regine Klapmpf
Risikomanagement und Kontrag (von Wolf)	Ronald Topf-Maurer
Zarathustras Schatten (von Wolf)	Klaus-Maria Hirte
Mysterium Wasser (von Wolf)	Adriane Moosbacher
The 1/2 Lives Captain Bluebear (von Moers)	Karoline Meisbichel
Von der Partizipationsmodellen der 70er Jahre	Roger Kern
Theater von Anfang an! (von (Hg.))	
Die Sogenannte Thronfolgegeschichte Davids	
Ulrich Zasius (von Rowan)	Regine Klapmpf
Der Geisterseher (von Schiller)	Karoline Meisbichel
Saadiah Gaon (von Katz)	
Sonnets (von Shakespeare)	Klaus-Maria Hirte
Romeo and Juliet (von Shakespeare)	
The Tragedy Macbeth (von Shakespeare)	
King Henry (von Shakespeare)	Roger Kern

Figure 27: Screenshot of final Leseratte 2.0 – Timeline and book search pane.

3.2.6. TA

In order to test newly developed controls and components we created an application called *Leseratte* 2.0 which contains the new controls on the one hand and resembles the general structure of typical Boom applications on the other hand. We invited nine users to participate in our TA test (one pilot test user and eight test users). The test users were mostly relatively inexperienced in using computers, but did at least operate computers at work. This user type is the usual target user of the tested enterprise software.

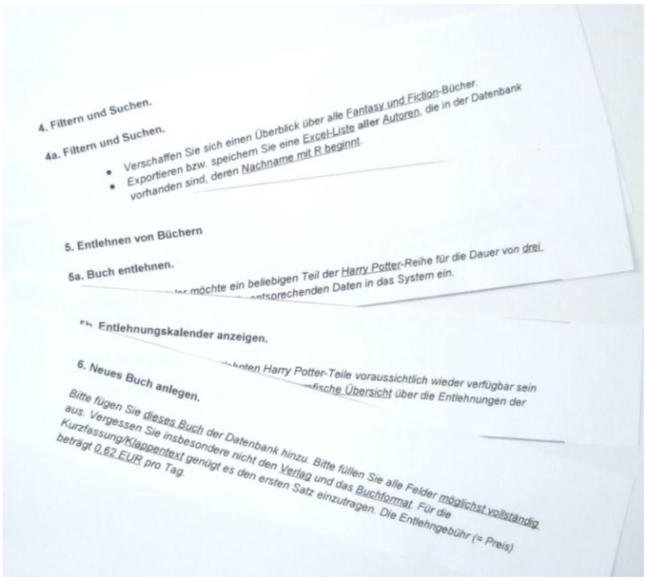
The test procedure for each test was the following:

1. Reception
2. Short explanation what the test is about and explanation of the test procedure.
3. Request for reading the information document.
4. Request for signing the confidentiality agreement and the agreement sheet.
5. Request for filling out the questionnaire about background knowledge.
6. Check for proper computer settings suitable for the test candidate (mouse speed, keyboard position, screen position)

7. Activation of the audio, video and screen recording devices.
8. Practical part of the TA test (execution of the predefined tasks).
9. Interview.
10. Request for filling out the feedback questionnaire.

The duration of one test session was approximately 60 minutes. We mainly followed Andrew’s suggested procedure for performing thinking aloud tests (Andrews, 2011).

Before the practical part of the test started, we asked the test users to fill out a form about their background knowledge related to computer usage and usage of business software (step 5). Then, the predefined tasks (step 8) were handed out as single paper sheets, one for each task. Figure 28 shows a photo of some of the task sheets. The test users had a certain amount of time to execute the task. If the predefined time was exceeded the current task was aborted and the next task sheet was handed out. In the subsequent interview (step 9, Figure 30) the test user was asked some general questions as well as questions about the specific events of his or her test. Finally, the test users were asked to fill out a questionnaire about the tested software and the thinking aloud test itself. All communication between the usability test team and the test users and all interaction with the application were recorded for later analysis (audio, video, screen capturing; see Figure 29).
The usability team created protocols for each test session and analyzed the recordings taken during the tests. All mentioned issues but also all positive feedback was regarded. In a next step the issues were ranked by subjective severity and by the number of people who reported the issue. Finally, a comprehensive report was created; the results were presented and discussed with Boom Software.


4. Filtern und Suchen.

4a. Filtern und Suchen.

- Verschaffen Sie sich einen Überblick über alle Fantasy und Fiction-Bücher.
- Exportieren bzw. speichern Sie eine Excel-Liste **aller** Autoren, die in der Datenbank vorhanden sind, deren Nachname mit R beginnt.

5. Entlehnen von Büchern

5a. Buch entlehnen.

…ler möchte ein beliebigen Teil der Harry Potter-Reihe für die Dauer von drei
…entsprechenden Daten in das System ein.

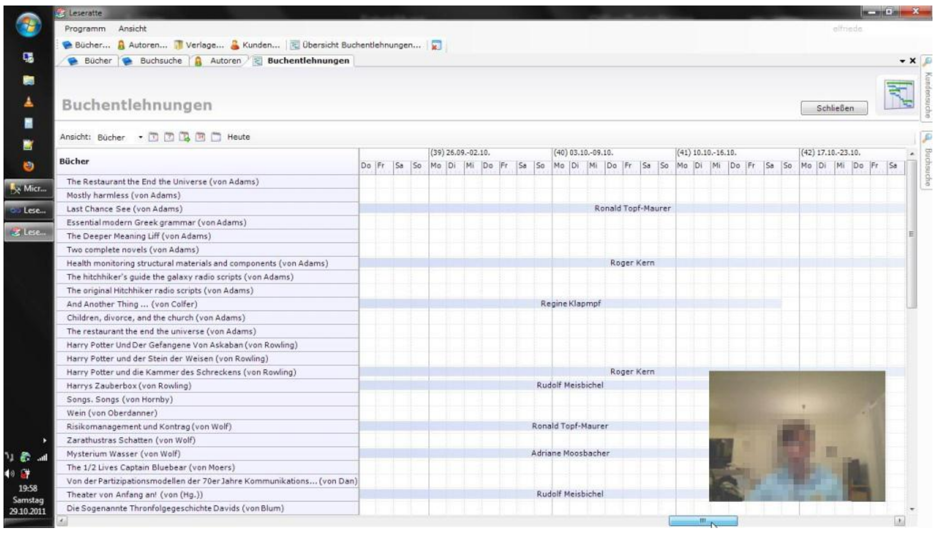
5b. Entlehnungskalender anzeigen.

…lehnten Harry Potter-Teile voraussichtlich wieder verfügbar sein
…fische Übersicht über die Entlehnungen der

6. Neues Buch anlegen.

Bitte fügen Sie dieses Buch der Datenbank hinzu. Bitte füllen Sie alle Felder möglichst vollständig aus. Vergessen Sie insbesondere nicht den Verlag und das Buchformat. Für die Kurzfassung/Klappentext genügt es den ersten Satz einzutragen. Die Entlehngebühr (= Preis) beträgt 0,62 EUR pro Tag.

Figure 28: Task cards for the thinking aloud tests.



Buchentlehnungen

Bücher	(39) 26.09.-02.10. – (42) 17.10.-23.10.
The Restaurant the End the Universe (von Adams)	
Mostly harmless (von Adams)	
Last Chance See (von Adams)	Ronald Topf-Maurer
Essential modern Greek grammar (von Adams)	
The Deeper Meaning Liff (von Adams)	
Two complete novels (von Adams)	
Health monitoring structural materials and components (von Adams)	Roger Kern
The hitchhiker's guide the galaxy radio scripts (von Adams)	
The original Hitchhiker radio scripts (von Adams)	
And Another Thing ... (von Colfer)	Regine Klapmpf
Children, divorce, and the church (von Adams)	
The restaurant the end the universe (von Adams)	
Harry Potter Und Der Gefangene Von Askaban (von Rowling)	
Harry Potter und der Stein der Weisen (von Rowling)	
Harry Potter und die Kammer des Schreckens (von Rowling)	Roger Kern
Harrys Zauberbox (von Rowling)	Rudolf Meisbichel
Songs. Songs (von Hornby)	
Wein (von Oberdanner)	
Risikomanagement und Kontrag (von Wolf)	Ronald Topf-Maurer
Zarathustras Schatten (von Wolf)	
Mysterium Wasser (von Wolf)	Adriane Moosbacher
The 1/2 Lives Captain Bluebear (von Moers)	
Von der Partizipationsmodellen der 70er Jahre Kommunikations... (von Dan)	
Theater von Anfang an! (von (Hg.))	Rudolf Meisbichel
Die Sogenannte Thronfolgegeschichte Davids (von Blum)	

Figure 29: Thinking aloud screen capturing with face recording.

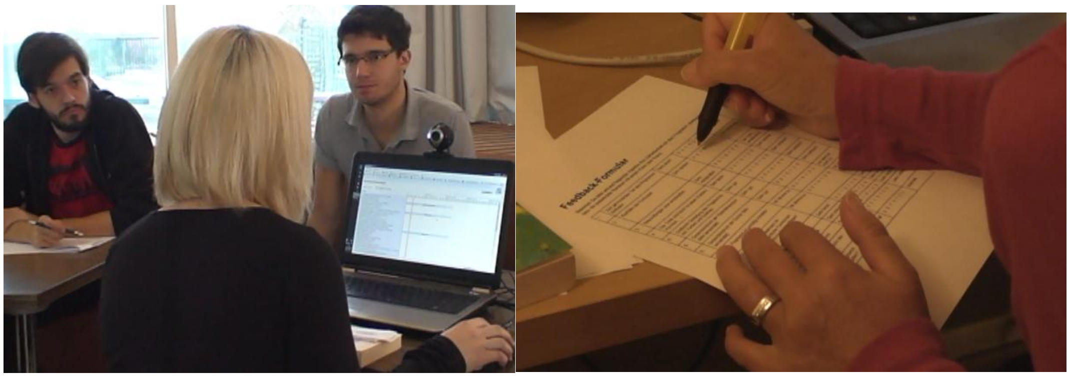


Figure 30: Thinking aloud interviews with test user and the usability team and feedback form.

3.2.7. Usability Guidelines

One goal of the project of evaluating Boom software was to create usability guidelines for current and future developments. The test results of both the heuristic evaluation and the thinking aloud tests were finally discussed with Boom and each reported issue from HE and TA was categorized into one of the following categories: Product specific issue, BORA specific issue and Guideline.

Product specific issues are issues which are directly related to the tested customized product and are only present in the tested version of the product. Fixing this issue in the tested product has no effect on other products.
BORA specific issues are issues of the base framework and therefore affect all Boom applications as all Boom applications build on BORA. Fixing this issue affects all products using the corresponding feature.
Guideline-classified issues are issues which should be included in the usability guidelines for software developers in order to make them aware of the usability problem. The guidelines should be applied by every Boom developer who creates or customizes software based on the BORA framework now or in the future.

The consolidated results from HE and TA and the elaborated guidelines were presented to the Boom developers during a workshop in March 2012. The result of the consolidation of the evaluation results can be found in Section 4.1.4.

3.3. AUIs vs. Non-AUI Experiment

Good usability of applications on mobile devices is important due to relatively small screen sizes and possible operation in harsh environment or in stressful situations. However, the importance of good usability of mobile applications is not higher than for desktop applications. The requirements of mobile usability are simply different. The reasons range from other input methods over different screen sizes up to the high mobility of smartphones or tablets.

Therefore, these special properties must be considered when engineering mobile software. Smart adaptive user interfaces might be one solution for user interfaces in such special conditions. Smart adaptive UIs use the available space more efficiently than conventional user interfaces and display the simplest user interface appropriate for the current context. Therefore, according to the hypothesis stated in the introduction, smart adaptive UIs make the interaction with the end-user more efficient.

An experiment was designed to measure the performance of an adaptive user interface in contrast to a non-adaptive user interface on mobile devices in order to test the hypothesis that that simpler user interfaces created by smart adaption enhance the performance of end users. The adaption is made by regarding the current state of the application, i.e. the previous input. The input button array is adapted by reducing the selection space, i.e. only offering appropriate options for the current context (Schmidt, 2000). The resulting reduction of the maximum number of simultaneously visible buttons makes it possible to make the single buttons larger. A reduction of the needed number buttons is also reached by hiding less frequently used functions.

The keystrokes made by the participant users were recorded and sent to a web service for later remote analysis of the performance of the participants of the experiment. A more detailed description of the test method follows later in this subsection.

The non-adaptive user interface provides a static array of buttons. These buttons are always visible, regardless of the current context. This resembles a classic calculator device with physical buttons.

It is expected that the adaptive user interface is perceived as simpler than the non-adaptive user interface by the users operating the calculator. Further, it is expected that the performance is higher when using the adaptive UI.

For the experiment a smartphone application for the Android operating system (app) was developed. The application is a calculator for basic mathematical expressions. The user interface mainly consists of (a) a TextView (i.e. an area to display text) for displaying the entered mathematical expression at the top of the screen, (b) another TextView for displaying the result and (c) an array of buttons which - when pressed – append the pictured number, operator or function name to the mathematical expression and display the new expression in (a) (Figure 31). If the currently displayed expression is valid (e.g. balanced parenthesis, all binary operators having two operands, and other criteria) the result is calculated and displayed immediately in (b). Otherwise "Invalid Expression" is displayed.

Two different versions of the button array area (c) were implemented: a non-adaptive version and an adaptive version. On first start of the application a random UI is selected. In a message box the user can decide by unchecking a checkbox not to participate in the experiment. In the preferences screen the user can choose to switch the user interface from adaptive to non-adaptive or vice-versa. Also, when using the adaptive user interface the screen can be rotated for using the application in landscape mode. In order to save space in landscape mode the TextViews (a) and (b) are positioned side by side.

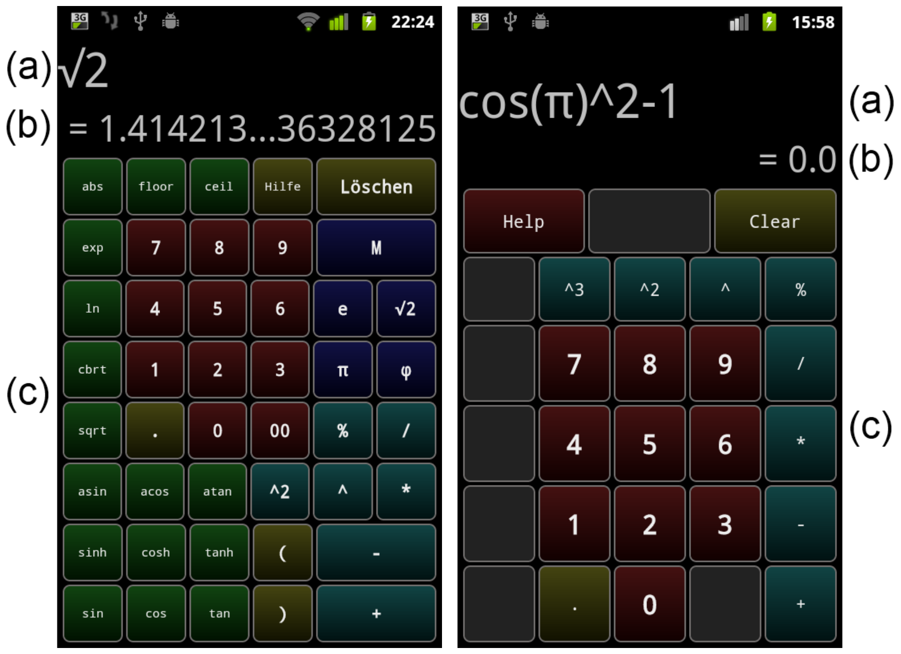


Figure 31: AdaptiveCalc – non-adaptive and adaptive UI

Design and Development. There was a choice to be made when designing the experiment whether to let the user make the decision of choosing the user interface or to disallow the change of the UI. The reasons for the decision to allow the user to change the UI were the following. First, by letting the user choose the user's UI preferences can be found out. If users use a certain UI more often it is likely that the more often used UI is the one which is better accepted by the users. Secondly, as the app should be distributed via the Android Market, the goal was to make the app attractive for as many potential end users as possible. Restricting the functionality to one UI, however, does not support the attractiveness of an application.

Using this model of letting the end-user choose the desired UI might, however, bias the performance measurements. The knowledge of using the adaptive or the non-adaptive user interface might influence the behavior of the end users and lead to unnatural typing behavior.

The goal of the experiment is to find out the favored UI by the end users and to take and to compare performance measures, keeping in mind the biasing issues mentioned above.

The application was developed using an agile software development process, as the effort for creating the app should be kept low. However, for some Android-independent components (classes) test-driven development was applied. Unit tests were written before the implementation of the single methods. The goal was to pass all tests after the implementation was done. The user interface was developed afterwards. Android unit tests were written for testing the proper functionality of the UI. Also a short thinking aloud test with two test users were performed in order to reveal misbehavior and design issues of the user interface before publishing the app on the Android Market (see later).

For evaluating the mathematical expressions the open-source library de.congrace.exp4j[29] was used. In order to improve the precision of the mathematical operations Java's BigDecimal type was used instead of double, which is used in the default implementation of exp4j. According to the project's website the library uses the shunting-yard algorithm for parsing mathematical expressions.

[29] exp4j project page: *http://projects.congrace.de/exp4j/*

For network communication Apache’s HttpComponents Client library was used. Network connection is needed for sending the recorded keystrokes to the web service (see later).
The user interface basically consists of two TextViews (a) and (b) (see above). TextView (a) was put inside a HorizontalScrollView. The array of buttons was arranged using a TableLayout and TableRows.

