

The User-Centred Prototyping of Mobile Applications by Example of the Third Computing Paradigm

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

eingereicht an der

Fakultät für Angewandte Informatik der Universität Augsburg

von

Frau Dipl. Medieninf. Karin Bee

Juni 2012

Erstgutachter:	Prof. Dr. Elisabeth André
Zweitgutachter (extern):	Prof. Dr. Albrecht Schmidt
Drittgutachter:	Prof. Dr. Rudi Knorr
Termin der mündlichen Prüfung:	29.06.2012

Für meine Schwester Sandra.

Abstract

This dissertation covers the user-centred prototyping process of mobile applications in the context of ubiquitous computing systems - *the Third Computing Paradigm*.

First, this dissertation addresses a requirement analysis and requirement specification of the tool-supported user-centred prototyping process. To meet these requirements, new conceptual and technical approaches are introduced and discussed. These approaches also were implemented within the development of the user-centred prototyping (UCP) tool MoPeDT in order to validate their feasibility.

Another part of the dissertation introduces a method to validate the meeting of the non-functional requirements. By means of the introduced tool MoPeDT and the evaluation method, a tool study is also presented to reveal benefits and problems of a tool, such as MoPeDT.

The last part of the dissertation aims at the tool-supported conduction of user evaluations in different test settings. Among others, a new setting - the tool-supported hybrid world simulation - is hereby introduced and discussed.

Zusammenfassung

Diese Dissertation beschäftigt sich mit der nutzerzentrierten Prototypenentwicklung von mobile Anwendungen im Kontext von allgegenwärtigen Computersystemen - dem dritten Computerparadigma.

Zunächst beschreibt die Dissertation eine Anforderungsanalyse und -spezifikation zur werkzeugunterstützten, nutzerzentrierten Prototypenentwicklung. Darauf folgend werden neue konzeptionelle und technische Ansätze vorgestellt und diskutiert, die bei der Einhaltung der Anforderungen helfen können. Diese wurden auch im Rahmen der Entwicklung des Tools MoPeDT implementiert, um ihre Realisierbarkeit zu überprüfen.

Ein weiterer Teil der Arbeit stellt eine Methode zur Validierung der Einhaltung der nichtfunktionalen Anforderungen vor. Mit dem vorgestellten Werkzeug MoPeDT und der Evaluationsmethode wird dann eine Werkzeugstudie durchgeführt, um Vor- und Nachteile eines Werkzeugs wie MoPeDT zu ermitteln.

Der letzte Teil der Arbeit stellt die werkzeugunterstützte Durchführung von Nutzerevaluationen in unterschiedlichen Testumgebungen vor. Dabei wird unter anderem eine neue Umgebung - die werkzeugunterstützte Hybride Weltsimulation - vorgestellt und diskutiert.

Danksagung

Mein besonderer Dank gilt meiner Betreuerin und Erstgutachterin Frau Prof. Dr. Elisabeth André, die mir diese Promotion ermöglicht hat und mich immer hilfreich unterstützt hat. Ich habe sehr viel von ihr gelernt. Auch gilt mein Dank meinem zweiten Gutachter Herrn Prof. Dr. Albrecht Schmidt, der mir immer sehr wichtige Hinweise und Verbesserungen für meine Promotion geben hat. Bedanken möchte ich mich außerdem bei Herrn Prof. Dr. Rudi Knorr für die Drittbegutachtung der Arbeit.

Danke auch an meine Kollegen vom Lehrstuhl für Human-Centered Multimedia für die gute Zusammenarbeit und die anregenden Gespräche. Ein besonderes Dankschön verdient vor allem meine langjährige Zimmerkollegin Thurid Vogt, die immer ein offenes Ohr für mich hatte. Wir hatten wahrlich immer sehr viel Spaß. Neben Thurid möchte ich mich auch besonders bei Stephan Hammer und Ekaterina Kurdyukova bedanken. Wir haben zusammen für das Verständnis von Usability Engineering gekämpft. Auch Matthias Rehm bin ich sehr dankbar für die tolle Zusammenarbeit im Besonderen bei der Erforschung der hybriden Weltsimulation. Im Rahmen meiner Arbeit in den Projekten e-CIRCUS, IRIS und OC-Trust habe ich sehr viel gelernt. Danke an alle Kollegen in diesen Projekten.

Eine Promotion ist ohne die Unterstützung von Studenten kaum möglich. Insbesondere Dennis Erdmann, Eva Lösch, Sebastian Thomas, Michael Ailinger, Katrin Schmitt und Julia Karcher danke ich vielmals für ihre tolle Arbeit.

Ein ganz besonderer Dank gilt meinem Mann Nikolaus, der mich nicht nur in der Arbeit als Kollege sondern vor allem privat unterstützt. Danke, dass du immer für mich da bist. Neben Nikolaus gilt mein Dank auch meinen Eltern Erich und Helga und meinen Geschwistern Andreas, Corinna und Mathias. Ihr gebt mir den bestmöglichen Rückhalt. Zum Schluss möchte ich mich noch bei meinen Freundinnen Patrizia, Manuela und Renate bedanken, die mich während der Promotionszeit aufgebaut haben und die mir immer viel Kraft und Freude geben. Danke!

Contents

- 1 Introduction 1**
 - 1.1 Research Questions 3
 - 1.2 Content Overview 5

- 2 Basic Concepts 7**
 - 2.1 The User-Centred Development 7
 - 2.1.1 The Human-Centred Design Process 8
 - 2.1.2 The User-Centred Prototyping Process 18
 - 2.2 The Third Computing Paradigm 19
 - 2.2.1 Ubiquitous and Pervasive Computing 19
 - 2.2.2 Mobile Phones as Interaction Devices 21
 - 2.3 Software Support 26
 - 2.3.1 IDEs with Software Modules and Middleware 26
 - 2.3.2 Classical Prototyping Tools 27
 - 2.3.3 Content and Behaviour Tools 28
 - 2.3.4 Evaluations and Analysis Tools 29
 - 2.3.5 User-Centred Prototyping Tools 30
 - 2.4 Summary 31

- 3 Requirement Engineering 33**
 - 3.1 The Mobile Application Development 33
 - 3.1.1 Mobile Interaction for Ambient-Assisted Living 34

3.1.2	Mobile Interaction for Mobile Outdoor and Indoor Aides . . .	38
3.1.3	Mobile Interaction for Mobile Learning and Entertainment . . .	41
3.2	Requirements	47
3.2.1	Non-Functional Requirements for Prototypes (NFR-P)	48
3.2.2	Functional Requirements for Prototypes (FR-P)	48
3.2.3	Non-Functional Requirements for the Tool (NFR-T)	50
3.2.4	Functional requirements for the tool (FR-T)	52
3.3	Related Tools	59
3.3.1	Tools for the Design Specification and Implementation	59
3.3.2	Tools for the Evaluation and Analysis	62
3.3.3	Tools for all Phases	66
3.3.4	The Tool Comparison	69
3.4	Summary	73
4	The Tool Development	75
4.1	The automatic Generation of Prototypes	75
4.1.1	The Software Architecture and its Components	76
4.1.2	The Software Modules	82
4.2	The User-Centred Prototyping Tool	94
4.2.1	The Design of Prototypes	94
4.2.2	The Evaluation of Prototypes	101
4.2.3	The Analysis of Evaluations	105
4.3	Summary	106
5	The Empirical Tool Validation	109
5.1	Related Studies	109
5.1.1	Related Methods	110
5.1.2	Discussion	111
5.2	Tool Study with MoPeDT	112

5.2.1	The Experimental Setting	112
5.2.2	Conducting the Pilot Tool Study	117
5.2.3	Conducting the Main Tool Study	118
5.2.4	Results of the Tool Study	119
5.3	Summary	126
6	Tool-Supported Empirical Evaluations	129
6.1	A Real World Evaluation	130
6.1.1	An Example of a Real World Evaluation	130
6.1.2	A Discussion about Real World Evaluations	133
6.2	A Real World Simulation	135
6.2.1	An Example of a Real World Simulation	135
6.2.2	A Discussion about Real World Simulations	138
6.3	A Virtual World Simulation	139
6.3.1	The Concept of Hybrid World Simulations	140
6.3.2	Tool-supported Hybrid World Simulations	141
6.3.3	Studying the Concept of Hybrid World Simulation	144
6.3.4	A Discussion about Hybrid World Simulations	152
6.4	Summary	153
7	Final Conclusion and Outlook	155
7.1	Research Contribution	155
7.2	Future Work	161
A	Interface Guidelines	163
B	Task Description	165
B.1	Task Description for the traditional Approach	165
B.2	Task Description for MoPeDT	168
C	Protocol	171

D Questionnaire	173
------------------------	------------

Bibliography	185
---------------------	------------

List of Figures

- 2.1 The five process steps of the Human-Centred Design process [Maguire, 2001]. 9
- 2.2 High-fidelity design specification of the appearance and behaviour of a mobile application. 14
- 2.3 Four main steps of the user-centred prototyping process. 17
- 2.4 The figure illustrates the three different computing paradigms with the mainframe era as *the First Computing Paradigm* where several users used together a single computer unit. This *First Computing Paradigm* started in the 1950s and was followed by the personal computing (PC) era - *the Second Computing Paradigm* - in the 1980s. Since the 2000s, *the Third Computing Paradigm* enters the user’s life. . 20
- 2.5 An example of the Data Matrix (left) and the QR Code (right). Both visual markers represent the text: *Karin*. 23
- 2.6 Screen view of the GUI builder from Netbeans for the development of the mobile application’s appearance. 27
- 2.7 Flow view of the GUI builder from Netbeans for the development of the mobile application’s behavior. 28
- 3.1 A user who performs *touching* [Rukzio et al., 2006]. 34
- 3.2 A user who performs *pointing* [Rukzio et al., 2006]. 35
- 3.3 Different screens of the low-fidelity prototype [Leichtenstern et al., 2006]. 36
- 3.4 Different screens of *the Restaurant Guide* [Schmitt, 2008]. 40
- 3.5 The first row on the left part shows Setting 1 and the *The Role Assignment Condition* while the right part shows Setting 2 where a single phone is available for the group. The second row shows Setting 3 with a phone per group member and all functionalities on the phones. 44

3.6	A screen of the application which displays the question and the four different answers (left). The public display which provided additional information to the children (right).	45
3.7	Impressions of the user evaluation performed with <i>the World Explorer</i>	46
4.1	The software architecture of MoPeDT consisting of physical objects, mobile users, a main server and a database as well as sensors, actuators and evaluators.	76
4.2	Examples of <code>ItemScreens</code> which are supported by the module <code>MOBILE CLIENT</code> . These screens can be used to generate different kinds of menu screens.	87
4.3	Examples of <code>MediaScreens</code> which are supported by the module <code>MOBILE CLIENT</code> . These screens can be used to generate different input and output screens for multimedia content.	88
4.4	Examples of <code>AlertScreens</code> which are supported by the module <code>MOBILE CLIENT</code> . These screens can be used to generate different alert screens: info, waiting or error screens.	89
4.5	Examples of <code>TabScreens</code> which are supported by the module <code>MOBILE CLIENT</code> . This screen block contains several other types of screens: <code>ItemScreens</code> , <code>MediaScreens</code> , <code>AlertScreens</code> or <code>SketchScreens</code>	89
4.6	The screens which are generated based on the static content specification of the prototype's appearance (see Program 5): <code>ItemScreen</code> (left) and <code>MediaScreen</code> (right).	91
4.7	Visual representation of Program 6: Screen 2 is loaded once the user has selected <code>SELECT</code> by means of the keyboard.	92
4.8	The dynamically generated screen of Program 8. A <code>ItemScreen</code> is generated that displays different physical objects of the just selected pervasive computing environment.	93
4.9	The graphical user interface of MoPeDT for the specification of the prototype's general settings. The upper part of the component displays the settings for the prototype (e.g. application name and icon) while the lower part aims at the specification of the server - server address and port.	95

-
- 4.10 The graphical user interface of MoPeDT for the specification of the mobile interaction styles. The left part of the component displays the tree view of all specified mobile interaction styles and their defined context values while the middle part of the component provides options to modify the mobile interaction styles and their values - add, update and remove. The right part provides options to define context values for other components of the architecture - other mobile users and sensors - which are of interest to the intended prototype. 96
- 4.11 The graphical user interface of MoPeDT for the specification of a static and dynamic appearance and behaviour by means of a state-chart view. The left part of the component displays the state-chart view of the prototype which contains the display of screen states and transitions while the right part of the design component provides a preview of the selected screen and options to modify the appearance and behaviour. 98
- 4.12 The graphical user interface of MoPeDT for the specification of a static and dynamic appearance and behaviour by means of a tree view. The left part of the component displays the tree view of the prototype which contains the screens, their widgets and transitions while the right part of the design component provides a preview of the selected screen and options to modify the appearance and behaviour. 99
- 4.13 The graphical user interface of MoPeDT for the specification of the database. The left part of the component displays the tree view of the database elements - the environments, objects, services and service entries - while the right part of the component provides options to modify - add, update and remove - the content of the corresponding database element. 100
- 4.14 The graphical user interface of MoPeDT for the conduction of a user evaluation. The upper part displays the two selected cameras while the lower part shows the selected mobile user and sensors as well as incoming messages from the server. The right part displays the cloned screen view of the selected mobile user. 102
- 4.15 The graphical user interface of MoPeDT for the recording of live comments and details about the tasks. The upper part displays the supported tasks while the lower part provides the input of live comments. 103

4.16	The graphical user interface of MoPeDT for the synchronous display of all recorded data. The upper part displays the captured videos and screen shots of the prototype's appearance (right). The lower part provides a time-line-based visualisation of the audio track as well as the labelling - annotation - by means of the logged quantitative data.	105
5.1	User ratings of effectiveness and efficiency. The test users rated the provided statements based on a scale from 1 (strongly disagree) to 5 (strongly agree). In terms of design specification, efficiency and time gain were significantly better rated for MoPeDT compared to the traditional approach (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).	120
5.2	Screen shots of a prototype that was developed with MoPeDT.	122
5.3	Screen shots of a prototype that was developed with the traditional approach.	122
5.4	User ratings of satisfaction, user control and learnability. The test users rated the provided statements based on a scale from 1 (strongly disagree) to 5 (strongly agree). In terms of design specification, satisfaction and user control were significantly better rated for the traditional approach compared to MoPeDT while learnability was better rated for MoPeDT (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).	123
5.5	Distribution of the preferred approach for the design specification. Most test users liked both approaches (8) and MoPeDT (7) for the design specification phase.	125
5.6	Distribution of the preferred approach for the evaluation and analysis. Most test users liked MoPeDT (11) or both approaches (5) for the evaluation and analysis phases.	125
5.7	Distribution of the preferred component of MoPeDT. Most test users liked all components of MoPeDT: the design, evaluation and analysis component.	126
6.1	Screens of the prototype that was developed with MoPeDT. The first screen is the main menu followed by the profile and help screen. The last two screens can either be used to load the stored content of a user and his community or to add new multimedia content.	132

-
- 6.2 Screens of the prototype to load multimedia content of a specific user. In this case a video content is loaded. 133
- 6.3 Screens of the prototype to store new multimedia content. In this case a new video content is stored. 133
- 6.4 Screens of the prototype that was developed with MoPeDT. The user can apply the application in order to get knowledge about a DVD and later on pay it. 136
- 6.5 A user while interacting with the keyboard-based (left) or NFC-based (right) mobile interaction style. 137
- 6.6 The setting of the real world simulation. The test user sits in front of the DVDs (left) while the evaluator controls MoPeDT's evaluation component (right). 137
- 6.7 The extended software architecture of MoPeDT consists of a main server, a database as well as evaluators alike the general architecture of MoPeDT. The mobile users have a virtual representation: the avatars. The physical objects are virtualised. Sensors and actuators can be virtually and/or really. 141
- 6.8 Degression model for signal strength of Wifi access points. Example for access point 2. 143
- 6.9 The virtual world simulation of the living room in Second Life. 147
- 6.10 Once the user has selected the physical object by one of the mobile interaction styles, the service entries for the selected object are displayed that then can be requested and changed. 147
- 6.11 Screens which are displayed when performing the keyboard-based mobile interaction style. The user first selects the environment and then the physical object. 148
- 6.12 Performing the laser beam-based mobile interaction style. The user applies the avatar and the virtual mobile phone to point at the virtual physical object in order to select it. 149
- 6.13 Performing the NFC-based mobile interaction style. The user applies the avatar and the virtual mobile phone to get very close to the virtual physical object in order to touch and select it. 150

List of Tables

3.1 The table compares the different introduced tools (see Section 3.3) and presents how well they meet the functional requirements towards a tool-support for the user-centred prototyping process (-: poor, 0: medium, +: good). These tools primarily assist during the mobile application development. **[FR-T01]** Support the design specifications during the entire development process. **[FR-T02]** Support the automatic generation of evolutionary software prototypes with low and high level of fidelity for the intended interaction and presentation device - which meet all non-functional [NFR-P] and functional requirements [FR-P] towards the resulting prototype. **[FR-T03]** Support the conduction of local and remote empirical evaluations - synchronously and asynchronously during the entire development process. **[FR-T04]** Support analytical evaluations - model-based and inspection-based - during the entire development process. **[FR-T05]** Support analyses of the captured data during the entire development process. 68

3.2 The table compares the different introduced tools (see Section 3.3) and presents how well they meet the functional requirements for the design phase (-: poor, 0: medium, +: good). **[FR-T01]** Support the design specifications during the entire development process. **[FR-T01.1]** Support the static and dynamic specification of the prototype’s appearance and behaviour. **[FR-T01.2]** Support the specification and use of different mobile interaction styles. **[FR-T01.3]** Support the automatic compliance of approved HCI guidelines. **[FR-T01.4]** Support the specification of remote data. 70

- 3.3 The table compares the different introduced tools (see Section 3.3) and presents how well they meet the functional requirements for the evaluation phase (-: poor, 0: medium, +: good). **[FR-T03]** Support the conduction of local and remote empirical evaluations - synchronously and asynchronously during the entire development process. **[FR-T03.1]** Support the display and the synchronised logging of quantitative data which are rather objective. **[FR-T03.2]** Support the display and the capturing of qualitative data which are rather objective. **[FR-T03.3]** Support the display and the synchronised logging of qualitative and quantitative data which are rather subjective. **[FR-T03.4]** Support the capturing of several users. 71
- 3.4 The table compares the different introduced tools (see Section 3.3) and presents how well they meet the functional requirements for the analysis phase (-: poor, 0: medium, +: good). **[FR-T05]** Support analyses of the captured data during the entire development process. **[FR-T05.1]** Support the synchronised display of all captured data as well as the automatic pre-annotation of the qualitative data by means of the quantitative data. **[FR-T05.2]** Support interactivity to analyse and modify the data. **[FR-T05.3]** Support the statistic analysis of the captured data. **[FR-T05.4]** Support the loading and display of several captured sessions. 72
- 6.1 Results of the referenced real world simulation (first row) and the hybrid world simulation (second row) (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$). 151

Chapter 1

Introduction

In terms of mobile computing there is currently one clear trend towards the use of smart phones and mobile applications. In November 2010¹, 23 percent of all Germans owned a smart phone which meant an increase by 65 percent between November 2009 and November 2010.

There are several reasons for the success of smart phones². One important reason is that the users are now tending to use the Internet mobile, such as for online shopping, navigating, social networks or just for browsing. On the one hand this trend is caused by the new price policy of the mobile phone carriers and the affordable mobile Internet fees. On the other hand this is also caused by the improved hardware capabilities of the mobile devices which enable a much better Human-computer interaction. Smart phones, for instance, characteristically provide larger displays compared to former mobile phones. Grounded on these displays and their larger presentation area, more information can be provided to mobile users more detailed with a higher resolution which powers a smart phone to a promising presentation device. Further, smart phones offer new and more intuitive types of input techniques which also make smart phones to interesting interaction devices. The display, for instance, can be used as touch screen in order to input text with a virtual keyboard. Also, the built-in microphone, accelerometer or camera can be used as input channels.

In South Korea, for instance, mobile users can use their mobile phones and their cameras to order different products on the go in a subway³. The concept is as followed: a virtually displayed shelf presents different products of an online shop. The users simply have to use their smart phones to take a picture of a QR code that

¹http://blog.base.de/boom_im_deutschen_smartphone-markt/

²<http://bazonline.ch/digital/mobil/Zehn-Gruende-fuer-den-Erfolg-des-iPhone/story/12511902>

³<http://www.zeit.de/digital/mobil/2011-07/homeplus-ubahn-onlineshop>

is presented together with a product (see Section 2.2.2) which enables the identification and order of the corresponding product. Later on, the selected products will be delivered to the users' home. Based on this service, the online shop could increase its sales by 130 percent between November 2010 and January 2011. Besides these visual identification techniques (e.g. QR codes), there is also a trend to use Near Field Communication (NFC) (see Section 2.2.2) in order to select objects by means of RFID tags. One idea is to use these NFC-based interactions for mobile payment⁴. Users simply apply their smart phones as a kind of credit card. They touch a reader with their phone and then the price of a product is automatically charged at the user's account.

The example from South Korea just illustrates one possible mobile application. For the different types of smart phones, there is currently a huge number of available mobile applications on markets. In June 2011, the market for the Apple iOS platform, for instance, provided 367.334 so-called Apps for a free advertising-financed or charged download while the market for the Google Android platform offered 206.143 Apps⁵. Mobile application development means a new branch of industry with a promising future. But at that point a serious problem arises. Developers of these mobile applications require comprehensive Software Engineering skills to quickly generate highly reliable and functional applications. To solve that problem, a trend goes towards supporting developers with software tools which limit the developer's required experiences and skills. Google, for instance, developed the tool called App Inventor⁶ which enables the quick and easy generation of applications for the Google Android platform. One concept of the tool is to use a visual programming language called Open Blocks developed by [Roque, 2007] which hardly requires programming skills.

But not only Software Engineering skills are required to build competitive products, also Usability Engineering skills are needed since a further and very important reason for the success of mobile applications is their good usability and attractive design⁷. Knowledge about the intended users, their tasks and their context of use is inevitable to build highly user-centred products (see Section 2.1). Mobile applications, for instance, can be used under different contextual conditions (e.g. indoor or outdoor) which can cause different user requirements towards the applications⁸. Developers of mobile applications have to take these aspects under consideration in order to develop a profitable product. A tool can also provide

⁴<http://www.computerwoche.de/netzwerke/mobile-wireless/2486721/>

⁵<http://www.zdnet.de/news/41552245/app-markt-android-ueberholt-iphone-bis-juli.htm>

⁶<http://appinventor.googlelabs.com/about/>

⁷<http://www.useit.com/alertbox/mobile-usability.html>

⁸http://www.eresult.de/studien_artikel/forschungsbeitraege/app-usability.html

assistance with regards to Usability Engineering, such as by supporting the conduction of user evaluations to find typical problems with regards to usability.

All in all, more and more people use smart phones and different mobile applications. This trend can provide new market opportunities and sources of income for industry. For the development of these mobile applications, there seems to be a tendency towards using a tool-based assistance for the quick and easy design and implementation of mobile applications as well as the conduction and analysis of user evaluations. By this tool-support, time and money can be saved to generate a highly functional, robust and usable application. However, less knowledge is provided for developers of such tools about their generic requirements and approaches to meet the requirements. Further, less experience is available about concrete benefits and problems of such tools.

This dissertation aims at the investigation of the tool-supported user-centred prototyping of mobile applications in the context of ubiquitous computing systems - *the Third Computing Paradigm*. Mobile applications in the context of *the Third Computing Paradigm* (see Section 2.2) are characteristically applications where the mobile phone is used as an interaction and presentation device to a so-called pervasive computing environment. The virtual shelf with the displayed products and QR codes in South Korea is an example of such a pervasive computing environment where mobile phones are used for the interaction and presentation.

1.1 Research Questions

When investigating the tool-supported user-centred prototyping of mobile applications in the context of ubiquitous computing systems - *the Third Computing Paradigm*, the following different research questions need to be answered.

- 1. How can developers of mobile applications in the context of *the Third Computing Paradigm* be supported during the design specification and implementation of mobile prototypes?**

The design and the implementation of mobile applications in the context of *the Third Computing Paradigm* require elaborated knowledge of Software and Usability Engineering. For instance, the developers have to consider the correct application of well-established interface guidelines during the generation of their prototypes in order to reduce usability errors.

Within this dissertation different conceptual and technical approaches are investigated and implemented which enable the tool-supported automatic

generation of prototypes. The use of screen templates, for example, is investigated in order to ensure the compliance with approved interface guidelines. All of the applied approaches are expected to reduce the developer's required programming and interface design skills and simultaneously to increase the developers' efficiency and effectiveness.

The different approaches and their feasibility are illustrated based on the implementation of the user-centred prototyping (UCP) tool called MoPeDT - Pervasive Interface Development Toolkit for Mobile Phones. Also, a tool study with MoPeDT sheds light on benefits and problems of the used approaches.

2. How can developers of mobile applications in the context of *the Third Computing Paradigm* be supported during the conduction and analysis of evaluations?

MoPeDT does not only illustrate conceptual and technical approaches approaches for the design and implementation of prototypes in the context of *the Third Computing Paradigm* but also for the conduction and analysis of evaluations with the generated prototypes.

The goal is to also improve the developer's efficiency and effectiveness when conducting and analysing evaluations. Further, the developer's required skills should also be reduced in this domain. For example, approaches are required that provide options to systematically track and capture an end-user while interacting with the system even if the test user is spatially or temporally separated from the evaluator.

3. How can such a support also be provided for prototypes that need to be evaluated in different test settings?

A special challenge of this dissertation is to support software assistance for the conduction of evaluations in wild and on the go. Mobile users apply the application under different contextual constraints, such as alone at home or together in a group on a festival. Developers of mobile applications need to have knowledge about the realistic use of an application in order to comprehensively address the users' needs. This knowledge can primarily be acquired by executing real world evaluations in the field. Conducting only these real world evaluations, however, can be very time-consuming and expensive. Thus, real or virtual simulations of the real world setting are also often conducted since even thereby knowledge can be acquired about the users and their application use, such as a user preference towards an interaction style.

For the different mentioned test settings, the use of user-centred prototyping (UCP) tools is investigated and discussed within this dissertation. By this means, knowledge is provided to tool users how UCP tools can support the conduction of real world evaluations in the field and real world simulations in the laboratory. Further, a tool-support for a new kind of test setting is investigated: the hybrid world simulation. A hybrid simulation means a combination of the real and virtual world. The user interacts with a virtual simulation of the pervasive computing environment by means of a real interaction and presentation device - a real mobile phone. Thus, at first glance, comparable knowledge might be provided about the user by a hybrid world simulation while money can be saved since the pervasive computing environment is only virtually available. This assumption was investigated within this PhD project as well as further benefits but also problems of a tool-supported hybrid world simulation.

The questions one and two are considered in Chapter 4 and Chapter 5. The last question is covered in Chapter 6.

1.2 Content Overview

After having described the research questions, the remaining dissertation aims at answering the questions. The dissertation is structured as followed.

Chapter 2 presents relevant basic concepts. First, the Human-centred Design process and its different steps are illustrated. In this context, the user-centred prototyping process is covered which addresses the last two steps of the Human-centred Design process: the step to produce design solutions - prototypical applications - and the step to evaluate the design solutions against requirements. Besides aspects of the user-centred development of applications, Chapter 2 also provides details about the basic idea of *the Third Computing Paradigm* and how mobile devices can be used as interaction and presentation devices to a pervasive computing environment. Finally, different categories of software support are presented in order to assist the user-centred prototyping process of mobile applications.

Chapter 3 aims at the analysis and specification of functional and non-functional requirements towards a user-centred prototyping (UCP) tool and its resulting prototypes. In order to finally define these requirements, different mobile applications in the context of *the Third Computing Paradigm* are presented and analysed

on typical requirements. Further, related software tools for the development of prototypes are discussed as well as either tools for the conduction and analysis of evaluations or for all phases of the user-centred prototyping process.