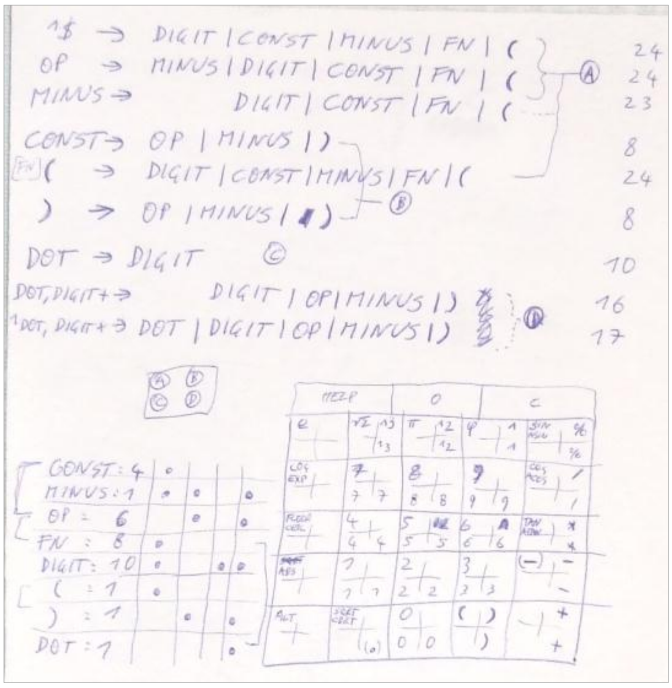
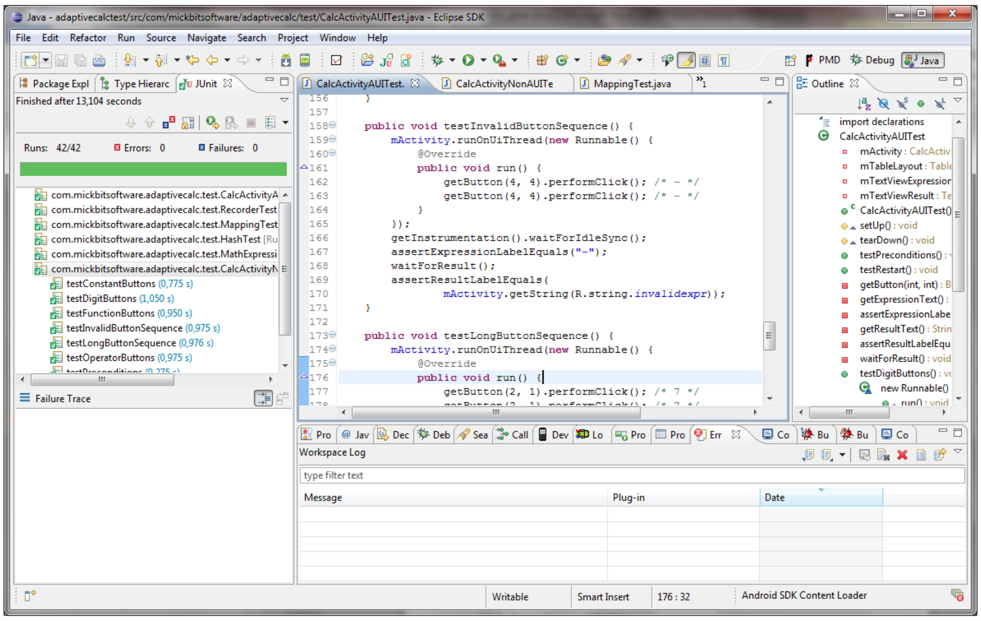

^$ → DIGIT | CONST | MINUS | FN | (24
OP → MINUS | DIGIT | CONST | FN | (24
MINUS → DIGIT | CONST | FN | (23
CONST → OP | MINUS |) 8
(→ DIGIT | CONST | MINUS | FN | (24
) → OP | MINUS |) 8
DOT → DIGIT 10
DOT, DIGIT+ → DIGIT | OP | MINUS |) 16
^DOT, DIGIT+ → DOT | DIGIT | OP | MINUS |) 17
A
B
C
D
CONST: 4
MINUS: 1
OP: 6
FN: 8
DIGIT: 10
(: 1
) : 1
DOT: 1
HELP
0
C

Figure 32: Design of the adaptive button array on paper

Figure 32 shows the first design of the adaptive user interface on paper. It also shows the context-related rules for displaying the single buttons. The rules are of the form

```
<previous symbol(s)> → <possible next symbol> (\|<possible next symbol>)*
```

where the term *symbol* corresponds to digits, operators, function names, parenthesis, or dots. The numbers next to the single rules in Figure 32 refer to the required number of visible buttons for the corresponding button configuration.
The app was developed using the Eclipse IDE (Figure 33) and Google’s Android SDK. JUnit and Android’s JUnit extensions were used to implement the unit tests.


Java - adaptivecalctest/src/com/mickbitsoftware/adaptivecalc/test/CalcActivityAUITest.java - Eclipse SDK
File Edit Refactor Run Source Navigate Search Project Window Help
PMD Debug Java
Package Expl Type Hierarc JUnit
CalcActivityAUITest. CalcActivityNonAUITe MappingTest.java
Outline
Failure Trace
Pro Jav Dec Deb Sea Call Dev Lo Pro Pro Err Co Bu Bu Co
Workspace Log
type filter text
Message Plug-in Date
Writable Smart Insert 176 : 32 Android SDK Content Loader

```
Finished after 13,104 seconds

Runs: 42/42    Errors: 0    Failures: 0

com.mickbitsoftware.adaptivecalc.test.CalcActivityA
com.mickbitsoftware.adaptivecalc.test.RecorderTest
com.mickbitsoftware.adaptivecalc.test.MappingTest
com.mickbitsoftware.adaptivecalc.test.HashTest [Ru
com.mickbitsoftware.adaptivecalc.test.MathExpressi
com.mickbitsoftware.adaptivecalc.test.CalcActivityN
    testConstantButtons (0,775 s)
    testDigitButtons (1,050 s)
    testFunctionButtons (0,950 s)
    testInvalidButtonSequence (0,975 s)
    testLongButtonSequence (0,976 s)
    testOperatorButtons (0,975 s)
```

```java
    }

    public void testInvalidButtonSequence() {
        mActivity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                getButton(4, 4).performClick(); /* - */
                getButton(4, 4).performClick(); /* - */
            }
        });
        getInstrumentation().waitForIdleSync();
        assertExpressionLabelEquals("-");
        waitForResult();
        assertResultLabelEquals(
                mActivity.getString(R.string.invalidexpr));
    }

    public void testLongButtonSequence() {
        mActivity.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                getButton(2, 1).performClick(); /* 7 */
```

```
import declarations
CalcActivityAUITest
    mActivity : CalcActiv
    mTableLayout : Table
    mTextViewExpression
    mTextViewResult : Te
    CalcActivityAUITest()
    setUp() : void
    tearDown() : void
    testPreconditions() :
    testRestart() : void
    getButton(int, int) : B
    getExpressionText() :
    assertExpressionLabe
    getResultText() : Strin
    assertResultLabelEqu
    waitForResult() : void
    testDigitButtons() : vo
        new Runnable()
```

Figure 33: AdaptiveCalc development using Eclipse

Test users. As stated before, the application was published on the Android Market[30]. Anyone who was interested could download and use the application. Measurements were taken automatically during normal operation of the calculator. These measurements were sent to a web service which stores the collected results for subsequent analysis. End users are informed about the background measurement activities via the app description and via an AlertBox on first start and can disable the measurements without affecting the features of the calculator.

Collecting results. The Android application records the keystrokes made by the users, if not deactivated. The records are sent via HTTP POST request to a webserver where the records are saved for later analysis. For collecting and evaluating the test data a server-side script was written in PHP. The PHP script receives and stores the data received from the app in a mysql database. The script also creates a HTML report from the stored records when called with the corresponding GET parameter. The report contains values calculated from the single records, such as the average time needed between key presses and the average number of times the clear button was pressed per

[30] AdaptiveCalc on the Android Market:
https://market.android.com/details?id=com.mickbitsoftware.adaptivecalc (published in February 2012)

calculation. For more information about the measured values and about the results of the experiment, please refer to Section 4.2.

The following measurements are taken during operation of the calculator: Pressed buttons, time between button presses, selected mode (adaptive or non-adaptive), and result of the calculation. A string of the form

```
<uimode>(;<ms>:<button>)+;<ms>;<result>
```

is recorded for each session. A session starts when bringing the calculator activity into view or when starting a new calculation (i.e. a new "session" or expression) and ends when the activity is left by the user or when the entered expression is cleared. <uimode> is a placeholder for a string representing the current user interface mode ("auiport" (AUI portrait), "auiland" (AUI landscape), or "nonaui" (non-AUI)), <button> is a string representing the pressed button (for example "1" or "+") and <ms> is the timestamp of the key press. The last <ms> represents the time when clearing the expression and therefore resetting the display and ending the session. <result> is either the string "ok" or "nan" (not a number) and represents the result of the corresponding calculation.

After a session has ended, the recorded session measurement string is sent to the web service if a network connection is available. Otherwise, the string is stored in a local database on the device for later transmission. The next time the app tries to send a session measurement string to the web service it also checks the local database for previously unsent data and also sends all previously unsent data, if possible.

Within a time period of approximately one month 408 single calculations (sessions) were recorded. For an analysis of the results all 408 single session strings (records) were separated into four groups. The first group only contains records from the adaptive user interface (portrait) version; the second group only contains records from the adaptive user interface (landscape) version; the third group only contains records from the non-adaptive user interface version; the fourth group contains invalid records.

Then, certain values were calculated from the collected records in order to evaluate the end-user's performance and acceptance of the user interface types.

As performance measures the average time between button presses was calculated for the single user interface types as well as the error rate. For evaluating the user acceptance interviews were conducted and the number of calculations made with each user interface was counted. To be more precise, for measuring the users' performance, the median time between button presses was calculated first for each session; then, the average of the medians was calculated for each of the groups AUI (portrait), AUI (landscape), and non-AUI. Using the median has the advantage that spike values are flattened. Spike values might result from thinking times or waiting times of the end users. These times must not be considered in the performance evaluation. The time between single keystrokes is used as performance measurement because the single tasks and therefore the total time needed for one session differs as there were no predefined tasks to accomplish for the end users. A more detailed explanation and the experiment's results are presented in Section 4.2.

3.4. HE Checklist for Mobile Devices

As we saw during the HE of Boom software, heuristic evaluation can be a powerful instrument for detecting usability flaws. However, heuristics and especially detailed checklists tailored to current smartphones or tablets are rare. The aim of this part of the work was to fill this gap and to provide a checklist which can be used as basis for future extension to create a more comprehensive set of rules.

As stated in Section 2.3.4, usability heuristics are principles or rules for user interface design. There exist relatively vague rule sets but also very detailed and concrete rule sets. Vague rule sets are applicable in many different situations and for different systems. Very concrete rules, however, might not be applicable in certain situations or for certain scenarios. But more detailed rules might be more beneficial for usability testers because the hints what issues to check for are more concrete. Nielsen's ten main usability principles, for example, are applicable to many different types of user interfaces. Detailed checklists, in contrast, must be tailored to certain types of systems, such as enterprise desktop systems or mobile applications. However, also within detailed checklists certain items might be applicable for different systems.

Most of the existing usability heuristics concentrate on graphical or even textual desktop systems where users typically use mouse and keyboard for interacting with the system. Recent heuristics regard mobile devices such as classic mobile phones. The newest generation of mobile devices combines mobility with relatively large finger-operated (multi) touchscreens and several more hardware features such as GPS sensors, cameras or RFID readers.

The goal in this part of the work is not to develop general usability heuristics for mobile applications but a detailed checklist for modern smartphone and tablet applications, similar to the rule set for text-bases desktop applications by Pierotti (2000). Therefore, existing checklists and guidelines for desktop computers were revised and end-user feedback of different mobile apps was considered. Applicable checklist items based on heuristics used for the work with Boom Software AG were included as well as rules from other sources, such as from Microsoft’s, Apple’s, and Google’s usability guidelines. The developer websites of all major companies which provide OS for mobile systems such as smartphones, provide usability guidelines for software developers. However, these guidelines are either very general or very system-specific – the degree of detail differs greatly between the single guidelines. Also, the foci of the guidelines differ. The following list mentions some of the sources for usability or user experience guidelines by different companies.

- Windows Mobile 6.5: Design Guidelines: *http://msdn.microsoft.com/en-us/library/bb158602.aspx*
- Windows Phone 7: User Experience Design Guidelines for Windows Phone: *http://msdn.microsoft.com/en-us/library/hh202915%28v=VS.92%29.aspx*
- Android: *http://developer.android.com/guide/practices/ui_guidelines/index.html*
- Apple iOS: iOS Human Interface Guidelines: *http://developer.apple.com /library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introducti on/Introduction.html*
- Nokia: Guidelines for Mobile Interface Design: *http://www.developer.nokia.com /Community/Wiki/Guidelines_for_Mobile_Interface_Design*

The consolidated usability checklist for mobile applications created during this work can be found in the Appendix A. Section 4.3 picks out some checklist items in order to briefly discuss them.

4. Results

This section describes the results of the work described in Section 3. This includes the work with Boom Software AG concerning the Android mobile client as well as the usability evaluation of the desktop software (HE and TA). Also the resulting guidelines are briefly discussed. Additionally, the results of the experiment about adaptive user interfaces (AdaptiveCalc) introduced in Section 3.3 are presented and a short report about the created usability checklist for mobile applications introduced in Section 3.4 is made.

4.1. Usability Consultancy

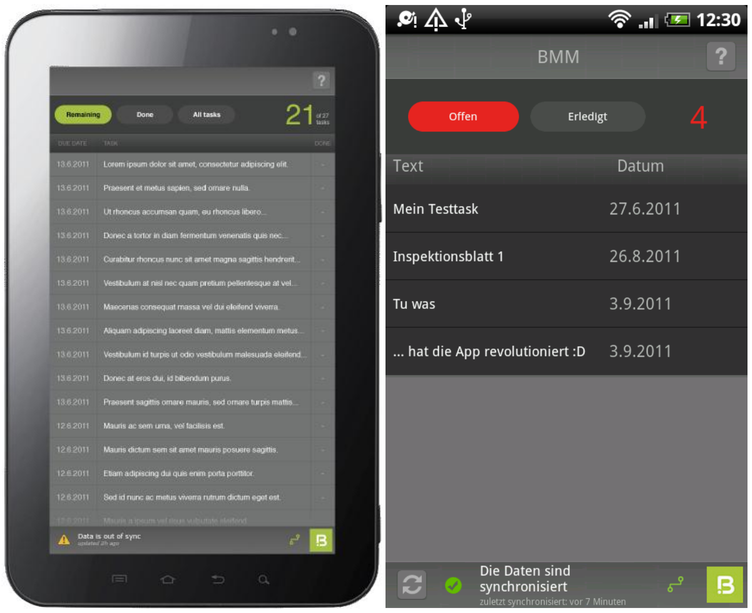
This section presents the outcome of the usability consultancy activities and the evaluation of Boom software. First, in Section 4.1.1, the latest prototype of the mobile application is reviewed and suggested improvements are presented. Secondly, in Sections 4.1.2 to 4.1.4, the results of the usability evaluation of Boom's desktop software are presented, as well as Boom's implementation of usability guidelines.

4.1.1. Mobile App

During a meeting about the mobile application for task sheets (see Section 3.2.1) potential usability issues were discussed. The latest version of the app at that time (August 2011) was reviewed.

The issues discussed include the size of the list entries of the task list, the display of the status bar and issues related to screen rotation. In the following paragraphs the found issues and the implementation of the fixes are discussed.

In the first versions of the app the UI was not properly adapted in landscape orientation. In our feedback we reported that in landscape mode the app should adjust the layout in a way that space is used more efficiently. The latest, improved version of the app hides the filter buttons at the top of the screen (captions "*Offen*" (*open*) and "*Erledigt*" (*done*)) and shows an additional column on the left instead (Figure 35). The column indicates the state of the task according to the two possibilities *open* and *done*.



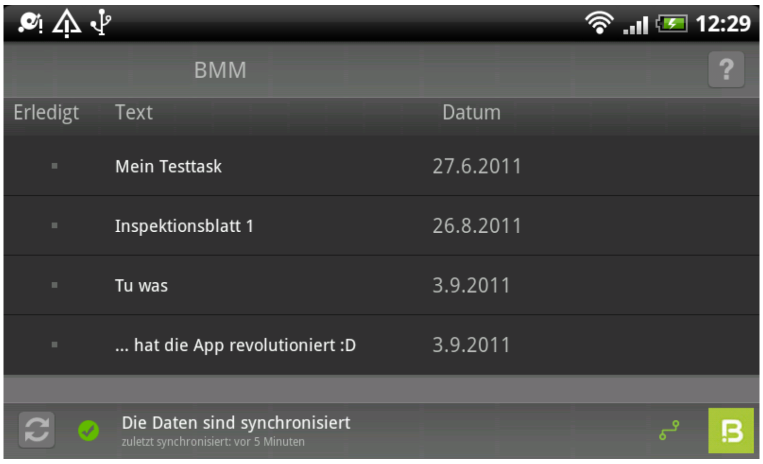
DUE DATE	TASK	DONE
13.6.2011	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	-
13.6.2011	Praesent et metus sapien, sed ornare nulla.	-
13.6.2011	Ut rhoncus accumsan quam, eu rhoncus libero...	-
13.6.2011	Donec a tortor in diam fermentum venenatis quis nec...	-
13.6.2011	Curabitur rhoncus nunc sit amet magna sagittis hendrerit...	-
13.6.2011	Vestibulum at nisl nec quam pretium pellentesque at vel...	-
13.6.2011	Maecenas consequat massa vel dui eleifend viverra.	-
13.6.2011	Aliquam adipiscing laoreet diam, mattis elementum metus...	-
13.6.2011	Vestibulum id turpis ut odio vestibulum malesuada eleifend...	-
13.6.2011	Donec at eros dui, id bibendum purus.	-
13.6.2011	Praesent sagittis ornare mauris, sed ornare turpis mattis...	-
12.6.2011	Mauris ac sem urna, vel facilisis est.	-
12.6.2011	Mauris dictum sem sit amet mauris posuere sagittis.	-
12.6.2011	Etiam adipiscing dui quis enim porta porttitor.	-
12.6.2011	Sed id nunc ac metus viverra rutrum dictum eget est.	-

Data is out of sync
updated 2h ago

Text	Datum
Mein Testtask	27.6.2011
Inspektionsblatt 1	26.8.2011
Tu was	3.9.2011
... hat die App revolutioniert :D	3.9.2011

Die Daten sind synchronisiert
zuletzt synchronisiert: vor 7 Minuten

Figure 34: Task list - Mockup and screenshot of latest app version



Erledigt	Text	Datum
▪	Mein Testtask	27.6.2011
▪	Inspektionsblatt 1	26.8.2011
▪	Tu was	3.9.2011
▪	... hat die App revolutioniert :D	3.9.2011

Die Daten sind synchronisiert
zuletzt synchronisiert: vor 5 Minuten

Figure 35: Task list in landscape mode (screenshot of latest app version)

Also, the proper size of the list entries was discussed. Originally, the height of the list entries was low. During the discussion it was figured out that only few list entries are expected to be shown in the list at the same time. Therefore, we suggested to make the list entries larger in order to use more of the free space on the one hand and to make the list entries easier touchable on the other hand (Figure 34, Figure 35).

Figure 36 shows a form created dynamically by the XML-based UI description. However, the form as shown in the figure was created for testing purposes only. There is a guideline rule which disallows nesting of captioned sections (see Section 4.1.4). This rule is intentionally violated in the screenshot for testing purposes.

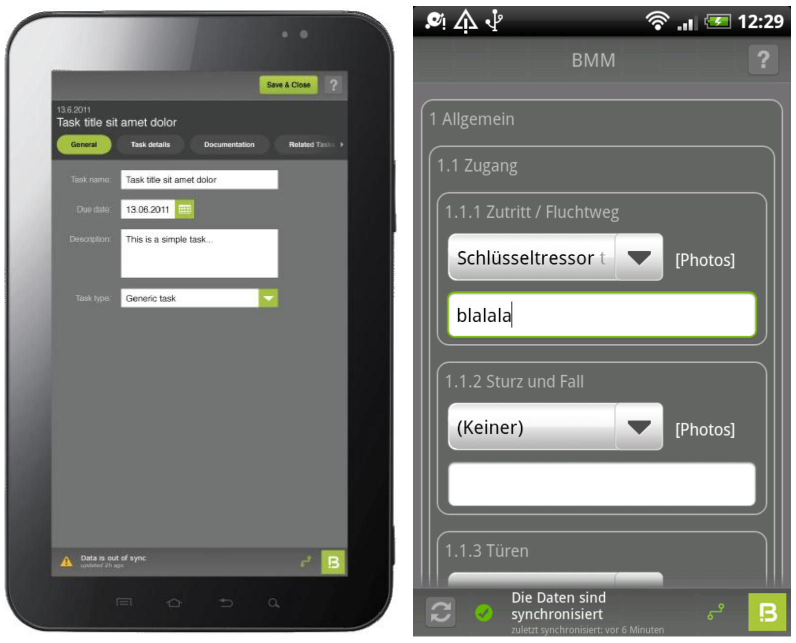


Figure 36: Task form - Mockup and latest version screenshot

Further, it was discussed whether to show Android's status bar or not. The advantage of showing the status bar is that notifications can be seen during operation of the software. A disadvantage is that the status bar needs some space on the screen. In the case of a business application where e-mails or messages might be received frequently we came to the conclusion that it is feasible to show the status bar.

During operation of the software we also discovered a technical error message which did not provide information for solving the problem. Similar issues were also reported while testing Boom's desktop software, as the reader will see later. Therefore we suggested that such error messages are replaced with error messages which do provide suggestions for solving the problem on the one hand and do blame the system wherever possible – not the user. This rule is based on heuristics from different sources about error handling, for example from Andrews (2006).

4.1.2. Results of HE

The usability of the desktop applications *Leseratte*, Boom Maintenance Manager (BMM, native Windows client and web client) and Target Manager (TM, native Windows client) was reviewed during the heuristic evaluation of Boom software. In this section the results of the evaluation are described. As a complete report of the results would go beyond the scope of this work, some exemplary results are presented at this point. The complete, approximately 100 pages report consisting of several single documents (checklists and detailed reports for each application and a summary) was provided to Boom Software AG.

The overall usability of the tested applications was good. However, there are several minor issues and some issues concerning the simplicity and the comfort in certain situations. The most problematic applications were *Leseratte* and the BMM web client. The issues in *Leseratte* can be explained with certain inconsistencies in the tutorial but also with the fact that the tutorial should not be too complicated for beginners. Ignoring the *Leseratte* and tutorial-specific issues and focusing on the features provided by the framework instead, only very few issues can be reported due to the simplicity of the application.

The BMM web client has issues with missing feedback and error messages as well with the general complex layout of combined tree and tab navigation. Due to the complexity of the problem to solve with the software, a certain complexity of the software must be accepted. However, there were certain inconsistencies found which could be removed and therefore simplify the handling.

BMM windows client handled the complexity very well and was easy operable. TM also provided good overall usability, however, there were certain issues with waiting times without feedback, unclear controls and some other minor limitations.

Positive points were the good overall consistency and the nice look and feel, flexible window arrangement mechanisms, in general good forms and error handling. Nice details such as direct opening of detail pages in case of only one single search result or drag & drop support in certain screens emphasize the positive impression. Some more negative points include technical error messages at some places (users are not interested in too much information or even confused), error messages which do not give hints how to solve the problem, and sometimes unclear icons and buttons which are in unexpected locations on certain screens.

Figure 37 and Figure 38 illustrate two more problems found during the heuristic evaluation. All found issues were reported as illustrated as shown in Figure 39. Every single issue got a unique identifier and a title as well as a description. The location of the issue was recorded or, if necessary how to reproduce the error was explained. The heuristics that led to the identification of the issue were reported and in many cases a screenshot was attached to the corresponding issue.

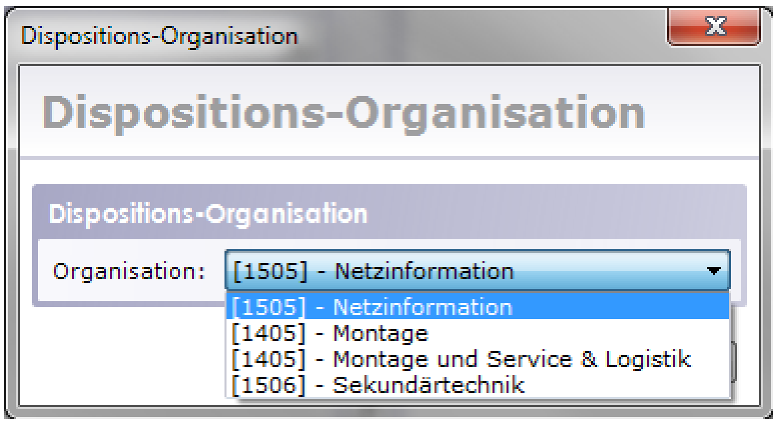

Dispositions-Organisation
Dispositions-Organisation
Dispositions-Organisation
Organisation:
[1505] - Netzinformation
[1505] - Netzinformation
[1405] - Montage
[1405] - Montage und Service & Logistik
[1506] - Sekundärtechnik

Figure 37: Issue found via HE: Dropdown box for only four items. Better: Radio buttons.

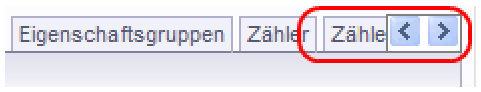
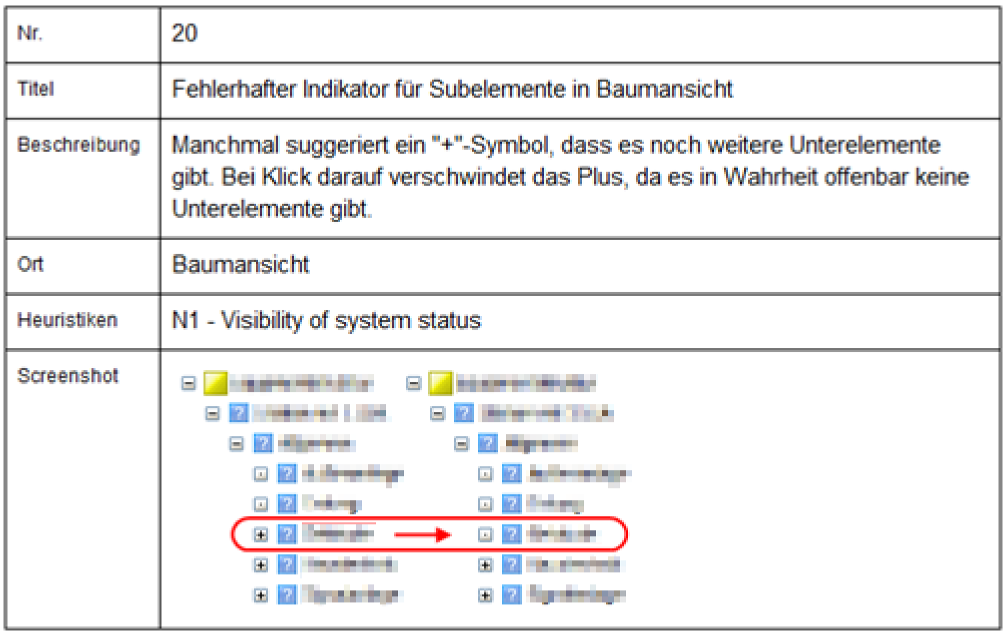


Figure 38: Issue found via HE: Too many tabs and last tab not visible.



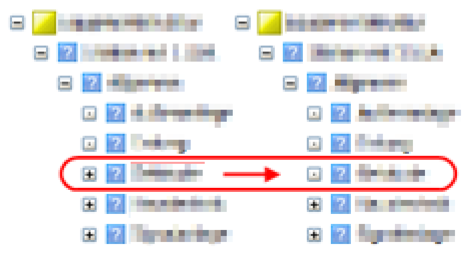
Nr.	20
Titel	Fehlerhafter Indikator für Subelemente in Baumansicht
Beschreibung	Manchmal suggeriert ein "+"-Symbol, dass es noch weitere Unterelemente gibt. Bei Klick darauf verschwindet das Plus, da es in Wahrheit offenbar keine Unterelemente gibt.
Ort	Baumansicht
Heuristiken	N1 - Visibility of system status
Screenshot	

Figure 39: HE – Example for one of the issues in the report for BMM

All found issues were finally ordered by subjective severity. The severity rating was calculated by the average rating of the independent ratings of the usability test team. The severity rating ranged from zero to five, five representing the highest severity.

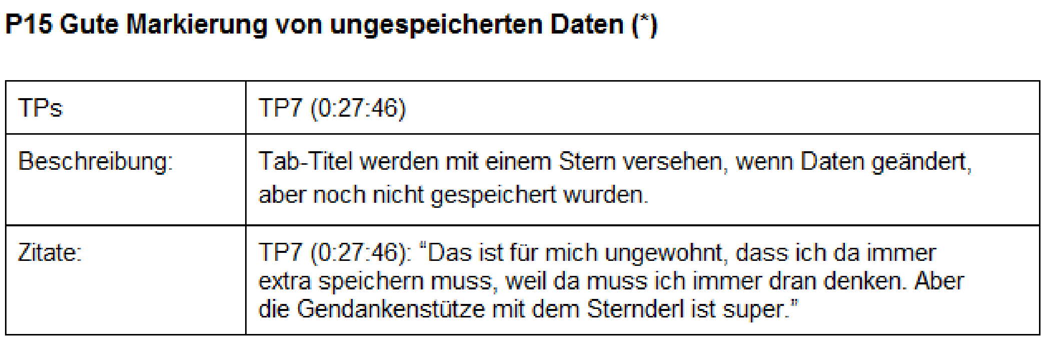
However, the issues we reported should also be evaluated by Boom and their customers in order to figure out the concrete relevance for Boom and their customers.

4.1.3. Results of TA

As for the HE report, more than 100 pages were created during the TA tests (50 pages final report plus protocols, filled out forms and instruction documents for the test users). Discussing every aspect in detail would go far beyond the scope of this work, therefore only some exemplary results are presented at this point. The complete report was provided to Boom Software AG and all collected material was shared.

The overall comments of the test users were positive. Especially the simplicity and the simple navigation were honored by the test users. The modern look-and-feel and the possibility of detailed searches got positive feedback as well. Issues were reported in connection with filter functions of lists and tables. Also the sometimes cumbersome workflow when adding new data was mentioned in a negative context. Some potential helpful functions were not recognized by some users. Other functions were viewed as not particularly useful but at least “nice to look at”.

Figure 40 and Figure 41 show examples of test user feedbacks and how they were reported in the report.



P15 Gute Markierung von ungespeicherten Daten (*)

TPs	TP7 (0:27:46)
Beschreibung:	Tab-Titel werden mit einem Stern versehen, wenn Daten geändert, aber noch nicht gespeichert wurden.
Zitate:	TP7 (0:27:46): “Das ist für mich ungewohnt, dass ich da immer extra speichern muss, weil da muss ich immer dran denken. Aber die Gendankenstütze mit dem Sternderl ist super.”

Figure 40: Thinking aloud test - example of positive feedback.

N19 Zurücksetzen-Button bei der Suche schwer zu erkennen

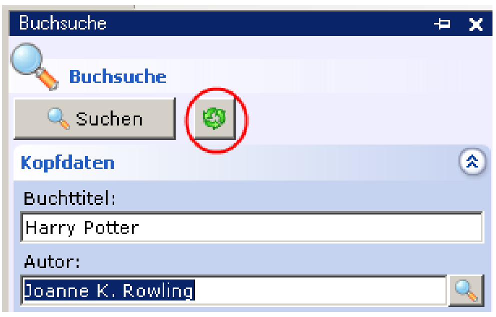
Severity:	2
TPs	TP2 (0:13:16), TP3 (0:16:31), TP4 (0:09:09), TP6 (0:10:29), TP8 (0:11:18)
Beschreibung:	Der Zurücksetzen-Button wurde von den meisten Testusern nicht erkannt. In ein paar Fällen wurde die Bedeutung des Buttons erst durch den Mouse-over Tooltip erkannt, in den meisten jedoch gar nicht - Die gesetzten Felder wurden einzeln geändert oder gelöscht.
Zitate:	TP2 (0:13:16): "Da muss man alles wieder zurücksetzen" (Feldinhalte wurden einzeln gelöscht) TP2 (Interview): "Der Zurücksetzen-Button sieht eher aus wie "Nochmal Suchen" oder so."
Screenshot:	 Buchttitel: Harry Potter Autor: Joanne K. Rowling

Figure 41: Thinking aloud test – example of negative feedback.

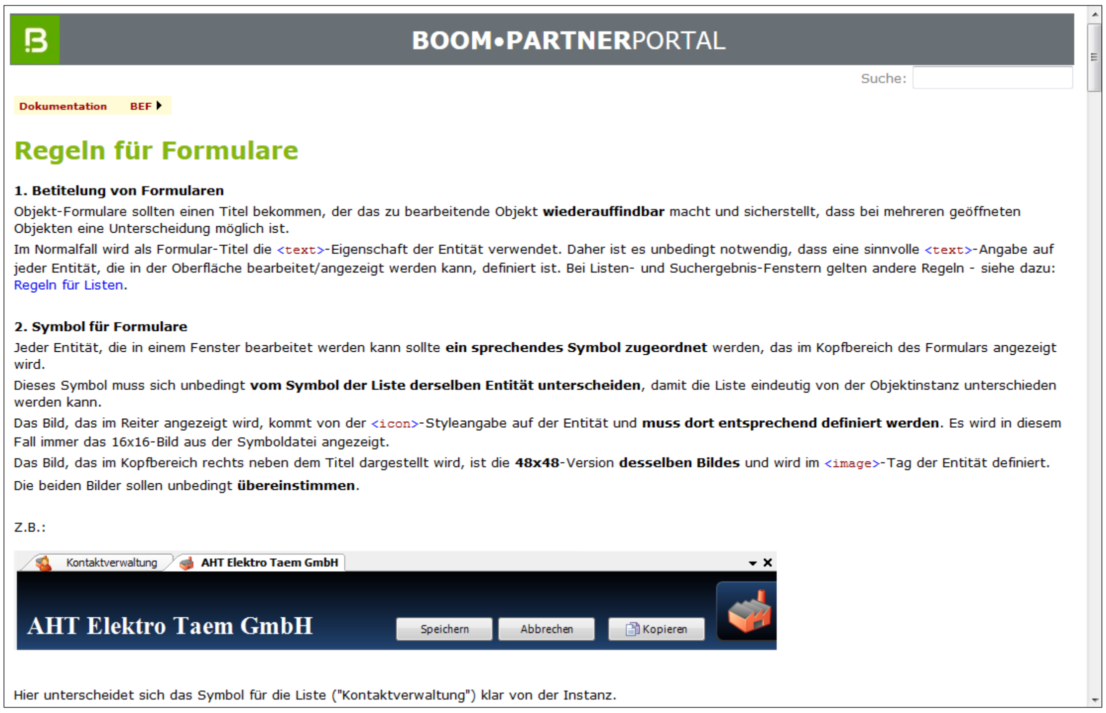
All found issues were rated by a subjective severity rating. The rating ranging from zero to four, where zero means no issue and four means big issue, was made by the usability test team independently. Finally an average value was calculated. The severity ratings were not only influenced by the personal opinion but also by the opinion perceived from the test users. Also the number of people who reported an issue was counted. All issues were finally ranked by the average severity rating and the number of reporting test users. For further analysis of the sessions each test session was protocolled and analyzed later by the usability experts.

All in all 15 distinct positive impressions, 41 negative impressions, 17 recommendations, and 6 remarks were collected. Although there were more negative impressions than positive impressions, it doesn't mean that the application has bad usability. As we saw during the thinking aloud tests, people are way more likely to mention negative impressions than positive ones. This is also shown by the fact that users answered the question “Would you use the program for yourself?” with a clear “yes” (1.56 of 7, where 1 means “yes” and 7 means “no”).

4.1.4. Implementation and Guidelines

In the last step during the usability evaluation of Boom’s desktop software all reported issues from HE and TA were put together and analyzed with the goal of improving the software by implementing certain changes immediately on the one hand, and by creating usability guidelines for Boom’s software developers on the other hand. Based on a cost estimate by Boom and based on the severity ratings of the usability team the found issues were ordered from “big issue and easy to solve” to “small issue and hard to solve”. Additionally, the issues were categorized into three groups: Product specific, framework specific, and guideline.