Chapter 4 goes into the details how the different mainly functional requirements towards the tool and the resulting prototypes can be fulfilled by means of conceptual and technical approaches. First, the automatic generation of prototypes is aimed and thereby Software Engineering aspects, such as an architecture is introduced that can be used to generate mobile applications in the context of *the Third Computing Paradigm*. Then, MoPeDT and the development of its different components are described. MoPeDT is a user-centred prototyping tool that can be used to perform the user-centred prototyping process in the context of mobile HCI: the design specification of prototypes and their automatic generation as well as the conduction and analysis of evaluations.

Chapter 5 describes a method to validate the meeting of the non-functional requirements. This method is aimed to provide a quality-based validation of the tool-supported user-centred prototyping process as well as the resulting prototypes. Based on this method, MoPeDT was validated to find typical benefits and problems of a user-centred prototyping tool. Further, new approaches of Chapter 4 are also validated.

Chapter 6 discusses the use of a tool for the conduction of empirical evaluations in different test settings and their typical benefits and problems. First, common settings are discussed: a real world evaluation in the field and a real world simulation in a laboratory. Then, the hybrid world simulation and its implementation are presented. This setting was investigated in a first user study on its typical benefits and limitations. Chapter 6 provides details about this user study and the findings.

Chapter 7 finally presents a conclusion on the research questions of Chapter 1. Thereby, the contributions of this dissertation are highlighted for the research community. In the end, an outlook is given on future work.

Chapter 2

Basic Concepts

This chapter presents basic concepts of the user-centred development of interactive systems, different steps of the Human-Centred Design process and their purposes. Based on the Human-Centred Design process, the concept of the user-centred prototyping process is introduced as well. Additionally, the basic idea of *the Third Computing Paradigm* is presented where mobile phones are used as interaction and presentation devices to a pervasive computing environment. Finally, this chapter also introduces categories of software support that might be useful for simplifying the user-centred prototyping of mobile applications in the context of *the Third Computing Paradigm*.

2.1 The User-Centred Development

When developing interactive computing systems, users need to be taken into account in order to get knowledge about their goals and desires as well as mental and physical skills and limitations. Hereby, the interactive system can be tailored exactly to the users' requirements which, in turns, can increase the users' efficiency, effectiveness and satisfaction when interacting with the system. If a system comprehensively supports an efficient, effective and satisfying use, it is meant to provide a high level of usability (DIN EN ISO 9241 part 11) which is an important quality criterion of a system. Besides the DIN EN ISO 9241 specification, there are further definitions of usability. [Nielsen, 1993], for instance, considers the system's ease of learnability and memorability as well as a low error rate as further attributes of a proper usability in addition to a high level of efficiency and satisfaction.

A high level of usability can provide several benefits for the end-users and the developers of the interactive system [Maguire, 2001]. It can increase the produc-

tivity of end-users since they do not need to concentrate on the system and its use but instead on their concrete tasks. A high level of usability can also reduce errors while completing a task. A further benefit is a reduced need for service support and training periods if a system provides a good user-friendliness by being self-explanatory. Also, the user's acceptance and willingness to spend money and time for the system can be improved if a system follows an appealing design and usability. Finally, a good usability can enhance the reputation of the product and its vendors which, in turns, can increase sale figures.

Usability Engineering processes [Faulkner, 2000] can help develop interactive systems with a high level of usability. Characteristically, they actively involve the end-users into the development process of a system to get a clear picture of the users' requirements and their tasks, but the kind of user involvement can be differently. The Participatory Design process (e.g. [Kensing and Blomberg, 1998] or [Muller et al., 1993]) and the Human-Centred Design process (e.g. [Hanington, 2003] or [Maguire, 2001]) are examples of Usability Engineering processes which can be applied to reach a good usability.

In the Participatory Design process, the end-users are characteristically part of the development team and equal partners. They participate all steps of the development process which, in particular, includes the implementation and evaluation phase of the system. The application of the Participatory Design process provides continuous feedback and input of the end-users, their goals and desires, however, this continuous involvement of end-users can be very difficult to organise and coordinate as well as very expensive. A compromise is the application of the Human-Centred Design process which only iteratively involves end-users during the phases to analyse the context of use and requirements as well as the evaluation phase of the interactive system. Thus, compared to the Participatory Design process, development costs and the time effort can be reduced since the end-users are not involved during all steps, such as not during the implementation step and the specification step. At the same time, useful knowledge about the end-users can still be acquired. As a consequence of its benefits, the Human-Centred Design process is an often used approach to built user-friendly interactive systems.

2.1.1 The Human-Centred Design Process

[Maguire, 2001] provides a comprehensive overview about the Human-Centred Design process and its five different phases (see Figure 2.1) which need to be considered and conducted: (1) plan the Human-Centred Design process, (2) under-

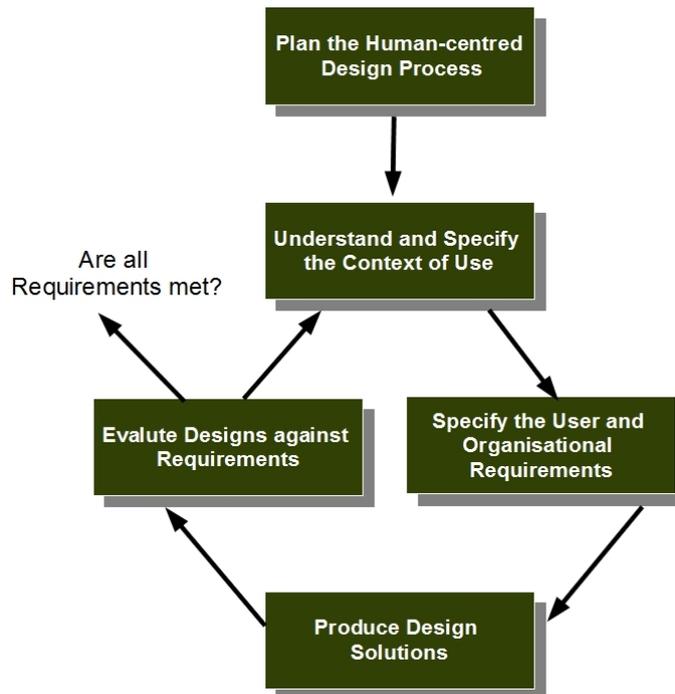


Figure 2.1: The five process steps of the Human-Centred Design process [Maguire, 2001].

stand and specify the context of use, (3) specify the user and organisational requirements, (4) produce design solutions and (5) evaluate the design solutions against requirements until all requirements are met. In the following, these five phases are presented and typical methods are discussed which can be applied during the different process phases.

First Step: Plan the Human-Centred Design process

Before starting the iterative Human-Centred Design (HCD) process, several questions need to be considered and answered which address the usability planning and scoping as well as the usability cost-benefit analysis [Maguire, 2001].

For the usability planning and scoping, the potential users and their expected objectives, skills and experiences need to be estimated and defined. Additionally, a first assumption has to be specified about the users' potential tasks and their environments. With regards to the users' environments, for instance, the emerging options and limitations towards the intended product need to be taken into account, such as about existing networks in the environments. Finally, usability goals should be considered as well. For instance, the importance of the ease of use or the minimisation of errors could be fixed as the most important usability goals.

An important goal of the development process is the reduction of development

costs and time in order to keep an economic cost-benefit value. Thus, the development team also has to review possible process steps and methods based on their probable costs with their expected benefits. Based on this analysis, the different selected methods have to be fixed for the following process phases.

Second Step: Understand and Specify the Context of Use

After having specified methods for the different phases of the HCD process and sketched a first picture of the intended target group, an in-depth knowledge about the users and the overall context of use has to be acquired and specified in the next step of the Human-Centred Design process.

To achieve this aim, one approach is to first specify the different stakeholders of the system and later on to get in touch with them in order to receive information about their objectives, skills and experiences as well as their workplaces. By having this knowledge, the development team's assumptions of the first phase either are proved or need to be corrected and extended. One option of getting in touch with the users is the execution of a survey with them. [Rogers et al., 2002] provide information about how to collect detailed user data based on a survey, such as by conducting interviews. When executing a face-to-face interview, an interviewer asks a test user pre-defined or spontaneous questions about different topics, such as about the user's goals and desires. Typically, surveys easily and quickly provide data about the users, but the data are just subjective attitudes which might be less valuable since they might not correspond to the users' actual objective behaviour.

Instead of survey methods, observation methods [Rogers et al., 2002] can provide objective data about the users' actual behaviour. Typically, the users are unobtrusively observed (e.g. by stationary cameras and microphones) while interacting under real contextual constraints which provides very realistic and valuable data about the users, their tasks and their environments. Despite this promising benefit, the classical observations, however, are often difficult to organise and execute in the user's real environment as well as time-consuming and error-prone to analyse since the recorded data are often qualitative (e.g. audio and video files). These qualitative data need to be interpreted on specific behaviour to enable a quantified analysis, such as how often the users behaved in a certain manner. A solution to simplify the interpretation of the recorded qualitative data is using *the Method of Thinking Aloud* [Nielsen, 1993] where users verbalise their thoughts while interacting in their environments which provides valuable knowledge about the user's mental and physical skills and limitations in different situations. The method, however, also has drawbacks since verbalising thoughts is less intuitive and unobtrusive which can lead to a changed user behaviour and thus biased data.

In contrast to the classical observation methods, there are also methods where the users themselves capture data. These methods can be easier to organise and conduct compared to classical rather stationary observation methods as well as less expensive. Diary studies are an example where users document their everyday life by recording rather objective and qualitative data about specific situations, such as by making pictures or notes about their eating habits. One well-known diary study is called *Cultural Probes* [Gaver et al., 1999] where users are equipped with a bag that contains different objects for the documentation, such as a camera or a pen and paper. Mobile phones [Hammer et al., 2010] also can be used as devices to execute a diary study by using the phone's built-in hardware for the documentation, such as the camera and microphone. These kind of studies are typically so-called remote studies which are characterised by a spatial and temporal separation of the probands and evaluators in a field setting [Andreasen et al., 2007]. A further well-known remote diary study is called *Experience Sampling* [Consolvo and Walker, 2003]. In contrast to *Cultural Probes*, *Experience Sampling* is event-driven by the evaluators. Thus, not the proband decides on situations for capturing data but the system and the evaluator. The probands receive reminders from time to time in order to capture data about their current situation. For instance, the reminder can occur every morning and evening. A benefit of *the Experience Sampling Method* is that the users hardly will forget to capture data since they are reminded. These alarms, however, also might happen in inappropriate situations, such as when being in the cinema. A further problem of *the Experience Sampling Method* is that very interesting situations might be missed due to the fact that not the participant of the study decides on the right moment for capturing data but instead of the system and the evaluator.

In practice, surveys and observations are often together sequentially or parallelly executed by first unobtrusively recording objective data about the user behaviour and later on by jointly reviewing the recorded data and asking the users about subjective explanations of a specific behaviour. Another often applied method called *Contextual Inquiry* [Holtzblatt and Beyer, 1996] also combines a survey with an observation method. Test users are interviewed while interacting under real contextual constraints in their environment.

Results of the second phase of the Human-Centred Design process are the identification and documentation of the users' significant attributes and their tasks. Additionally, technical, physical and organisational aspects of the environments are documented, such as information about the equipment (e.g. hardware and network availabilities), disturbances (e.g. loudness) or communication structures beyond the users.

Third Step: Specify the User and Organisational Requirements

Followed by the second phase, the third phase of the HCD process considers the specification of the user and organisational requirements. This phase can start with a further user survey to get additional details about user requirements. Interviews or *Focus Groups* [Nielsen, 1993] are examples to discuss about requirements that need to be fulfilled by the system. *Focus Groups* bring together a group of about five users for discussions simultaneously which can save time and money compared to a face-to-face interview where only one user can be interviewed at the same time. Despite this benefit, however, opinion makers can appear in a *Focus Group*. These opinion makers can influence other participants, such as useful attitudes of more reserved users can be lost. Thus, a moderator needs to consider that problem. In addition to the conduction of such a survey method, existing systems of a competitor or a previous system can be used and analysed together with the users in order to detect further user requirements towards the intended product.

Scenarios of use (e.g. [Nielsen, 1993]) and *Personas* (e.g. [Cooper et al., 2007]) are often used techniques to specify the users and their requirements. Scenarios describe the typical use of the system, knowledge about the user, task goals and the environments as well as details about objects and contexts that emerge during the use. They provide a common language for all members of the multi-disciplinary development team. *Personas* give a clear picture about the different user groups. Each *Persona* illustrates a fictive representative of a single group with their typical demographic data, objectives, skills and experiences. Apart from *Personas* and scenarios of use, a further document can contain a concrete list of user and organisational requirements. User requirements, for instance, address the functional (e.g. the desired task-support) and non-functional requirements (e.g. usability requirements) towards the systems that need to be fulfilled to meet the users' needs and goals.

Fourth Step: Produce Design Solutions

After having insights about the context of use and the requirements, design solutions can be produced. Typically for the first iteration of the HCD process, the phase starts with discussions about the existing specifications (e.g. the list of requirements) to come along with a first idea about the implementation of a product. This idea is often specified by making use of *Storyboards* [Landay and Myers, 1996]. *Storyboards* visualise the basic concept of the application and its use by a sequence of images. After having specified the concept of the application and validated

it with end-users, two subtasks are typically conducted during the fourth phase of the HCD process. First a design solution of the intended product is specified and later on the specification is implemented in order to generate a prototypical solution of the design idea.

Design specification: Before implementing a design solution, the application's appearance and behaviour are often specified by making use of different kinds of diagrams or models. The process of a design specification can help understand the domain, keep the application simple and prevent implementation errors since a clear structure of the application is provided. *Affinity Diagrams* [Beyer and Holtzblatt, 1997] or *Card Sorting* [McDonald and Schvaneveldt, 1988], for instance, can be used to visualise the hierarchical application flow of the intended product. For both techniques, the use of sticky notes is often used in order to document screens and functions and later on to group them.

For mobile applications, the design specification is often done by defining the application's different screens and the application flow which typically looks comparable to a *State-Transition Diagram* [Wellner, 1989]. In this diagram, each state represents a screen of the mobile application and from each of these screen states user interactions can call other screen states. Consequently, user interactions (e.g. the execution of a keyboard command) are represented by transitions in the model. With respect to the level of detail, the design specification can be modelled with low or high fidelity. Low-fidelity models are often specified with pen and paper (e.g. *Card Sorting*) since thereby the models can quickly and easily be changed. High-fidelity models are often designed with the support of graphical tools [Rogers et al., 2002]. Their look and feel is usually quite similar to the final product. At early stages of the Human-Centred Design process, the design specification characteristically is rather low in detail as in later iterations. Figure 2.2 illustrates a high-fidelity design specification of a mobile application.

Implementation: The implementation of the previous design specification generates prototypes. In terms of interface design, a prototype represents a partial simulation of a product with respect to its final appearance and behaviour [Houde and Hill, 1997]. Prototypes can be classified by their level of detail, range of supported functions and their reusability.

Similar to the design specification, the prototypes can be implemented with a low or high level of detail (e.g. [Nielsen, 1993, Rogers et al., 2002]). Low-fidelity prototypes are prototypes with sketched screens which can be implemented by pen and paper - paper prototypes [Rettig, 1994] - whereas high-fidelity prototypes have a very similar look-and-feel compared to the final product. Typically, high-fidelity prototypes are running applications for the intended interaction and presentation

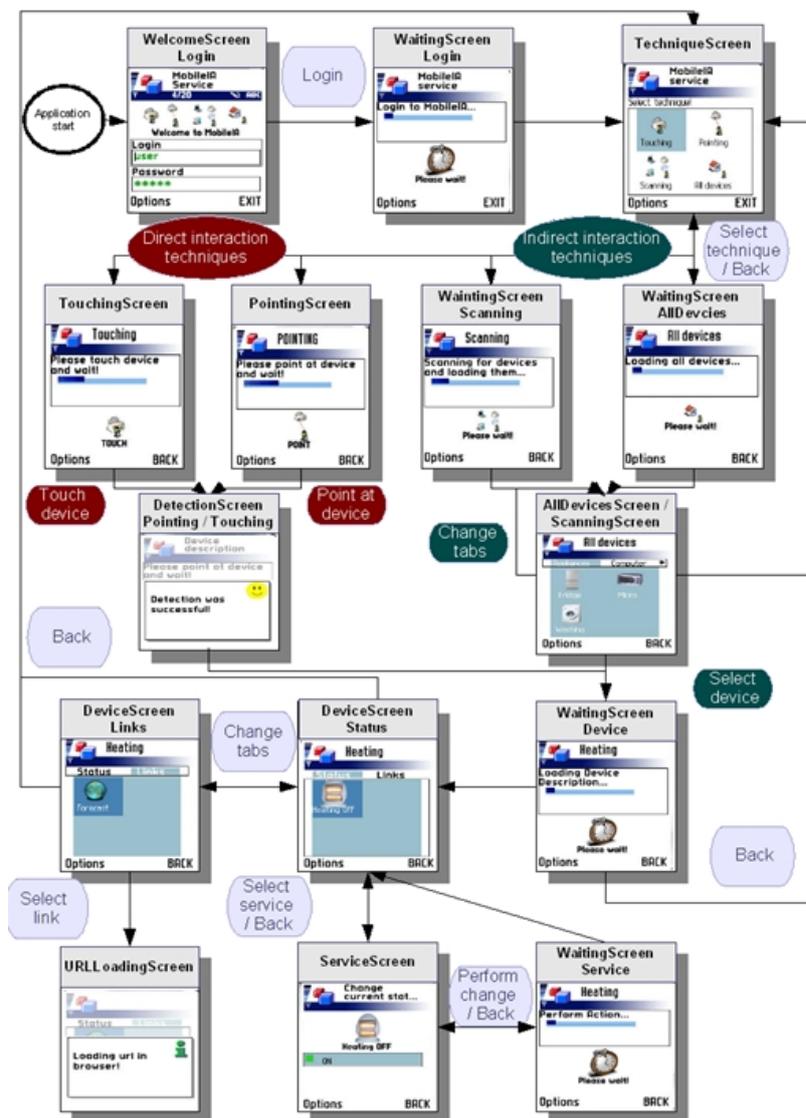


Figure 2.2: High-fidelity design specification of the appearance and behaviour of a mobile application.

device (e.g. mobile phones). For mobile applications, prototypes can be implemented for different platforms (e.g. J2ME, Android or Windows Mobile) with the support of an integrated development environment (e.g. Eclipse, Netbeans or Visual Studio).

Low and high-fidelity prototypes are usually restricted by their range of supported functions. Their supported functions can be limited vertically or horizontally (e.g. [Nielsen, 1993, Rogers et al., 2002]). Horizontal prototypes provide an overview about all functionalities but do not provide these functionalities in-depth whereas vertical prototypes provide functionality in-depth but not for all functionalities. If developers, for instance, want to test the overall navigation and the supported scope of features (e.g. for a website), a horizontal prototype is typ-

ically used (e.g. the homepage of a website). The different features, however, are not completely implemented and cannot be comprehensively tested during user evaluations. Horizontal prototypes are often used in very early phases of the HCD process whereas vertical prototypes are usually used later once a specific supported task needs to be optimised. Highly functional prototypes typically focus on providing both the overview about all supported functionalities and the functionalities implemented in-depth.

A further prototype classification aims at the prototype's reusability in the following iterations [Davis, 1992] of the Human-Centred Design process. Throwaway prototypes are not reused and modified. For example, paper prototypes are often thrown away after a single iteration. Instead of reusing the prototype, knowledge of the evaluation is used to generate a complete new prototypical solution. In contrast to throwaway prototypes, evolutionary prototypes are reused, modified and retested with experts and end-users in several iterations until they meet all requirements of the end-users [Davis, 1992, Davis et al., 1988]. Software prototypes are characteristically used as evolutionary prototypes.

In practice, throwaway prototypes in the form of paper prototypes are often applied at the beginning of the development process to reduce inhibition thresholds to change prototypical solutions whereas evolutionary prototypes in the form of software prototypes are normally used later on in the process once the developers have a clearer picture about the user requirements. Then, the developers can concentrate on iteratively optimising the software prototypes towards the user's needs.

Fifth Step: Evaluate Design against Requirements

After having implemented a prototypical design solution, it is evaluated against the specified requirements. This fifth and last phase of the Human-Centred Design process also can be split in two subtasks. First an evaluation has to be conducted which is followed by the analysis of the captured data whether all user and organisational requirements are fulfilled.

Evaluation: The implemented prototype is either evaluated with end-users - empirical evaluations - or with experts - analytical evaluations [Somervell and McCrickard, 2004]. The selection of appropriate empirical and analytical evaluation techniques [Duh et al., 2006] often depends on the evaluation goals and the used prototypes (e.g. their level of detail and supported level of functionality).

Empirical evaluations: When conducting empirical evaluations, the end-users usually apply the implemented prototypes to either execute pre-defined

tasks or to freely use them. During this use, observation techniques [Rogers et al., 2002] are applied in order to record different objective data of the user evaluation, such as quantitative data about the user behaviour by means of cameras and microphones or quantitative data (e.g. the user interactions) by means of logging mechanisms. These methods are similar to observation methods which are applied in the second and third phase of the Human-Centred Design process if knowledge about the context of use and the requirements need to be determined. In contrast to these phases, during the last phase of the Human-Centred Design process, the objective is often to test an existing system - the prototypical solution of the intended product - to find problems in terms of the specified requirements which need to be fixed by means of a new iteration of the HCD process. In combination with observation techniques, survey techniques (e.g. interviews or questionnaires) are also often applied to gather subjective data [Rogers et al., 2002]. The subjective data can help interpret the gathered objective data, such as why the users preferred a specific functionality of the prototype.

Empirical evaluations can be conducted in different test settings: (1) as a real world evaluation in the user's real setting - the field, (2) as a real world simulation in a laboratory by physically re-building the real world setting or (3) as a virtual world simulation in a virtual environment by virtually re-building the real world setting. The tool-supported execution of empirical evaluations in these different test settings is one main objective of this dissertation. More about the different test settings, their benefits and problems and how they can be conducted based on a tool-support is described in Chapter 6.

Analytical evaluations: Besides empirical evaluations, analytical evaluations are also typically applied when developing an interactive system by means of the HCD process. Experts with regards to Usability Engineering and/or the application domain either use inspection methods [Nielsen, 1993] or methods which are rather formal-analytical and base on empirically validated models in order to find usability problems. GOMS [John and Kieras, 1996] (Goals, Operators, Methods and Selection Rules) is a well-known example of a method that makes use of an empirically validated model. Based on this model, the required execution time can be determined for each supported task of a prototype and thus problems with regards to efficiency. Therefore, the goals of the users are identified together with the methods to fulfil the goals and selection rules which methods are preferred in which situation. Finally, the operators are defined which need to be completed for each method. By means of the operators, the completion time of a task can be estimated for the different methods. The empirically validated model provides the required knowledge about the completion time for each operator, such as the required time to press a key or to select a button.

Inspection methods can either be executed guideline-based or walkthrough-based. *The Heuristic Evaluation* [Nielsen, 1993], for instance, is an guideline-based inspection method. Based on a list of well-known heuristics, the prototypes are reviewed to find violations against them and thus usability problems. After the inspection, the problems and possible solutions are communicated as direct input to the interface developers. The other kind of inspection methods are walkthrough-based methods where interface developers typically simulate empirical evaluations by interacting with the system like an end-user and completing different tasks in order to reveal usability problems. *The Cognitive Walkthrough* [Nielsen, 1993] is such a walkthrough-based inspection method. In practice, guideline-based and walkthrough-based methods are often conducted together (e.g. the Heuristic Walkthrough [Sears, 1997]) to get a better picture about the prototype and its potential usability problems.

Analysis: The last step of the HCD process' fifth phase is the analysis of the objective and subjective data which were gathered during the conduction of empirical and analytical evaluations [Rogers et al., 2002]. The objective of the analysis is to reveal whether all user and organisational requirements are fulfilled, such as whether the prototype provides a high level of usability or not.

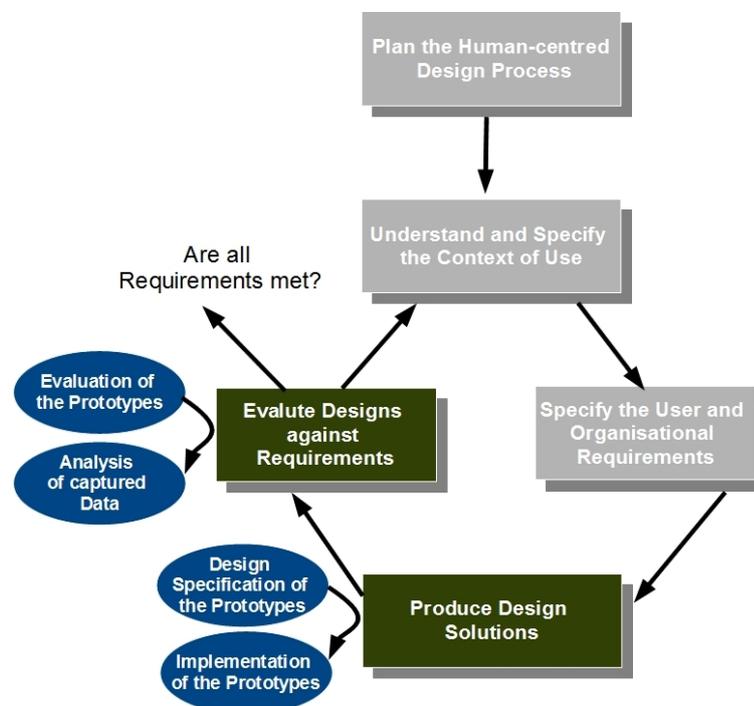


Figure 2.3: Four main steps of the user-centred prototyping process.

The gathered objective and subjective data either provide qualitative or quantitative content. Quantitative content (e.g. time or error measurements) can easily

be analysed by different analysis techniques, such as statistical analyses. Consequently, by means of quantitative data, different questions of the development team can be answered, such as whether the end-user is significantly more efficient when using the new prototype for the completion of a task compared to an older version. In contrast to quantitative content, qualitative content (e.g. audio and video files) frequently needs to be quantified by using so-called annotation schemas. During the process called annotation, these schemas provide information about the characteristics which need to be identified and labelled for the qualitative data, such as specific user behaviour (e.g. body movements). The results of the annotation reveal where, when, how often and how long the intended characteristics were recognised in the qualitative data. Now, the quantified data also can be analysed statistically in order to find significant differences between two prototypes, such as a different level of user involvement.

After the analysis and interpretation of the gathered data, a new iteration of the HCD process is started if the requirements of the users and organisation are still not completely fulfilled otherwise the HCD process terminates.

2.1.2 The User-Centred Prototyping Process

The different phases of the Human-Centred Design process can be split in two main tasks for interface developers: (1) the context of use and requirement analysis and specification as well as (2) the iterative user-centred prototyping.

The second and third phase of the HCD process - (i) understand and specify the context of use and (ii) specify the user and organisational requirements - aim at the context of use and requirement analysis and specification. Users are typically observed and surveyed without the intended product in order to get knowledge about their goals and desires, their tasks and their environments. This knowledge and the resulting requirements are specified as input for the user-centred prototyping (UCP) process.

The user-centred prototyping process aims at the last two phases of the Human-Centred Design process: (i) the production of a design solution as well as (ii) its evaluation against the requirements. Thus, the user-centred prototyping process is characterised by the iteratively conducted rapid implementation and evaluation of design solutions with the involvement of end-users and experts. Characteristically, four subtasks need to be completed during the conduction of the user-centred prototyping process (see Figure 2.3): (a) the design specification and (b) implementation of prototypes as well as (c) the evaluation of the prototypes and (d) the analysis of the resulted data of the evaluation.