Based on the results of our research, based on the collected material about usability, and based on the results of the usability evaluations Boom’s web documentation for BORA was extended with the chapter *UI-Design Guidelines* and therefore included in the BORA standard. This new chapter consists of several subchapters, such as *Common UI Principles* and *UI-Design using BORA*. The latter includes subchapters for with rules for designing forms, lists, menus and common design rules. Also a checklist was added for validating the standard-conformity of the created UI. Figure 42 shows a part of the chapter *Rules for Forms*, which contains, amongst many others, rules such as “do not use nested captioned sections in forms” or rules about the proper usage of form titles and icon design.



Regeln für Formulare

1. Betitelung von Formularen

Objekt-Formulare sollten einen Titel bekommen, der das zu bearbeitende Objekt **wiederauffindbar** macht und sicherstellt, dass bei mehreren geöffneten Objekten eine Unterscheidung möglich ist.

Im Normalfall wird als Formular-Titel die `<text>`-Eigenschaft der Entität verwendet. Daher ist es unbedingt notwendig, dass eine sinnvolle `<text>`-Angabe auf jeder Entität, die in der Oberfläche bearbeitet/angezeigt werden kann, definiert ist. Bei Listen- und Suchergebnis-Fenstern gelten andere Regeln - siehe dazu: Regeln für Listen.

2. Symbol für Formulare

Jeder Entität, die in einem Fenster bearbeitet werden kann sollte **ein sprechendes Symbol zugeordnet** werden, das im Kopfbereich des Formulars angezeigt wird.

Dieses Symbol muss sich unbedingt **vom Symbol der Liste derselben Entität unterscheiden**, damit die Liste eindeutig von der Objektinstanz unterschieden werden kann.

Das Bild, das im Reiter angezeigt wird, kommt von der `<icon>`-Styleangabe auf der Entität und **muss dort entsprechend definiert werden**. Es wird in diesem Fall immer das 16x16-Bild aus der Symboldatei angezeigt.

Das Bild, das im Kopfbereich rechts neben dem Titel dargestellt wird, ist die **48x48**-Version **desselben Bildes** und wird im `<image>`-Tag der Entität definiert.

Die beiden Bilder sollen unbedingt **übereinstimmen**.

Z.B.:

Hier unterscheidet sich das Symbol für die Liste ("Kontaktverwaltung") klar von der Instanz.

Figure 42: Web page of Boom's new usability documentation – rules for forms

According to Boom, BORA-specific issues are already partly solved based on our feedback. In the near future Boom aims for implementing the remaining issues which were sufficiently high prioritized.

Many common rules elaborated for Boom's desktop software are also applicable for mobile software, for example for tablet PCs. However, in the future the rules must be extended to cover the special properties of mobile devices. The usability checklist for mobile applications described within this work (Section 4.3) is a first step to this goal.

4.2. AUIs vs. Non-AUI Experiment

This section covers the presentation of the results of the experiment about adaptive user interfaces (AdaptiveCalc) described in Section 3.3. The experiment was about a performance comparison between an adaptive user interface and a non-adaptive user interface. The aim of the experiment was to support the hypothesis that the performance on simple adaptive user interfaces is higher than on complex user interfaces providing no adaption. A calculator application with two different user interfaces was developed and published in the Android market for being able to reach many users within a short period of time.

Both the user acceptance and the performance of each user interface were evaluated during the experiment. The acceptance was evaluated by interviews and by recording the number of calculations made with each of the user interfaces. The performance was measured by calculating the average time between button presses (TBBP) on the one hand, and by calculating the error rate by counting the number of clear button presses on the other hand.

Figure 43 and Figure 44 show the cumulated values Average of the medians of the TBBPs and Error rate for the AUI (portrait) and the non-AUI. The figures illustrate that after about 240 calculations both values stabilized around the presented values (i.e. the cumulated values after 408 calculations).

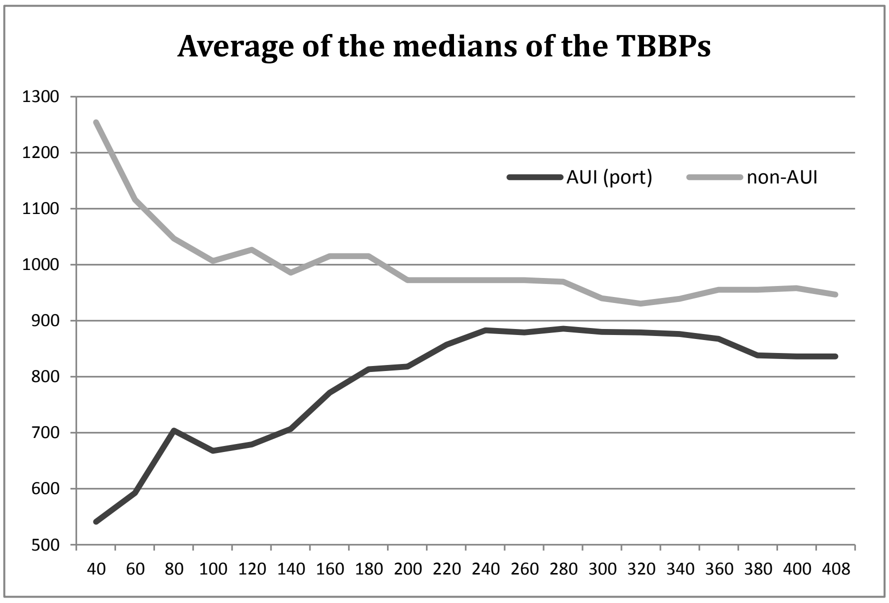


Figure 43: Time between button presses.

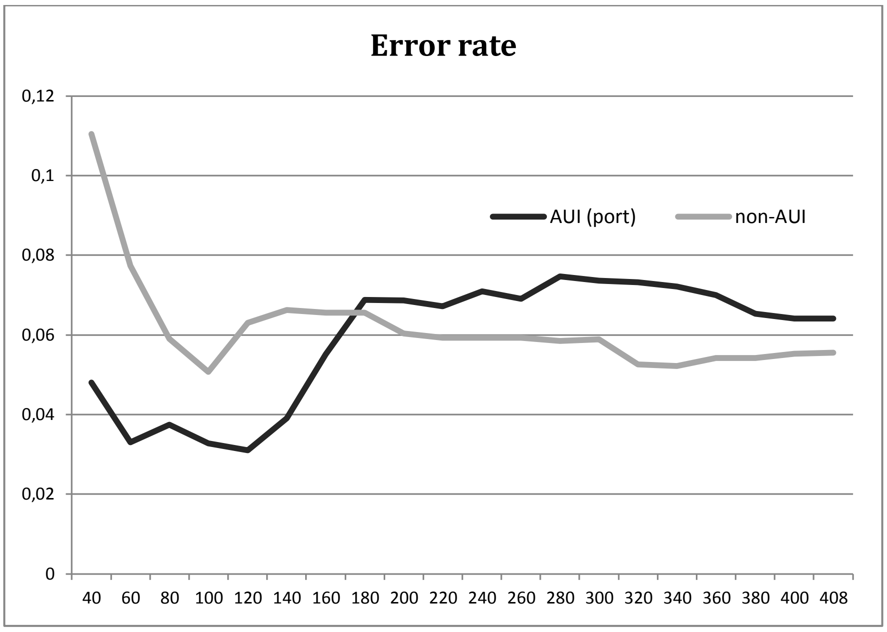


Figure 44: Error rate (clear presses / total number of button presses).

User acceptance. Interviews with six test users were conducted for gaining information about the strengths and the weaknesses of the two user interface types. Before interviewing the six test users, three were asked to do several calculations with the non-adaptive UI, the other three test users were asked to do the same calculations with the adaptive UI (Figure 45). When finished, the UI was switched and the users were asked to do several more calculations with the other user interface. The predefined calculations included simple summations as well as calculations using functions.

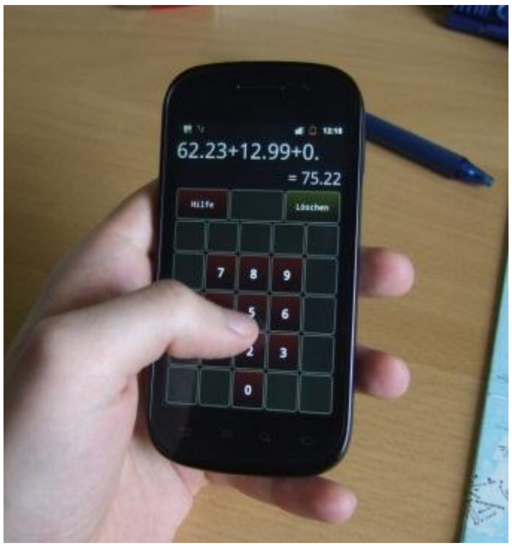


Figure 45: Test user calculating with AdaptiveCalc.

During the interviews the users were asked to describe which user interface they liked more and why. Four users reported to prefer the AUI, one user reported to prefer the non-AUI, and one did not decide for one certain UI. Table 12 summarizes the thoughts of the users. In the table, a plus sign refers to a positive impression while a minus sign refers to a negative impression. The column "#users" contains the number of test users who gave the according feedback.

Relates to (+/-)	Feedback	#users
AUI (+)	The buttons are larger	6
AUI (+)	I like that you only see what is currently relevant.	4
AUI (-)	It is confusing that the buttons (dis)appear.	2
Non-AUI (+)	All buttons are always visible – it is clear what functions are available.	2
Non-AUI (-)	Buttons are quite small.	2

Table 12: Summary of the user's answers during the interview

Number of calculations. During the test period 408 calculations were reported to the server. 198 of the counted calculations were made with the AUI in portrait mode (8 in landscape mode) while 133 calculations were made with the non-AUI (only portrait mode possible). 69 calculations were filtered out because of very short and/or invalid calculations (for example when users only entered an opening parenthesis or only one number). These numbers are illustrated in Figure 46. Including the invalid calculations 249 sessions were started in AUI (portrait) mode (61 %), 9 in AUI (landscape) mode (2.2 %), and 150 in non-AUI mode (36.8 %). The 408 calculations were made by approximately 25 different users. The number of users can only be estimated because no user data was recorded. The estimate is based on the number of personally contacted test users and on the number of installations reported by Android's publishing service.

As on the first run of the application the user interface was selected randomly with an equally distributed likelihood, it can either be concluded that

- users tend to switch to and stay in the AUI mode or
- that the users who started with the AUI had significantly more to calculate or simply preferred calculating with AdaptiveCalc than with another calculator.

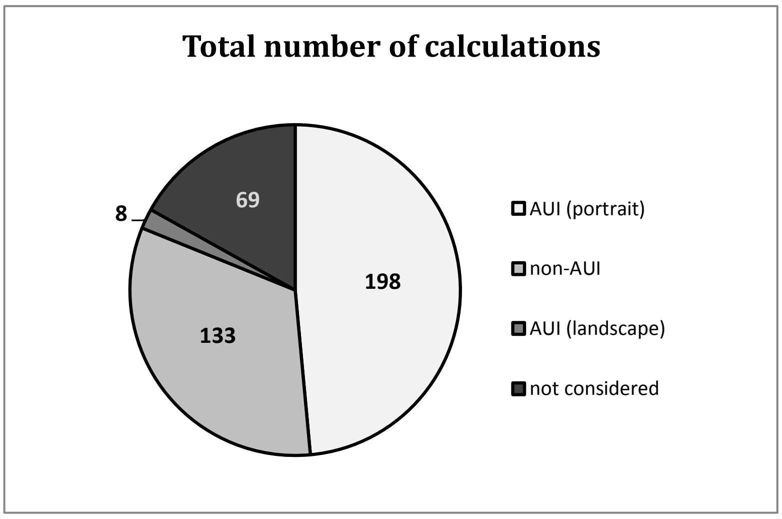


Figure 46: Total number of calculations with AdaptiveCalc.

Performance. The performance of the calculations made was measured remotely and reported to a server. A server-side script recorded and evaluated the incoming results. As performance measures both the error rate e and the typing speed was used. The error rate e was calculated in number of clear button presses $|c|$ divided by the total number of button presses $|b|$.

$$e = \frac{|c|}{|b|} \tag{1}$$

The typing speed s was determined by calculating the average of the medians of the times between single button presses b of each single recorded calculation R_i.

$$s = \frac{\sum_{i=1}^{|R|} \tilde{x}(R_i)}{|l|} \tag{2}$$

where

$$R_i = [t_{b_2} - t_{b_1}, \dots, t_{b_{|b|}} - t_{b_{|b|-1}}]_i \tag{3}$$

and $\tilde{x}$ refers to the median function. t_{b_j} refers to the time when button press j was made during a calculation R. b_1 denotes the first button press within one calculation, $b_{|b|}$ denotes the last button press within one calculation. The difference $t_{b_{j+1}} - t_{b_j}$ is the time between single button presses (TBBP) b_j and b_j+1. One single calculation is also called record and identified by a unique index i. $|R|$ represents the total number of calculations (records).

Using the median has the advantage that spike values are flattened. Spike values might result from thinking times, reading times or waiting times. These times must not be considered in the performance evaluation. Additionally, very large TBBPs (> 5000 ms or 3500 ms, see later) were ignored. Very large TBBPs are associated with long thinking times or breaks by the user but certainly not with looking for certain buttons. The time between single button presses was used as performance measure because the single tasks and therefore the total time needed for one calculation differed, as there were no predefined tasks (and therefore no predefined expression length) to accomplish for the end users.

Figure 47 and Figure 48 show examples of the TBBPs of two different calculations. In the diagrams the single TBBPs are placed along the x-axis in chronological order. So the leftmost bar of the bar chart represents the TBBP of the button presses b_1 and b_2. The height of the bars (y-axes) represents the corresponding TBBP in milliseconds $(t_{b_{j+1}} - t_{b_j})$.

The nineteen bars in Figure 47 show the time needed between the twenty button presses of the corresponding calculation. The calculation was 45.95 + 29.45 + 38.95 - 20 with the result 94.35. The corresponding record was

```
auiport;1328716355011:4;1328716355581:5;1328716356431:.;1328716357069:9;13287
16357787:5;1328716359378:+;1328716360167:2;1328716360790:9;1328716361441:.;13
28716361985:4;1328716362487:5;1328716363452:+;1328716364328:3;1328716364840:8
;1328716365464:.;1328716366014:9;1328716366496:5;1328716367613:-
;1328716368135:2;1328716368653:0;1328716371512;ok
```

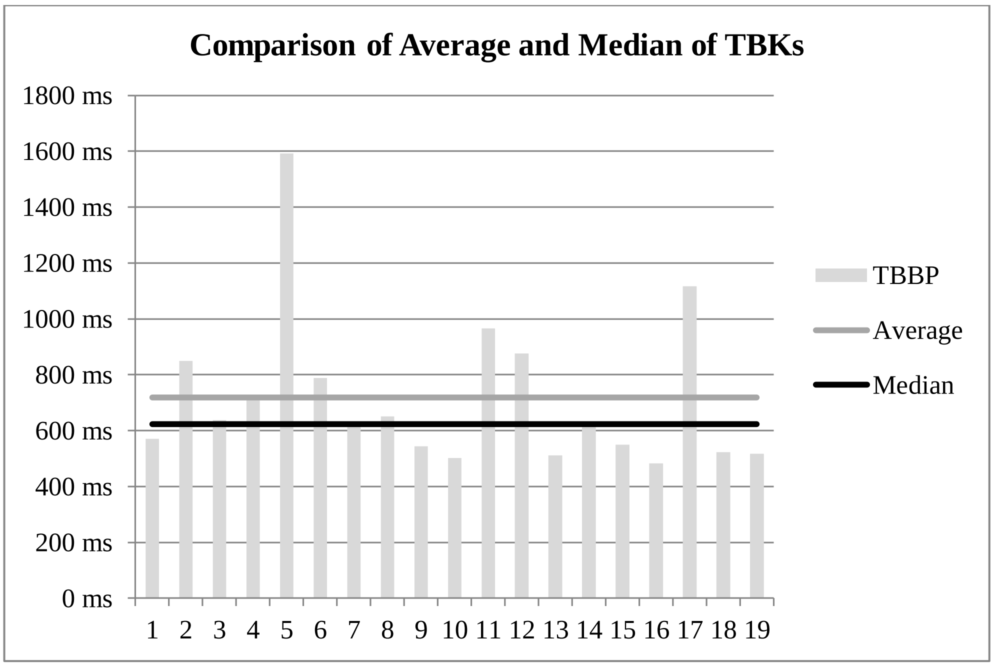


Figure 47: Comparison of average and median of TBBPs (long calculation)

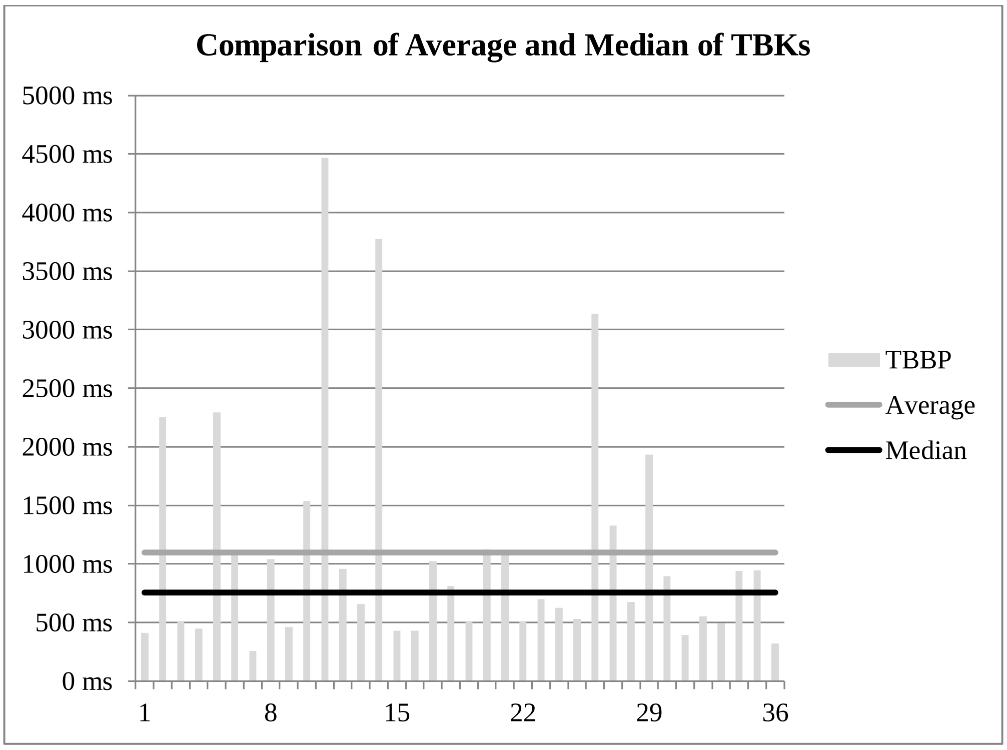


Figure 48: Comparison of average and median of TBBPs (short calculation)

As mentioned before, incomplete calculations were filtered out. The following example shows an incomplete calculation. Here, the user only entered an opening parenthesis and then pressed the clear button two times. "nan" (not a number) in the end of the line indicates that there was no valid result for the corresponding calculation.

```
auiport;1328699930271:(;1328699931276:[clear];1328699931486:[clear];132869993
1487;nan
```

Typing speed. The average times of the median times between single button presses (TBBPs) were calculated as described above. The results were 836.1 ms for AUI (portrait) mode, 947.5 ms for non-AUI mode (Figure 49). The values for AUI (landscape) mode are not included as there were only eight results collected. In average the TBBPs were 111 ms lower in AUI mode than in non-AUI mode. Therefore an average calculation including 10 button presses is performed more than one second faster in AUI mode than in non-AUI mode. A cutoff of 5000 ms means that all TBBPs larger than 5000 ms were ignored as larger TBBPs can be considered as thinking times or calculation breaks. When using a lower cutoff of 3500 ms the values were 803.4 ms for AUI (port) and 917.9 ms for non-AUI.

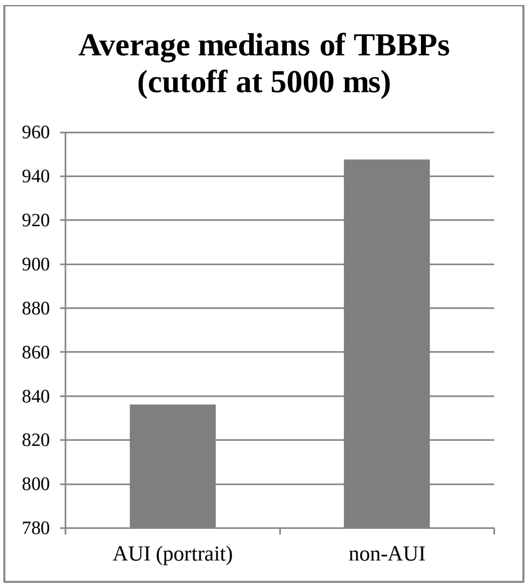


Figure 49: Average medians of TBBPs (cutoff at 5000 ms).

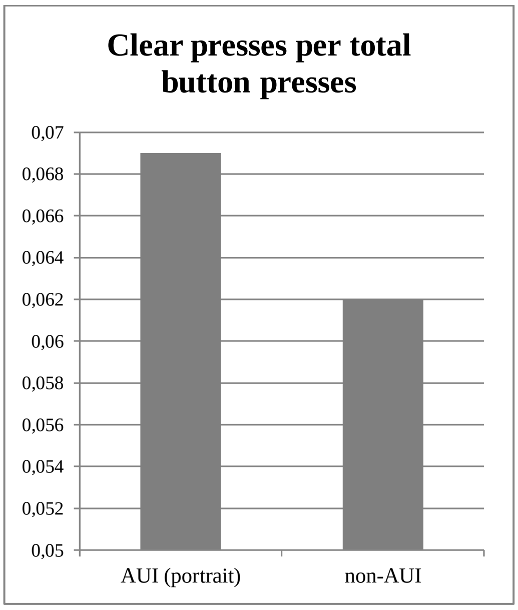


Figure 50: AdaptiveCalc – clear presses per total button presses.

Error rate. The error rate (number of clear button presses divided by total number of button presses) was slightly higher in AUI mode: 0.069 in AUI (portrait) mode and 0.062 in non-AUI mode (Figure 50). This means that on 100 button presses the clear button is – in average - pressed 6.9 times in AUI mode, while in non-AUI mode clear is only pressed 6.2 times.

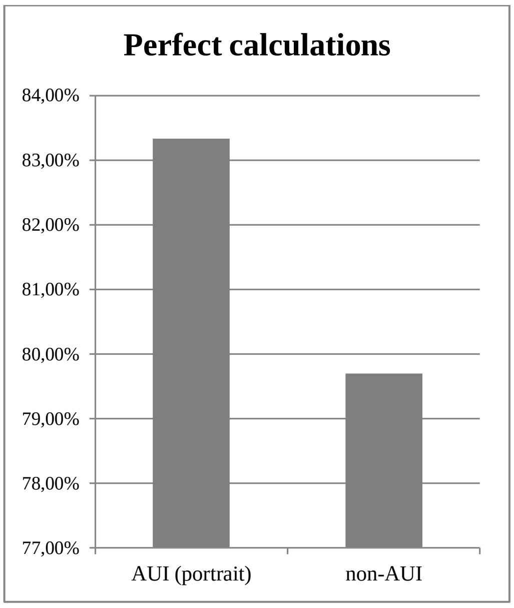


Figure 51: Perfect calculations.

Looking at the number of "perfect calculations" (calculations where no "clear" button press was involved, except for clearing the whole calculation in the end of the calculation) the AUI (portrait) UI has 83.33 % perfect calculations while the non-AUI only has a number of 79.7 % perfect calculations (Figure 51).

Conclusion. The results of the experiment suggest that the overall acceptance of the simple AUI is better than the acceptance of the complex non-AUI. Also the typing speed when using the AUI was better in average. The error rate was slightly higher in AUI mode. Due to the high similarity of the error rate results we can say that the overall performance of the AUI at least was not worse than when using the non-AUI.

4.3. HE Checklist for Mobile Devices

Based on different sources and based on the experience of the author of this thesis a usability checklist for smartphone and tablet applications was created. The resulting list is a list which covers general guidelines as well as more specific guidelines. However, the degree of specificity is limited in order to keep the check items applicable to most platforms. For detailed, platform-specific guidelines the reader shall refer to the developer documentation of the respective mobile platform. Platform-specific guidelines include icon design guidelines as well as guidelines on handling system notifications or similar system-specific points.

The guidelines developed for smartphones and tablets are as general as possible but target applications similar to applications of the two currently most spread operating systems Android and iOS, plus Windows Phone. Generalized rules might have to be adapted to the current situation or to the capabilities of the operating system. Hardware buttons, for example, are not present on all devices. Therefore rules mentioning hardware buttons have to either be ignored or applied to always visible software buttons.

The single rules were categorized into groups of thematically similar issues (see also Section 2.3.4). This section only gives some examples of checklist items. For the full checklist, please refer to Appendix A of this work.

C1 Feedback		
4	*Apple*	Is feedback subtle, but clear?
15	*Ji et al.*	Scroll bar: Is it possible to predict the quantity of contents through the scroll bar?

Table 13: Examples of checklist items for C1 Feedback

Both checklist items for C1 Feedback (Table 13) were found in checklists or guidelines for mobile devices. However, both rules are probably valid for desktop applications as well.

C2 Input		
1	-	Is the need for text input reduced to a minimum?
6	*Pierotti*	Are the valid answers for a question listed? (Nielsen: Provide lists of choices and picking from lists)

Table 14: Examples of checklist items for C2 Input

Table 14 lists examples of checklist items for C2 Input. The rule to reduce text input to a minimum was inspired by the statement of Longoria (2001) that text input annoys users. The second example also aims to reduce typing requirements.

C3 Match (Match between system and the real world)		
6	*Nielsen*	Is the conceptual model familiar to the user?
15	*fluid*	Is vocabulary familiar to the intended user, avoiding system-oriented terms?
26	-	Are user interface elements (buttons, …) large enough?
27	-	Are gestures like swiping used?

Table 15: Examples of checklist items for C3 Match

Concepts used in everyday life support the understanding of user interfaces. The rules categorized in C3 Match (Table 15) support this idea. Swiping gestures, often used for switching screens, resembles turning over paper pages.

C4 User control, Flexibility		
3	*Ji et al.*	Connect: Is there a way to stop the process of connection?
20	*fluid*	Is there a home/start page link?
33	-	Is it possible to deactivate vibration or sound feedback?
38	*Android*	Is the state of the app preserved during unexpected interruptions and properly restored when resuming (e.g. phone calls)?

Table 16: Examples of checklist items for C4 User Control, Flexibility

Table 16 (C4 User control, Flexibility) presents some example items for supporting the ability to control the application. Especially for mobile devices the ability of controlling the mobile data transfer is important. Also proper handling of unexpected interruptions such as incoming phone calls is required.

C5 Undo/Reversal		
3	*Pierotti*	Do functions that can cause serious consequences have an undo feature?
6	*Nielsen*	Is it obvious how to undo actions?

Table 17: Examples of checklist items for C5 Undo/Reversal

The rules for C5 Undo/Reversal (Table 17) are mostly applicable to both mobile apps and desktop applications.

C6 Consistency (and standards)		
5	-	If gestures are used, are common gestures used instead of proprietary gestures? (Also, don't override default system gestures)
10	*(Pierotti)*	Is the number of different colors limited?
30	*Microsoft*	Button control text should be concise and typically be a verb
40	*Android*	Do context menus identify the selected item? (e.g. "Edit contact", not just "Edit")

Table 18: Examples of checklist items for C6 Consistency

Rules for C6 Consistency (Table 18) are also mostly applicable for both desktop and mobile applications. However, the rule set of this category may be expanded by several mobile device or touchscreen specific items. Rules originally specifying concrete values ("*Are there no more than four to seven colors?*"), were changed a to more unspecific phrasing ("*Is the number of different colors limited?*"). The concrete values must be revised and validated for mobile devices, as the original rule was designed for text-based desktop systems.

C7 Error handling and prevention		
3	*Pierotti*	Are data inputs case-blind whenever possible?
10	*fluid*	Is confirmation required when an action is difficult or impossible to undo?

Table 19: Examples of checklist items for C7 Error handling and prevention

C7 Error handling and prevention (Table 19) is important for most systems regardless the device type the application is running on. As the example checklist items show, case-blindness and undo features are helpful even on mobile devices.

C8 Short-term memory (Recognition rather than recall)		
10	-	If gestures are not obvious, is the number of different gestures limited and are they easy to remember?
20	*Apple*	Do controls look tappable?
11	*Microsoft*	Do map controls and long lists fill the whole available screen space in order to avoid excessive scrolling?
30	-	Are sliders, radio buttons or other touchable controls preferred to text input?
37	-	Do all screens (activities) support both portrait and landscape screen orientation?
42	*Apple*	Is scrolling preferred over reducing the content size?

Table 20: Examples of checklist items for C8 Short-term memory

Supporting the short term memory (C8, Table 20) is recommended for mobile applications as well as for non-mobile applications. However, there are special aspects to regard for mobile applications. Therefore, rules for small screen sizes as well as touch input related rules were added.

C9 Design (Aesthetic and minimalist design)		
7	-	Are animations and transitions smooth?
32	*Pierotti*	Are the most frequently used functions in the most accessible positions?

Table 21: Examples of checklist items for C9 Design

Many rules in C9 Design (Table 21) are applicable to both desktop systems and mobile systems. However, most of the rules have to be implemented differently. The animations and transitions, for example, must comply with the platform-specific design guidelines. The "most accessible position" on smartphones might be the lower right corner of the screen if the user operates the phone with only one hand and one thumb. On other devices the most accessible position might differ.

C10 Documentation (Help and documentation)		
10	*fluid*	Is help information focused on the user's task?
18	*Pierotti*	Can users easily switch between help and their work?

Table 22: Examples of checklist items for C10 Documentation

Examples for rules for checking the documentation quality can be found in C10 Documentation (Table 22). In the rules can be summarized a few words by stating that the documentation should be helpful and not hinder the usage of the actual application. These rules are applicable not only for desktop applications; however, comprehensive on-device documentation for mobile software is rare. When selecting “help” in Android applications, often simply the web browser is opened displaying a help website.

C11 Testing (see Appendix A) stresses the importance of usability tests. These rules are actually not used during the heuristic evaluation but should be regarded all the time. The rules were included for completeness because they were included in the source documents as well.

5. Lessons Learned

During the work with Boom Software AG we had the opportunity to gain insight into the processes and the methods of a real business company. It was interesting to see how business software engineering works from an elevated point of view. This means that we got insight into many different areas of the software development at Boom – starting from the core framework development up to customer relations and application development. What we learned was that, when developing for customers, the customer preferences have top priority, even though the personal preferences or even objective criteria suggest something different. Of course, customers are always informed about that, but they are the ones who have the last word. Additionally, there is always a certain dependency from the customers, which always results in a certain risk for every project.

During the thinking aloud tests we learned that people's behavior and personal preferences may differ to a very high degree. For one person an application cannot be too sober, for another person the same application is way too "boring" and colorless. Also the "thoughts-to-speech" behavior of test users differs greatly. Some were very motivated in speaking out their thoughts; others had to be motivated to talk again and again. We also learned that detailed usability evaluations can be very time consuming. The analysis of the thinking aloud tests took longer than expected and the heuristic evaluation is time consuming as well when inspecting complex applications. When comparing the test results of HE and TA it could be seen that HE is more detailed in some cases, however TA probably focuses more on the issues end users really find disturbing.

Another interesting effect was observed during the TA tests. For many people it seems to be much easier to talk about problems and things they don't like than about things they like. We collected 15 distinct positive impressions, 41 distinct negative expressions, and 17 distinct recommendations, although the overall usability was rated "very good" by most of the test users. Example: When reviewing a text with some spelling errors the readers will probably criticize every single misspelled word. However, it is very unlikely that readers will positively mention every correctly spelled word.

During the creation of the mobile-specific checklist it could be seen that the basic rules for usability seem to be valid for different kinds of systems. For covering certain scenarios or features of modern devices, however, the rules have to be altered or expanded. Some rules are even not applicable at all. However, it was interesting to see that some ancient rules, originally created for text-based user interfaces, still have validity for modern touch-based mobile user interfaces.

During literature studies it was found that adaptive user interfaces are in general seen as good tool for improving usability. However, in practice only few applications can be found which extensively implement the concept of adaptivity. This shows that effort must be put into bringing good scientific ideas into practice. For the performance comparison of the non-adaptive and the adaptive user interface the design, the conduction, and the evaluation of a scientific experiment was practiced – a new experience for me as software engineer.

According to literature and according to discussions with people coming from the enterprise area, usability is still underestimated during software engineering. For Boom Software AG usability has a high significance – this is not only proven by the conduction of extensive usability tests during this work, but also by the overall good results. However, Boom admits that there is even more work to be done in this area.

Additionally, in software engineering education usability should get a higher value. At Graz University of Technology there are courses dealing with Web usability, user-centered design and data visualization. However, a broader education in this area might help to increase the usability awareness amongst students. As one-year guest student at the Faculty of Engineering (LTH), Lund University it could also be seen that there was a strong focus on all sorts of technical aspects. Usability engineering issues were, however, disregarded.

Summarizing, it can be said that during this work I had the opportunity to get insight into areas I would never have gotten insight into without this work. The work differed from the classic software engineering and informatics topics learned during my studies not only because I got insight into the work of a real mid-sized company. It also showed that in software engineering there is more to be done than data modeling, programming and functional testing of software. And, in one part of the work, even scientific work was practiced.

6. Conclusion

In this work several aspects of user interface design and testing were discussed. The terms user experience, usability and accessibility were discussed as well as different usability evaluation methods. The greater and greater importance of mobile devices and the importance of good usability of mobile applications was stressed. It was also pointed out that in the enterprise sector usability should be an integral component in application development as well. The advantages of adaptive user interfaces in terms of usability on mobile, hence usually small devices were explained and confirmed by a small experiment. The results of the experiment suggest that adaptive user interfaces may help to improve the usability of applications and the performance of end users. This result coincides with other literature sources.
The usability of enterprise desktop software by Boom Software AG was evaluated using the methods heuristic evaluation and thinking aloud. Comparing the results of both evaluations it can be seen that both methods found similar issues, however, HE seems to focus more on details while TA focuses more on general and practically relevant issues. It is unsure whether the issues found via HE affect the perceived usability by the end-users, i.e. are practically relevant. Despite the support of a checklist, HE probably did not find all issues. When applying TA it cannot be said as well that all issues were found because the perception of certain issues might not be spoken out by the test users. The results of both applied methods cannot be directly compared within this work because different applications and different application versions were evaluated.
Based on the results of the usability evaluations of enterprise software, usability guidelines for Boom software were elaborated. The guidelines will help to ensure the quality of current and future developments by Boom Software AG.
For conducting the heuristic evaluation a checklist tailored to enterprise desktop software was elaborated. This checklist was further extended and modified in order to create a basic checklist for evaluating applications for mobile devices – specifically for current smartphones and tablets. The created checklist is not complete, but provides a basic reference point for heuristic evaluations of modern mobile applications. As Boom is about to expand their technology to mobile devices, this checklist might be the basis for future heuristic evaluations of Boom's mobile applications.

7. Future Work

Future work to be done in the area of usability evaluation on mobile devices include the development of usability evaluation methods tailored to mobile devices regarding the special properties of such devices – especially of current smartphones and tablets.

Heuristic evaluation tailored to mobile devices must be further elaborated for current smartphones and tablet computers, especially detailed checklists for such devices must be extended. The usability checklist for mobile applications developed during this work must be improved and expanded in order to get the highest benefit. The rules listed in Appendix A are a good starting point.

In software development, engineers must always keep in mind that usability is a key factor for success of their application. Therefore, the importance of usability must be stressed more – not only in education but also in the professional area. Usability testing should be an integrated component in every software development process.

Specifically for Boom Software AG, the elaborated usability guidelines and improvement suggestions must be further applied to their software and the awareness for usability issues must be raised amongst the application developers. In the future, regular usability tests will help to maintain a good user experience with Boom software.