This dissertation aims at the second main task of the Human-Centred Design process: the user-centred prototyping process for mobile applications by example of *the Third Computing Paradigm*.

2.2 The Third Computing Paradigm

This section presents the basic concept of *the Third Computing Paradigm* and the use of a mobile phone as interaction and presentation device to a pervasive computing environment.

2.2.1 Ubiquitous and Pervasive Computing

[Weiser, 1991] presented the concept of *Ubiquitous Computing*. Similar concepts to *Ubiquitous Computing* are *Pervasive Computing*, *Ambient Intelligence* or *the Internet of Things* (e.g. [Satyanarayanan, 2001, Aarts et al., 2001, Mattern and Flörkemeier, 2010, Gershenfeld et al., 2004]). The basic idea of all these concepts is the omnipresence of computers and computerised objects in the user's everyday environment. Based on [Kindberg et al., 2002], computerised objects can be classified as people, places and things. The users should be enabled to interact with these objects or more general with everything, everywhere at any time which also means a paradigm shift (see Figure 2.4) from a one-to-one relationship between a user and his personal computer - *the Second Computing Paradigm* - to a many-to-one relationship between a single user and many different computers or computerised objects - *the Third Computing Paradigm*.

When applying the concept of *the Third Computing Paradigm*, it is differed how users interact with their computerised - pervasive computing - environment and its objects. Users can either explicitly interact with the system based on, for instance, direct manipulations of objects in the pervasive computing environment or implicitly by changing the context of the environment [Schmidt, 2000].

Explicit interactions [Schmidt, 2000] are actively executed by the users with the system, such as the selection of a button by clicking it with a computer mouse. These explicit interactions can also be categorised whether they are executed directly or indirectly. Users, for instance, can directly interact with objects in the pervasive computing environment if they provide an interactive user interface (e.g. an interactive table) for interactions and presentations. If an object does not have an interactive user interface, a user has to apply an interaction and presentation device as a medium which provides the required interactive user interface in order

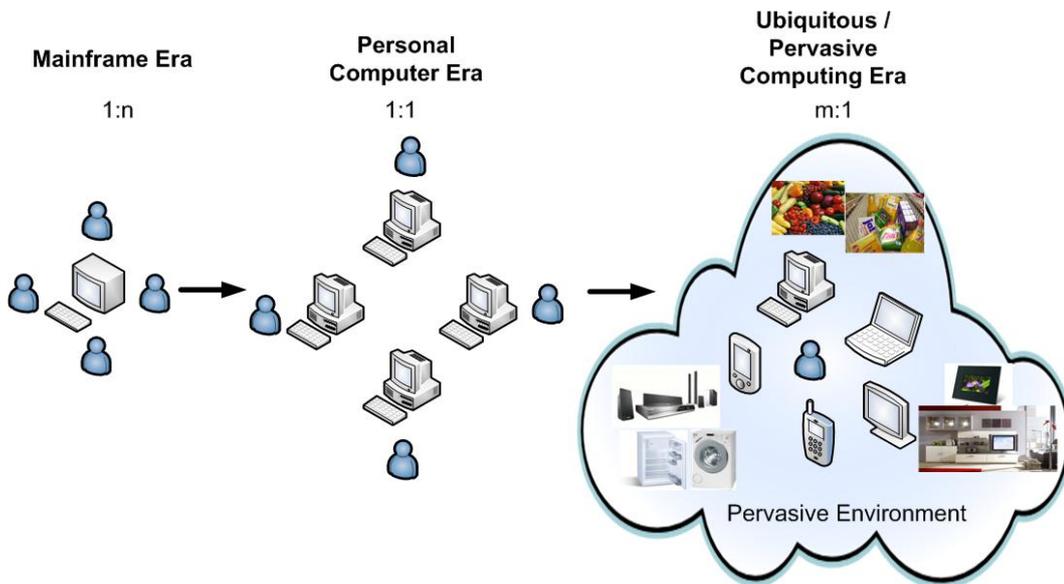


Figure 2.4: The figure illustrates the three different computing paradigms with the mainframe era as *the First Computing Paradigm* where several users used together a single computer unit. This *First Computing Paradigm* started in the 1950s and was followed by the personal computing (PC) era - *the Second Computing Paradigm* - in the 1980s. Since the 2000s, *the Third Computing Paradigm* enters the user's life.

to indirectly interact with the intended object. A mobile device can be used as such a medium. It can be used for explicit interactions with the pervasive computing environment and its objects as well as for presentations of objects' user interfaces, such as for the display of services of an intended object. Interactions which are executed by means of a mobile device are called mobile interactions [Rukzio et al., 2007].

Mobile phones cannot only be used to execute explicit but also implicit interactions [Schmidt, 2000]. For the *Implicit Human Computer Interaction* [Schmidt, 2000] provided the following definition.

"Implicit Human Computer Interaction is an action, performed by the user that is not primarily aimed to interact with a computerized system but which such a system understands as input."

Thus, in contrast to explicit interactions, implicit interactions typically base on recognised context of the user and his environment which is often caused by aware or unaware interactions of the user with the environment [Schmidt, 2000], such as the changed position or orientation of the user in the environment.

Based on the definition of [Dey, 2001], context is

"... any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

Contexts are typically built by means of data from sensors which help form an information which, in turns, can trigger events. The user's presence at a certain location, for instance, can be recognised by means of data of a GPS receiver and trigger an event, such as the display of information on a user's mobile phone.

In this dissertation, the user-centred prototyping process of applications is focused which make use of mobile phones to enable explicit and implicit interactions with primarily things: physical objects in the pervasive computing environment.

2.2.2 Mobile Phones as Interaction Devices

The use of mobile or smart phones has many benefits. Almost everybody owns a mobile device and takes it around constantly which makes them not only to a widespread interaction but also to a presentation device of information. Also, recent phones support novel hardware and network facilities which enable different mobile interaction styles that can be used for mobile interactions with physical objects and the pervasive computing environment. Before these different mobile interaction styles are covered more detailed, a typical mobile interaction scenario with a pervasive computing environment is described. This scenario bases on [Kindberg et al., 2002].

Pervasive museum scenario: The user John Smith visits a museum that presents different objects of arts. He wants to interact with these objects of arts in order to receive detailed information, such as information about their original context. With regards to *the Third Computing Paradigm*, the museum is a pervasive computing environment that includes different physical objects - objects of arts - which provide services, such as a service that offers detailed information. John Smith needs to explicitly select a physical object in order to receive and interact with its services and service entries (e.g. multimedia content). Since the objects of arts are not directly addressable, John Smith applies his mobile phone as indirect interaction and presentation device to the physical objects. For the selection of the intended physical object and the interaction with its services and service entries, John Smith can make use of different mobile interaction styles, such as he might apply the mobile phone's keyboard to first navigate through a list of displayed

objects of arts - the physical objects - and later on to select the intended one. This keyboard-based mobile interaction style, however, does not seem to be comfortable in each situation, such as if the list of objects is extremely long and the navigation is time-consuming and annoying. In this situation a more direct mobile interaction with the physical object would be preferable, such as a camera-based interaction style where the user simply has to capture a picture of the intended physical object for its identification and selection. Thus, John Smith uses his phone in order to take a picture of *Mona Lisa*. Now, the system detects the selection of the correct physical object and loads and displays a list of the available services for *Mona Lisa* on John Smith's phone. He navigates through the list of services and selects the service called *Original Context* to load and display detailed information about the object of art. All in all, the scenario illustrated that a mobile phone can be used to perform different explicit mobile interactions: selections and navigations. Also, a phone can be used to present information on the phone's display, such as services and service entries.

Mobile interaction styles: [Ballagas et al., 2006] give a comprehensive overview about the mobile phone's hardware facilities (e.g. the mobile phone's camera) which enable different mobile interaction styles. In summary, the mobile phone's keyboard or touch-sensitive display can be used for mobile interactions as well as a phone's camera, microphone, GPS receiver, accelerometer or NFC reader. Some phones even support proximity, light, orientation or position sensors. In the context of *the Third Computing Paradigm*, most of the mentioned mobile interaction styles are primarily used for interactions with physical objects - their selection - while other mobile interaction styles are mainly used for navigations, such as navigations through displayed menus. Besides this category, mobile interaction styles also can be categorised by whether they primarily support implicit or explicit interactions. In the following some examples of mobile interaction styles are described and categorised.

Keyboard-based interaction style: Users can apply the mobile phone's keyboard to interact with the graphical user interface that is presented on the mobile phone's display. In particular, the so-called softkeys of the mobile phone can be used for the keyboard-based interaction style. Softkeys are directly located below the mobile phone's display. They provide the opportunity to select a displayed option (e.g. the virtual representation of a physical object) by means of the fire-key or to navigate through a menu by means of the four navigation keys. Mobile interactions which are executed by the keyboard are usually explicit interactions since users actively interact with the graphical user interface of the phone in order to indirectly interact with the physical object.

Touch-display-based interaction style: The touch-sensitive display of a mobile phone supports similar mobile interactions compared to the keyboard. They are also explicit and primarily can be used for selections of objects as well as for navigations through menus. These navigations and object selections, however, are more directly compared to keyboard-based interactions. For the selection of a physical object, for instance, a user can directly touch the touch-sensitive display and thus a displayed virtual object that represents a physical object whereas when using the keyboard a cursor first needs to be navigated towards the intended virtual object in order to highlight and then select it.

Microphone-based interaction style: Mobile interactions can also be performed based on the mobile phone's microphone (e.g. [Lim et al., 2009]). Grounded on speech input, the user can explicitly select a physical object. Therefore, the user simply has to input a corresponding keyword, such as the name of the intended physical object. This word is then recognised by a speech recogniser. Besides interactions with physical objects, the microphone-based interaction style also can be used in order to either manipulate the mobile phone's graphical user interface, such as to play, pause and stop a video player or to navigate through a list by speaking commands (e.g. up, down, right or left). The microphone-based interaction style cannot only be used for explicit but also for implicit interactions. Based on the microphone, context can be detected (e.g. a high level of noise) and thus events can be triggered, such as the display of a notification that speech input is currently impossible because of strong background noise.

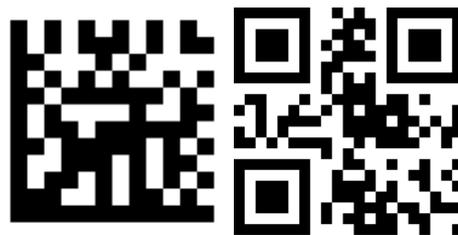


Figure 2.5: An example of the Data Matrix (left) and the QR Code (right). Both visual markers represent the text: *Karin*.

Camera-based interaction style: Similar to the microphone-based interaction style, the camera-based interaction style can typically be used for the explicit selection of a physical object. To do that, one option is to capture an image of the intended physical object and use an image recogniser to identify it (e.g. [Föckler et al., 2005]). Another option is to use visual markers which represent the corresponding identifier (e.g. an URL) of the physical object. These markers also can be interpreted by means of an image recogniser. *The Data Matrix* and *the QR Code* are two prominent examples of 2D visual markers (see Figure 2.5). Visual

markers cannot only be used for an object selection but also as position sensors (see Position-based interaction style) to recognise the tilting position of the phone (e.g. [Rohs and Zweifel, 2005]). Apart from these rather explicit interactions, the camera also can be applied for implicit interactions, such to detect a user's current lighting condition in the environment by automatically capturing and interpreting images.

NFC-based interaction style: The built-in NFC reader of a mobile phone can also be applied to enable explicit interactions with physical objects. For the NFC-based interaction style, the physical object needs to be augmented with an RFID tag that provides information about an object's identifier. Now, once the phone has reached a short distance to an RFID tag, an NFC connection is built and the identifier is transmitted to the mobile phone. Several user studies have been conducted where the NFC-based interaction style was used in the context of *the Third Computing Paradigm*. Results show many benefits for the system's usability (e.g. [Rukzio et al., 2007] or [Broll and Hausen, 2010]). The NFC-based interaction style, for example, is meant to be more robust, quicker and easier to use compared to the camera-based interaction style [Mäkelä et al., 2007].

GPS-based interaction style: The GPS receiver can primarily be used to perform implicit interactions since it provides data to detect the user's location context. This location context can trigger the execution of specific services - location-based services -, such as the loading and display of location-dependent content. There are several applications which make use of location-based services, such as games (e.g. Savannah [Benford et al., 2004]) or outdoor guides (e.g. [Kaasinen, 2003]). Geocaching¹ is a further well-known application domain where GPS-based mobile interaction are used. The idea is that some users hide geo-tagged treasures and other users try to find this treasures based on their GPS devices.

Orientation-based interaction style: The orientation sensor bases on the idea of a digital compass since it provides information about changes of the phone's and the user's orientation. The orientation-based interactions can be both implicit and explicit. If the user is aware that an orientation sensor exists as well as how orientation gestures (e.g. a 180 degree rotation) influence the system, the orientation-based interaction is rather explicit whereas if the user is either unaware of the sensor itself and its influence towards the system or how the orientation sensor influences the system, the interactions are rather implicit. Explicit orientation-based interactions are typically used for navigation tasks. In combination with GPS, the orientation sensor, for instance, can help explicitly navigate

¹<http://www.geocaching.de/>

a user to a certain location. With respect to *the Third Computing Paradigm*, the orientation sensor can be used to explicitly navigate through displayed lists or to implicitly personalise and optimise the application based on the user's orientation.

Position-based interaction style: A position sensor provides information about the phone's tilt in 2D-axes. By making use of the sensor, a user can explicitly navigate up, down, right or left through different displayed elements (e.g. [Mäntyjärvi et al., 2006]). By navigating through letters, even text can be entered [Wigdor and Balakrishnan, 2003]. Mobile games sometimes also apply the position sensor to control game elements, such as a virtual character in the game world. Similar as for the orientation sensor, the position sensor is primarily used for navigation tasks that are executed explicitly. Therefore, the user must be aware of the sensor as well as the 2D position gestures and their influences on the system. Implicit interactions can also emerge if the application makes use of the sensor data in order to adapt the system which sometimes even remains hidden for the user.

Accelerometer-based interaction style: Data of the mobile phone's accelerometer and its 3D-axes can be used for accelerometer-based interactions, such as implicitly for the recognition of a user state (e.g. [Iso and Yamazaki, 2006]), such as whether the user is sitting, standing, lying, walking or running or explicitly by the recognition of a particular 3D gesture (e.g. [Mäntyjärvi et al., 2004], [Rehm et al., 2010] or [Kela et al., 2006]). Recognised explicitly performed gestures, for instance, can be mapped to the identification of a physical object or to a navigation command, such as up, down, right or left.

Proximity-based interaction style: Proximity sensors typically enable implicit interactions. By means of the sensor, the mobile phone can detect whether users hold their phones in their hands or not. This information can be used as contextual information for an application. For example, the phone can automatically switch between a visual notification mode to an audio and vibration notification mode once the user state has changed. By this means, users can perceive important notification even if they do not have their phones in their visual attention.

Light-based interaction style: A light sensor also primarily generates implicit interactions since it provides contextual information (e.g. [Gellersen et al., 2002]), such as whether the user is in a bright or dark environment. As result of a changed situations, for instance, the display light can be adapted.

2.3 Software Support

This dissertation mainly aims on the question how the user-centred prototyping process of mobile applications in the context of *the Third Computing Paradigm* can be improved by making use of a software assistance. This section provides details about different categories of software assistance. Based on [Hull et al., 2004] developers can employ different types of software support when developing mobile applications: middleware or software modules in combinations with IDEs as well as classical prototyping tools or content and behaviour tools. These categories, however, primarily support in the design and implementation of application but do not additionally focus on a support for the conduction and analysis of user evaluations. To support assistance for the user-centred prototyping process in all stages, approaches are required which also assist the conduction and analysis of user evaluations. Thus, two further types of tools are added to Hull's categories: evaluation and analysis tools as well as user-centred prototyping tools.

2.3.1 IDEs with Software Modules and Middleware

Eclipse and Netbeans are two well-known *Integrated Development Environments* (IDE) that can be used for the software implementation of mobile applications (e.g. for the platforms J2ME and Android). These IDEs make use of so-called emulators of mobile phones (e.g. from Nokia or Sony Ericsson) in order to enable emulations and tests of mobile applications on a desktop computer during the development process. Typically, these emulators also support a set of different so-called *Application Programming Interfaces* (APIs). APIs typically provide software access to different functionalities of a mobile phone. For instance, most modern emulators of J2ME (Java Microedition) support the *JSR 135 Mobile Media API* that provides interfaces and classes to address the mobile phones built-in camera and microphone as well as to run audio and video files. IDEs also often provide a so-called graphical user interface (GUI) builder that supports the specification of a prototypes's appearance (see Figure 2.6) and behaviour (see Figure 2.7). The result of specifications with the GUI builder is automatically generated source code that can be modified by the interface developer.

Middleware and software modules are also software components that typically provide an API to enable access to different functionalities, such as access to different hardware capabilities of a device or a network layer. Usually, they also can be added to a project of an IDE. Equip [Greenhalgh, 2002] is an example of a middleware that supports developers of mobile applications. It offers a software platform for mixed reality user interfaces. In contrast to middleware, modules typi-

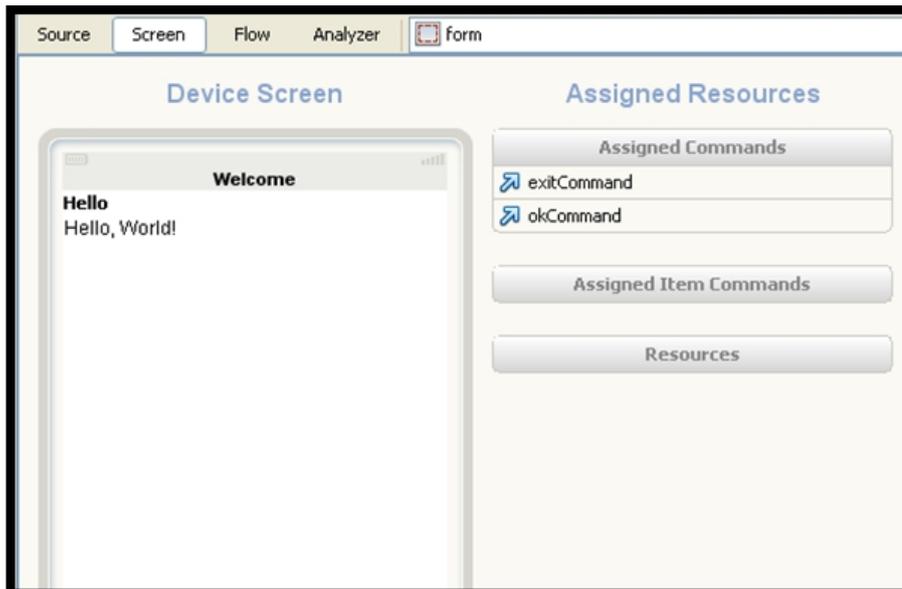


Figure 2.6: Screen view of the GUI builder from Netbeans for the development of the mobile application's appearance.

cally provide software for different components of software architectures, such as a server component of a client-server architecture. Using software modules, developers do not need to re-implement the corresponding component anymore but instead can re-use it in different contexts. For instance, the Context Toolkit [Salber et al., 1999] provides software modules to address different sensors.

Middleware and software modules provide developers with valuable support when developing applications which can save time and money as well as prevent errors but even if they are used in combination with an IDE that provides a GUI builder, the interface developers still require comprehensive programming skills. Additionally, middleware and software modules typically just support in the design and implementation and not in the evaluation and analysis of applications. Thus, this type of software support does not seem to be an appropriate assistance for interface developers which conduct the user-centred prototyping process.

2.3.2 Classical Prototyping Tools

In contrast to middleware and software modules, classical prototyping tools require less programming skills but fail in other requirements. They typically provide a graphical user interface to build up a sketch of an application based on its early concept but often do not assist in the generation of functional high-fidelity prototypes that directly run on the intended devices, such as mobile phones. Visio, Adobe Fireworks and PowerPoint are examples of well-known commercial

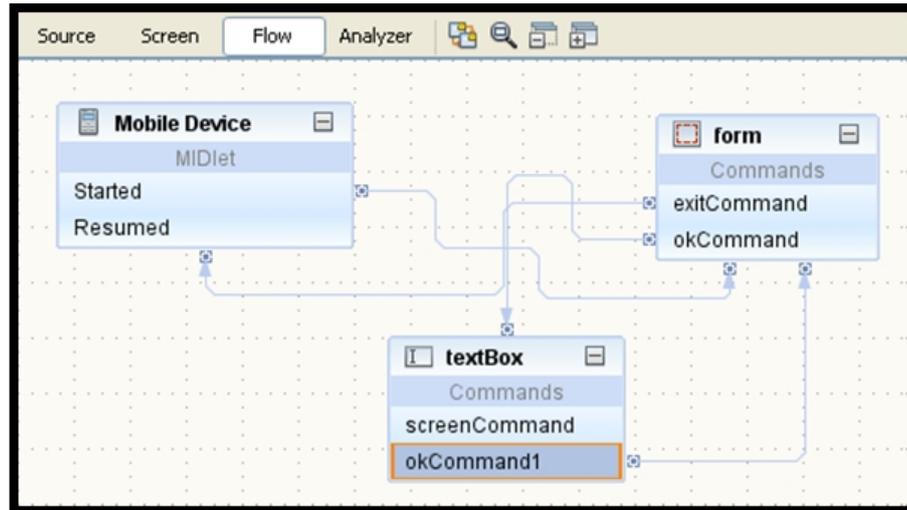


Figure 2.7: Flow view of the GUI builder from Netbeans for the development of the mobile application's behavior.

tools that can be used to illustrate a sketch of an application. MockUp² is a further example that can be used to sketch an iPhone application whereas Mockflow³ can be used for sketching a webpage.

All in all, classical prototyping tools are very helpful to communicate ideas for discussions in an interdisciplinary project team. They also can be used to generate first versions of an application in order to conduct user evaluations in early stages of the user-centred prototyping process. Despite their benefits, classical prototyping tools also do not seem to be an appropriate software support for the entire user-centred prototyping process (see Section 2.1) since, later on in the process, functional high-fidelity prototypes often cannot be generated that directly run on the intended device.

2.3.3 Content and Behaviour Tools

Content and behaviour tools provide a graphical user interface (GUI) for the design specification of a prototype that is, later on, automatically generated as a result of the specification task. Usually, the developers just need to specify the content and the behaviour as well as the appearance of the intended application by means of the tool's GUI. The content specification addresses multimedia content (e.g. text and images) that should be presented at the runtime of the application whereas the appearance specification covers the specification of all screens

²<http://mockapp.com/>

³<http://www.mockflow.com/>

including their content and control elements. Finally, the behaviour specification considers the specification of the application flow and logic.

Compared to middleware and software modules, content and behaviour tools require no or less implementation skills since prototypes are automatically generated as a result from the GUI-based design specification task. The generated prototypes directly run on the intended interaction devices. Thus, in contrast to classical prototyping tools, content and behaviour tools can generate functional high-fidelity prototypes and thereby they also can support during later phases of the user-centred prototyping process. The App Inventor from Google⁴ is a commercial example of a content and behaviour tool that helps generate prototypes for the Android platform by means of a GUI-based design specification whereas MScape [Hull et al., 2004] is a research example that supports developers with a tool in order to generate location-based applications, such as Savannah [Benford et al., 2004]. The result of the design specification with MScape are XML files. These XML files are interpreted by a software module in order to run the prototype on a PDA.

Content and behaviour tools seem to appropriately support the design specification and implementation phase of the user-centred prototyping process. A GUI is provided to specify the prototype while the implementation phase is not required anymore since the prototype is automatically generated. This aspect potentially reduces the development time as well as errors caused by a lack of an interface developer's proper programming skills. Content and behaviour tools, however, do not support the conduction and analysis of evaluations.

2.3.4 Evaluations and Analysis Tools

Evaluation and analysis tools assist in the conduction and analysis of user evaluations. They typically provide a graphical user interface to record test users once they are interacting with an application. Later on, the tools also assist in the analysis of the captured data. Userfly⁵ is a commercial example that can be used to log all interactions while users are surfing on a website. Afterwards, the system can be used to playback the logged sessions as a movie that displays the corresponding website content and the logged keyboard and mouse interactions. Thus, the evaluator can exactly replicate a session in order to find user problems. A similar system is M-Pathy⁶ that also can be used to log mouse-based user interactions and

⁴<http://appinventor.googlelabs.com>

⁵<http://userfly.com/>

⁶<http://www.m-pathy.com>

later on analyse the logged data by providing the website content and the logged mouse events. A well-known further example is Google Analytics⁷ that also can be used to analyse the web traffic on a website. There are several further tools available that support the evaluation and analysis of websites but there are less tools for the mobile context.

When conducting evaluations in this context, challenges occur due to the fact that mobile users are often in the wild and on the go. Mobile users usually do not only use their devices alone at home which would be similar to the use of a desktop PC but instead mobile devices are used in different indoor and outdoor environments with different social context. Tools have to support the conduction of evaluations in these different situations in order to get a clear picture about the users. However, tool-support in the wild and on the go is not always easy to provide. An audio-visual capturing of the user, for instance, is difficult to perform in an environment that changes constantly. MyExperience [Froehlich et al., 2007], for instance, supports developers in the mobile context by automatically logging user interactions that are conducted with mobile phones. After the conduction of a user evaluation, MyExperience additionally assists in the analysis by providing a graphical user interface which displays all logged user data.

All in all, neither classical evaluation and analysis tools for the desktop setting nor tools for the mobile setting support all phases of the user-centred prototyping process because the design and implementation of prototypes are not supported. Nevertheless, these tools provide important insights about how a tool can support during the conduction and analysis of user evaluations.

2.3.5 User-Centred Prototyping Tools

As a conclusion, content and behaviour tools support during the design specification task and the automatic generation of prototypes while evaluation and analysis tools support during the task to conduct and analyse user evaluations. In a combination they can cover all stages of the user-centred prototyping process. Tools of these two categories, however, are often not compatible and therefore a fluent process of the different phases can be interrupted. Moreover, developers need to learn different tools with different handling which can needlessly require a period of time.

User-centred prototyping (UCP) tools support developers with an all-in-one-software assistance in all stages of the user-centred prototyping process [Hartmann et al., 2006, Klemmer et al., 2000]. They assist developers in the tool-based

⁷<http://www.google.com/analytics/>

design specification, evaluation and analysis of user interface prototypes. Consequently, UCP tools are an all-in-one tool combination of a content and behaviour tool as well as a evaluation and analysis tool. Usually, UCP tools have a strong link between the design, evaluation and analysis component which can prevent interruptions of the process, such as the evaluation component supports the conduction of user evaluations with prototypes that were generated during the tool-based design specification. The analysis component assists interface designers with the interpretation of synchronously captured data of the evaluation component.

One question, however, is which further benefits and problems emerge when using such an all-in-one tool for the user-centred prototyping process. This and other questions (see Chapter 1) will be addressed in the next chapters of this dissertation.

2.4 Summary

This chapter introduced the user-centred development of interactive computing systems, *the Third Computing Paradigm* and different categories of software support that might assist developers during the user-centred prototyping process.

First, aspects of usability and its benefits were presented. Then, the Human-Centred Design (HCD) process and its different phases were introduced. While the first three phases of the HCD process primarily address aspects of a requirement analysis and specification, the last two phases of the process focus on the production of prototypical solutions and their evaluation and analysis against the specified requirements. These last two phases can be split in four subtasks which together result in the user-centred prototyping process: (1) design specification and (2) implementation of prototypes as well as (3) the evaluation of the generated prototypes and (4) the analysis of the evaluation results. This user-centred prototyping process is objective of the dissertation.

Besides the user-centred development, the second part of the chapter introduced the concept of *the Third Computing Paradigm*. The basic idea of *Ubiquitous Computing* and similar concepts were presented. A focus of the dissertation is the development of applications which make use of mobile phones as interaction and presentation device to a pervasive computing environment. Therefore, different aspects of such mobile applications were introduced with a focus on the applied mobile interaction styles.