As adaptivity might be a good way for improving usability, especially for applications running on mobile devices with relatively small screens, more use cases for better practical applicability should be developed in order to bring AUIs to real world applications. New developments in the area of smartphones and tablets, such as new sensors and the upcoming NFC technology must be integrated in future usability considerations.

Due to a lack of good software tools for supporting usability tests such as thinking aloud on mobile devices, better tools must be developed. Usability testing software for mobile devices similar to desktop software which allows screen capturing, accessing webcams, and key logging is required. Such software should allow using the device's internal front camera for recording the face of the test user and integrate touch logging as well as screen capturing and audio recording and recording of the current device's context (orientation, position, ambient light, …).

8. List of Figures

Figure 1: Global ICT developments, 2001-2011. Source: ITU 18
Figure 2: Simple vs. complex user interfaces (found in Lorz (2010)) 20
Figure 3: Tandy ZOOMER, source: 8bit-micro.com 27
Figure 4: The first Smartphone “Simon” by IBM 1994 28
Figure 5: Smartphone Penetration and OS Share, third quarter 2011, USA. Source: Nielsen 30
Figure 6: Ranking of technology trends 2012-2014. Source: Forrester Research, Inc. via http://www.cio.de/knowledgecenter/bi/2292300/ 31
Figure 7: World-wide smartphone mobile OS market share. Image source: Wikimedia Commons, own work by user milominderbinder2; data originally from Gartner. 38
Figure 8: Concept sketch of LucidTouch: a pseudotransparent device (Wigdor et al., 2007) 46
Figure 9: Basic layouts in Android (left) and Windows Phone (right). 53
Figure 10: Users, end-users and test users. 61
Figure 11: Meaning of SUS values (Bangor et al., 2008). 67
Figure 12: Proportion of usability problems found by aggregates of size 1 to 30 (Nielsen and Molich, 1990) 69
Figure 13: Total Customizing with Boom’s BORA framework (source: Boom) 82
Figure 14: BORA framework - main components (source: Boom) 82
Figure 15: Detailed view on BORA’s core components (source: Boom) 84
Figure 16: Boom’s three development areas (source: Boom) 87
Figure 17: Boom’s simplified process model for software development (source: Boom) 87
Figure 18: Mobile Android app for task sheets 90
Figure 19: Leseratte tutorial – screenshot of Boom BORA Designer 94
Figure 20: Leseratte tutorial – UI definition 95
Figure 21: Screenshot of the tutorial version of Leseratte 95
Figure 22: Mockup for Leseratte 2.0 – Start page and book overview. 98
Figure 23: Mockup for Leseratte 2.0 – customer list. 99

Figure 24: Data model of Leseratte 2.0 99
Figure 25: Screenshot of final Leseratte 2.0 – Start page and book overview. 100
Figure 26: Screenshot of final Leseratte 2.0 - List of authors. 100
Figure 27: Screenshot of final Leseratte 2.0 – Timeline and book search pane. 101
Figure 28: Task cards for the thinking aloud tests.. 103
Figure 29: Thinking aloud screen capturing with face recording. 103
Figure 30: Thinking aloud interviews with test user and the usability team and feedback form. 103
Figure 31: AdaptiveCalc – non-adaptive and adaptive UI 106
Figure 32: Design of the adaptive button array on paper 108
Figure 33: AdaptiveCalc development using Eclipse.. 109
Figure 34: Task list - Mockup and screenshot of latest app version 114
Figure 35: Task list in landscape mode (screenshot of latest app version) 114
Figure 36: Task form - Mockup and latest version screenshot..................................... 115
Figure 37: Issue found via HE: Dropdown box for only four items. Better: Radio buttons. 117
Figure 38: Issue found via HE: Too many tabs and last tab not visible. 117
Figure 39: HE – Example for one of the issues in the report for BMM 117
Figure 40: Thinking aloud test - example of positive feedback. 118
Figure 41: Thinking aloud test – example of negative feedback................................. 119
Figure 42: Web page of Boom's new usability documentation – rules for forms........ 121
Figure 43: Time between button presses. 122
Figure 44: Error rate (clear presses / total number of button presses). 123
Figure 45: Test user calculating with AdaptiveCalc. 124
Figure 46: Total number of calculations with AdaptiveCalc. 125
Figure 47: Comparison of average and median of TBBPs (long calculation)............. 127
Figure 48: Comparison of average and median of TBBPs (short calculation)............ 128
Figure 49: Average medians of TBBPs (cutoff at 5000 ms). 129
Figure 50: AdaptiveCalc – clear presses per total button presses. 129
Figure 51: Perfect calculations. 130

9. List of Tables

Table 1: Comparison of Notebook and Smartphone 25
Table 2: Worldwide Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units). 29
Table 3: Positive and negative aspects of smartphones compared to classic (cell) phones. 37
Table 4: Differences between native apps and HTML5 apps. 40
Table 5: Analysis of permissions needed by 237 randomly selected apps 48
Table 6: Summary of the usage of phone features of current apps. 50
Table 7: Analogies between user anxiety and metrics (Holzinger et al., 2008b) 58
Table 8: Comparison of usability evaluation techniques (Holzinger, 2005) 62
Table 9: Guidelines for the appropriateness of different evaluation techniques for context-aware mobile user interfaces (Lumsden, 2008) 65
Table 10: Assignment of different heuristics to categories 72
Table 11: Examples of checklist items from different categories. 97
Table 12: Summary of the user’s answers during the interview 124
Table 13: Examples of checklist items for C1 Feedback 131
Table 14: Examples of checklist items for C2 Input 131
Table 15: Examples of checklist items for C3 Match 132
Table 16: Examples of checklist items for C4 User Control, Flexibility 132
Table 17: Examples of checklist items for C5 Undo/Reversal 132
Table 18: Examples of checklist items for C6 Consistency 133
Table 19: Examples of checklist items for C7 Error handling and prevention 133
Table 20: Examples of checklist items for C8 Short-term memory 134
Table 21: Examples of checklist items for C9 Design 134
Table 22: Examples of checklist items for C10 Documentation 135

10. Glossary

Adaptive user interface. User interface which is altered at runtime in order to satisfy the the special needs of the end user by regarding the current context.

App. Short for application. However, in everyday speech the term app is often used especially when meaning mobile applications for mobile platforms which can usually be bought in an app store or app market.

Context. All circumstances which influence the current situation.

Control. A basic user interface component such as a text box or a button. A user interface is usually composed of several controls.

End user. The target users of software. End users are the people who use the software at their workplaces or for getting benefits from the usage. Test users in contrast are people who use software for the purpose of testing software. Therefore test users might be end users at the same time if a usability evaluation method such as field observation is applied.

Heuristic evaluation. Usability inspection method conducted by usability experts. Usability experts review software by a set of heuristics or guidelines. Also detailed checklists can be used in order to support the reviewing process.

Input method. An independent component (typically software) which makes data input to a UI control possible. Example: a soft keyboard which allows text input to text boxes.

Mobile device. Computer which is small and lightweight enough to be carried around. Examples of mobile devices are smartphones and tablet computers, but also Netbooks and music players.

Soft keyboard. A software on-screen keyboard which allows entering text without a physical keyboard. Often used on smartphones or tablet computers.

Test user. Users who use software during a usability study for evaluating the usability of the application.

Thinking aloud. A usability test method where test users are encouraged to speak out loud all their thoughts in order to get insight into the thinking processes and the reasons for the single actions.

Total Customizing. A concept followed by Boom Software AG which makes it possible to create highly customizable software and allows quick reactions to changed customer needs.

Usability. The degree of effectiveness, efficiency, and satisfaction when using a product which is used to reach a specific goal.

User (software). Any user of software. This includes end users as well as test users or even software developers who try newly developed features.

User experience. Relates to the whole experience a user has while using a product. User experience goes beyond usability and also includes emotions and feelings.

User interface. The interface between the system's internals and the user. Via the user interface the user can interact with the system. Additionally, the system gives feedback to the user via the user interface.

11. References

ANDREASEN, M. S., NIELSEN, H. V., SCHRÖDER, S. O. & STAGE, J. What happened to remote usability testing?: An empirical study of three methods. CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems, 2007 New York, NY, USA. ACM, 1405-1414.

ANDREWS, K. 2006. *Andrews Allgemeine Usability Heuristiken* [Online]. Available: *http://courses.iicm.tugraz.at/hci/practicals/materials/de/he/heuristiken.pdf* [Accessed 10 January 2012].

ANDREWS, K. 2011. *Human Computer Interaction - Thinking Aloud* [Online]. Available: *http://courses.iicm.tugraz.at/hci/practicals/materials/en/ta* [Accessed 10 January 2012].

BAKER, J. M., DENG, L., GLASS, J., KHUDANPUR, S., LEE, C.-H., MORGAN, N. & O'SHAUGHNESSY, D. O. 2009. Research Developments and Directions in Speech Recognition and Understanding, Part 1. *IEEE Signal Processing Magazine,* 26**,** 75-80.

BANGOR, A., KORTUM, P. T. & MILLER, J. T. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction,* 24**,** 574-594.

BAUDISCH, P. & CHU, G. Back-of-device interaction allows creating very small touch devices. Proceedings of the 27th international conference on Human factors in computing systems, 2009 Boston, MA, USA. 1518995: ACM, 1923-1932.

BERTINI, E., GABRIELLI, S. & KIMANI, S. Appropriating and assessing heuristics for mobile computing. Proceedings of the working conference on Advanced visual interfaces, 2006 Venezia, Italy. 1133291: ACM, 119-126.

BEVAN, N. 1995. Measuring Usability as Quality of Use. *Software Quality Journal,* 4**,** 115-130.

BEVAN, N. 2001. International standards for HCI and usability. *International Journal of Human-Computer Studies,* 55**,** 533-552.

BEVAN, N. What is the difference between the purpose of usability and user experience evaluation methods? Proceedings of the Workshop UXEM'09 (Interact 2009), 2009 Uppsala, Sweden.

BEVAN, N. & AZUMA, M. Quality in Use: Incorporating Human Factors into the Software Engineering Lifecycle. Proceedings of the 3rd International Software Engineering Standards Symposium (ISESS '97), 1997. IEEE Computer Society, 169.

BEVAN, N. & CURSON, I. Methods for Measuring Usability. Proceedings of the sixth IFIP conference on human-computer interaction, 1997.

BEVAN, N. & MACLEOD, M. 1994. Usability measurement in context. *Behaviour & Information Technology,* 13**,** 132-145.

BIEL, B., GRILL, T. & GRUHN, V. 2010. Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application. *Journal of Systems and Software,* 83**,** 2031-2044.

BILLI, M., BURZAGLI, L., CATARCI, T., SANTUCCI, G., BERTINI, E., GABBANINI, F. & PALCHETTI, E. 2010. A unified methodology for the evaluation of accessibility and usability of mobile applications. *Universal Access in the Information Society,* 9**,** 337-356.

BROOKE, J. 1996. SUS: A quick and dirty usability scale. *Usability evaluation in industry*.

BRUUN, A., GULL, P., HOFMEISTER, L. & STAGE, J. Let your users do the testing: a comparison of three remote asynchronous usability testing methods. CHI '09: Proceedings of the 27th international conference on Human factors in computing systems, 2009 New York, NY, USA. ACM, 1619-1628.

CASTILLO, J. C., HARTSON, H. R. & HIX, D. Remote usability evaluation: can users report their own critical incidents? CHI 98 conference summary on Human factors in computing systems, 1998 Los Angeles, California, United States. 286736: ACM, 253-254.

CHIN, J. P., DIEHL, V. A. & NORMAN, K. L. Development of an instrument measuring user satisfaction of the human-computer interface. Proceedings of the SIGCHI conference on Human factors in computing systems, 1988 Washington, D.C., United States. 57203: ACM, 213-218.

COOKE, L. Is Eye Tracking the Next Step in Usability Testing? International Professional Communication Conference, 2006 IEEE, 23-25 Oct. 2006 2006. 236-242.

CROSBY, M. E., IDING, M. K. & CHIN, D. N. 2001. Visual Search and Background Complexity : Does the Forest Hide the Trees? *In:* BAUER, M., GMYTRASIEWICZ, P. & VASSILEVA, J. (eds.) *User Modeling 2001.* Springer Berlin / Heidelberg.

DIAPER, D. & STANTON, N. 2004. The Handbook of Task Analysis for Human Computer Interaction.

EL-BAKRY, H. M., RIAD, A. M., ABU-ELSOUD, M., MOHAMED, S., HASSAN, A. E., MASTORAKIS, N., KANDEL, M., ZADEH, L. & KACPROZYK, J. Adaptive User Interface for Web Applications. WSEAS International Conference Proceedings Recent Advances in Computer Engineering, 2010. WSEAS, 190-211.

ERICSSON, K. A. & SIMON, H. A. 1980. Verbal Reports as Data. *Psychological Review,* 87**,** 215–251.

FARRINGTON, J. 2011. Seven plus or minus two. *Performance Improvement Quarterly,* 23**,** 113-116.

FINSTAD, K. 2010. The Usability Metric for User Experience. *Interacting with Computers,* 22**,** 323-327.

FORLIZZI, J. & BATTARBEE, K. Understanding Experience in Interactive Systems. Proceedings of the 2004 conference on Designing Interactive Systems (DIS 04), 2004. ACM, 261-268.

FORMAN, G. H. & ZAHORJAN, J. 1994. The Challenges of Mobile Computing. *COMPUTER,* 27**,** 38-47.

FOULK, E., HAY, R., SCOTT, K., SQUIERS, M. D., TESAR, J., COHEN, C. J. & JACOBUS, C. J. 2009. *Method for Controlling a Graphical User Interface for Touchscreen-Enabled Computer Systems*. United States patent application.

GELLERSEN, H. W., SCHMIDT, A. & BEIGL, M. 2002. Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts. *Mobile Networks and Applications,* 7**,** 341-351.

GERMANAKOS, P., TSIANOS, N., LEKKAS, Z., MOURLAS, C. & SAMARAS, G. 2009. Realizing Comprehensive User Profile as the Core Element of Adaptive and Personalized Communication Environments and Systems. *Comput. J.,* 52**,** 749-770.

GERMANAKOS P., T. N., LEKKAS Z., MOURLAS C., BELK M., & SAMARAS G. 2009. Towards an Adaptive and Personalized Web Interaction using Human Factors. *In:* ANGELIDES, M. (ed.) *Advances in Semantic Media Adaptation and Personalization.* Taylor & Francis Group.

GHAHRAMANI, B. 1998. *Method for measuring the usability of a system*. Sep 15, 1998.

HASSENZAHL, M. & TRACTINSKY, N. 2006. User experience - a research agenda. *Behaviour & Information Technology,* 25**,** 91-97.

HEO, J., HAM, D.-H., PARK, S., SONG, C. & YOON, W. C. 2009. A framework for evaluating the usability of mobile phones based on multi-level, hierarchical model of usability factors. *Interacting with Computers,* 21**,** 263-275.

HILBERT, D. M. & REDMILES, D. F. 1999. Extracting usability information from user interface events. *ACM Computing Surveys (CSUR),* 32**,** 384-421.

HIMBERG, J., KORPIAHO, K., MANNILA, H., TIKANMAKI, J. & TOIVONEN, H. T. T. Time series segmentation for context recognition in mobile devices. Data Mining,

2001. ICDM 2001, Proceedings IEEE International Conference on, 2001 2001. 203-210.

HOLZ, C. & BAUDISCH, P. Understanding touch. Proceedings of the 2011 annual conference on Human factors in computing systems, 2011 Vancouver, BC, Canada. 1979308: ACM, 2501-2510.

HOLZINGER, A. User-Centered Interface Design for Disabled and Elderly People: First Experiences with Designing a Patient Communication System (PACOSY). Proceedings of the 8th International Conference on Computers Helping People with Special Needs, 2002. 684035: Springer-Verlag, 33-40.

HOLZINGER, A. Finger instead of mouse: touch screens as a means of enhancing universal access. Proceedings of the User interfaces for all 7th international conference on Universal access: theoretical perspectives, practice, and experience, 2003 Paris, France. 1765461: Springer-Verlag, 387-397.

HOLZINGER, A. 2005. Usability engineering methods for software developers. *Communications of the ACM,* 48**,** 71-74.

HOLZINGER, A. 2006. Thinking-aloud eine Königsmethode im Usability Engineering. *OCG Journal,* 31**,** 4-5.

HOLZINGER, A. 2010. *Process Guide for Students for Interdisciplinary Work in Computer Science/Informatics: Instructions Manual - Handbuch für Studierende*, Books on Demand.

HOLZINGER, A. & BROWN, S. Low cost prototyping: Part 1, or how to produce better ideas faster by getting user reactions early and often. Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 2, 2008a Liverpool, United Kingdom. British Computer Society, 213-214.

HOLZINGER, A. & BROWN, S. Low cost prototyping: Part 2, or how to apply the thinking-aloud method efficiently. Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 2, 2008b Liverpool, United Kingdom. 217-218.

HOLZINGER, A. & ERRATH, M. 2007. Mobile computer Web-application design in medicine: some research based guidelines. *Universal Access in the Information Society International Journal,* 6**,** 31-41.

HOLZINGER, A., ERRATH, M., SEARLE, G., THURNHER, B. & SLANY, W. From Extreme Programming and Usability Engineering to Extreme Usability in Software Engineering Education (XP+UE->XU). Proceedings of the 29th annual international conference on Computer software and applications conference, 2005a. IEEE Computer Society, 169-172.

HOLZINGER, A., HÖLLER, M., SCHEDLBAUER, M. & URLESBERGER, B. 2008a. An Investigation of Finger versus Stylus Input in Medical Scenarios. *Proceedings of the Iti 2008 30th International Conference on Information Technology Interfaces***,** 433-438.

HOLZINGER, A., NISCHELWITZER, A. & MEISENBERGER, M. 2005b. Lifelong-learning support by m-learning: example scenarios. *eLearn,* 2005**,** 2.

HOLZINGER, A., SAMMER, P. & HOFMANN-WELLENHOF, R. Mobile computing in medicine: designing mobile questionnaires for elderly and partially sighted people. Proceedings of the 10th international conference on Computers Helping People with Special Needs, 2006 Linz, Austria. 2098051: Springer-Verlag, 732-739.