The last aspects of the chapter was the introduction of different categories of software support for interface developers. The categories middleware and module

were introduced as well as the categories classical prototyping tools, content and behaviour tools and evaluation and analysis tools. Finally, the category user-centred prototyping (UCP) tools was presented which potentially supports all phases of the user-centred prototyping (UCP) process. This dissertation investigates the use of all-in-one UCP tools for the user-centred prototyping process of mobile applications in the context of *the Third Computing Paradigm*. The first research question is to reveal typical requirements towards a UCP tool and the resulting prototypes of the tool.

To find these requirements, the next chapter describes and discusses different mobile applications that were developed by means of user-centred prototyping process. Also, related content and behaviour tools, evaluation and analysis tools as well as UCP tools are reflected on their tool features.

Chapter 3

Requirement Engineering

The objective of this chapter is to define a list of typical functional and non-functional requirements regarding software assistance for the user-centred prototyping process as well as functional and non-functional requirements for the resulting prototypes. To find characteristic requirements, a bottom-up approach was undertaken by analysing the user-centred development of mobile applications in three different application domains on representative challenges for interface developers.

Besides the requirement analysis, the chapter also presents a literature overview about related software tools that provide assistance for the different phases of the user-centred prototyping process. Based on the previous defined functional requirements, a tool comparison is also described in order to receive insights to strengths and weaknesses of the existing software support.

3.1 The Mobile Application Development

One goal of the chapter is to analyse the user-centred prototyping process in context of *the Third Computing Paradigm* in order to find typical challenges for interface designers when conducting the user-centred prototyping process without software assistance. By this means, functional and non-functional requirements should be revealed for a software support in the different process phases: the design specification and automatic generation of prototypes as well as the conduction of evaluations and their analyses.

Characteristically for mobile applications in the context of *the Third Computing Paradigm*, the applications make use of mobile phones to enable interactions with objects of a pervasive computing environment and their services (see Section 2.2).

These services usually provide access to read and/or write object-related data (e.g. multimedia content or system states). Thus, as a first requirement, the software assistance for the user-centred prototyping process should support the generation of mobile applications which enable the following functionalities: the remote access and the dynamic loading and display of an environment's objects, their services and content of the services. Typical services in the context of *the Third Computing Paradigm* are either information requests of multimedia content or services to change the multimedia content (e.g. images or videos). The specification of the remote content (e.g. database content) should also be enabled.

A further characteristic of mobile applications in the context of *the Third Computing Paradigm* is the use of different mobile interaction styles as also described in Section 2.2. Consequently, resulting prototypes of the software-assisted user-centred prototyping process are additionally expected to support different mobile interaction styles as a second functional requirement. The required mobile interaction styles (e.g. keyboard-based or accelerometer-based) often strongly depend on the application and the context of use.

To get further insights in typical requirements for the software assistance of the user-centred prototyping process and the resulting prototypes, case studies of three different application domains are discussed together with their user-centred prototyping iterations. These three domains can be distinguished as followed: ambient-assisted living, mobile outdoor and indoor aides and mobile learning and entertainment.

3.1.1 Mobile Interaction for Ambient-Assisted Living



Figure 3.1: A user who performs *touching* [Rukzio et al., 2006].

The user's home contains several objects, such as different home appliances (e.g. the cooker or refrigerator) or home entertainment devices (e.g. the TV set or DVD player). By using a mobile phone, these different physical objects can become addressable (e.g. by means of a UPnP network) to load information about the objects

and their system state as well as to change the state, for instance, by switching the objects on or off. Such an application - *MobiMote* - was user-centred developed in former work [Rukzio et al., 2006]. The main objective of its user-centred development was to get knowledge about user preferences for a mobile interaction style in different situations in terms of context. Three different kinds of mobile interaction styles were supported by *MobiMote*: a NFC-based interaction style called *touching*, a Laser-beam based interaction style called *pointing* and a keyboard-based interaction style called *scanning*. Figure 3.1 shows how the user performs the NFC-based interaction whereas Figure 3.2 shows the laser-beam based interaction. This application also highlights the need and requirement to support several mobile interaction styles for mobile applications.

Before the user evaluation and later on the analysis of the user evaluation were conducted, the mobile application first was designed and implemented. These different steps of the user-centred prototyping process show further requirements towards a tool-support and the resulting prototypes.

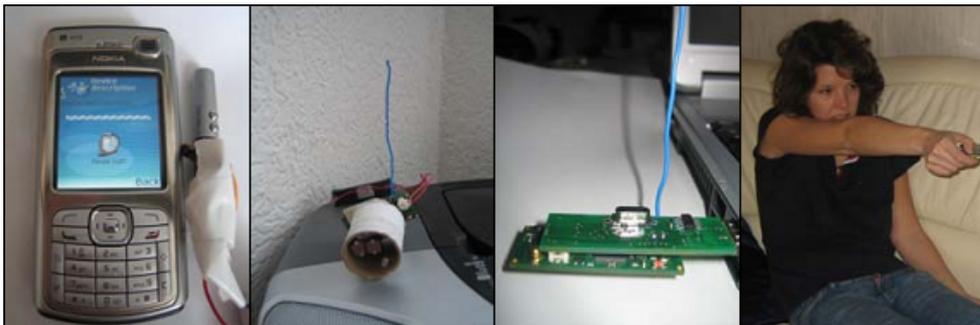


Figure 3.2: A user who performs *pointing* [Rukzio et al., 2006].

The design and implementation of the application: To get a first impression about *MobiMote*, its screen contents and control elements, a state-chart (see Section 2.1) with the relevant screen states and the transitions - the application flow - were specified to finally generate a low-fidelity prototype of *MobiMote* - a paper prototype. Figure 3.3 shows some of the paper prototype's screens which were implemented in the first iteration of the user-centred prototyping process [Leichtenstein et al., 2006].

At first view, the specification of the prototype's state-chart and the generation of a paper prototype are quickly and easily to conduct due to the fact that these steps do not require any software skills, such as the software implementation of an underlying architecture. These steps, however, are a challenge with regards to user interface design since the resulting prototype is expected to meet a high user-friendliness (e.g. see [Myers, 1993]). In order to do so, interface guidelines for

mobile phones (e.g. see Guidelines of *Nokia's Design and User Experience Library*¹) can be applied since guidelines provide comprehensive knowledge about good practice. Interface designers, however, need to know how to apply the guidelines in an efficient and comprehensive way, such as how to easily ensure a consistent layout. Consequently, the correct application of interface guidelines is a challenge for inexperienced interface developers. [Myers, 1993] also highlights the problem to correctly apply interface guidelines due to the need to first correctly interpret them. This problem is even increased because interface designer are usually under time pressure to quickly generate the product.

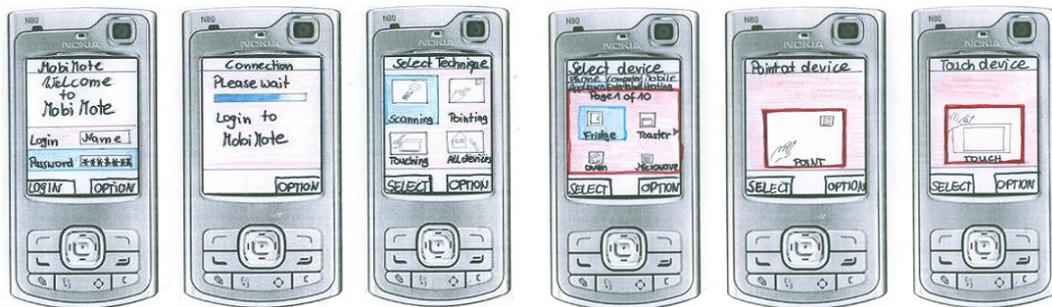


Figure 3.3: Different screens of the low-fidelity prototype [Leichtenstern et al., 2006].

In order to tackle the challenge to correctly apply interface guidelines, a software assistance of the user-centred prototyping process should, if possible, automatically support the correct application of guidelines, such as by automatically checking their compliance. Since the automatic verification of guidelines is not always possible, the analytical verification of the prototypes by usability and/or domain experts should also be assisted by the tool, such as by supporting a guideline-based and a walkthrough-based analytical evaluation (see Section 2.1).

Further, the software assistance should not only support the generation of high-fidelity software prototypes but also the generation of low-fidelity prototypes with sketched screens both by means of a state-chart view due to the fact that low-fidelity prototypes rather increase a user's willingness to provide critical feedback than high-fidelity prototypes with a meaningful look-and-feel. This detailed feedback is very helpful in particular in early iterations of the user-centred prototyping process when interface developers have less knowledge about the end-users of a mobile application (see Section 2.1).

The design and implementation of the *MobiMote's* high-fidelity prototype in JAVA required a client-server architecture [Rukzio et al., 2006] and the implementation of the architecture's components. The components were implemented as fol-

¹<http://library.forum.nokia.com>

lowed. The server component either communicated with the physical objects by means of the UPnP server (e.g. to forward status requests) or replied information about the objects (e.g. their user manual) which were persistently stored in a database. The service requests were initiated by the application that ran on the mobile phone. This mobile client could connect to the server component in order to load and display data about the objects (e.g. status information).

The implementation of *MobiMote's* high fidelity prototype required a lot of programming and Software Engineering skills, such as knowledge about server-client programming and knowledge about how to implement the different mobile interaction styles. The implementation of such an application can be very time-consuming and error-prone for inexperienced interface designers. The development of *MobiMote*, for instance, took almost two month as a one-man-project. Moreover, the complex implementation process distracts too much from the original objective: the evaluation of the prototypes and its analysis.

In summary, the design and implementation of low-fidelity and high-fidelity prototypes require both Software and Usability Engineering skills to prevent long development times and an error-prone product. A software assistance is strongly required for the design and the implementation of prototypes to keep the focus on the user and not on the prototype generation. Thus, as a further requirement the software assistance indispensably needs to decrease the required programming skills by providing an architecture and implemented software modules for at least the typical generic components: the mobile client, the server and the database.

The evaluation and analysis of the application: After the generation of the high-fidelity prototype, the user evaluation of the prototype was conducted by means of an inquiry method: a questionnaire [Rukzio et al., 2006]. Test users had to apply *MobiMote* in some contextual situations to complete pre-defined tasks. For instance, they had to select the CD player by one of three supported mobile interaction styles to load its current status and switch it on if necessary. After each situation they had to fill-in the questionnaire by documenting their selected interaction style. The different situations were specified by the user's location and activity. In some situations the users had line-of-sight to the intended object or even were in touch with it whereas in other situations they did not have line-of-sight, such as when the object was located in another room. For all these location contexts, it also was differed whether the user's activity was sitting, lying or standing. Followed by the user evaluation, the results of the questionnaire were analysed which revealed context-dependent user preferences.

As a further requirement towards a tool-support, the recording of subjective data seems to be important, such as by means of storing the quantitative and qualita-

tive answers of a questionnaire. For mobile applications, the recording of subjective data can be conducted directly with questionnaires that are displayed on the phone (e.g. see [Vääätäjä and Roto, 2010]).

In addition to an inquiry method, the user evaluation conducted for *MobiMote* also can be executed based on an observation method (see Section 2.1). The evaluation and analysis phase of observation methods are potentially more time-consuming than inquiry methods, such as they require the analysis of objective and typically qualitative data (e.g. captured videos). Despite these problems, observation methods often provide much more valid data since user behaviour is measured and not just the user's attitude (see Section 2.1). Challenges of executing observation methods and thus potential requirements for a software-assisted evaluation and analysis still need to be revealed by means of an application that was evaluated with an observation method.

3.1.2 Mobile Interaction for Mobile Outdoor and Indoor Aides

Not only applications for ambient-assisted living but also for mobile outdoor and indoor aides need to be addressed which typically provide pervasive services not primarily to a single user but instead often to several users. Museums or shopping stores, for instance, are mobile indoor aides where objects of arts or products are physical objects. In these environments, several users separately interact with the system. These users do not only request information but they also can generate new multimedia content (e.g. user feedback about physical objects) which afterwards can be requested by other users. Thus, the service to add new multimedia content or change existing content is even more relevant if such user communities exist.

One mobile outdoor aide that was developed was called *Gastronomy Guide* [Schmitt, 2008]. It was located in the environment of a city and the city's restaurants were the physical objects of the application. The idea of *Gastronomy Guide* was to provide information about restaurants including official critics and critics of the user community. Thus, the users could not only request information about physical objects - the restaurants - but also provide new multimedia data (e.g. video and textual information) about them. The NFC-based and keyboard-based mobile interaction styles were supported by *Gastronomy Guide* to either select a restaurant by touching an RFID-tagged city guide or by selecting the restaurant from a list of all restaurants which were displayed on the mobile phone's screen.

Since the former design and implementation of *MobiMote* revealed, in particular, challenges when designing and implementing a mobile application in the context

of the *Third Computing Paradigm*, *Gastronomy Guide* was developed with a first version of software assistance: an implemented architecture and its software modules (see Section 4.1.2). By means of template-based XML specification files, the mobile interaction styles could be specified as well as the appearance - the screens, their content and control elements - and the behaviour - the logic and flow of the application - of *Gastronomy Guide*. For *Gastronomy Guide*, the appearance and the behaviour could be specified statically or dynamically. Static specifications can be used if the appearance and behaviour are completely known at the specification time, such as the content of a help screen. If the appearance and the behaviour are just known at the runtime, they have to be specified dynamically with a scripting language and script prompts. The support of both, static and dynamic specifications of the application's appearance and behaviour, seems to be a further important requirement towards a software tool.

The available software module for the mobile phone should be able to interpret the different design specification files at runtime in order to automatically generate the functional high-fidelity prototype. The software modules provide this ability which were used for *Gastronomy Guide* (see Section 4.1.2). The required skills for the implementation phase seem to be reduced by the approach to enable the XML-based and script-based specifications of a software module. The interface designer only needs to design the application by making use of the corresponding XML files. A separate implementation phase with the use of an object-oriented programming language (e.g. JAVA) is not required anymore due to the fact that the prototype is automatically generated from the design specification files. The question is whether the mentioned challenges of *MobiMote* - primarily in terms of Software Engineering - can comprehensively be tackled for the design and implementation phase and whether new challenges emerge when making use of software modules as well as the XML-based and script-based design specification.

The design and implementation of the application: Figure 3.4 shows screens of *Gastronomy Guide* which was developed by means of the software modules described in Section 4.1.2. By using the software modules, challenges could be partly tackled regarding the design and implementation phase. The required interface design skills, for instance, were reduced due to the fact that the compliance of well-established guidelines from Nokia² were considered. The compliance of the guidelines could be ensured since corresponding rules are integrated in XML-based screen templates and in the implementation of the software module: MOBILE CLIENT (see Section 4.1.2). Thus, the developers do not need to spend time

²<http://library.forum.nokia.com>

anymore to think about how to correctly apply the mentioned guidelines. They simply have to use and edit the pre-defined screen templates which also reduce the required programming skills.

Network programming, for instance, remains hidden for them. The editing of the XML files, however, still requires knowledge, such as how to add items or multimedia content to a screen template. For the editing of the XML files, the use of a graphical user interface (GUI) seems to be an important new requirement to even more reduce the required skills which, in turn, also reduces the number of errors and the required development time to generate a prototype.

In conclusion, software assistance for the user-centred prototyping process seems to be an insufficient support if it just bases on software modules. Instead of software modules, either content and behaviour tools in combination with evaluation and analysis tools or all-in-one user-centred prototyping tools should be used for the user-centred prototyping process since these software assistances provide a graphical user interface and thereby even more prevent required Software and Usability Engineering skills (see Section 2.3).



Figure 3.4: Different screens of *the Restaurant Guide* [Schmitt, 2008].

The evaluation and analysis of the application: Similar as for *MobiMote*, the user evaluation of *Gastronomy Guide* was conducted by means of an inquiry method in a laboratory setting. First, the test users made use of the prototype to complete different tasks. After the use, they answered questions of a questionnaire mainly addressing usability in general and user satisfaction with *Gastronomy Guide*.

Gastronomy Guide also can be evaluated in a real world evaluation - a field study. During this kind of evaluation, the evaluator and the test users are typically separated spatially and sometimes also temporally. This separation simplifies the conduction of mid-term and long-term studies due to the fact that the local presence of the evaluator is not always necessary anymore (see Section 2.1). During these so-called remote evaluations, the users can usually freely use the application

while all their actions and contexts are automatically recorded. Also, the users can be asked to document their daily life based on the application, such as by means of *the Experience Sampling Method* (see Section 2.1). Thereby, the users can either provide objective data about their current situation (e.g. images or videos) or subjective data (e.g. answers about questions).

As a new requirement towards a tool-support, the software needs to assist not only evaluations in the laboratory setting - a real world simulation - but also in a field setting - a real world evaluation. To enable mid-term and long-term evaluations in these settings, the tool should also support both, local and remote evaluations, where the evaluator either directly or indirectly observes the test users. For remote evaluations, it is important to enable a remote communication between the evaluator and the test users, such as to provide the evaluator with just-in-time details about the test users and their current context. Further, it is important that the prototype is highly reliable and functional in order to comprehensively ensure the uncontrolled and unaided use of the application. Finally, it is important to enable the logging of subjective and objective data both, quantitatively and qualitatively.

3.1.3 Mobile Interaction for Mobile Learning and Entertainment

Apart from mobile applications for ambient-assisted living and mobile outdoor and indoor aides, mobile interaction can also be performed for mobile learning and entertainment, such as described by [Rehm et al., 2010, Rehm and Leichtenstern, 2012]. The concept of the application described by [Rehm et al., 2010, Rehm and Leichtenstern, 2012] was as followed. Two users could first separately watch videos with their mobile devices and learn how to perform different gestures of different countries, such as the greeting gesture in Japan. Later on, they could practice the learned gestures in a competitive two-user mode by first executing the gestures by means of their mobile phone and its accelerometer-based interaction style and then by comparing their performances.

A further mobile learning and entertainment application that supports several users is ORIENT [Lim et al., 2009, Lim et al., 2011]. The application was a computerised educational role-play game. The main objective of ORIENT was intercultural learning that is to improve the user's acceptance and appreciation of foreign cultures. In ORIENT, three children could jointly interact with a virtual culture - the Sprytes characters - of a foreign planet called ORIENT. This planet and the different characters were presented with a projection of 3D content. For the communication with the virtual characters, three interaction devices were available: a mobile phone, a WiiMote and a dance mat. The mobile phone was used for

speech input - microphone-based interaction style - and object input - NFC-based interaction style. The children could use the phone as a kind of *tricorder device* to either select a character of ORIENT - speech input - by speaking his name or an intended object by scanning its real world representation - object input - and making it available in the virtual 3D world of the Sprytes. For instance, the NFC reader of the mobile phone was used to transmit objects such as soil, seed and green drink which were required at certain points of the ORIENT story.

The WiiMote was used as a kind of *magic stick* to perform gestures which were mapped to actions, such as greeting, eating or apologising. Thus, the speech, object and action input together could be used to build sentences, such as to ask a Spryte character questions about the meaning of different objects in the ORIENT culture. For navigation within the 3D world of ORIENT, the dance mat was applied. A challenge of ORIENT was to handle the synchronous interactions of several users who had a direct relation, such as it was important to correlate speech input - the name of the intended Spryte -, the object input - the intended object - and the action input - the intended verb - to communicate with the virtual world and the Sprytes. Thus, in contrast to many classical applications in the context of ambient-assisted living or mobile outdoor and indoor aides, applications for mobile learning and entertainment are very often multi-user settings with collaborative or competitive modes. Also, they often need to support asynchronous or synchronous user interactions. All in all, the application domain seems to arise new challenges for interface developers compared to the other two application domains.

The World Explorer [Leichtenstern and André, 2009a] is a perfect example to illustrate further challenges for interface developers of applications in the context of mobile learning and entertainment. Its user-centred development does not only highlight challenges for the prototype generation as ORIENT, but also for the conduction and analysis of an empirical evaluation due to the fact that *the World Explorer's* evaluation was executed by means of an observation method. ORIENT, in contrast to *the World Explorer*, was evaluated similar to the other introduced applications with inquiry methods.

Similar to ORIENT and in contrast to *MobiMote* and *Gastronomy Guide*, *the World Explorer* is a multi-user application with synchronous user interactions due to the fact that children always could play synchronously in groups of three and their interactions had a direct influence on each other. The main task for users of *the World Explorer* was to learn knowledge about different countries together in the group and correctly answer as many questions about these countries as possible. In contrast to ORIENT, *the World Explorer* only made use of mobile phones as interaction devices to complete the main task.

These mobile phones could differently be assigned to the three children. Applying the role assignment approach [Leichtenstern et al., 2007] every child had its own mobile phone which represented a role in the game and was based on a corresponding functionality. During the game of *the World Explorer* each child needed to accomplish a subtask to successfully complete the main task as group. Subtasks in *the World Explorer* included *Initiation* - the selection of a country and a corresponding topic -, *Evaluation* - the evaluation of the received information - and finally *Execution* - the answering of a question. Thus, based on the main task there were three different actions which could be assigned to three different children via their mobile phones and enabled the following scenario.

The first child could use a mobile phone with a built-in NFC reader to first select a country (Initiation). She simply needed to touch the RFID tag of the board that represented the country. Afterwards, the corresponding flag had to be found and picked as well. The children could then choose one out of four different topics corresponding to the selected country (e.g. about geography or music). After that point the second child and her mobile phone came into the play. Via her mobile phone's graphical user interface the child had to indicate whether information should be requested or whether the question should be loaded about the topic. When the child had selected the option *information*, the third child automatically got information in form of a multimedia presentation (video, audio, text, image) (Evaluation). The third child could replay the received information several times to evaluate its content. This information gave useful input to correctly answer the questions. Once the second child had selected the option *question*, a question and four answers were loaded on the phone of the second child. Now, this child could scroll through the four options to select an answer (Execution).

In addition to the just described *Role Assignment Condition* (Setting 1), children could also play *the World Explorer* in two further multi-user settings. In one setting, the user-group was equipped with a single phone (Setting 2). As a consequence of this setting, the children had to organise the completion of the three subtasks by themselves which could lead to uncontrolled situations, such as a dominant user who performed all of the three subtasks alone. In a further setting, all three users were equipped with a separate phone which supported the entire set of functionalities (Setting 3). In this setting each child was able to perform all three subtasks alone which also required a group organisation to prevent competitions among the users. Figure 3.5 shows the three different group settings.

The World Explorer did not only have mobile phones as presentation device but also supported another audio-visual output channel because results of former studies [Leichtenstern et al., 2007] showed that children found the game more realistic when they received direct feedback from a projected presentation of a public

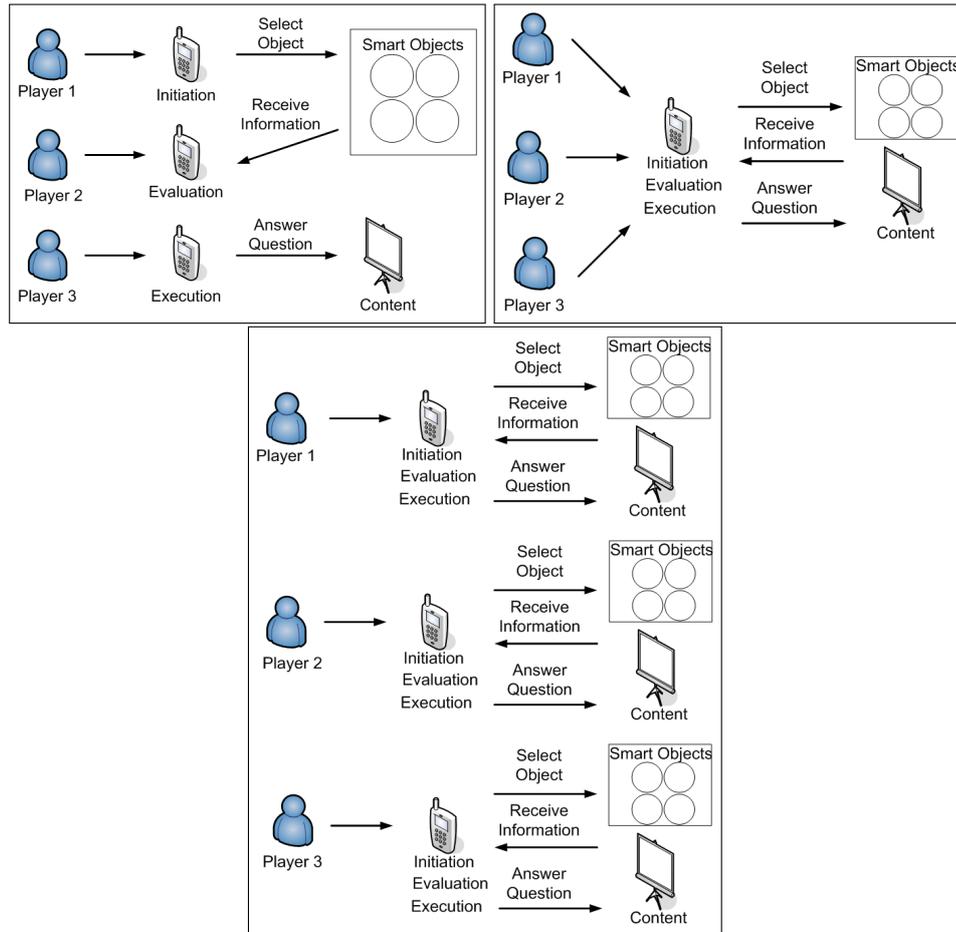


Figure 3.5: The first row on the left part shows Setting 1 and the *The Role Assignment Condition* while the right part shows Setting 2 where a single phone is available for the group. The second row shows Setting 3 with a phone per group member and all functionalities on the phones.

display. Therefore, *the World Explorer* projected videos which gave hints during the game, such as informing the children as to whether they had successfully performed an action or not. Additionally, these videos introduced the different countries and their topics as well as the questions and answers. Figure 3.6 shows a screen of the mobile application and the content of the public display.

All in all, the applications *the World Explorer* and *ORIENT* potentially provide knowledge about further requirements for the tool-supported user-centred prototyping. Despite the main focus on mobile phones as interaction and presentation devices, the tool should additionally enable the generation of prototypes with further presentation (e.g. a public display) and interaction devices (e.g. a WiiMote). In some cases, these additional devices might increase the user experience.

Further requirements are discussed by means of the design and implementation of *the World Explorer* and in particular by means of *the World Explorer's* conducted



Figure 3.6: A screen of the application which displays the question and the four different answers (left). The public display which provided additional information to the children (right).

evaluation and analysis of the results. The original main objective of the evaluation and analysis of *the World Explorer* was to answer the question which of the three mentioned multi-user settings would best fulfil the requirements to support social group behaviour. Therefore an observation method was used. Later on, the captured data were analysed on group activities. In the following the evaluation and analysis are presented after the description of the design an implementation of *the World Explorer*.

The design and implementation of the application: Similar to *Gastronomy Guide*, prototypes for *the World Explorer* were generated by means of the software module for mobile phones described in Section 4.1.2. The generation of prototypes for Setting 2 revealed no additional challenges since only one phone was available. Thus, the specification of the prototype was quite similar to the specification of *Gastronomy Guide's* prototype. In contrast to Setting 2, Setting 1 and Setting 3 consisted of three synchronised phones. In Setting 1, for instance, the second phone had to automatically load information once the first child had selected a country and topic. Consequently, in this setting, events triggered by mobile interactions had to be distributed among the phones in order to handle the complex interplay between the different roles. In Setting 3, synchronous mobile interactions could also happen, such as that all three children used their phone to select a country at the same time. In such situations, the system had to synchronously handle multiple mobile interactions to load and display the same content for all three users and their corresponding phones. Technically this was done by only accepting the first user request at the server and distributing the resulting content to all three phones. The interplay between the phones was handled by specifying context relations by

means of a further XML specification file of the software module: *MOBILE CLIENT*. Details are provided in Section 4.1.2. The non-interactive public display also was generated by means of an XML file and the software module called *ACTUATOR* (see Section 4.1).

All in all, *the World Explorer* required much more elaborated programming skills compared to the former mentioned applications due to the complex interplay and the need for synchronised devices. As already mentioned, the used software assistance does not only need to support aspects of single-user settings but also aspects of multi-user settings and their requirements: asynchronous and synchronous user interactions. The use of XML files for the specification of the interplay enormously reduces the required skills, however, developers still need to know how to specify these files. The knowledge gathered by the design and implementation of *the World Explorer* confirms the need for a graphical user interface to tackle challenges in terms of XML specifications.

The evaluation and analysis of the application: After the generation of the three multi-user settings and the corresponding prototypes, a user evaluation was conducted and later on analysed on aspects of *the Level of Activity*, *Off-task Behaviour* and *Social Interaction* [Leichtenstern and André, 2009a]. During the user evaluation, all user groups with a total of 18 children played each setting. While they were interacting with the system they were audio-visually recorded. Figure 3.7 shows some impressions of the user evaluation.



Figure 3.7: Impressions of the user evaluation performed with *the World Explorer*.

After the evaluation, the analysis of the video followed. For analysing *the Level*

of Activity, the number of game-related activities was counted which were performed by means of a mobile phone. For *the Off-task Behaviour*, the user gaze was counted to reveal how often the user looked away from the game setting longer than two seconds. Additionally, off-task conversation and other activities were calculated. Finally, *Social Interaction* was analysed by identifying conversations about the task and mutually-supportive behaviour.

The investigation of the user's behaviour based on an objective observation method is probably more meaningful than investigating the user's attitude with a subjective inquiry method due to the fact that observation methods are usually more difficult to manipulate and thus more reliable compared to inquire methods. Observation methods, however, generate new challenges for the interface developer and thus requirements for the software assistance of the user-centred prototyping process. The basic requirement is the support to synchronously capture audio-visual content and user interactions performed with the mobile device or another interaction device.

Further, since *the World Explorer* is a multi-user application, it seems to be important to assist the conduction of evaluations of several users who interact synchronously. By this means, the complex interplay between various users can be analysed correctly. Also grounded on experience acquired during the development of *the World Explorer*, it would be desirable to enable evaluators to record comments just-in time during the evaluation, such as comments about interesting incidents or comments about the current task, such as their begin and end.

When analysing the captured data, the tool should also support the display of all recorded data in an appropriate way in order to easily find interesting sequences. For *the World Explorer*, the complex interplay of the three group members was of interest. Thus, the display of test sessions of several test users also seems to be useful in order to enable the comparison between the different test users. Additionally, the pre-annotation of the videos by means of the logged user data (e.g. user interactions) is expected to be a useful further feature that should be supported by a tool. Finally, since most analyses end with an statistical verification, the software assistance is also expected to support the statistical analysis. *The World Explorer*, for instance, required the statistical analysis about aspects of *the Level of Activity*, *the Off-task Behaviour* and *Social Interaction*.

3.2 Requirements

The following section sums up non-functional and functional requirements for user-centred prototyping tools and their resulting prototypes. Several research

papers address functional and non-functional requirement engineering (e.g. [Thayer et al., 1997, Chung et al., 1999, Nuseibeh and Easterbrook, 2000]). One common definition is that non-functional requirements typically address the needed quality attributes of the system in different domains (e.g. usability and reliability). Thus, non-functional requirements do not contain details about what the system will do but instead of how - good - the system will do it. Functional requirements, in contrast, provide insights to the needed task support of the system and therefore rather specify the behaviour of the system.

3.2.1 Non-Functional Requirements for Prototypes (NFR-P)

In the following, the non-functional requirements for the resulting prototypes of a tool-assisted generation are listed. The focus of these requirements are towards mobile Human-Computer-Interaction (HCI). Thus, they only base on requirements which are perceptible by the end-users of the intended application: usability, reliability and functionality.

[NFR-P01] Support a high level of usability: The prototypes should provide a high level of usability in order to increase the end-user's efficiency, effectiveness and satisfaction when using the system. One option to fulfil this requirement is to make sure that approved interface guidelines are met.

[NFR-P02] Support a high level of reliability: A further non-functional requirement is the support of a high level of reliability. The resulting prototypes are expected to directly run on the intended interaction device with a high level of robustness and a short response time on user requests.

[NFR-P03] Support a high level of functionality: A high level of functionality is a last important requirement towards the resulting prototype. This non-functional requirement can be met by supporting a large number of the following functional requirements.

3.2.2 Functional Requirements for Prototypes (FR-P)

The functional requirements towards the resulting prototype are typically for the application domain of *the Third Computing Paradigm* where mobile phone are used as interaction and presentation device.

[FR-P01] Support the use of mobile interaction styles: There is an indicator [Rukzio et al., 2006] that the users apply different mobile interaction styles in different contextual situations. Thus, the resulting prototype should support different explicit and implicit mobile interaction styles (see Section 2.2).

[FR-P02] Support the remote access to dynamically load and display remote content: A further functional requirement is the support to enable the remote access to a server and its database in order to load and display information about the pervasive computing environment, its objects and services as well as the multimedia content of the service entries (e.g. video files).

[FR-P03] Support the remote access to dynamically capture and modify remote content: The content of the service entries should not only be loadable and displayable but also modifiable. The user should be able to capture different multimedia content (e.g. audio and text files) in order to add a new service entry or to change an existing service entry.

[FR-P04] Support the use of mobile interactions for several users: Since some mobile applications in the context of *the Third Computing Paradigm* address multi-users, the prototype should be able to handle synchronous and asynchronous interactions of several users which also might have a direct relationship (see Section 3.1).

[FR-P05] Support the use of additional interaction and presentation devices: Some applications do not only require mobile phones as interaction and presentation devices but also require additional interaction (e.g. a WiiMote) or presentation devices (e.g. a public display), such as the mobile learning and entertainment applications described in Section 3.1. Thus, as a further functional requirement, the resulting prototypes should support the option to add additional devices.

[FR-P06] Support the storage of qualitative and quantitative data which are either objective or subjective during the evaluation phase: Another functional requirement arises when conducting user evaluations with the prototype. In this phase, the prototype needs to support an option to log qualitative and quantitative data which can be either objectively (e.g. images which provide knowledge about the user's current contextual situation) or subjectively (e.g. answers about questions of a questionnaire).

[FR-P07] Support the remote communication between the test user and evaluator during the evaluation phase: Finally, the resulting prototype is expected to support a remote communication between the test user and the evaluator during the evaluation phase, such as to enable the transfer of messages (e.g. for diary studies).

3.2.3 Non-Functional Requirements for the Tool (NFR-T)

Besides the resulting prototype, there are several non-functional requirements towards the tool-based software assistance of the user-centred prototyping. Besides knowledge from the user-centred development of mobile applications, the literature also provides some insights on non-functional requirements for software tools.

[Li et al., 2004], for instance, expect that software tools improve the easy access for even inexperienced non-technical users by reducing the required software skills. Also, the application of the tool should reduce the development time as well as provide early user feedback in the design process. [Myers, 1995] describes two further non-functional requirements for a software support that appropriately supports in a development process. Firstly, the tool needs to improve the result of the tool-supported process: the resulting prototype. Secondly, the tool should support the ease of use and efficiency to run through a development process. [Klemmer et al., 2000] also see a need to reduce the required programming skills. Additionally, they highlight two further requirements. The tool has to be easy to learn. Further, it needs to enable the efficient generation and evaluation of prototypes as well as their modification in a new iteration. [Hartmann et al., 2006] confirm the requirements of Klemmer and colleagues since they also highlight the need to provide tools that assist multiple user-centred prototyping iterations and the modifications of evolutionary prototypes.

Based on this knowledge from literature and experience collected during the development of the previous described mobile applications, the following non-functional requirements are summed up with regards to a tool's usability, reliability and functionality.

[NFR-T01] Support a high level of usability: The non-functional requirement about usability covers aspects of efficiency, effectiveness, learnability and satisfaction as well as the tool's user-friendliness.

[NFR-T01.1] Highly improve the efficiency of the tool user: One non-functional usability requirement towards a tool assistance is the need to improve the efficiency of interface developers to quickly run through an iteration of the user-centred prototyping process. One option to improve the interface developer's efficiency is to reduce the physical and mental effort of the interface developer, such as by reducing the barriers by means of less required skills [NFR-T01.4] or by providing a highly use-friendly tool [NFR-T01.5] and [NFR-T01.6]. The efficiency of an interface developer is potentially also improved by executing automatic processes by means of the system, such as the automatic generation of a

functional high-fidelity prototype based on a design specification or the automatic pre-annotation of the captured qualitative data based on the quantitative data in the analysis phase.

[NFR-T01.2] *Highly improve the effectiveness of the tool user:* The core task of the interface designer is to user-centred develop a prototype of the intended mobile application that highly meets its own functional (FR-P) and non-functional requirements (NFR-P). Consequently, the effectiveness of the interface developers can be improved if the tool supports the generation of prototypes with a high level of usability, functionality and reliability.

[NFR-T01.3] *Highly improve the satisfaction of the tool user:* Apart of aspects of efficiency and effectiveness, the interface developer additionally needs to be satisfied when using the tool. When having good attitudes towards the tool, the interface designer will probably continue using it. Satisfaction with the system is therefore an important quality criterion that needs to be met. Satisfaction can be influenced by all other aspects of non-functional requirements. For instance, if the system is not reliable and less functional the interface developer probably will not be satisfied.

[NFR-T01.4] *Highly improve the learnability and reduce the required skills:* Learnability is strongly linked to the required skills of an interface developer as well as the required training periods when using the system. In best case, an interface developer can immediately use the system without requiring comprehensive knowledge about Software Engineering and programming as well as Usability Engineering and interface design.

[NFR-T01.5] *Support a high level of intuitiveness and transparency:* Further, the tool needs to provide a high level of intuitiveness and transparency by providing a highly user-friendly graphical user interface (GUI) for all components: the design specification, the evaluation and analysis component. To provide a good user-friendliness, a very important aspect is to always provide details and easy perceivable information about the executable actions of the system as well as the system state after a performed action [Norman, 2002]. By this means, *the Gulf of Execution* and *the Gulf of Evaluation* can be reduced which are often reasons of usability problems, such as that the user cannot execute the intended action or the user cannot interpret the system change. For the design specification component, the graphical user interface should enable the specification of the prototype including its appearance and behaviour in different views. As one view, the state-chart view should be provided. For the evaluation component, the GUI should provide an appropriate way to conduct the evaluation

which includes a view that displays the cameras, the current appearance of the prototype and the logged user events. In the analysis phase, the graphical user interface should enable the quick and easy inspection of the captured data also in several views. One important view is the time-line based display in order to jump between different sequences of interest.

[NFR-T02] Support a high level of reliability: As a further non-functional requirement, the tool indispensably needs to be highly reliable when using it. The tool has to be robust. It should not crash or provide any system errors.

[NFR-T03] Support a high level of functionality: As last non-functional requirement, the system is expected to provide a large scope of functionalities for the interface developers. One aspect is to support all functional requirements for the three tool components: the design specification, the evaluation and analysis component.

3.2.4 Functional requirements for the tool (FR-T)

For the tool-supported user-centred prototyping process, the following functional requirements base on the experiences which were acquired by developing several mobile applications in the context of *the Third Computing Paradigm* (see Section 3.1).

[FR-T01] Support design specifications during the entire development process: The first functional requirement is to support the design specification of the intended prototype which mainly addresses the specification of the application's appearance, behaviour and mobile interaction styles.

[FR-T01.1] Support the static and dynamic specification of the prototype's appearance and behaviour: The design component of a user-centred prototyping tool is expected to support the static and dynamic specification of the prototype's appearance and behaviour. The specification of the prototype's appearance includes the specification of the application's different screens, their content and control elements whereas the specification of the prototype's behaviour addresses the specification of the application flow and logic. When specifying an application statically, the concrete appearance and behaviour are known at the design time. The interface designers, for instance, can directly assign content (e.g. images or text) to the screens and define the invariable application flow and logic. For many applications, however, the application flow and logic as well as the content of the screens and their control elements are variable and depend on the use history or the current context, such as the user's location. Thus, the tool should

not only support the static but also the dynamic design specification by enabling the definition of rule sets for the appearance and behaviour, such as by means of a scripting language.

[FR-T01.2] Support the specification and use of different mobile interaction styles: Since applications should be developed for *the Third Computing Paradigm*, the tool also needs to support the specifications of different mobile interaction styles (see Section 2.2.1) and their possible values (e.g. SAVE as a value of the keyboard-based interaction style). After this specification, the specified mobile interaction styles and their values should be available for the definition of the prototype's behaviour. Thus, the occurrence of a mobile interaction style's value (e.g. SAVE) can be mapped to a change the application's appearance (e.g. the display of a new screen). The occurrence of a mobile interaction style's value, however, not only might be relevant for the dynamic specification of the application flow but also for the dynamic specification of the application logic. The presence of a user at a certain location, for instance, could be used for a dynamically specified application logic.

[FR-T01.3] Support the automatic compliance of approved HCI guidelines: In order to reduce the required interface designer's Usability Engineering skills, the tool assistance should additionally support the automatic compliance of approved HCI guidelines. By this means, the quality of the resulting prototype can potentially be improved with respect to usability. The compliance of approved HCI guidelines, for instance, can be achieved by automatically checking the interface designer's specification as well as by providing warnings about violations.

[FR-T01.4] Support the specification of remote data: Further, mobile applications in the context of *the Third Computing Paradigm* often do not only display local content which is stored on the mobile phone but also remote content of a database. This database contains information about the pervasive computing environment, its objects, services and service entries. During the design specification, the tool is therefore expected to also support the specification of data of a database.

[FR-T02] Support the automatic generation of evolutionary software prototypes with low and high level of fidelity for the intended interaction and presentation device - which meet all non-functional [NFR-P] and functional requirements [FR-P] towards the resulting prototype: Another functional requirement is the need to support the automatic generation of a prototype which bases on the design specification. The generated prototype is expected to automatically run on the

intended interaction device - the mobile phone - as well as meets all non-functional [NFR-P] and functional requirements [FR-P] of a resulting prototype, such as a high level of usability, functionality and reliability. Only by this means, a realistic local or remote evaluation can be executed (see Section 2.1).

Moreover, in early development iterations not only functional high-fidelity prototypes should be producible but also functional low-fidelity prototypes with sketched screens. The use of low-fidelity prototypes has benefits, such as they potentially do not limit the user's willingness to provide detailed and, in particular, rather critical feedback about the prototype. Finally, the resulting prototype should be an evolutionary prototype (see Section 2.1) which means that an interface designer should be able to easily modify this prototype in the following process iteration, such as by easily adding new aspects (e.g. a new screen) or by easily changing existing elements of the prototype (e.g. the application flow). This last aspect is also highlighted by [Hartmann et al., 2006] and [Klemmer et al., 2000].

[FR-T03] Support the conduction of local and remote empirical evaluations - synchronously and asynchronously during the entire development process:

After the generation of a prototype, the tool should also support the conduction of empirical evaluations by using the resulting prototype. These empirical evaluations should be conducted during the entire development process in order to acquire knowledge about the users and their needs towards the intended application even at the very beginning of the development process. Having this knowledge can help prevent misleading assumptions of the development team about the application's required functionalities.

Empirical evaluations of mobile applications are often conducted in the real environment of the end-users - the field - since comprehensive and realistic knowledge about usability problems and user preferences can only be acquired under real contextual constraints. To enable field studies for a large number of users as well as over a longer period of time, a tool-support should assist in the conduction of remote empirical evaluations because this approach reduces organisational problems, such as the presence of the evaluator in the field.

Due to the fact that some user evaluations need to be conducted under controlled context (e.g. controlled loudness and lighting conditions), the tool-support should not only support the conduction of field studies but also user evaluations in a laboratory setting. Evaluations in the laboratory are rather local evaluations since the evaluator and the test users are normally not far away located from each other. The evaluator and the test users can either be in the same test room or the evaluator is located next to the test room in a control room.

Further, local evaluations in the laboratory are typically conducted as synchrony evaluations which means that the test users and the evaluator operate simultaneously. Usually, the test user interacts with the prototype while the evaluator simultaneously tracks the recorded data. By means of synchrony evaluations, the evaluator always can control the evaluation and also might intervene in some situations, such as if the test user has problems with the system and therefore needs help.

Some remote evaluations in the field also need to be conducted as synchrony evaluations, such as if an alarm-triggered Diary Study should be executed (see Section 2.1). A UCP tool, however, should not only support synchrony but also asynchrony evaluations if there are not any temporal constraints towards the evaluation. During asynchrony evaluations, the system simply logs all data while the test user interacts with the system. After the conduction of the evaluation, the evaluator loads the data for interpretation and analysis. Asynchrony evaluations do not require the presence of the evaluator and therefore there are more meaningful when conducting long-term evaluations. In asynchrony evaluations, however, the evaluator cannot control the evaluation. Both kinds of evaluations seem to be useful for evaluators of mobile applications and therefore a UCP tool should support the conduction of local and remote evaluations which can be conducted synchrony and asynchrony during the entire development process.

[FR-T03.1] Support the display and the synchronised logging of quantitative data which are rather objective: As a further and essential functional requirement towards a tool that supports local and remote evaluations, explicit and implicit mobile interactions of the user should be logged synchronously. Having captured these quantitative data can help get a clear picture about the use of the different mobile interaction styles, such as the frequency of their use. Additionally, it is also important to have knowledge about the user and environmental context, such as information about the user activity or the loudness level. These details might provide very important knowledge why the user has behaved in a particular way. The logging of user and environmental contexts is important in both local and remote evaluations, however, it seems to be difficult in remote evaluations if this requires the presence of the evaluator and means a disturbance for the user. An unobtrusive solution would be if all sensors for the measuring of user and environmental context are integrated into the mobile phone itself or into the user's clothes.

In summary, the tool should support the synchronised logging of different quantitative data: explicit and implicit mobile interactions as well as data from sensors (e.g. temperature or body sensors) which provide information about the user and

environmental context. When conducting synchronous evaluations, these data should be displayed for the evaluator in order to always provide knowledge about interesting situations, such as situations which need an alarm-triggered event to capture qualitative data.

[FR-T03.2] Support the display and the capturing of qualitative data which are rather objective: Primarily during the conduction of local evaluations in the laboratory, the tool is expected to also support the audio-visual capturing of the user as well as his environment by means of the system and its stationary cameras and microphones. These qualitative data can simplify the analysis and interpretation of a user evaluation. For example, the data can help acquire knowledge why a user has a certain preference towards an mobile interaction style. During the conduction of remote evaluations, the systematic capturing of audio-visual content by means of stationary cameras and microphones is difficult in a mobile setting. In order to also get qualitative data during the conduction of a remote evaluation, the users should be able to capture qualitative data about their context by themselves, such as the test users should be able to take pictures or capture videos about their environments. If this kind of qualitative capturing should be conducted alarm-triggered by the evaluator, the tool additionally needs to support a remote communication between the evaluator and test user. Finally, when conducting a user evaluation in the field or the laboratory, it is further very important to provide details about the different states of the prototype by logging the prototype's appearance: the currently displayed screen.

All in all, the tool is expected to not only support the logging of quantitative data (e.g. the logging of explicit and implicit interactions) but also the capturing of qualitative data during both, the conduction of local and remote evaluations. Similar as for quantitative data, the incoming qualitative data should also be displayed for evaluators of synchronous evaluations to always enable the control during the conduction of a remote or local evaluation.

[FR-T03.3] Support the display and the synchronised logging of qualitative and quantitative data which are rather subjective: When conducting a synchrony remote or local evaluation, the evaluator should further be able to log data which are rather subjective, such as details about a task that is currently conducted or details about incidents that might be of interest during the analysis. These data can be rather qualitative or quantitative.

In addition to the evaluator, the test users also should be able to log qualitative data, such as user comments, and quantitative data, such as ratings about statements that are displayed from time to time. Due to the fact that an audio-visual

capturing of qualitative data is difficult during the conduction of remote evaluations, the capturing of user feedback and their ratings are a very important source of knowledge to get details about the user's situation while performing a task.

In summary, not only objective but also subjective data can be important for a later on analysis and therefore the logging of subjective data - qualitative and quantitative - should be supported by a tool assistance for both the evaluator and the test users.

[FR-T03.4] Support the capturing of several users: Since mobile applications cannot only address single-users but also multi-users (see Section 3.1), the tool additionally requires to assist the capturing of several users while they are interacting together with the system. Thus, the evaluator can later on analyse the relationship between different users and their behaviour.

[FR-T04] Support analytical evaluations - model-based and inspection-based - during the entire development process: Besides the conduction of empirical evaluations, the tool should also assist the conduction of analytical evaluations. Analytical evaluations are conducted with experts in terms of the domain and/or usability (see Section 2.1). Normally, analytical evaluations are executed prior to an empirical evaluation in order to find and eliminate obvious problems of usability which would distract from hard to find problems in empirical evaluations. An analytical evaluation either can be conducted with an empirically validated model to investigate the user's efficiency when completing a corresponding task (e.g. KLM) or by means of guideline-based and/or walkthrough-based inspections of the generated prototype (e.g. *the Heuristic Evaluation*). As a further functional requirement, the tool-support should support the mentioned model-based and inspection-based analytical evaluations.

[FR-T05] Support analyses of the captured data during the entire development process: When having finished the conduction of an empirical evaluation, the tool should support the analysis and interpretation of the logged data.

[FR-T05.1] Support the synchronised display of all captured data as well as the automatic pre-annotation of the qualitative data by means of the quantitative data: As a first requirement for the tool-based analysis, the tool is expected to support a graphical user interface that displays all logged qualitative and quantitative data synchronously. One often used visualisation approach is the application of a time-line based presentation of the qualitative data (e.g. videos) together with the logged quantitative data (e.g. explicit and implicit mobile interactions).

The tool is further expected to pre-annotate the qualitative data based on the logged quantitative data. The automatic pre-annotations cannot only prevent errors and save time, they also can provide evaluators with hints about sequences that might be of interest, such as a sequence which contains a lot of explicit mobile interactions or user comments. Without the automatic pre-annotation, the evaluators have to review the different video sequences and label them by events of interest in order to enable an interpretation and quantification of the qualitative data (see Section 2.1).

[FR-T05.2] Support interactivity to analyse and modify the data: As a further requirement, the graphical user interface of the analysis component needs to be interactive in order to enable the evaluator to freely review the captured qualitative and quantitative data to get a clear picture for interpretations and a basis for a decision-making in a new iteration. The evaluator, for instance, might jump to a certain sequence of the video in order to verify the user behaviour and the user and environmental context. The evaluator also needs to be able to modify the annotations and thus the system should enable the addition of new annotations as well as the removal or change of existing annotations.

[FR-T05.3] Support the statistical analysis of the captured data: The statistical analysis of quantitative data or quantified qualitative data is an often executed task during the analysis phase and thus should be assisted by the tool as well. The evaluator, for instance, can verify whether the frequent use of a certain mobile interaction style occurred randomly or significantly. Thereby, the evaluator can decide on the importance to consider a found user behaviour (e.g. a user preference) in further iterations.

[FR-T05.4] Support the loading and display of several captured sessions: As a last requirement towards the analysis component, the tool is expected to enable the loading and display of several captured sessions, such as sessions of different test users. Grounded on this feature, the evaluator can compare different sessions. For instance, the evaluator can verify whether each user has problems with a specific task.

[FR-T06] Support all phases of user-centred prototyping process with a strong linking: Finally, a user-centred prototyping tool should assist all phases of the user-centred prototyping process: the design specification and automatic generation of a prototype as well as the conduction of an evaluation and finally its analysis. Further, a strong link shall additionally be supported between all phases, such as all logged data of the evaluation component can easily be loaded and displayed by the analysis component.

3.3 Related Tools

In the last section, challenges were identified for interface developers of mobile applications in the context of *the Third Computing Paradigm*. These challenges revealed typical requirements for a software assistance of the user-centred prototyping process. In this section existing software tools are introduced and investigated based on the functional requirements in order to highlight strength and weaknesses of the corresponding tools.

One challenge found for developers of mobile applications is the need to have appropriate programming skills. Even if software modules are used that can be specified by means of XML files, interface developers still need to have comprehensive Software Engineering skills. A solution to tackle the challenge is to only make use of tools that provide a graphical user interface (GUI) for the specification and automatic generation of prototypes as well as for the conduction and analysis of evaluations. As a consequence, the following analysed related software tools are either content and behaviour tools which support the design specification and implementation of prototypes or evaluation and analysis tools. Some of the tools are all-in-one user-centred prototyping tools which support all phases of the user-centred prototyping process. The focus is not on the other kinds of software support which were presented in Section 2.3. The focus of the literature review is further on tools which assist the development of mobile applications.

3.3.1 Tools for the Design Specification and Implementation

There are several tools available which help generate prototypes of an application. Some of these tools focus on the generation of sketches of the intended application - classical prototyping tools - whereas other tools support the generation of functional high-fidelity prototypes which directly run on the intended platform - content and behaviour tools (see Section 2.3). Axure³ and iplotz⁴ are examples of classical prototyping tools which help easily generate mock-ups of websites whereas MockApp⁵ is a classical prototyping tool that can be used for the generation of mock-ups of mobile applications. These tools aide to get a first impression of the intended application which is important for discussions of the multi-disciplinary development team as well as for user evaluations of different design ideas in early process iterations. The final application, however, is often difficult to implement with classical prototyping tools. This can often be done by so-called content and

³<http://www.axure.com/>

⁴<http://iplotz.com/>

⁵<http://mockapp.com/>

behaviour tools (see Section 2.3). These tools are on the focus of the dissertation and therefore the following software assistances are rather tools of this software category.

The App Inventor from Google⁶ is an example that not only supports the GUI generation as a GUI builder (see Section 2.3) but also the entire generation of the application. This can be done due to the fact that the App Inventor makes use of a visual programming language which hardly requires programming skills. Developers can concentrate on the concept and evaluation of the applications and do not need to spend too much time on aspects of programming. There seems to be a trend to also enable people with less skills to design and create their own applications for different domains.

OIDE [McGee-Lennon et al., 2009] and iStuff Mobile [Ballagas et al., 2007], for instance, follow this trend. They are tools which support the use of multimodal sensor input. Results of these tools are not functional high-fidelity prototypes of the intended application which directly run on mobile devices but instead a GUI-based specification of the mobile interaction styles. Thus, both tools are not typical content and behaviour tools but they provide an elaborated graphical specification of interaction styles. OIDE is part of *the Open Interface platform*. It enables the addition and combination of different input and output components in order to specify the behaviour of the application. For this specification the interface developer uses a state-chart view where the states represent the different components and the transitions the relationships between the components. By this means, input components can be linked to output components, such as a specific value of the phone's accelerometer can be mapped to the display of a corresponding content on a public display. Studies [McGee-Lennon et al., 2009] showed that users found benefits in using OIDE and the graphical user interface for the specification of mobile interaction styles by means of a state-chart view. This visualisation seems to be a useful approach for the specification of a prototype's behaviour since iStuff Mobile also uses a very similar method compared to OIDE. Using iStuff Mobile, interface developers also can make use of a state-chart view in order to link context input to a specific system behaviour, such as that data of a mobile phone's position sensor can be mapped to the position of an image displayed on a public display.

Another related tool for the development of mobile applications is called TERESA [Chesta et al., 2004]. It provides assistance for the tool-based design of functional nomadic interfaces. The focus of nomadic interfaces is to generate applications for different platforms, such as desktop PCs or mobile phones. For the genera-

⁶<http://appinventor.googlelabs.com/about/>

tion of applications with TERESA, different phases of design specifications exist which start with the specification of high level tasks and end with the platform-dependent specification, such as the specification of the typical appearance of the corresponding device. As the result of the design specification, functional high-fidelity prototypes are automatically generated for the different platforms and thus an implementation phase is not required anymore.