HOLZINGER, A., SCHLÖGL, M., PEISCHL, B. & DEBEVC, M. Preferences of Handwriting Recognition on Mobile Information Systems in Medicine. Proceedings of the 2010 International Conference on e-Business (ICE-B), 2010. 120-123.

HOLZINGER, A., SEARLE, G., KLEINBERGER, T., SEFFAH, A. & JAVAHERY, H. 2008b. Investigating Usability Metrics for the Design and Development of Applications for the Elderly. *In:* MIESENBERGER, K., KLAUS, J., ZAGLER, W. & KARSHMER, A. (eds.) *Computers Helping People with Special Needs.* Springer Berlin / Heidelberg.

HOLZINGER, A., WACLIK, O., KAPPE, F., LENHART, S., ORASCHE, G. & PEISCHL, B. Rapid Prototyping On The Example Of Software Development In Automotive Industry. Proceedings of the 8th International Conference on electronic Business and Telecommunications, 2011a. SciTec, 57-61.

HOLZINGER, K., LEHNER, M., FASSOLD, M. & HOLZINGER, A. Archaeological Scavenger Hunt on Mobile Devices: From e-Education to E-Business - A triple adaptive mobile application for supporting Experts, Tourists and Children. ICETE 2011 8th International Joint Conference on e-Business and Telecommunications, 2011b Sevilla, Spain. SciTec, 131-136.

ISHII, H. & ULLMER, B. Tangible Bits: Towards Seamless Interfaces between People , Bits and Atoms. CHI '97 Proceedings of the SIGCHI conference on Human factors in computing systems, 1997.

JI, Y. G., PARK, J. H., LEE, C. & YUN, M. H. 2006. A Usability Checklist for the Usability Evaluation of Mobile Phone User Interface. *International Journal of Human-Computer Interaction,* 20**,** 207-231.

KIRAKOWSKI, J. & CORBETT, M. 1993. SUMI: the Software Usability Measurement Inventory. *British Journal of Educational Technology,* 24**,** 210-212.

KORHONEN, H. & KOIVISTO, E. M. I. 2006. Playability heuristics for mobile games. *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services - MobileHCI '06***,** 9.

KORPIPAA, P., MANTYJARVI, J., KELA, J., KERANEN, H. & MALM, E. J. 2003. Managing context information in mobile devices. *Pervasive Computing, IEEE,* 2**,** 42-51.

KRISHNAMURTHY, S., CHAKRABORTY, D., JINDAL, S. & MITTAL, S. Context-Based Adaptation of Mobile Phones Using Near-Field Communication. Mobile and Ubiquitous Systems - Workshops, 2006. 3rd Annual International Conference on, 17-21 July 2006 2006. 1-10.

LAUGWITZ, B., HELD, T. & SCHREPP, M. Construction and Evaluation of a User Experience Questionnaire. Proceedings of the 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for Education and Work, 2008 Graz, Austria. 1484388: Springer-Verlag, 63-76.

LEE, S. & ZHAI, S. 2009. The performance of touch screen soft buttons. *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09***,** 309.

LEMLOUMA, T. & LAYAIDA, N. Context-aware adaptation for mobile devices. Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on, 2004 2004. 106-111.

LEWIS, J. 1995. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction,* 7**,** 57-78.

LEWIS, J. R. Input rates and user preference for three small-screen input methods: Standard keyboard, predictive keyboard, and handwriting. PROCEEDINGS OF THE HUMAN FACTORS AND ERGONOMICS SOCIETY 43RD ANNUAL MEETING, VOLS 1 AND 2, 1999 PO BOX 1369, SANTA MONICA, CA 90406-1369 USA. HUMAN FACTORS AND ERGONOMICS SOC, 425-428.

LOBO, D., KASKALOGLU, K., FOX, H. & SRISANGKHAJORN, M. T. 2011. A Synergic Approach to Web Usability for Smartphones. *Engineering,* 6**,** 65-69.

LONGORIA, R. 2001. Designing Mobile Applications: Challenges, Methodologies, and Lessons Learned. *Usability Evaluation and Interface Design.*

LORZ, A. 2010. *Mobile Usability Testing (presentation slides)* [Online]. Available: *http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/mz/veranstaltungen/konferenzen/2010/5.wud_in_dresden_2010/world_usability_day_2010-Dateien/Praesentationen/WUD2010_Lorz.pdf* [Accessed 2 February 2012].

LOWRY, G. 2008. *Creating Heuristics for the Evaluation of Mobile Devices.* Napier University.

LUMSDEN, J. 2008. User Interface Design and Evaluation for Mobile Technology.

MARTIN, B., ISOKOSKI, P., JAYET, F. & SCHANG, T. 2009. *Performance of finger-operated soft keyboard with and without offset zoom on the pressed key,* New York, New York, USA, ACM Press.

MCCARNEY, R., WARNER, J., ILIFFE, S., VAN HASELEN, R., GRIFFIN, M. & FISHER, P. 2007. The Hawthorne Effect: a randomised, controlled trial. *BMC Medical Research Methodology,* 7**,** 30.

MELANSON, D. 2010. *Microsoft Adaptive Keyboard prototype debuts at center of UIST Student Innovation Contest* [Online]. Melanson, Donald. Available: *http://www.engadget.com/2010/08/12/microsoft-adaptive-keyboard-prototype-debuts-at-center-of-uist-s/* [Accessed 12 October 2011].

MERLIN, B. & RAYNAL, M. Evaluation of SpreadKey system with motor impaired users. ICCHP'10 Proceedings of the 12th international conference on Computers helping people with special needs, July 2010. 112-119.

MICROSOFT 2009. Xaml Object Mapping Specification 2009.

NIELSEN, J. 1994a. Enhancing the Explanatory Power of Usability Heuristics.

NIELSEN, J. 1994b. Estimating the number of subjects needed for a thinking aloud test. *Int. J. Hum.-Comput. Stud.,* 41**,** 385-397.

NIELSEN, J. 1994c. Heuristic evaluation. *In:* NIELSEN, J. & MACK, R. L. (eds.) *Usability Inspection Methods.* New York, NY: John Wiley & Sons.

NIELSEN, J. 2003. *Usability 101: Introduction to Usability* [Online]. Available: *http://www.useit.com/alertbox/20030825.html* [Accessed 10 February 2012].

NIELSEN, J. & MOLICH, R. Heuristic evaluation of user interfaces. Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people, 1990 Seattle, Washington, United States. 97281: ACM, 249-256.

PIEROTTI, D. 2000. *Heuristic Evaluation - A System Checklist* [Online]. Available: *http://www.stcsig.org/usability/topics/articles/he-checklist.html* [Accessed 18 January 2012].

PITTMAN, J. A. 2007. Handwriting recognition: Tablet PC text input. *COMPUTER,* 40**,** 49-54.

PO, S., HOWARD, S., VETERE, F. & SKOV, M. B. 2004. Heuristic Evaluation and Mobile Usability: Bridging the Realism Gap. *Work***,** 49-60.

RAYMOND, E. S. & LANDLEY, R. W. 2004. *The Art of Unix Usability* [Online]. Available: *http://catb.org/~esr/writings/taouu/html/index.html* [Accessed 11 February 2012].

RAYNAL, M. & VIGOUROUX, N. 2005. KeyGlasses: Semi-transparent keys to optimize text input on virtual keyboard. *ASSISTIVE TECHNOLOGY RESEARCH SERIES,* 16**,** 713-717.

RIEMAN, J., HIITOLA, K., HEINE, H., YLI-NOKARI, J., KALLIO, M. & KAKI, M. 2008. *Input On Touch User Interfaces*. United States patent application.

RODDEN, T., CHERVEST, K., DAVIES, N. & DIX, A. Exploiting Context in HCI Design for Mobile Systems. Workshop on Human Computer Interaction with Mobile Devices, 1998.

SCHILIT, B., ADAMS, N. & WANT, R. Context-Aware Computing Applications. Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on, 8-9 Dec. 1994 1994. 85-90.

SCHILIT, B. N. 2011. Mobile Computing: Looking to the Future. *COMPUTER,* 44**,** 28-29.

SCHMIDT, A. 2000. Implicit human computer interaction through context. *Personal Technologies,* 4**,** 191-199.

SCHMIDT, A., BEIGL, M. & GELLERSEN, H.-W. 1999. There is more to context than location. *Computers &amp; Graphics,* 23**,** 893-901.

SESSIONS, R. 2007. *A Comparison of the Top Four Enterprise-Architecture Methodologies* [Online]. Available: *http://msdn.microsoft.com/en-us/library/bb466232.aspx* [Accessed 7 February 2012].

SHNEIDERMAN, B. & PLAISANT, C. 2004. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition),* Pearson Addison Wesley.

SIEK, K., ROGERS, Y. & CONNELLY, K. Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs. 2005. 267-280.

SIGNER, B., GROSSNIKLAUS, M. & NORRIE, M. C. 2007. Interactive paper as a mobile client for a multi-channel web information system. *World Wide Web-Internet and Web Information Systems,* 10**,** 529-556.

STUBBLEFIELD, N. B. 1908. *Wireless Telephone*. United States patent application.

TAPPERT, C. C., SUEN, C. Y. & WAKAHARA, T. 1990. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12**,** 787-808.

TINWALA, H. & MACKENZIE, I. S. Eyes-free Text Entry on a Touchscreen Phone. Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference, 2009. IEEE, 83-88.

TSIMHONI, O., SMITH, D. & GREEN, P. 2004. Address entry while driving: Speech recognition versus a touch-screen keyboard. *HUMAN FACTORS,* 46**,** 600-610.

VARSALUOMA, J. 2009. Scenarios in the Heuristic Evaluation of Mobile Devices: Emphasizing the Context of Use. *Evaluation***,** 332-341.

VAUGHAN-NICHOLS, S. J. 2009. Will Mobile Computing's Future Be Location, Location, Location? *COMPUTER,* 42**,** 14-17.

VOGEL, D. & BAUDISCH, P. Shift: a technique for operating pen-based interfaces using touch. Proceedings of the SIGCHI conference on Human factors in computing systems, 2007 San Jose, California, USA. 1240727: ACM, 657-666.

WEINSCHENK, S. & BARKER, D. 2000. *Designing Effective Speech Interfaces,* Wiley.

WHITE, G. M. 1976. Speech Recognition: A Tutorial Overview. *Computer,* 9**,** 40-53.

WIGDOR, D., FORLINES, C., BAUDISCH, P., BARNWELL, J. & SHEN, C. Lucid touch: a see-through mobile device. Proceedings of the 20th annual ACM symposium on User interface software and technology, 2007 Newport, Rhode Island, USA. 1294259: ACM, 269-278.

YUAN-KAI, W. Context awareness and adaptation in mobile learning. Wireless and Mobile Technologies in Education, 2004. Proceedings. The 2nd IEEE International Workshop on, 2004 2004. 154-158.

A. Appendix

The following table shows the list of rules created for usability evaluation of mobile touch-based devices. The list must not be considered complete. However, the list can be used as starting point for further development of smartphone- and tablet-tailored rules. Most of the rules are based on existing rules; however, many are slightly changed in order to match the modern world of mobile user interfaces. The source column specifies the author of the corresponding rule or of the rule which was used as starting point for the adapted rule. It cannot be excluded that certain rules are also mentioned by other authors than the authors specified in the list. However, for the author of this thesis the author(s) mentioned in the list was/were the primary source(s) for the corresponding rule.

Some rules were altered in a way that very concrete values were removed in order to make the rule more generic. Example: "*Are there no more than twelve to twenty icon types?"* was changed to "*Is the number of icon types limited?"* As the original rule was valid for desktop systems, it is disputable whether these values are still valid for modern UIs of mobile devices.

Some rules are based on the personal experience of the author of this thesis. The personal experience was collected either during conversations with colleagues, during reading feedback to different apps or during practical work on mobile app development.

More than 300 rules which are applicable in most situations were collected for the following list. However, it might be necessary to adapt or to reinterpret certain rules in order to match the requirements or the platform the software is developed for. Some rules might not be applicable at all for certain software. The expert usability evaluator has to make the proper decision how to apply the single rules.

ID	Source	Rule
C1 Feedback		
1	*Nielsen*	Is feedback timely and accurate?
2	*Pierotti*	Is there some form of system feedback for every user action?
3	*Nielsen*	Is feedback provided for all actions?
4	*Apple*	Is feedback subtle, but clear?
5	*Nielsen*	Is status information provided? / Is the user kept informed about what is giong on?
6	*Pierotti*	After the user completes an action (or group of actions), does the feedback indicate that the next group of actions can be started?
7	*Pierotti*	If there are observable delays in the system's response time, is the user kept informed of the system's progress?
8	*Pierotti*	Are response times appropriate to the task?
9	*Nielsen*	Is shown that input has been received?
10	*Nielsen*	Indicate progress in task performance
11	-	Are all and only useful features available for the current context?
12	*Pierotti*	Does the system provide visibility? That is, by looking, can the user tell the state of the system and the alternatives for action?
13	*Ji et al.*	Slider: Is the extent of increase and decrease that slider represents easy to recognize?
14	*Ji et al.*	Tabs: Is the visual discrimination between selected item and unselected item clear?
15	*Ji et al.*	Scroll bar: Is it possible to predict the quantity of contents through the scroll bar?
16	*Ji et al.*	Connect: Is connection success or failure properly notified?
17	*Ji et al.*	Connect: Is there a visual display to show the process of loading?

18	*Ji et al.*	Connect: Is loading/connection success or failure properly notified?
Captions and messages		
19	*Pierotti*	Does each window have a title?
20	*Pierotti*	Does every screen have a heading that describes the screen contents?
21	*Pierotti*	Do menu instructions, prompts, and error messages appear in the same place(s)?
22	*Pierotti*	In multipage data entry screens, is each page labeled to show its relation to others (page numbers, ...)?
23	*fluid*	Does a page's title accurately describe its purpose?
24	*Pierotti*	Is the menu-naming terminology consistent with the user's task domain?
25	*(fluid)*	Does every list entry have a visible unique ID (where applicable)?
Icons		
26	*Nielsen*	Are icons and other visual indicators shown?
27	*Pierotti*	Is there a consistent icon design schema and stylistic treatment across the app?
28		Does the icon design match the platform's icon design?
29	*Pierotti*	Are icons labeled?
30	*(Pierotti)*	Is the number of icon types limited?
Selections and visibility of UI state		
31	*Pierotti*	If multiple options can be selected in a menu or dialog box, is there visual feedback about which options are already selected?
32	*Pierotti*	Is there visual feedback when objects are selected or moved?
33	*Pierotti*	Is the current status of an UI object clearly indicated?
34	*Pierotti*	Do GUI menus make obvious which item has been selected?
35	*Pierotti*	Do GUI menus make obvious whether deselection is possible?

36	*Nielsen*	Direct manipulation: visible objects, visible results (Apple: Directly manipulate onscreen objects instead of using separate controls)
37	-	Is vibration or sound used to get feedback when a touch-button is pressed?
38	*Pierotti*	Is a single, selected item clearly visible when surrounded by unselected icons?

C2 Input		
1	-	Is the need for text input reduced to a minimum?
2	-	Are default values prefilled wherever possible?
3	-	Can suggestions for text entry fields be edited before submitting the query?
4	-	Are hardware keyboards supported?
5	-	Do all text boxes provide metadata which describes the expected content of the text box (email address, telephone number, normal text, …)
6	*Pierotti*	Are the valid answers for a question listed? (Nielsen: Provide lists of choices and picking from lists)
7	*Ji et al.*	When input text, is the indication of current input location appropriate?
Usage of hardware capabilities and sensors		
8	-	Are default values taken from external sensors (e.g. GPS coordinates, street names, etc.)?
9	-	Are useful suggestions proposed for text entry fields?
10	-	Is QR tag reading supported for text input?
11	-	Is Barcode reading supported for numeric input?
12	-	Is the GPS sensor used for prefilling text input boxes or for preselection?
13	-	Is direct camera access used to speed up image input?
14	-	Can voice recognition be used for text entry?

15	-	Is the device's hardware, such as proximity sensors or microphones, used for enhancing usability, where applicable?

C3 Match (Match between system and the real world)		
Conventions		
1	*Pierotti*	If shape is used as a visual cue, does it match cultural conventions?
2	*Pierotti*	Do the selected colors correspond to common expectations about color codes?
3	*Nielsen*	Follow real-world conventions
Familiarity		
4	*Pierotti*	Are icons concrete and familiar?
5	*Pierotti*	On data entry screens, are fields described in terminology familiar to users?
6	*Nielsen*	Is the conceptual model familiar to the user?
7	*Nielsen*	Is the user’s background knowledge used?
8	*Nielsen*	Learnable through natural, conceptual model
9	*Nielsen*	Does the screen representation match the real world?
Language, texts, and numbers		
10	*Pierotti*	For question and answer interfaces, are questions stated in clear, simple language?
11	*Pierotti*	When prompts imply a necessary action, are the words in the message consistent with that action?
12	*Nielsen*	Is the user’s language spoken and are familiar terms and natural language used?
13	*Nielsen*	Are metaphors from the real world used? (Apple: e.g. files and folders, playback controls, dragging gestures)
14	*fluid*	Is plain language used?
15	*fluid*	Is vocabulary familiar to the intended user, avoiding system-oriented terms?
16	*Pierotti*	Does the system automatically format numbers consistently?

Menus		
17	*Pierotti*	Are menu choices ordered in the most logical way, given the user, the item names, and the task variables?
18	*Pierotti*	If there is a natural sequence to menu choices, has it been used?
19	*Pierotti*	Do menu choices fit logically into categories that have readily understood meanings?
Forms and dialogs		
20	*Pierotti*	Do related and interdependent fields appear on the same screen?
21	*Pierotti*	Do GUI menus offer activation: that is, make obvious how to say "now do it"?
22	*Pierotti*	Has the system been designed so that keys with similar names do not perform opposite (and potentially dangerous) actions?
23	*fluid*	Is the order of information natural and logical?
Environment		
24	-	Are compass and GPS sensors used for establishing a relation between screen content and the real world?
25	-	Is augmented reality used?
Accessibility		
26	-	Are user interface elements (buttons, …) large enough?
27	-	Are gestures like swiping used?
28	-	If multi-touch gestures are used are the corresponding functions also reachable with single touches?
29	-	Are multi-touch gestures used for example spreading fingers for zooming?
30	-	Are available gestures obvious?
31	-	Is ambient light considered when displaying information on the screen?