By means of TERESA, interface developers cannot only specify the behaviour of the application but also the appearance of different screens as well as their static content (texts, images, audio and video files). This specification is done by a state-chart view: screens are represented by states whereas the mobile interactions are represented by transitions. Thus, at the runtime, an explicitly or implicitly executed mobile interaction triggers a specified transition and leads to another state and the display of another screen. For instance, a keyboard-based mobile interaction (e.g. the selection of the command `SAVE`) can trigger a transition and thus a changed appearance - the display of a new screen.

MScape [Hull et al., 2004] is another tool that supports the automatic generation of mobile applications based on a design specification. Since the resulting prototypes are location-based applications, maps can be used to specify the behaviour of the intended application. With MScape's graphical user interface and the used maps, points of interest (POI) and implicit interactions can be defined, such as which event should be triggered once the user has entered a corresponding POI. As results of triggered events, different multimedia content can be displayed. In contrast to the other related tools, MScape also enable the specification of dynamic content by means of a scripting language. At runtime, MScape's used software modules can interpret the rule sets which were defined for the application in order to correctly display content. Using a scripting language increases the required software skills but it also enables much more use possibilities and realistic applications since the behaviour of the application can be specified more detailed.

Topiary [Li et al., 2004] is a further tool that supports the design specification of location-based applications for mobile devices. The tool is quite similar to MScape but in contrast to it, content specifications are specified just statically. As a further difference, Topiary does not only support functional software tests of the generated prototypes as MScape but also empirical remote evaluations (see Section 2.1) where the evaluator and the test users are spatially separated.

The tool-supported evaluations with Topiary are conducted as Wizard-of-Oz studies [Dahlbäck et al., 1993]. In these studies an evaluator controls the location contexts of the user in the hidden background in order to conduct a study as realistic as possible for the participant even though if the application is not

completely functional. The test users normally do not realise the evaluator in the background. They simply can use a PDA to see the application's screens and their static content (e.g. images or text) dependent on the user's location context. During the user evaluation, the evaluator uses a graphical user interface that presents the user's location on the map as well as the current active screen of the user. Thus, the evaluator can always trace the test users.

Similar to Topiary, MakeIT [Holleis and Schmidt, 2008] is also a tool which supports both the generation of prototypes and the conduction of evaluations. But in contrast to Topiary, MakeIT does not assist the conduction of empirical evaluations but instead model-based analytical evaluations which ground on the Keystroke-Level Model (KLM) for mobile interaction [Holleis et al., 2007]. KLM is a simplified version of GOMS (see Section 2.1). It can be used to determine the time to complete a specific task by means of the required operations (e.g. keyboard-based mobile interactions). Thus, MakeIT assists in calculating the time for a task completion and therefore it can estimate the user's efficiency.

3.3.2 Tools for the Evaluation and Analysis

The following section discusses tools which support the evaluation and analysis phase. [Carter et al., 2007] see a need to provide evaluation and analysis tools for interface designers which help improve prototypes even though if they are not completely finished.

Compared to design and implementation tools, there are fewer tools which assist primarily objective methods (see Section 2.1) during the conduction and analysis of user evaluations. Most available evaluation and analysis tools support in the logging of web traffic and the visualisation of the logged data (e.g. Google Analytics⁷ or [Arroyo et al., 2006]).

Other tools do not only enable the logging of user events and the presentation of the logged data by means of charts but also the playback of the user interface's appearance - the webpage content - as a video file in the analysis phase in order to also get insights about content of interest (e.g. Userfly⁸). Some tools even combine objective user loggings with subjective inquiries, for instance, by asking for answers about short questions from time to time during the conduction of the evaluation in order to also get user feedback (e.g. Usabilla⁹).

⁷<http://www.google.com/analytics/>

⁸<http://userfly.com/>

⁹<http://usabilla.com/>

When analysing the captured data, the just mentioned tools provide very interesting insights to user behaviour and preferences by means of the logged quantitative data. Qualitative data are often not captured which show the users and their environments (e.g. by means of audio or video files). This aspect, however, might cause a problem for the interpretation of the logged data. The following focus is therefore not on tools which only log user events or subjective data but more elaborately support the conduction and analyses of user evaluations, such as by also recording audio-visual content.

Moare¹⁰ is an industrial tool that fulfils these requirements. It can be used for remote web evaluations if the evaluator and the test users are spatially separated but not temporally. User data are logged similar to the other tools but audio-visual content of the user is also captured. After the evaluation, the tool helps analyse the captured data by providing a chart representation of the logged data. DRUM [Macleod and Rengger, 1993] is a prominent evaluation and analysis tool from research that supports similarly compared to Moare when conducting evaluations for desktop applications. DRUM was developed to support audio-visual recordings of user sessions as well as the logging of user events and comments of the evaluator. In the analysis phase, the evaluator can use DRUM in order to diagnose the captured video data on usability problems as well as on the performance data of the users (effectiveness and efficiency), such as the level of task completion. In this analysis phase the evaluator can manipulate the logged data or add new data, such as new comments of the evaluator which occur retrospectively.

Other tools also help capture physiological data while users are interacting with the system in order to get more insights about the users and their context, such as about their engagement and attention when using a system. SSI [Wagner et al., 2009]), a research tool, and Observer XT¹¹, an commercial example from Noldus, are evaluation tools which support the capturing of different objective data including physiological data (e.g. the heart rate or skin conductance). SSI, for instance, was used to analyse trust-related behaviour based on the user's heart rate and eye fixation while using a website [Leichtenstern et al., 2011b].

Most of the mentioned evaluation and analysis tools concentrate on desktop applications. The following evaluation and analysis tools focus on mobile applications. For this domain, user evaluations can locally be conducted in a laboratory if controlled experiments are to be conducted with controlled contextual situations. Controlled experiments can be very helpful to answer different open questions, such as which mobile interaction style is preferred in a specific contextual situation (see Section 3.1). This kind of user evaluation can be conducted in a very

¹⁰<http://www.techsmith.com/morae.asp>

¹¹<http://www.noldus.com/human-behavior-research/products/the-observer-xt>

similar way as for desktop settings, such as user events can be logged while the user is audio-visually captured.

In practice, however, mobile applications are often used in different situations in terms of context, such as at home or outdoor. Knowledge about these different environments and emerging contexts is inevitable to address all user requirements. Thus, user studies for mobile devices often also need to be conducted under real contextual constraints in-situ in order to get useful insights to user behaviour and preferences as well as usability problems (e.g. [Häkkinen and Mäntyjärvi, 2006, Palen and Salzman, 2002b]). One option to conduct user studies in the field are direct observation methods (e.g. [Kellar et al., 2005]). Direct observation methods are evaluations where the evaluator is located at the same place as the test users which is difficult to organise and conduct for field studies. Consequently, for being cost-efficient, only a small number of users can be observed for a short time. To evaluate a larger number of test users over a longer period of time, user evaluations for mobile applications are therefore often conducted as remote evaluations where the evaluator and the test users are separated spatially and sometimes also temporally.

ContextPhone [Raento et al., 2005] is a tool which provides assistance for the conduction of remote evaluations if the evaluator and the test users are spatially and temporally separated. During the conduction of an evaluation, ContextPhone can be used to log all user events (e.g. text messages) as well as different information about the user and his context, such as his GPS locations or activity state.

The developers of Momento [Carter et al., 2007] also mention the need to support local and remote evaluations, a synchronised data storage of user events together with user context as well as later on the analysis of the captured data. Moreover, they also see a need to combine the recording of quantitative and qualitative data. Qualitative data (e.g. videos) are difficult to capture in a field setting and therefore the developers of Momento see a need to log qualitative user data similar to diary studies (see Section 2.1). The users receive alarm-triggered messages from time to time which remind them to capture pictures and thereby provide qualitative details about their situations.

If the diary-study-feature of Momento should be used during the conduction of an evaluation, the evaluation setting requires the temporal presence of the evaluator in order to enable the timed transfer of reminders. For these just-in-time evaluations, a graphical user interface is provided to the evaluator which displays the just logged data in a time-line based GUI together with the captured pictures of the user.

MyExperience [Froehlich et al., 2007] is a further tool that supports the conduc-

tion of evaluations but, in contrast to Momento, also the analysis of the captured data. Apart from that aspect, MyExperience is quite similar to Momento since it also helps remotely log all user events and user context as well as qualitative data. Qualitative data are received by making use of *the Experience Sampling Method* (see Section 2.1). Thus, the users reply data (e.g. images) to their situation which are triggered by events of the evaluator. During the analysis phase, MyExperience supports interface developers with an analyser tool that displays the stored data. This analyser tool, however, only displays a list of the logged data which is difficult to read and to understand. More useful would be the appropriate data visualisation of all stored data in order to quickly and easily find usability problems or information about the users, such as their performances and preferences.

There are further tools which help when analysing captured data. ANVIL [Kipp, 2001], for instance, supports the display and the playing of audio-visual content - the captured video files - as well as the visualisation and modification of annotations at various freely definable time-line based tracks. Annotation means that the interface developers can load a captured video file, label it based on their interest (e.g. user activities) in order to enable quantifiable analyses of the qualitative data, such as the analysis how often the user performed a certain gesture (see Section 2.1). If a statistical analysis is desired, the labelled video data can also be exported to formats of a statistic tool, such as SPSS.

A similar tool called ModelUI [Wagner et al., 2010] can also be used to annotate captured data (e.g. audio and video files together with screen captures) by means of a time-line based graphical user interface. As a difference to ANVIL, ModelUI does not only enable the loading of qualitative audio-visual content but also the loading of all logged quantitative data of a user evaluation (e.g. the user events and physiological data) which were synchronously captured by means of the evaluation tool SSI [Wagner et al., 2009]. Thus, the tracks of ModelUI are automatically filled with labels. That is a pre-annotation of the captured qualitative data with the captured quantitative data. Pre-annotations can save time and reduce errors since the interface developers do not need to do this step themselves for the logged data (e.g. user events) anymore but instead the interface developers can concentrate on the core task of the analysis: the identification of usability problems and user requirements. Noldus' Observer XT¹² also supports similar features compared to ModelUI.

¹²<http://www.noldus.com/human-behavior-research/products/the-observer-xt>

3.3.3 Tools for all Phases

The only known tools which support all phases of the user-centred prototyping are OmniSCOPE [de Sá and Carriço, 2009], SUEDE [Klemmer et al., 2000] and d.tools [Hartmann et al., 2006].

OmniSCOPE is a tool which assists the generation of software artefacts for patients of therapists. By means of the tool, static multimedia content can be defined and prototypes can be generated for mobile devices. The patient can use these prototypes as part of their therapy, such as they can either generate multimedia content by themselves (e.g. video files) in order to document their life or they can watch provided multimedia content of the therapists in order to get support in their daily life. While the users interact with the system, the prototype automatically records all user interactions which later on can be analysed and interpreted by the therapist. Thus, OmniSCOPE's generated applications enable remote evaluations with a temporal and spatial separation of the test user and evaluator.

During the analysis phase, the captured data of the user evaluation can be analysed by means of a graphical user interface that presents the recorded objective data, such as video files. The synchronised display of all logged qualitative and quantitative data, however, is not supported.

Another UCP tool called SUEDE assists in the iterative development of speech interfaces. User interface developers can use SUEDE to design dialogue examples, evaluate the examples in a Wizard-of-Oz study [Dahlbäck et al., 1993] and later on to analyse the evaluation, such as the user's used dialogue path during the test. Therefore, different graphical user interfaces are supported. By means of a state-chart view the different dialogues are modelled. When switching to the test mode, audio files are generated for the modelled dialogues in order to enable a simulation of the speech interface. Thus, SUEDE does not generate prototypes for the final setting but just prototypes for the Wizard-of-Oz study.

In the evaluation phase, the wizard is provided with a graphical user interface. This GUI can be used to simulate a running speech interface that recognises speech input from the user and replies appropriate dialogues. In order to increase the realism of the system, the evaluator also can simulate errors of speech recognition, time-outs of the speech recogniser and barge-ins which means the overlay if the system reply starts before the user has finished its speech input. During the evaluation phase, all of the user's dialogues are recorded as audio files as well as the reaction of the Wizard are logged for the later on analysis. In the analysis phase, SUEDE presents all graphs of the design phase and highlights the used path. It also provides assistance for the statistical analysis, such as to analyse the

time required to complete a task or the number of users who preferred a certain path compared to alternative paths.

SUEDE shows different benefits of user-centred prototyping tools. A decisive advantage of all-in-one user-centred-prototyping (UCP) tools is the strong link between the design, evaluation and analysis component. For instance, the evaluation component supports the conduction of evaluations with prototypes which were automatically generated during the tool-based design whereas the analysis component assists interface designers with the interpretation of synchronously captured qualitative and quantitative data of the evaluation. Consequently, problems of compatibility between the different components can be prevented that would typically happen when using separate tools for the different phases. A further benefit is that interface developers do not need to learn different tools for the different phases since they can use a single software to run all phases which can reduce training periods. The user evaluation in SUEDE are typically conducted as local evaluations. Thus, aspects of other tools (e.g. remote communication) are not considered.

The UCP tool d.tools [Hartmann et al., 2006] supports similar features as SUEDE but addresses a different application domain. D.tools can be primarily applied to develop, test and analyse new information appliances, such as new media players or cameras and their buttons and sliders. Having a tool support for this application domain is very helpful since interface developers even more require hardware and programming skills in order to address the different hardware components. During the design specification with d.tools, a state-chart view can be used to define the appearance of the application - its screens and their static content - as well as the application behaviour. Furthermore, sensor input of the intended hardware can be specified. In the evaluation phase, audio-visual content can be logged together with the user events. For these test sessions, the interaction devices (e.g. a media player), however, need to be connected to the interface designer's desktop PC. Thus, the generated prototypes do not directly run on the intended device that, however, is a precondition for a tool-supported assistance to execute remote evaluations.

After the conduction of the evaluation, the captured data can be analysed by means of a time-line based visualisation of the audio-visual data and the logged user events. The evaluator can switch to a certain logged event which is directly mapped to a screen state of the design GUI. Thus, the interface designer has a direct link between the analysis component and the design component. This also can be done vice-versa. The evaluator can select a screen state of the state chart in order to see the sessions of the captured data which address this screen state.

	[FR-T01]	[FR-T02]	[FR-T03]	[FR-T04]	[FR-T05]	[FR-T06]
App Inventor	+	+	-	-	-	-
TERESA	+	+	-	-	-	-
OIDE	0	-	-	-	-	-
MakeIT	0	-	-	0	-	-
MScope	+	+	-	-	-	-
Topiary	+	0	+	-	-	-
iStuff Mobile	0	-	-	-	-	-
ContextPhone	-	-	+	-	-	-
MyExperience	-	-	+	-	0	-
Momento	-	-	+	-	-	-
ANVIL	-	-	-	-	+	-
ModelUI	-	-	0	-	+	-
OmniSCOPE	+	+	+	-	0	0
d.tools	+	0	0	-	+	+
SUEDE	+	0	0	-	+	+

Table 3.1: The table compares the different introduced tools (see Section 3.3) and presents how well they meet the functional requirements towards a tool-support for the user-centred prototyping process (-: poor, 0: medium, +: good). These tools primarily assist during the mobile application development. **[FR-T01]** Support the design specifications during the entire development process. **[FR-T02]** Support the automatic generation of evolutionary software prototypes with low and high level of fidelity for the intended interaction and presentation device - which meet all non-functional [NFR-P] and functional requirements [FR-P] towards the resulting prototype. **[FR-T03]** Support the conduction of local and remote empirical evaluations - synchronously and asynchronously during the entire development process. **[FR-T04]** Support analytical evaluations - model-based and inspection-based - during the entire development process. **[FR-T05]** Support analyses of the captured data during the entire development process.

3.3.4 The Tool Comparison

Based on the acquired practical knowledge (see Section 3.1), functional requirements towards a UCP tool were defined (see Section 3.2). The following section compares the related tools by means on these functional requirements.

Table 3.1 shows a comparison of the main functional requirements towards a user centred prototyping tool. First of all, a tool is expected to support the GUI-based specification of a prototype (see Requirement [FR-T01]). Most related tools (e.g. TERESA or d.tools) support that requirement. The requirement is just partly fulfilled if the design specification is assisted (e.g. by assisting the specification of the interaction styles) but the tool still requires comprehensive programming skills (e.g. OIDE or iStuff Mobile).

Compared to a support of the design specification, less tools automatically generate functional prototypes which run on the intended device (see Requirement [FR-T02]). Only the App Inventor, TERESA, MScape and OmniScope generate such functional and evolutionary prototypes. Generated prototypes of Topiary, d.tools and SUEDE are less functional. Prototypes of Topiary, for instance, require a human simulator (see Section 3.3) in order to generate location context during the conduction of a user evaluation since the GPS-based interaction style is only partly functional. Prototypes of SUEDE and d.tools do not run on the intended interaction device. Thus, remote evaluations cannot be conducted by these tools.

The support to conduct both kinds of user evaluations - local and remote evaluations - is a further requirement (see Requirement [FR-T03]). This requirement is fulfilled by several tools (e.g. MyExperience and OmniScope). Due to that support, these tools can be used to run real evaluations - field studies - and simulated evaluations - laboratory studies - in the real world. Simulations in the virtual world are not supported by the related tools.

Further, a tool should not only support the conduction of empirical but also analytical evaluations (see Requirement [FR-T04]). From the introduced tools, only MakeIT assists analytical evaluations - the model-based evaluations. None of the tools provide assistance for both kinds of analytical evaluations: model-based and inspection-based.

For the analysis phase, some tools (e.g. ANVIL, ModelUI) offer a graphical user interface to present all logged qualitative and quantitative data (see Requirement [FR-T05]) while other tools just present a list of the logged data - MyExperience - or a option to display the captured videos - OmniScope. All three phases of the user centred prototyping process (see Requirement [FR-T06]) - the design specifi-

cation, the conduction of evaluations and the analysis of captured data - are just supported by OmniScope, d.tools and SUEDE.

In summary, the table shows that most related tools focus on one or two requirements and do not consider all requirements to support an interface developer during the entire phases of the user-centred prototyping process.

	[FR-T01]	[FR-T01.1]	[FR-T01.2]	[FR-T01.3]	[FR-T01.4]
App Inventor	+	+	+	-	-
TERESA	+	0	-	-	-
OIDE	0	-	+	-	-
MakeIT	0	0	0	-	-
MScope	+	+	+	-	-
Topiary	+	0	+	-	-
iStuff Mobile	0	-	+	-	-
OmniSCOPE	+	0	0	0	-
d.tools	+	0	+	-	-
SUEDE	+	0	0	-	-

Table 3.2: The table compares the different introduced tools (see Section 3.3) and presents how well they meet the functional requirements for the design phase (-: poor, 0: medium, +: good). [FR-T01] Support the design specifications during the entire development process. [FR-T01.1] Support the static and dynamic specification of the prototype's appearance and behaviour. [FR-T01.2] Support the specification and use of different mobile interaction styles. [FR-T01.3] Support the automatic compliance of approved HCI guidelines. [FR-T01.4] Support the specification of remote data.

The Design Specification Table 3.2 shows how the related tools meet more concrete requirements towards the tool-supported design specification. Most tools support the GUI-based specification of the prototype's appearance and behaviour (see Requirement [FR-T01.1]) but less tools - just the App Inventor and MScope - support the dynamic specifications of the prototype by means of a scripting language. For the GUI-based specification of the interaction styles (see Requirement [FR-T01.2]), some tools provide very elaborated approaches to specify values of the styles. Topiary and MScope, for instance, provide a map to specify points of interest.

For the automatic appliance of approved HCI guidelines (see Requirement [FR-T01.3]) of the corresponding application domain (e.g. mobile HCI), the tool assistance is low for the introduced tools. Just OmniScope provides assistance by making use of a layout manager.

	[FR-T03]	[FR-T03.1]	[FR-T03.2]	[FR-T03.3]	[FR-T03.4]
Topiary	+	+	-	-	-
ContextPhone	+	0	-	-	+
MyExperience	+	0	0	0	+
Momento	+	+	+	0	+
ModelUI	0	+	+	-	-
OmniSCOPE	+	0	0	0	+
d.tools	0	+	+	-	-
SUEDE	0	+	-	-	-

Table 3.3: The table compares the different introduced tools (see Section 3.3) and presents how well they meet the functional requirements for the evaluation phase (-: poor, 0: medium, +: good). **[FR-T03]** Support the conduction of local and remote empirical evaluations - synchronously and asynchronously during the entire development process. **[FR-T03.1]** Support the display and the synchronised logging of quantitative data which are rather objective. **[FR-T03.2]** Support the display and the capturing of qualitative data which are rather objective. **[FR-T03.3]** Support the display and the synchronised logging of qualitative and quantitative data which are rather subjective. **[FR-T03.4]** Support the capturing of several users.

The Evaluation Apart from the requirements towards a tool-supported design specification, also the assistance of a tool-supported conduction of a user evaluation is compared more detailed (see Table 3.3). A basic function of a tool-supported evaluation is the support to synchronously log quantitative data which are rather objective (see Requirement [FR-T03.1]), such as all explicit and implicit interactions. All tools address that requirement. Some of them, however, do not additionally provide a graphical user interface to display the incoming data (e.g. Context Phone and MyExperience).

A further requirement aims at the logging of objective data which are rather qualitative (see Requirement [FR-T03.2]). Momento, ModelUI, and d.tools provide that assistance as well as the display of the data. My Experience and OmniScope also provide the assistance to log that data but do not additionally present the captured data during the evaluation.

An audio-visual capturing can be conducted by means of the evaluator component during local evaluations whereas when executing a remote evaluation (e.g. in the field), audio-visual data can be captured by means of the users. The related tools either focus on the capturing in remote settings by means of the user - MyExperience, Momento and OmniScope - or on the capturing in local settings by

means of the evaluator - ModelUI and d.tools.

Since also subjective data are of interest for a later on analysis, not only objective but also subjective data should be logged (see Requirement [FR-T03.3]). Usually, tools that address remote evaluations also support options to support the logging of subjective data by means of the user. None of the tools, however, display that data during the evaluation phase.

Finally, the tools should also support the capturing of several users (see Requirement [FR-T04]). This feature is of particular concern if the application is a multi-user setting and the interplay between the users should be analysed. Context-Phone, MyExperience, Momento and OmniScope enable the synchronous logging of several users.

	[FR-T05]	[FR-T05.1]	[FR-T05.2]	[FR-T05.3]	[FR-T05.4]
MyExperience	0	-	-	-	-
ANVIL	+	0	+	+	-
ModelUI	+	+	+	+	-
OmniScope	0	-	-	-	-
d.tools	+	+	+	+	+
SUEDE	+	+	+	-	-

Table 3.4: The table compares the different introduced tools (see Section 3.3) and presents how well they meet the functional requirements for the analysis phase (-: poor, 0: medium, +: good). **[FR-T05]** Support analyses of the captured data during the entire development process. **[FR-T05.1]** Support the synchronised display of all captured data as well as the automatic pre-annotation of the qualitative data by means of the quantitative data. **[FR-T05.2]** Support interactivity to analyse and modify the data. **[FR-T05.3]** Support the statistic analysis of the captured data. **[FR-T05.4]** Support the loading and display of several captured sessions.

The Analysis When having conducted a user evaluation, the analysis component supports the analysis of the captured qualitative and quantitative data. Table 3.4 shows the detailed requirements towards the analysis phase. Most tools - except MyExperience and OmniScope - provide a synchronised display of all logged data as well as the pre-annotation of the qualitative data by means of the quantitative data (see Requirement [FR-T05.1]). Just ANVIL does not provide the automatic pre-annotation of the logged data. For the tool-supported analysis, another typical feature is the support of interactivity to analyse and modify the data (see Requirement [FR-T05.2]). Most tools meet that requirement as well as the Requirement [FR-T05.3]: the support to statistically analyse the data. The display

of several captured test sessions (see Requirement [FR-T05.4]), however, is only supported by d.tools.

3.4 Summary

The objective of this chapter was to reveal typical requirements towards a user-centred prototyping tool and its resulting prototypes.

To get knowledge about these requirements, the chapter presented the user-centred development of mobile applications in different domains: ambient-assisted living and mobile outdoor and indoor aides as well as mobile learning and entertainment. These single and multi-user applications revealed several challenges for developers of mobile applications in the context of *the Third Computing Paradigm* and thus derived functional and non-functional requirements.

After the specification of the requirements, related software tools were introduced and discussed on whether they meet the requirements or not. By this means, strengths and weaknesses of the corresponding tools were highlighted.

The next chapter goes into the details about the development of a user-centred prototyping tool. The chapter aims at the question how the different rather functional requirements can be fulfilled and which challenges emerge for developers of user-centred prototyping tools when trying to meet as many requirements as possible. It also presents new conceptual and technical approaches and highlights differences to the related software tools.

Chapter 4

The Tool Development

Besides the specification of different requirements towards user-centred prototyping (UCP) tools and their resulting prototypes, a further objective of this thesis is to research for conceptual and technical approaches to meet these requirements. In order to acquire the necessary experience and demonstrate the feasibility of the used approaches, a tool called MoPeDT - Pervasive Interface Development Toolkit for Mobile Phones - was developed within this dissertation.

In this chapter the tool development of MoPeDT is presented with a focus on approaches how to meet the functional requirements towards a UCP tool and its resulting prototypes. First, software aspects of the automatic generation of software prototypes are discussed and then aspects of the GUI-based UCP tool. Finally, the used approaches are compared with the related tools.

4.1 The automatic Generation of Prototypes

The automatically generated prototypical applications must meet all non-functional and functional requirements towards a tool-generated prototype. Thus, the resulting prototypes should be highly usable (see Requirement [NFR-P01]), reliable (see Requirement [NFR-P02]) and functional (see Requirement [NFR-P03]) as well as they should enable the use of mobile interaction styles (see Requirement [FR-P01]), the remote access to load and display (see Requirement [FR-P02]) or capture and modify content (see Requirement [FR-P01]) for several users (see Requirement [FR-P04]). Apart from the mobile devices, further presentation and interaction devices should be addable (see Requirement [FR-P05]). Finally, when executing evaluations, the prototype should support the storage of quantitative and qualitative data (see Requirement [FR-P06]) as well as a remote

communication between the test user and the evaluator (see Requirement [FR-P07]).

In order to enable the automatic generation of evolutionary software prototypes which meet all mentioned requirements, an appropriate software architecture and software components - software modules - are required. In the following, an approach of a client-server architecture is described that was used for the automatic generation of software prototypes by means of MoPeDT. These aspects also were published within this PhD project [Leichtenstern and André, 2009b, Leichtenstern and André, 2010, Leichtenstern and André, 2011, Leichtenstern et al., 2011a].

4.1.1 The Software Architecture and its Components

The needed client-server architecture of MoPeDT is expected to support the user-centred prototyping process of applications which use mobile phones as interaction and presentation device to a pervasive computing environment (see Section 2.2.1). To meet this application domain, the used component-based client-server architecture of MoPeDT (see Figure 4.1) enables the plug&play of the following components: physical objects, mobile users, a main server and a database as well as sensors, actuators and evaluators. In the following these different components and their meaning towards the resulting prototype are presented.

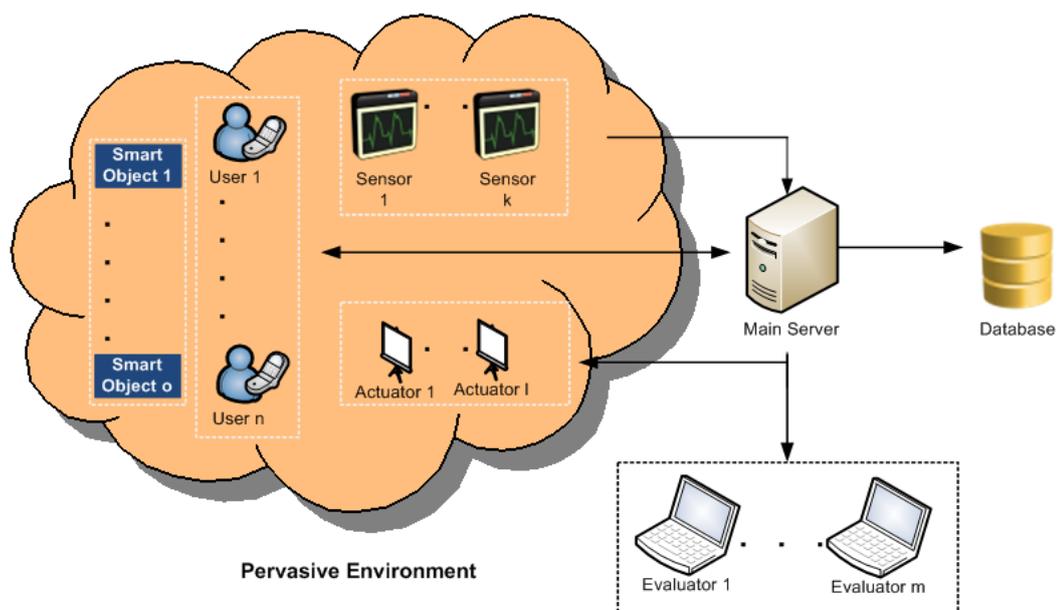


Figure 4.1: The software architecture of MoPeDT consisting of physical objects, mobile users, a main server and a database as well as sensors, actuators and evaluators.

The physical objects: As described in Section 2.2.1, physical objects (e.g. objects of arts) are everyday objects which are located in a pervasive computing environment (e.g. a museum) that can be rather public, private or game-like (see Section 3.1). A characteristic of physical objects is that they provide electronic services, such as detailed information about an object of art and its artist. Physical objects are often not directly addressable in order to call and use the electronic services. Instead, mobile phones can be used as a medium to interact with the physical objects - select and identify them - by means of different mobile interaction styles (see Section 2.2.2) as well as present the electronic services and other content on the mobile phone's display.

When using the architecture of MoPeDT, different physical objects can be added to a public, private or game-like pervasive computing environment by storing their representation (ID, name and icon), services, service entries and content into MoPeDT's database. In the context of MoPeDT, the identifier of a physical object (e.g. a specific object of art in a museum) accords with the identifier of the corresponding physical object's entry in MoPeDT's database table called `object`. By this means, MoPeDT's prototypical mobile applications are able to load and display all details which are stored in the database once the identifier of an intended object is known. This identifier is determined by the mobile users and their used mobile interaction styles.

The mobile users: The mobile users are the second component of the architecture if the application domain is concerning *the Third Computing Paradigm*. Mobile users are equipped with their mobile phones which can be used as user interfaces to pervasive computing environments and their different physical objects. On these mobile phones, mobile applications - the generated prototypes - run which base on the software module called `MOBILE CLIENT` (see Section 4.1.2). These applications can be applied for the interactions with the physical objects and for the access and utilisation of the provided services.

The generated prototypical applications of MoPeDT support different mobile interaction styles: keyboard-based, microphone-based, camera-based, NFC-based and GPS-based. Thus, the corresponding Functional Requirement [FR-P01] is met for resulting prototypes of MoPeDT: the support of different mobile interaction styles.

Keyboard-based interaction style: As described in Section 2.2.2, the keyboard can be used to perform explicit interactions in order to interact with the graphical user interface that is presented on the mobile phone's display. The keyboard use of MoPeDT's prototypes bases on *Nokia's Design and User Experience*

*Library*¹. The right softkey is applied for the supposed negative commands (BACK, CANCEL and EXIT) while the left softkey is used for the option menu and its supposed positive commands, such as HELP. The middle key is utilised to select the most important positive command for the current screen, such as SELECT or SAVE.

Based on the keyboard-based interaction style, a physical object, service or service entry can explicitly be selected to determine its corresponding database ID and load the associated database content. In order to do so, the mobile user navigates through a list of the corresponding list items and selects the intended GUI element that represents a physical object, service or service entry. Apart from these interactions, the keyboard-based interaction style can also be applied to either input text or control the camera and microphone as well as a displayed content (e.g. PLAY, STOP and PAUSE a video file).

Microphone-based interaction style: MoPeDT's generated prototypes also support the microphone-based interaction style. For selections of a physical object, the user has to speak the word OBJECT and the corresponding identifier of the physical object (e.g. OBJECT 1). MoPeDT's generated prototypes automatically capture an audio file which is sent to the main server for speech recognition. The replied recognised identifier is then used to automatically load the database content.

Camera-based interaction style: The camera-based interaction style is similar to the microphone-based interaction style. By means of the interaction style, the user explicitly selects a physical object but instead of saying the identifier of a physical object, a picture needs to be taken of the intended physical object. Once the picture has been captured with or without a marker, the image data are transmitted to the server for recognising the intended object. After the recognition process has been finished, the determined identifier is replied to the mobile user to automatically load database content of the physical object.

NFC-based interaction style: A further supported mobile user interaction style is the NFC-based interaction style. For this interaction style, the mobile phone needs to support a built-in NFC reader. Then, mobile users can select physical objects by touching their attached RFID tags which contain details about the object identifier (e.g. OBJECT | 1). After an identifier has been determined, information about the physical objects are automatically loaded similar to the other mobile interaction styles.

GPS-based interaction style: The last supported mobile interaction style is GPS-based. By means of the GPS receiver, the user's current longitude and lat-

¹<http://library.forum.nokia.com>

itude can be determined. These data might correspond with a relevant location context (e.g. HOME or OFFICE) that can be specified (see Section 4.1.2). If such a relevant location context has been recognised, the data are transmitted to the server in order to determine if also a pervasive computing environment has been reached, such as whether the location HOME provides physical objects. To process this information, the database provides the necessary knowledge about the location data of the different pervasive computing environments. Now, once the server replies the context that a pervasive computing environment has been entered, MoPeDT's generated applications automatically pre-select the identified pervasive computing environment (e.g. a specific shopping store).

In summary, MoPeDT's generated applications cannot only support rather explicit mobile interaction styles but also an implicit interaction style which makes use of GPS data. The recognised relevant location contexts (e.g. HOME or OFFICE) can also be synchronously logged when conducting a user evaluation with the prototypical application. The logging of these data provides very important details about the user context while using the system.

The main server: The main server is the central component of the component-based client-server architecture. By means of their mobile phones and a running prototypical application, mobile users can connect to the server in order use different kinds of server services. Apart from the mobile users, also sensors, actuators and evaluators can connect to the main server as clients and utilise the provided services. In the following, the different server services for the different client components are described.

Running on a mobile phone, MoPeDT's generated prototypical applications apply the main server for the interpretation of captured data which is necessary for the microphone-based, camera-based and GPS-based mobile interaction styles. For this server service, the corresponding data material (e.g. audio files or images) is sent to the server and a plugged-in recogniser is used for the interpretation of the sent data. After the interpretation process has been finished, the recognised data (e.g. the identifier of a physical object) are returned to the mobile phone for the further processing.

Apart from this server service, the main server also provides a service to save the interest of mobile users, actuators and evaluators in contexts of other mobile users and sensors. By having this knowledge at the runtime of the system, incoming context data (e.g. interactions of the different mobile users or sensor data) are automatically forwarded to the interested clients, once an interest has been detected for these data. This service is very important if synchronous and asynchronous interactions of several users should be supported (see Requirement [FR-P04]), such

as for the application *the World Explorer* (see Section 3.1). This server service is further important to store qualitative and quantitative data during the evaluation phase (see Requirement [FR-P06]) since by this service a remote communication is provided between the evaluator component and the mobile users - the test users (see Requirement [FR-P07]). By this communication channel, all data from the mobile users can be forwarded to the evaluator component to synchronously log them.

Finally, a further task of the main server is the support of the following server services: (1) process and reply information requests about all pervasive computing environments which are stored in MoPeDT's database, (2) process and reply information requests about all stored physical objects of a selected pervasive computing environment, (3) process and reply information requests about all stored services of a selected physical object and (4) process and reply information requests about all stored service entries of a selected physical object's service, such as all stored comments - the service entries - about a selected product. By providing these server services, the Functional Requirement [FR-P02] is fulfilled since a remote access is provided to load and display content of a database. Also, the Functional Requirement [FR-P03] is met since the server also enables (5) the insertion of new service entries and their content into MoPeDT's database as well as (6) the update of existing service entries and their content.

The database: MoPeDT's used database persistently stores all information about the different pervasive computing environments and their physical objects as well as the available services, the physical objects' service entries and their contents. In this way, MoPeDT's database enables the modelling of mobile applications in the context of *the Third Computing Paradigm* as illustrated in Section 2.2 where mobile phones are used as a medium to interact with a pervasive computing environment and its physical objects. The database model contains the following tables: *environment*, *object*, *service*, *entry* and *content*. The table *environment* contains all information about the available pervasive computing environments including details about their locations whereas the table *object* provides information about the different physical objects together with a reference to the responsible pervasive computing environment. The table *service* provides information about existing services and the table *content* contains all multimedia contents of the service entries (e.g. text or video files). The last table called *entry* provides references between services and contents to physical objects - the service entries.

The sensors: Sensors (e.g. temperature sensors) are an optional component of the component-based plug&play architecture of MoPeDT. They can be considered as a further interaction device (see Requirement [FR-P05]) since they provide a

new input channel to the system. Sensors can connect to the main server in order to provide context data that might be of interest to other components (e.g. mobile users or evaluators), such as details about the current temperature, loudness or lighting conditions in a pervasive computing environment. Thus, a particular incoming context (e.g. LIGHT.BRIGHT or LIGHT.DARK) might cause an adaptation of the mobile phone's appearance, such as the display of another screen content. At the evaluator component, incoming sensor data enable the synchronised logging of environmental context while conducting a user evaluation. Knowledge about the environmental context can help interpret a user behaviour in the analysis phase, such as why the users suddenly behave unusually.

MoPeDT's sensor component bases on an equally named software module - `SENSOR` - that handles the client-server communication between the sensor and the server. For instance, XML messages are automatically transmitted to the server whenever a new context value emerges. For the specification of the software module, XML files need to be used. These XML files contain information about the main server (e.g. the server address and port) and information about the sensor's context values that might be supplied. Also, the JAVA-based `SENSOR` module provides programming interfaces to add the sensor hardware and other required software components, such as for the context interpretation.

The actuators: The actuator is also an optional plug&play component of the architecture. The actuators can be used as an additional output channel for multimedia content - a further presentation device (see Requirement [FR-P05]). As illustrated by *the World Explorer* (see Section 3.1), MoPeDT's architecture and modules can be applied to generate an application that does not only contain multiple mobile users and their phones but also a public display that presents video content. The actuator component only needs to connect to the main server component in order to register the interest in contexts of mobile users and sensors. Once there is an incoming message from one of the interested components via the main server, the context is mapped to a multimedia content which is then automatically displayed and played.

For the actuator component there is also a JAVA-based software module - `ACTUATOR` - available that enables the client-server communication to the main server. This module can be configured by making use of XML files. For instance, one XML file is used for the mapping between incoming context values and the loading and displaying of multimedia content. Different players can be added by means of programming interfaces.

The evaluators: A special kind of actuator is the evaluator. This component is required whenever tool-supported user evaluations are executed. Several evalu-

ators can connect to the server and register their interest in other connected components: mobile users and sensors. Now, the evaluators can synchronously log all data of the selected mobile user (e.g. mobile interactions and user context) and sensors for a later on analysis. The utilisation of the evaluator is the prerequisite to execute remote evaluations since it enables the remote communication between the mobile user and the evaluator (see Requirement [FR-P07]) and thus the tracing of the mobile user independent on his location.

The evaluator component was implemented as a JAVA and a C# module. The corresponding module handles the client-server communication to the main server and the registration of interest in other components. Both modules also can be defined based on an XML file. Programming interfaces exist which need to be implemented in order to process the received messages, such as context messages from mobile users and sensors. The C#-based software module is part of the user-centred prototyping tool MoPeDT and its evaluation component (see Section 4.2).

4.1.2 The Software Modules

For most components of the architecture exist software modules. The software modules `SENSOR`, `ACTUATOR` and `EVALUATOR` were described in the previous Section. In the following, the central modules `MOBILE CLIENT` and `MAIN SERVER` are considered more detailed.

The `MOBILE CLIENT` software module: The J2ME-based software module of the `MOBILE CLIENT` (e.g. [Leichtenstern and André, 2009b, Leichtenstern and André, 2010]) handles the whole client-server communication between the mobile user and the main server as well as contains the implementation of different mobile interaction styles. Additionally, the use of screen templates and a scripting language are supported to enable the dynamic specification of the prototype's appearance and behaviour. In the following, these aspects of the `MOBILE CLIENT` are described with a primary focus on how the different functional requirements were fulfilled towards a tool-generated prototype.

The client-server communication: A central task of the module is to handle the whole client-server communication. In order to enable this communication at the runtime of the system, an essential task is to edit the JAD file of the `MOBILE CLIENT`. This file provides information about the main server - the server address and port - as well as information about the application's name, background and foreground colours. The `MOBILE CLIENT`'s client-server communication is mainly required whenever the mobile user requests information about a pervasive computing environment and its physical objects. That happens typically whenever a

user has selected a physical object, service or service entry by means of one of the provided mobile interaction styles (see Section 4.1). In this case, the software module automatically checks the need for a server request and, if necessary, sends an XML message to the server. Program 1 illustrates the request for a physical object.

Program 1 As an example of a server request, the XML message requests information about a physical object.

```
<get>
  <user_id>1</user_id>
  <user_name>user1</user_name>
  <request_id>1</request_id>
  <request_type>object</request_type>
</get>
```

The server replies the request about a physical object with information about its name and identifier as well as the available services of the intended physical object (see Program 2). The replied data, for instance, can then be used to display a list of all available services of the selected physical object. To enable this display, a corresponding dynamically specified screen is required. Similar to the illustrated example, XML messages are also sent if captured data should be transmitted and stored in the database, such as a new service entry. In summary, with the MOBILE CLIENT's supported client-server communication, the requirements [FR-P02] and [FR-P03] are fulfilled since remote access is provided to load or modify data of the database.

The client-server communication is also required for some mobile interaction styles. The microphone-based, camera-based and GPS-based mobile interaction styles require the interpretation of recorded data (e.g. of audio files) on the server. Once the user has performed a mobile interaction by means of one of these three interaction styles, the MOBILE CLIENT automatically transmits the recorded data and waits for a server reply in order to proceed, such as with a database request.

Different further XML messages are also automatically sent to the server which happens rather hidden to the users. These messages provide information about explicit and implicit mobile interactions, such as that the user has touched an RFID tag with the NFC reader of the mobile phone. Other XML messages are transmitted to the server whenever new screens are loaded and displayed on the phone. This information is important during the evaluation phase since it provides the opportunity to display a cloned screen view of the test user's phone on the evaluator's desktop computer (see Section 4.2). Program 3 illustrates a typical XML message that is sent when the user has performed a keyboard-based mobile inter-

Program 2 As an example of a server reply, the XML message replies information about a physical object and its supported services.

```

<object>
  <user_id>1</user_id>
  <object_id>1</object_id>
  <object_name>Titanic</object_name>
  <services>
    <service>
      <service_id>2</service_id>
      <service_name>Critics</service_name>
      <service_icon>icon</service_icon>
    </service>
    <service>
      <service_id>3</service_id>
      <service_name>Plot</service_name>
      <service_icon>icon</service_icon>
    </service>
  </services>
</object>

```

action. The same syntax is also used whenever the user performs another explicit or implicit mobile interaction or whenever the screen is updated. The data exchange of context messages is important to fulfil the requirement [FR-P06]: the support to store qualitative and quantitative data during the evaluation phase.

Program 3 As an example of a context message, the XML message provides details about the occurrence of a keyboard-based mobile interaction.

```

<context>
  <context_owner_id>1</context_owner_id>
  <owner_name>user1</owner_name>
  <owner_type>user</owner_type>
  <context_type>KEYBOARD</context_type>
  <context_interpretation>SELECT</context_interpretation>
</context>

```

The mobile interaction styles: The MOBILE CLIENT module supports different mobile interaction styles (see Section 4.1.1). Once the user has performed a mobile interaction by means of one of the supported mobile interaction styles, the MOBILE CLIENT recognises the context value (e.g. the ID of an intended physical

object or a relevant location context) and automatically uses this value for further processes, such as for a server request.

To configure the different mobile interaction styles, an XML file needs to be edited. This file provides several details about the mobile interaction styles. First of all, it provides knowledge about which mobile interaction styles should be activated by the MOBILE CLIENT. At the runtime of the system, the activation enables the transmission of context messages to the main server whenever context values emerge. As an example, Program 4 specifies that a context message should be sent to the server whenever the mobile user applies the keyboard to activate the command SELECT. Another kind of context message is also sent whenever the user has entered the location that is specified for HOME. The context messages are of special interest to connected evaluator components as well as other interested mobile users. The opportunity of a specification of the mobile interaction styles is therefore important for the Requirements [FR-P06] and [FR-P04]. It enables the automatic storage of qualitative and quantitative data during the evaluation phase (see Requirement [FR-P06]) as well as the use of mobile interactions for several mobile users (see Requirement [FR-P04]). The specified mobile interaction styles cannot only be used to generate context messages for evaluators and other mobile users, they also can be used for the specification of the prototype's application flow and logic as described in the following paragraph.

The dynamic specification of the appearance and behaviour: Besides the mobile interaction styles, the appearance and behaviour of a prototype can also be specified with XML files. For the specification of the prototype's appearance, the interface designer makes use of XML-based screen templates. These screen templates ground on 22 approved interface guidelines (see Appendix A) which were derived from *the Nokia Design and User Experience Library*². For instance, these templates consider a consistent layout, softkey use and navigation style. Each screen of the generated prototype has a heading, content and a softkey part. The left softkey is used for options, the middle key is used for confirmations and navigations and the right softkey is used for negative actions (BACK, CANCEL or EXIT). Additionally, each screen contains a help option and an option to return to the main menu. Moreover, from each screen the user can return to the previous screen automatically by selecting BACK or CANCEL. An assumption is that by using such screen templates, the usability of the resulting prototypes (see Requirements [NFR-P01] and [FR-T01.3]) can be increased. The validation of this assumption is presented in Chapter 5.

The expandable set of the MOBILE CLIENT's screen templates currently supports

²<http://library.forum.nokia.com/>

Program 4 As an example of a specification file for mobile interaction styles, the XML files defines the keyboard-based and GPS-based mobile interaction styles.

```

<context_specification>
  <context>
    <context_type>KEYBOARD</context_type>
    <values>
      <value>
        <value_interpretation>SELECT</value_interpretation>
      </value>
    </values>
  </context>
  <context>
    <context_type>OUTDOOR_LOCATION</context_type>
    <values>
      <value>
        <value_interpretation>HOME</value_interpretation>
        <value_context>
          <item type="latitude">48.352</item>
          <item type="longitude">10.886</item>
        </value_context>
      </value>
    </values>
  </context>
</context_specification>

```

ItemScreens (see Figure 4.2), MediaScreens (see Figure 4.3), AlertScreens (see Figure 4.4), screens with a tabulator functionality - TabScreens (see Figure 4.5) as well as SketchScreens which enable the display of place holders, such as a scanned image of the intended screen. SketchScreens are typically used at the very beginning of the development process if the layout of the prototype should be quick and dirty (see Section 2.1). Based on the MOBILE CLIENT's supported types of screen templates, a prototype's appearance can be generated which provides a high or low level of fidelity (see Requirement [FR-T02]).

For each type of screen, XML templates exist that need to be edited in order to specify the static or dynamic content of a screen. Program 5 shows the XML specification of static content for two types of screens. The first screen bases on the template of a ItemScreen whereas the second screen grounds on a MediaScreen. For both types of screens, a XML tag called widget can be specified in order

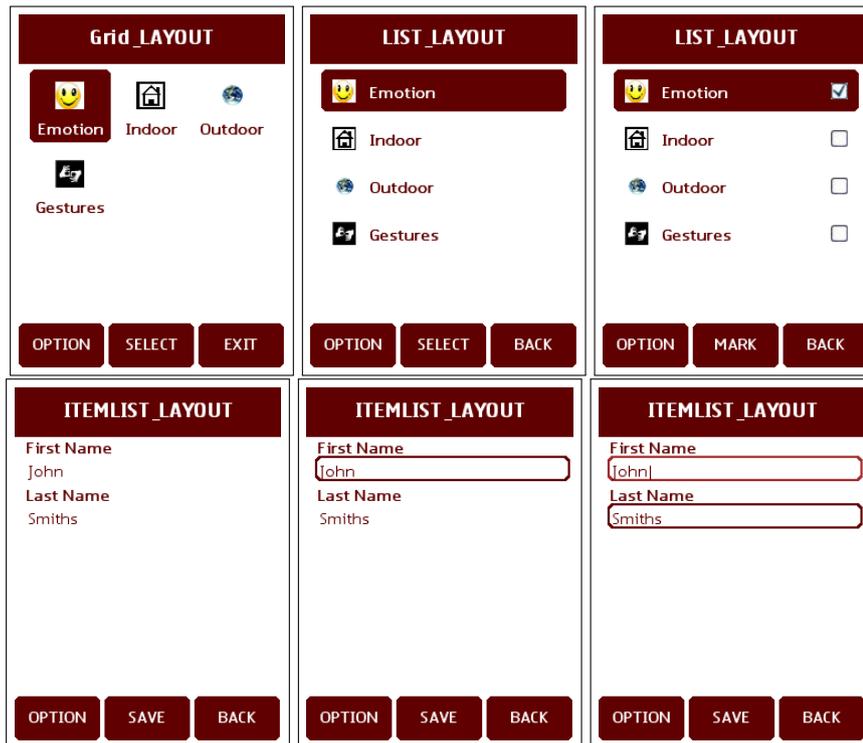


Figure 4.2: Examples of ItemScreens which are supported by the module MOBILE CLIENT. These screens can be used to generate different kinds of menu screens.

to define the content of the screen heading and help option. Further, for the ItemScreen, a layout (e.g. grid or list) and a list of items can be defined whereas the MediaScreen enables the specification of multimedia content (e.g. text, image, audio or video) also by means of the XML tag widget. Based on the XML specification of Program 5, Figure 4.6 shows the two screens which were generated with the MOBILE CLIENT.

Another XML file is used to specify the prototype's static or dynamic behaviour - the application flow and logic of the application. The idea is that the application flow of the intended prototype can also be represented by a state chart similar as done by related tools (see Section 3.3). Following this idea, the different screens of the intended prototype are screen states in the state chart. Whenever the application is in one screen state, the performance of mobile interactions (e.g. with the keyboard) can trigger transitions - the loading and display of another screen state. For instance, Program 6 specifies that the screen state with the identifier 2 should be loaded and displayed once the mobile user has applied the keyboard-based mobile interaction style and selected the value SELECT: KEYBOARD.SELECT. Figure 4.7 shows a visual representation of Program 6. For the XML-based specification of the application flow and logic, STATE CHART XML (SCXML)³ was used.

³<http://www.w3.org/TR/2005/WD-scxml-20050705/>

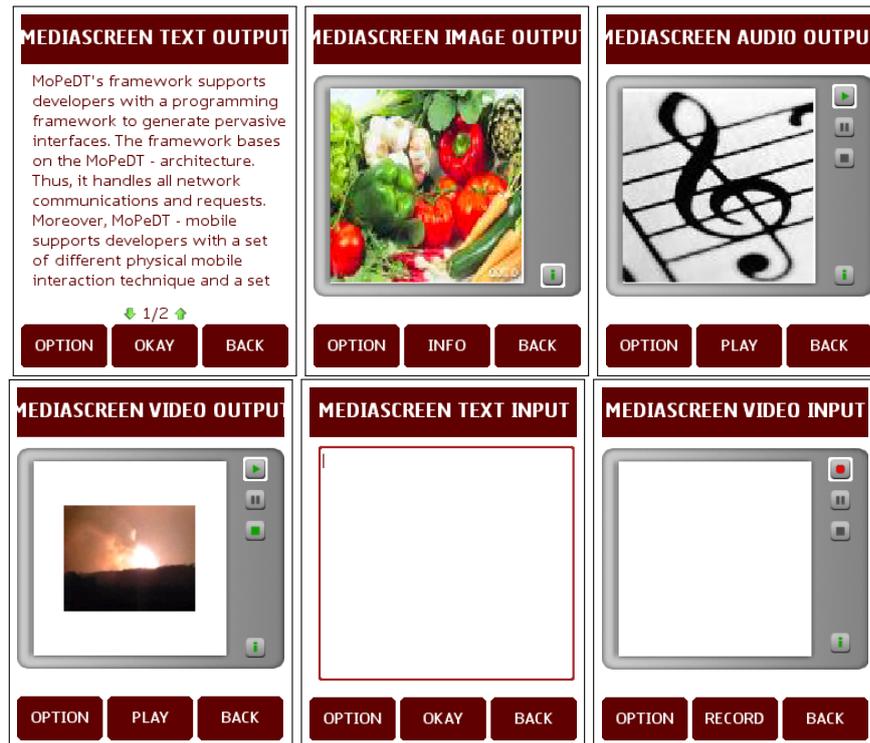


Figure 4.3: Examples of MediaScreens which are supported by the module MOBILE CLIENT. These screens can be used to generate different input and output screens for multimedia content.

Using the concept of state charts for the specification of a prototype's application flow, all supported mobile interaction styles (e.g. GPS-based or NFC-based) but also contexts from other components - other mobile users and sensors - can be used as transitions, such as the context TEMPERATURE.HOT as incoming context of the temperature sensor. In order to receive contexts of other sensors and mobile users, a further XML file needs to be specified (see Program 7). At the runtime, this XML file is sent to the main server to register the interest in context messages, such as in context messages of the type NFC or TEMPERATURE. This XML file is important to meet the Requirement [FR-P04]: the use of mobile interaction for several mobile users.

Contents of the screens are often unknown at the specification time of a prototype as well as the concrete behaviour of a prototype. For instance, if all physical objects of a just selected pervasive computing environment should be loaded and displayed at the runtime, it is impossible to know details about the physical objects (e.g. the number of the physical objects and their names) at the design time. Consequently, the specification of the appearance and behaviour is often not just statically but also dynamically (see Requirements [FR-P02] and [FR-P03]). For dynamic specifications by means of the MOBILE CLIENT module, the same XML files



Figure 4.4: Examples of AlertScreens which are supported by the module MOBILE CLIENT. These screens can be used to generate different alert screens: info, waiting or error screens.

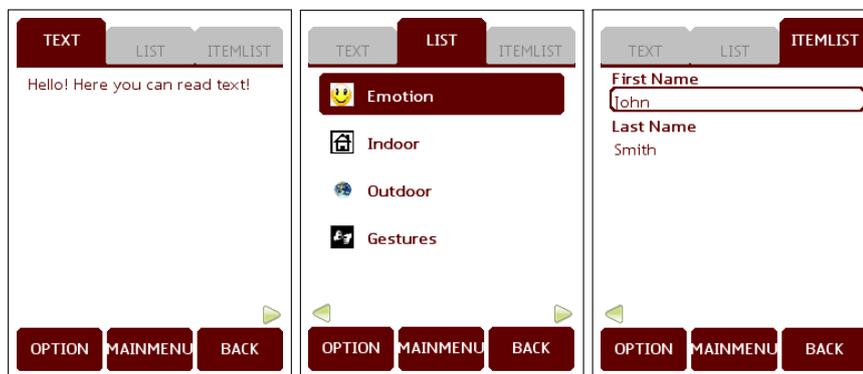


Figure 4.5: Examples of TabScreens which are supported by the module MOBILE CLIENT. This screen block contains several other types of screens: ItemScreens, MediaScreens, AlertScreens or SketchScreens.

can be used as for the static specification of the appearance and behaviour but this time prompts of a scripting language need to be used. At the runtime of the application, these prompts are interpreted by the MOBILE CLIENT in order to dynamically generate and display screens.