C4 User control, Flexibility		
1	*Pierotti*	If the system has multipage data entry screens, can users move backward and forward among all the pages in the set?
2	*Pierotti/Nielsen*	Can users cancel out of operations in progress and is it possible to resume the operation later?
3	*Ji et al.*	Connect: Is there a way to stop the process of connection?
4	*Pierotti*	Can users reduce data entry time by copying and modifying existing data?
5	*Pierotti*	Are menus broad (many items on a menu) rather than deep (many menu levels)?
6	*Pierotti*	If the system has multiple menu levels, is there a mechanism that allows users to go back to previous menus and is it possible to change the earlier menu choice?
7	*Pierotti*	Can users move forward and backward between fields or dialog box options? (Keyboard provides "Next" button?)
8	*Nielsen*	Are there clearly marked exits in the app or during long operations?
9	*Nielsen*	Modelessness: Is users allowed to do what they want?
10	*Pierotti*	If the system supports both novice and expert users, are multiple levels of error message detail available?
11	*Pierotti*	Does the system allow novice users to enter the simplest, most common form of each command, and allow expert users to add parameters?
12	*Nielsen*	Accelerators should be provided
13	*Nielsen*	User tailorability to speed up frequent actions
14	*Nielsen*	System should be efficient to use
15	*Nielsen*	User interface should be customizable
16	*Nielsen*	Hardware button core functions should be supported
17	*Nielsen*	Interaction with the system should feel natural
18	*fluid*	Can repetitive actions or frequent activities be made easier?

19	*fluid*	Are there features (e.g. a site map, navigation bar) that help users find content and navigate?
20	*fluid*	Is there a home/start page link?
21	*Pierotti*	Can users choose between iconic and text display of information, where applicable?
22	*Pierotti*	If the system supports both novice and expert users, are multiple levels of detail available.
23	*Pierotti/Nielsen*	Are users the initiators of actions rather than the responders?
24	*Pierotti*	Does the system perform data translations for users?
25	*Pierotti*	Do field values avoid mixing alpha and numeric characters whenever possible?
26	*(Pierotti)*	When fields have fixed lengths does the cursor automatically jump to the next field when all fixed-length data was entered?
27	*Pierotti*	Do the selected input device(s) match user capabilities?
28	*Pierotti*	Are important keys larger than other keys? (Also applicable for on-screen keyboards)
29	*Pierotti*	Does the system correctly anticipate and prompt for the user's probable next activity?
30	-	Can long lists quickly be scrolled through?
31	-	Is it possible to directly jump to certain entries (e.g. starting letters) in ordered lists?
32	-	If a data connection is used, can the user decide whether to use the mobile data connection, Wi-Fi or another data connection (if available)?
33	-	Is it possible to deactivate vibration or sound feedback?
34	-	Does the application ask the end user before it sets a - in terms of time, money battery life or data volume - potentially costly action?
35	-	Is it possible to deactivate/configure security questions?
36	-	Do the standardized buttons (search, menu, home, back, volume control) work as intended by the platform?

37	*Pierotti*	For data entry screens, are partially filled screens saved?
38	*Android*	Is the state of the app preserved during unexpected interruptions and properly restored when resuming (e.g. phone calls)?
39	-	Is form input saved as intended by the platform? (e.g. is form input saved on pressing back-button? Or should it only bes saved when pressing an OK button?)
40	-	If an application partly depends on a functioning internet/network connection, can the non-network dependent functions be accessed anyway?
41	*Pierotti*	Are vertical and horizontal scrolling possible in each window (where applicable)?
System integration		
42	*Nielsen*	Is the application well-integrated in the rest of the system? (see also C8 and C9)
43	-	Does the design of the application match the rest of the system?
44	-	Is direct access to other apps possible (dialler, navigation, ...)?
45	-	Is sending a text message and/or calling a telephone number possible directly from the application?
46	-	Is sending an email to an email address possible directly from the application?
47	-	Is navigation to addresses supported directly from the application by using the appropriate app?
48	-	Are home screen widgets provided, where useful?
49	-	Are operating system notifications/messages properly handled?

C5 Undo/Reversal		
1	*Pierotti*	Can users easily reverse their actions? (Nielsen: "Forgiveness: make actions reversible")
2	*Pierotti*	Is there an "undo" function at the level of a single action, for single data entries, and/or for a complete group of actions?

3	*Pierotti*	Do functions that can cause serious consequences have an undo feature?
4	*Pierotti*	If the system allows users to reverse their actions, is there a retracing mechanism to allow for multiple undos?
5	*Nielsen*	If undo is supported, redo should be supported as well (where applicable)
6	*Nielsen*	Is it obvious how to undo actions?

C6 Consistency (and standards)		
Standards		
1	*Pierotti*	Have industry or company formatting standards been followed consistently in all screens within a system?
2	*Pierotti*	Have industry or company standards been established for menu design, and are they applied consistently on all menu screens in the system?
3	*Nielsen*	Is the system conform to platform interface conventions?
4	*fluid*	Are well-known pictures and symbols used? (e.g. "?" for help, "<" for back, "i" for information).
5	-	If gestures are used, are common gestures used instead of proprietary gestures? (Also, don't override default system gestures)
Attention		
6	*Pierotti*	Are attention-getting techniques used with care?
7	*(Pierotti)*	Is blinking content avoided?
8	*(Pierotti)*	Is the number of font sizes and styles limited?
9	*Microsoft*	Always use the system font unless brand guidelines mandate divergence.
10	*(Pierotti)*	Is the number of different colors limited?
11	*Pierotti*	Sound: Are soft tones used for regular positive feedback, harsh for rare critical conditions?

12	*Pierotti*	Are attention-getting techniques used only for exceptional conditions or for time-dependent information?
13	*Pierotti*	Have pairings of high-chroma, or spectrally extreme colors been avoided?
14	*Pierotti*	Are high-value, high-chroma/intense colors used to attract attention?
Labels, texts, and numbers		
15	*Nielsen*	Consistency: Are same things expresses in the same way?
16	*Pierotti*	Has heavy use of all uppercase letters on a screen been avoided?
17	*fluid*	Is the use of all-upper-case text limited to acronyms?
18	*Pierotti*	If applicable, are integers right-justified and real numbers decimal-aligned?
19	*Pierotti*	Are field labels consistent from one data entry screen to another?
20	*Pierotti*	Do field labels appear to the left of single fields and above list fields?
21	*Pierotti*	Is a legend provided if color codes are numerous or not obvious in meaning?
22	*Pierotti*	Do on-line instructions appear in a consistent location across screens?
23	*Pierotti*	Is the most important information placed at the beginning of prompts or messages?
24	*Pierotti*	Are user actions named consistently across the system?
25	*Pierotti*	Are entities named consistently across the system?
26	*Pierotti*	If the system has multipage data entry screens, do all pages have the same title?
27	*Pierotti*	If the system has multipage data entry screens, does each page have a sequential page number?
28	*Nielsen*	Show similar info at same place on each screen
29	*fluid*	Do links match destination titles?

30	*Microsoft*	Button control text should be concise and typically be a verb
Menus, commands, and navigation		
31	*Pierotti*	Are menu items presented and aligned in the same way?
32	*Pierotti*	Are menu choice names consistent, both within each menu and across the system, in grammatical style and terminology?
33	*Pierotti*	Are commands used the same way and do they mean the same thing in all parts of the system?
34	*fluid*	Is navigation consistent and intuitive?
35	*fluid*	Are similar operations or tasks performed in similar ways?
36	*Nielsen*	Consistency: Do same things look the same?
37	*Android*	Are context menus (right click on desktop systems, long tap on many touch-based mobile OS) only used as shortcuts? I.e., are all context menu actions also available at other locations?
38	*Android*	In context menus: Are all selection-specific commands separated from global commands?
39	*Android*	In context menus: Are the most frequently used commands placed first?
40	*Android*	Do context menus identify the selected item? (e.g. "Edit contact", not just "Edit")
Platform consistency		
41	-	Do the device's (hardware) navigation buttons work as expected?
42	-	If available, do hardware buttons (search, volume, back) behave as expected / as intended by the platform? (Android: Don't take over the *back* key unless you absolutely need to)
43	*Android*	Notifications and App Widgets should provide consistent back behavior (mostly Android specific)
44	*Android*	Are the operating system's notification and messaging services used rather than self-developed systems?

C7 Error handling and prevention		
Error prevention		
1	*Pierotti/Nielsen*	Does the system prevent users from making errors whenever possible?
2	*Pierotti*	Are menu choices logical, distinctive, and mutually exclusive?
3	*Pierotti*	Are data inputs case-blind whenever possible?
4	*Pierotti*	Are the buttons that can cause the most serious consequences located far away from low-consequence and high-use buttons?
5	*Pierotti*	Does the system warn users if they are about to make a potentially serious error?
6	*Pierotti*	Does the system intelligently interpret variations in user input? (Nielsen: Understand the user's language)
7	*Pierotti*	Do text input fields indicate the number of characters left, if there is a maximum number of characters?
8	*Pierotti*	Do fields in data entry screens and dialog boxes contain default values when appropriate?
9	(fluid)	Do UI components provide help or hints about the expected input?
10	fluid	Is confirmation required when an action is difficult or impossible to undo?
11	*Pierotti*	Are users prompted to confirm commands that have drastic, destructive consequences?
12	*(Pierotti)*	Is it possible to add multiple datasets on a single screen?
Error messages		
13	*Pierotti*	Are prompts stated constructively, without overt or implied criticism of the user?
14	*Pierotti*	Do prompts imply that the user is in control?
15	*Pierotti*	Are prompts brief and unambiguous.
16	*Pierotti*	Are error messages worded so that the system, not the user, takes the blame?

17	*Pierotti*	If humorous error messages are used, are they appropriate and inoffensive to the user population?
18	*Pierotti*	Are error messages grammatically correct?
19	*Pierotti*	Do error messages avoid the use of exclamation points?
20	*Pierotti*	Do error messages avoid the use of violent or hostile words?
21	*Pierotti*	Do error messages avoid an anthropomorphic tone?
22	*Pierotti*	Do all error messages in the system use consistent grammatical style, form, terminology, and abbreviations?
23	*Pierotti*	Do messages place users in control of the system?
24	*Pierotti*	Do error messages inform the user of the error's severity?
25	*Pierotti*	Do error messages suggest the cause of the problem?
26	*Pierotti*	Do error messages indicate what action the user needs to take to correct the error?
Error recovery		
27	*Pierotti*	If an error is detected in a data entry field, does the system place the cursor in that field or highlight the error?
28	*Pierotti*	If the system supports both novice and expert users, are multiple levels of error-message detail available?
29	*fluid*	Can users easily recover from errors, unintended actions, or actions that did not lead to desired results (eg undo, back)?

C8 Short-term memory (Recognition rather than recall)		
1	*Nielsen*	Is the users' memory load minimized?
2	*fluid*	Does the application support user memory?
3	*(Nielsen)*	Is the principle of seeing-and-touching followed rather than remembering-and-typing?
4	*Nielsen*	Is direct manipulation supported? (visible objects, visible results)
5	*Pierotti*	Are prompts, cues, and messages placed where the eye is likely to be looking on the screen?

6	*Pierotti*	Is there an obvious visual distinction made between "choose one" menu and "choose many" menus/lists?
7	*Pierotti*	Have frequently confused entity/data pairs been eliminated whenever possible?
8	*Nielsen*	Is the repertoire of available actions made salient?
9	*fluid*	Are illustrations provided for important concepts?
10	-	If gestures are not obvious, is the number of different gestures limited and are they easy to remember?
11	-	Is there a quickly accessible documentation?
12	*(Nielsen)*	Does the application cooperate well with other applications? (e.g. direct dialling of phone numbers instead of remembering them and copying them manually, see also C4)
13	-	Are graphical representations preferred to textual representations, where applicable? (e.g. phone unlocking via drawing a visual pattern rather than entering a character sequence (password))
Forms and dialogs		
14	*Pierotti*	Is all data a user needs visible at each step in a multistep data entry screen?
15	*Pierotti*	Does the system gray out or hide inactive or useless UI controls? (Android: Dim or hide menu items that are not available in the current context)
16	*Pierotti*	Are items grouped into logical zones, and are headings used to distinguish between zones?
17	*Pierotti*	Are zones/groups clearly separated by borders, lines, color, letters, bold titles, rules lines, or shaded areas?
18	*Pierotti*	Are size, boldface, underlining, color, shading, or typography used to show relative quantity or importance of different screen items (if applicable)?
19	*Pierotti*	Do data entry screens and dialog boxes indicate when fields are optional? Are optional data entry fields clearly marked?

20	*Apple*	Do controls look tappable?
Labels, icons, and colors		
21	*Pierotti*	Is color coding consistent throughout the system?
22	*Pierotti*	Is color used in conjunction with some other redundant cue?
23	*Pierotti*	Are field labels close to fields?
24	*Nielsen*	Show icons and other visual indicators
25	*Pierotti*	Is the first word of each menu choice the most important?

C9 Design (Aesthetic and minimalist design)		
1	*Nielsen*	Is it easy to discriminate action alternatives?
2	*fluid*	Is no unnecessary material displayed ("clutter")?
3	*fluid*	Can users suppress distracting effects? (sound, penetrantly recurring message boxes, ...)
4	-	Can touchable regions (buttons, …) easily be recognized? / Is it easy to recognize that a certain area is touchable?
5	-	Is the contrast high enough to be able to read the display even outside or in a bright environment?
6	-	Does the user interface react immediately after input?
7	-	Are animations and transitions smooth?
8	*fluid*	Can foreground elements - either text or images - be easily distinguished from the background?
9	*fluid*	Is there adequate contrast within images?
10	*fluid*	Are related UI elements grouped?
11	*Microsoft*	Do map controls and long lists fill the whole available screen space in order to avoid excessive scrolling?
12	*Microsoft*	Is indicated that pages can be switched by swiping to the right or to the left by showing parts of the content which will appear after swiping to the corresponding direction?
13	*Nielsen*	Is the application well-integrated in the rest of the system and does the design match the rest of the system? (see also C4)

14	-	Is an adaptive user interface used which is adapted automatically at runtime based on the application's context?
15	*Apple*	Are standard UI controls preferred to customized controls wherever possible?
Lables and texts		
16	*Pierotti*	Is only (and all) information essential to decision making displayed on the screen?
17	*Pierotti*	Does each data entry screen have a short, simple, clear, distinctive title?
18	*Pierotti*	Are field labels brief, familiar, and descriptive?
19	*fluid*	Are sentences and paragraphs short and to the point?
20	*Pierotti*	Are prompts expressed in the affirmative, and do they use the active voice? (to continue with the desired operation the user should have to answer an "are you sure?" prompt with "yes")
21	*Pierotti*	Are menu titles brief, yet long enough to communicate?
Icons, images, and colors		
22	*Pierotti*	Are all icons in a set visually and conceptually distinct?
23	*Pierotti*	Does each icon stand out from its background?
24	*Pierotti*	Is each individual icon a harmonious member of a family of icons?
25	*Pierotti*	Has excessive detail in icon design been avoided?
26	*Pierotti*	Has color been used with discretion?
27	*Pierotti*	Has color been used specifically to draw attention, communicate organization, indicate status changes, and establish relationships?
28	*Apple, Android*	Are high resolution images available which are suitable even for large screen sizes? Therefore the images must usually be provided in different resolutions.
Input		
29	*Pierotti*	Are typing requirements minimal in forms?

30	-	Are sliders, radio buttons or other touchable controls preferred to text input?
31	-	Are radio buttons preferred to dropdown boxes in case of a low number of available choices?
32	*Pierotti*	Are the most frequently used functions in the most accessible positions?
33	*Pierotti*	Does the system complete unambiguous partial input on a data entry field?
Performance		
34	-	Does the app not waste battery power by performing unnecessary or unoptimized operations?
35	-	Does the app not waste battery power by not disabling unused sensors?
Screen orientation and size / support for different screen and device types		
36	-	Is the device's orientation sensor used to adapt the UI to the device's orientation?
37	-	Do all screens (activities) support both portrait and landscape screen orientation?
38	-	Are multiple screen sizes supported (including tablets and smartphones)?
39	-	Are uncommon screen sizes and ratios supported?
40	-	Are multiple (secondary) screens supported?
41	-	Is the application optimized all supported screen/device types?
42	*Apple*	Is scrolling preferred over reducing the content size?

C10 Documentation (Help and documentation)		
1	*Pierotti*	If menu items are ambiguous, does the system provide additional explanatory information when an item is selected?
2	*Pierotti*	Are there memory aids for commands, either through on-line quick reference or prompting?

3	*Pierotti*	Is the help function visible; for example, a key labeled *Help* or a special menu?
4	*Pierotti*	Is the help system interface (navigation, presentation, and conversation) consistent with the navigation, presentation, and conversation interfaces of the application it supports?
5	*fluid*	Is help information and/or documentation easy to search?
6	*Pierotti*	Help system's navigation: Is information easy to find?
7	*Pierotti*	Help system's presentation: Is the visual layout well designed?
8	*Pierotti*	Help system: Is the information accurate, complete, and understandable?
9	*Pierotti*	Help system: Is the information relevant?
10	*fluid*	Is help information focused on the user's task?
11	*Pierotti*	Help information: Goal-oriented (What can I do with this program?)
12	*Pierotti*	Help information: Descriptive (What is this thing for?)
13	*Pierotti*	Help information: Procedural (How do I do this task?)
14	*Pierotti*	Help information: Interpretive (Why did that happen?)
15	*Pierotti*	Help information: Navigational (Where am I?)
16	*Pierotti*	Is there context-sensitive help?
17	*Pierotti*	Can the user change the level of help information detail?
18	*Pierotti*	Can users easily switch between help and their work?
19	*Pierotti*	Is it easy to access and return from the help system?
20	*Pierotti*	Can users resume work where they left off after accessing help?
21	*fluid*	Are illustrations provided to make help instructions easier to understand?

C11 UI Testing		
1	*Raymond & Landley*	The interface isn't finished until the end-user testing is done.
2	*Apple*	Were custom UI elements user-tested thoroughly?