Program 8 shows the specification of a screen with dynamic content. An ItemScreen should be presented that should display a list of all physical objects of a selected pervasive computing environment. For each physical object, the corresponding icon and name should be shown. The heading of the screen is also dynamically. The name of the just selected pervasive computing environment should be displayed. Figure 4.8 shows the result of the dynamically specified screen at the runtime. Based on the data of the database, this screen displays three physical objects - *Titanic*, *Se7en* and *Pirates of the Caribbean* - with their names and icons. As heading, the name of the selected pervasive computing environment is shown - DVDs. By using scripting prompts, all supported screen templates -

Program 5 As an example of a static content specification of a prototype's appearance. Screen 1 defines a `ItemScreen` with static content whereas screen 2 specifies a `MediaScreen` with static content.

```
<presentation>
  <screenblock id="1" type="standard" main="true">
    <screen type="ItemScreen">
      <widget element="heading" value = "Main Menu"/>
      <widget element="help" value = "Please select an option!"/>
      <group type="grid">
        <item>
          <widget element="icon" value="/image.jpg" />
          <widget element="text" value="Image" />
        </item>
      </group>
    </screen>
  </screenblock>
  <screenblock id="2" type="standard">
    <screen type="MediaScreen">
      <widget element="heading" value="Text Screen"/>
      <widget element="help" value = "This screen shows..."/>
      <widget element="text" value="This is a text screen..."/>
    </screen>
  </screenblock>
</presentation>
```

`ItemScreens`, `MediaScreens`, `AlertScreens`, `TabScreens` or `SketchScreens` - can dynamically be defined. For instance, a `MediaScreen` can dynamically be specified to display different multimedia contents (e.g. text and video) of a service entry which are stored in the database.

The scripting language, however, cannot only be used to specify a dynamic appearance but also a dynamic behaviour. Program 9 shows that script prompts can also be used to define conditions for a transition and thus the application flow. In this example, the first transition can only be triggered once the value `SAVE` has been activated with the keyboard-based mobile interaction style as well as if a variable - camera - has a certain value - 1. If the transition and its condition are fulfilled, the screen state 1 is left and the screen state 2 is loaded and displayed. In this case, it is checked whether the screen state 1 has an XML tag called `onexit` and whether the screen state 2 has an XML tag called `onentry`. Both XML tags can include script prompts that should be executed if the state is loaded - `onentry` - or left - `onexit`.

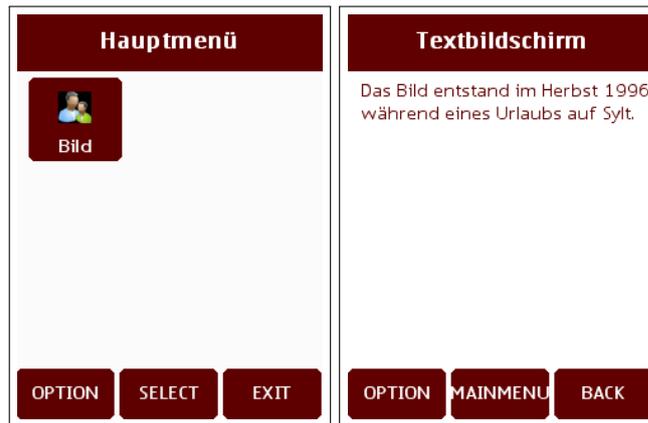


Figure 4.6: The screens which are generated based on the static content specification of the prototype's appearance (see Program 5): ItemScreen (left) and MediaScreen (right).

Program 6 As an example of the specification of a prototype's application flow: screen 2 is loaded once the user has selected `SELECT` by means of the keyboard.

```
<scxml xmlns="http://www.w3.org/2005/07/scxml"
  version="1.0"
  initialstate="Main">
  <state id="1">
    <transition event="KEYBOARD.SELECT" cond="" target="2"/>
  </state>
</scxml>
```

By using these XML tags and defining script prompts, the logic of the application can be defined much more dynamically. In the example of Program 9, the screen state 1 provides an XML tag `onexit` that includes scripting code. At the runtime of the system, this scripting code is interpreted and executed by the module `MOBILE CLIENT`. The objective of the code example is to save a new service entry and the just captured multimedia content in the database. The code first checks whether the screen state 1 has been left due to the fact that `SAVE` has been selected by means of the keyboard. Then the captured new contents (e.g. a video file and text) are loaded and finally a new service entry is created and its multimedia contents are stored in the database.

By using the scripting language that is supported by the `MOBILE CLIENT`, variables and constants can be used as well as different programming constructs (e.g. if-else statements or for-loops). Additionally, content of the database tables `environment`, `object`, `service`, `entry` and `content` can be loaded and manipulated.

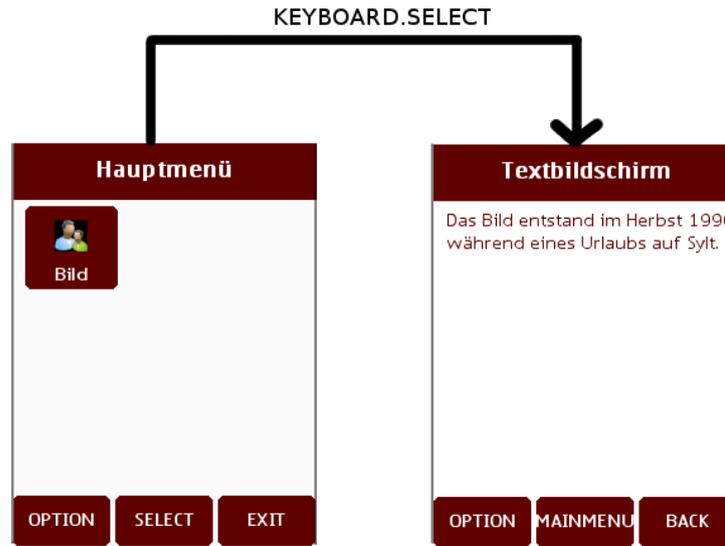


Figure 4.7: Visual representation of Program 6: Screen 2 is loaded once the user has selected `SELECT` by means of the keyboard.

Program 7 As an example of the interest specification of contexts. The mobile user is interested in the contexts `TEMPERATURE` and `NFC` from sensors and other mobile users.

```
<interest>
  <types_of_interest>
    <type>TEMPERATURE</type>
    <type>NFC</type>
  </types_of_interest>
</interest>
```

All in all, since the generated prototypes base on the `MOBILE CLIENT` module and its supported functionality, the resulting prototypes are supposed to always be highly robust (see Requirement [NFR-P02]) and functional (see Requirement [NFR-P03]). These two assumptions are also validated in Chapter 5 besides the usability of the generated prototypes (see Requirements [NFR-P01]).

The MAIN SERVER software module: The `JAVA`-based module called `MAIN SERVER` handles the whole client-server communication for the main server and provides programming interfaces for plug-in recognisers.

The client-server communication bases on `TCP/IP` sockets. Different clients - mobile users, actuators, sensors and evaluators - can connect to the server in order to use different services or provide context messages which need to be forwarded by

Program 8 As an example of the dynamic content specification of a screen. Screen 1 defines a dynamic ItemScreen.

```
<screen type="ItemScreen" script="true" value="environment.getObjects()">
  <widget element="heading" scripted="true" value="environment.getName()"/>
  <widget element="help" value="Please select an option!"/>
  <group type="grid">
    <item>
      <widget element="icon" value="object.getIcon()" scripted="true"/>
      <widget element="text" value="object.getName()" scripted="true"/>
    </item>
  </group>
</screen>
```



Figure 4.8: The dynamically generated screen of Program 8. A ItemScreen is generated that displays different physical objects of the just selected pervasive computing environment.

the main server (see Section 4.1.1). In order to reply server requests which aim at a pervasive computing environment, the MAIN SERVER module is also connected to MoPeDT's database. For the configuration of the MAIN SERVER, an XML file is required that contains information about the server port as well as information about the database (e.g. the database's address, name and login data). The MAIN SERVER module also can be extended to add recognisers for the microphone-based and camera-based mobile interaction style. Two different classes exist which provide a method that needs to be implemented. This method provides the raw data of the corresponding mobile interaction style (e.g. an audio file). As the result of the recognition process, the method replies a string value (e.g. the recognised ID of a physical object). For the GPS-based mobile interaction style, the incoming location context of the mobile user is automatically compared with the location of the pervasive computing environments which are stored in the database.

Program 9 As an example of the dynamic specification of a prototype's behaviour. A new service entry and its content should be persistently stored in the database once the user has selected `SAVE` by means of the keyboard-based mobile interaction style.

```

<state id="1">
  <transition event="KEYBOARD.SAVE" cond="camera=1" target="2"/>
  <transition event="NFC.OBJECT" cond="" target="3"/>
  <onexit>
    <script>
      if(world.getContextValue("KEYBOARD")== 'SAVE'){
        var newContent = world.getContentValues();
        var ent = service.createEntry(service.getName()+counter);
        counter++;
        ent.setContent(newContent);
      }
    </script>
  </onexit>
</state>

```

4.2 The User-Centred Prototyping Tool

Grounded on the component-based plug&play architecture and the software modules, the user-centred prototyping (UCP) tool called MoPeDT - Pervasive Interface Development Toolkit for Mobile Phones - was implemented in C# within this PhD. This tool supports interface developers of mobile applications in the context of *the Third Computing Paradigm* with a graphical user interface (see Requirement [NFR-T01.5] and [NFR-T01.6]) in the three phases of the user-centred prototyping process (see Requirement [FR-T06]): (1) the design specification and automatic generation of prototypes, (2) the conduction of evaluations and (3) the analysis of the captured data [Leichtenstern et al., 2008, Leichtenstern and André, 2010, Leichtenstern and André, 2011, Leichtenstern et al., 2011a]. In the following, the tool components are addressed for these three phases. The focus is primarily on how the functional requirements towards a UCP tool are fulfilled by MoPeDT.

4.2.1 The Design of Prototypes

The design component of MoPeDT graphically supports during the design specification of the mobile interaction styles as well as the specification of the static and

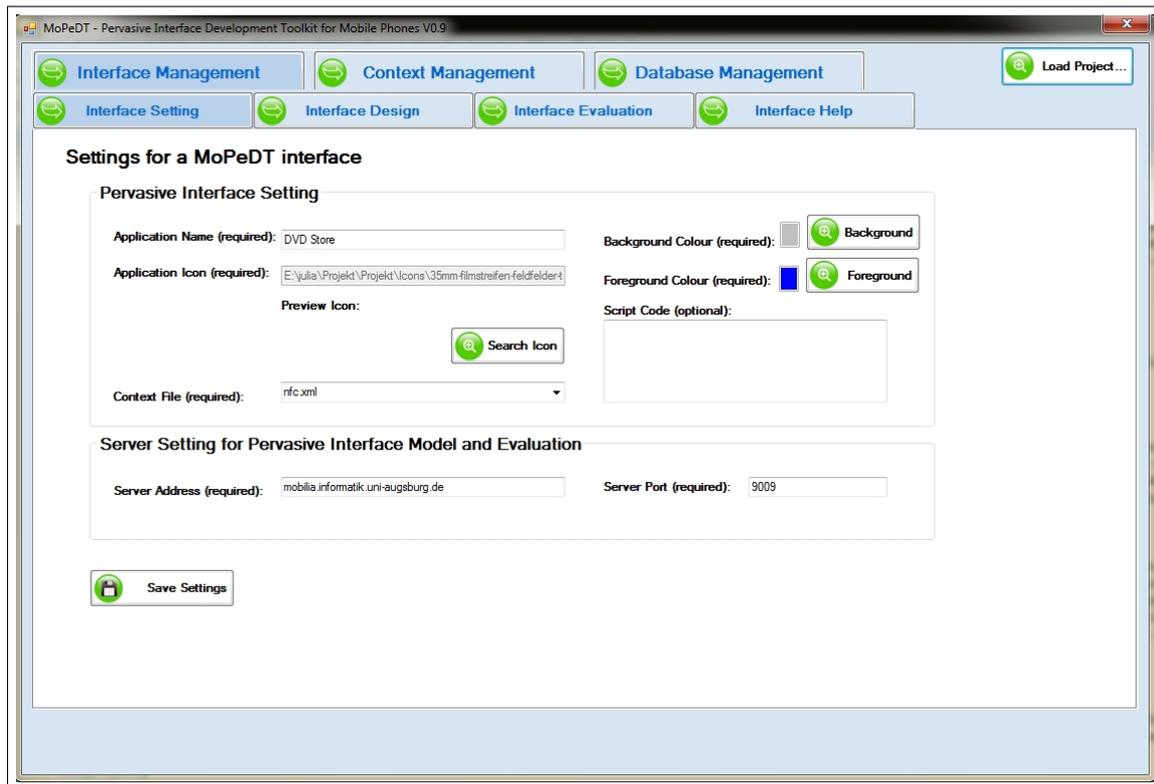


Figure 4.9: The graphical user interface of MoPeDT for the specification of the prototype's general settings. The upper part of the component displays the settings for the prototype (e.g. application name and icon) while the lower part aims at the specification of the server - server address and port.

dynamic appearance and behaviour of the intended prototype (see Requirement [FR-T01]). The result of a design specification with MoPeDT's design component is an executable JAR file that bases on the software module `MOBILE CLIENT` (see Section 4.1.2).

Before starting to specify details about the intended prototype, its general settings need to be defined. Figure 4.9 shows the GUI for the specification of these general settings. Using that GUI, for instance, the application name, icon and colours can be specified as well as details about the main server. Based on this input, the JAD file is generated for the `MOBILE CLIENT` (see Section 4.1.2).

The graphical specification of the mobile interaction styles: MoPeDT features the GUI-based specification of different mobile interaction styles (see Requirement [FR-T01.3]). In Section 4.1.2, an XML file was introduced that needs to be modified for the module `MOBILE CLIENT` in order to define the mobile interaction styles - the context types - and their context values. Figure 4.10 shows the GUI to specify the different context types and their values for this XML-based context

specification file.

Interface designers can add, update and remove several mobile interaction styles and provide context values for them. For instance, `SELECT`, `BACK`, `CANCEL` or other keyboard commands can be defined as values of the keyboard-based mobile interaction style. As described in Section 4.1.2, the specification of mobile interaction styles and their context values leads to their activation and the transmission of context messages to the server once one of the specified context values has emerged at the runtime. The specified types of contexts - the mobile interaction styles - and their context values also enable transitions when specifying the prototype's application flow. For instance, once the interface developer has graphically added the NFC-based mobile interaction style and its context value `BUY`, a transition `NFC.BUY` can be used when specifying the prototype's application flow.

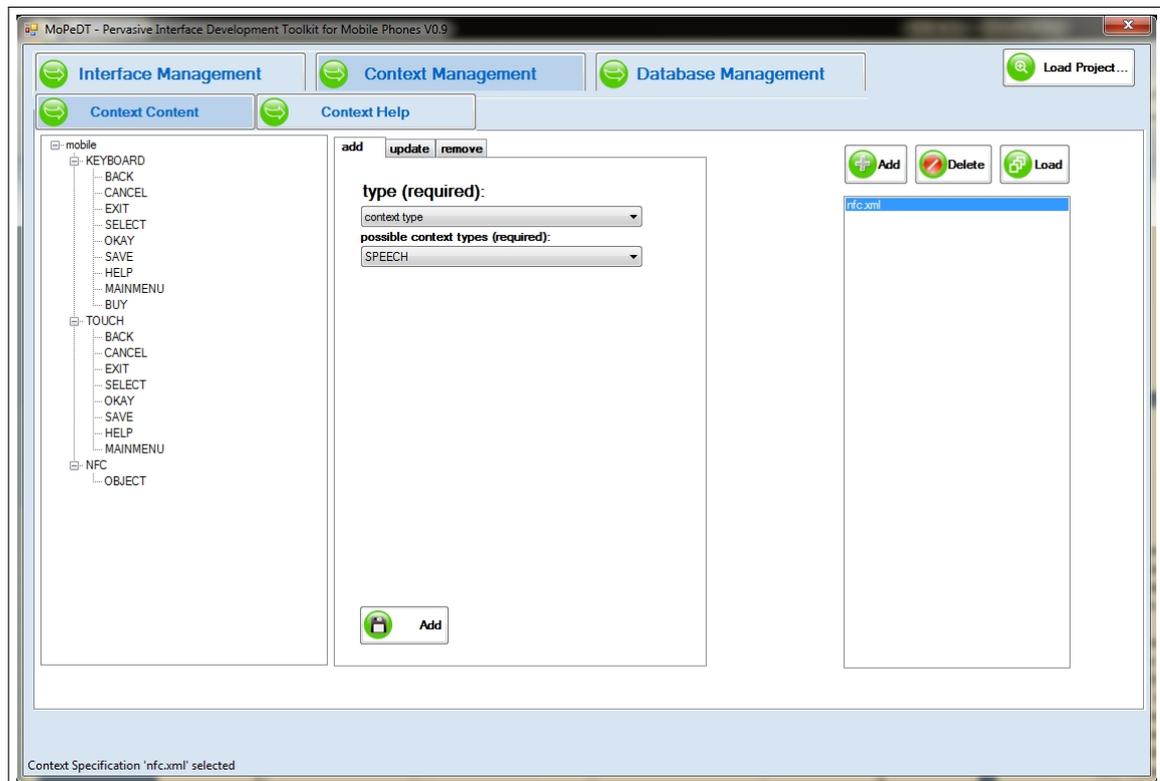


Figure 4.10: The graphical user interface of MoPeDT for the specification of the mobile interaction styles. The left part of the component displays the tree view of all specified mobile interaction styles and their defined context values while the middle part of the component provides options to modify the mobile interaction styles and their values - add, update and remove. The right part provides options to define context values for other components of the architecture - other mobile users and sensors - which are of interest to the intended prototype.

If the prototype should also make use of transitions which are based on context values of sensors or other mobile users, further XML-based context specification files need to be generated. Each of these files provides information about available context types and values of the corresponding sensor or mobile user. For instance, if the intended prototype should make use of the transition `TEMPERATURE.HOT`, an XML file needs to be generated for the temperature sensor with the interaction style `TEMPERATURE` and the context value `Hot`.

Later on, when specifying the behaviour of the intended prototype, transitions can be used which either base on the own context specification file or on strange context files. At the runtime of the prototype, the server will forward the required context messages to the mobile user which, in turns, will handle the incoming context message if a transition (e.g. `TEMPERATURE.HOT`) was specified for the current screen state. Based on the design specification of the different context files and the application of context values as transitions, the XML file of Program 7 is automatically generated by MoPeDT. This file provides the required information to the main server about a mobile user's interest in context messages from sensors and other mobile users (see Section 4.1.2).

The graphical specification of the prototype's static and dynamic appearance and behaviour: MoPeDT applies the approach to support the static and dynamic specification of a prototype's appearance and behaviour (see Requirement [NFR-T01.1]) with a state-chart view (see Figure 4.11). Since several views can potentially provide benefits for the interface developer (see Requirement NFR-T01), MoPeDT does not only support a state-chart view but also a tree view (see Figure 4.12). MoPeDT's tree view supports a better overview when working on a specific screen state whereas the state-chart view provides a better overview when defining the application flow. The state-chart view shows a preview of all screen states and easily enables the linking of these states. The tree view represents very detailed the XML tag structure of the appearance and behaviour's specification files (see Section 4.1.2). By this means, tree elements can be selected which, for instance, represent the XML tags `widget`, `item`, `onentry` or `onexit` in order to add, update or remove aspects of a single screen's appearance or behaviour.

When specifying a prototype of a mobile application with MoPeDT, the interface designer can first add, update and remove several screen states which base on the screen templates of the `MOBILE CLIENT` (see Section 4.1.2). After the specification of the screen states, the designer can link several screen states by adding, updating and removing transitions. For instance, a transition can be added to a screen state that, at the runtime, another screen state should be loaded and displayed once a user has selected a physical object by means of the mobile phone's NFC reader.

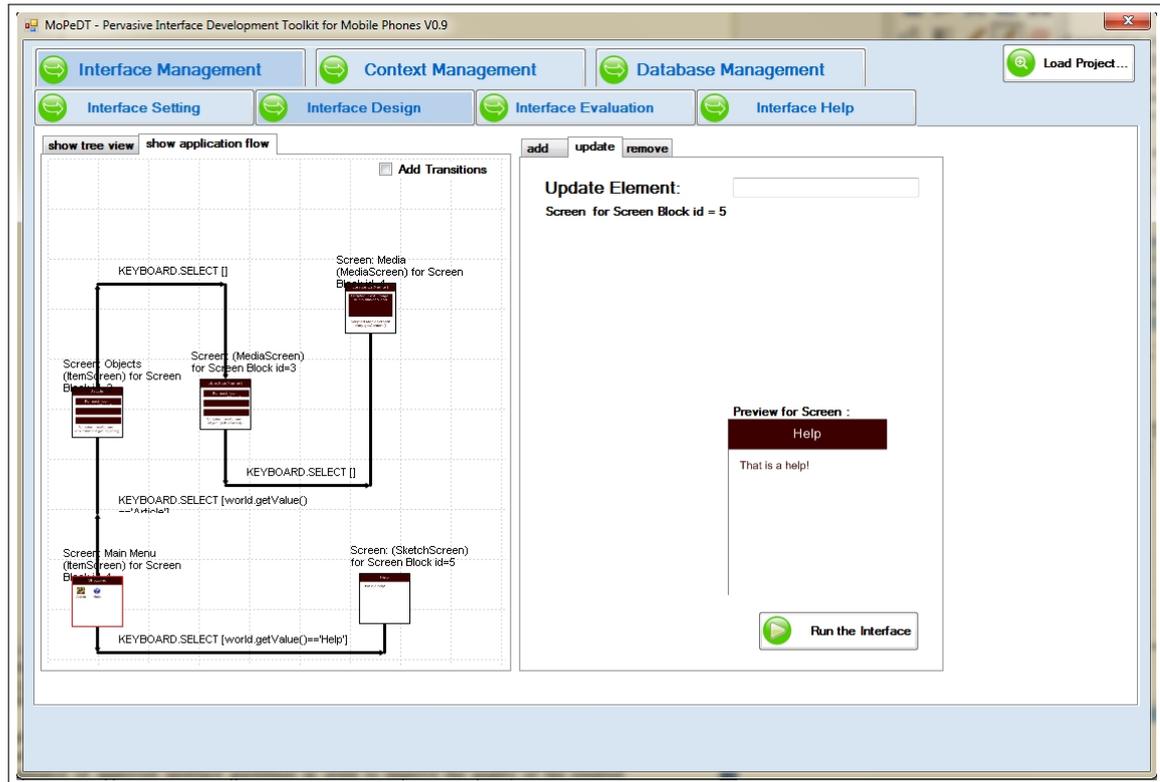


Figure 4.11: The graphical user interface of MoPeDT for the specification of a static and dynamic appearance and behaviour by means of a state-chart view. The left part of the component displays the state-chart view of the prototype which contains the display of screen states and transitions while the right part of the design component provides a preview of the selected screen and options to modify the appearance and behaviour.

The GUI of MoPeDT provides a list of all available transitions which can be used. This list bases on the own and strange context specification files which were generated during the specification of the mobile interaction styles and their context values.

A special feature of MoPeDT is the support to enable both the specification of a prototype's static and dynamic appearance and behaviour. Static content (e.g. images, videos or text) and behaviour can directly be defined at the specification time. For the specification of dynamic content and behaviour, the scripting language can be used as described in Section 4.1.2. The designer only needs to select whether a screen element (e.g. widget or screen) has static or dynamic content and add the corresponding static contents or alternatively script prompts. Also, scripting code can be defined for the XML tags `onexit` and `onentry` in order to specify the application logic. The GUI of the design component provides assis-

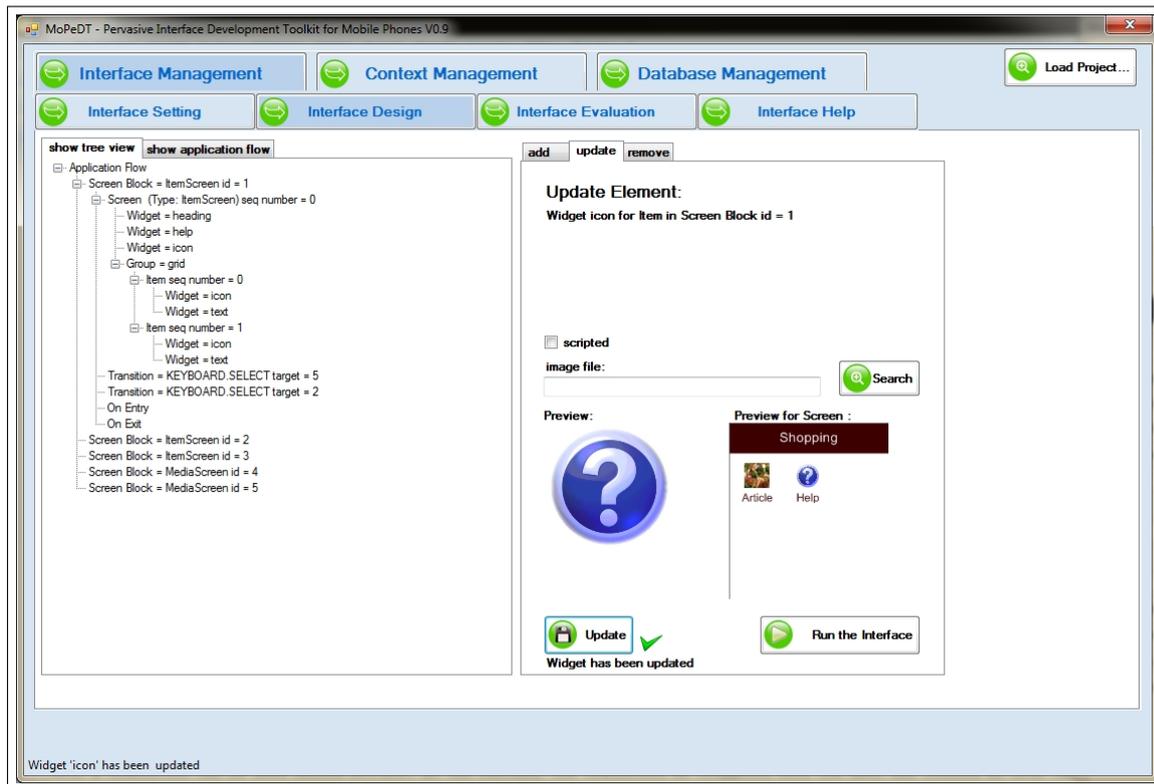


Figure 4.12: The graphical user interface of MoPeDT for the specification of a static and dynamic appearance and behaviour by means of a tree view. The left part of the component displays the tree view of the prototype which contains the screens, their widgets and transitions while the right part of the design component provides a preview of the selected screen and options to modify the appearance and behaviour.

tance when specifying static and dynamic content, such as it provides a preview of the specified screens (see Figure 4.11 and Figure 4.12 on the right part).

As described in Section 4.1.1, the scripting language also can be used to define the application's behaviour with regards to the remote content that is persistently stored in the database. For the specification of this database content, a further GUI component of MoPeDT can be used (see Figure 4.13). Environments, objects, services, entries and contents can be added, updated and removed. Thus, MoPeDT also enables the definition of remote data (see Requirement [FR-T01.4]).

The auto-generation of running prototypes as result of the design specification: After having specified the intended application, a prototype is automatically generated which directly run on the intended device (see Requirement [FR-T02]). Such highly functional prototypes can be produced at several stages of the development process. At early stages of the process, SketchScreens can be used

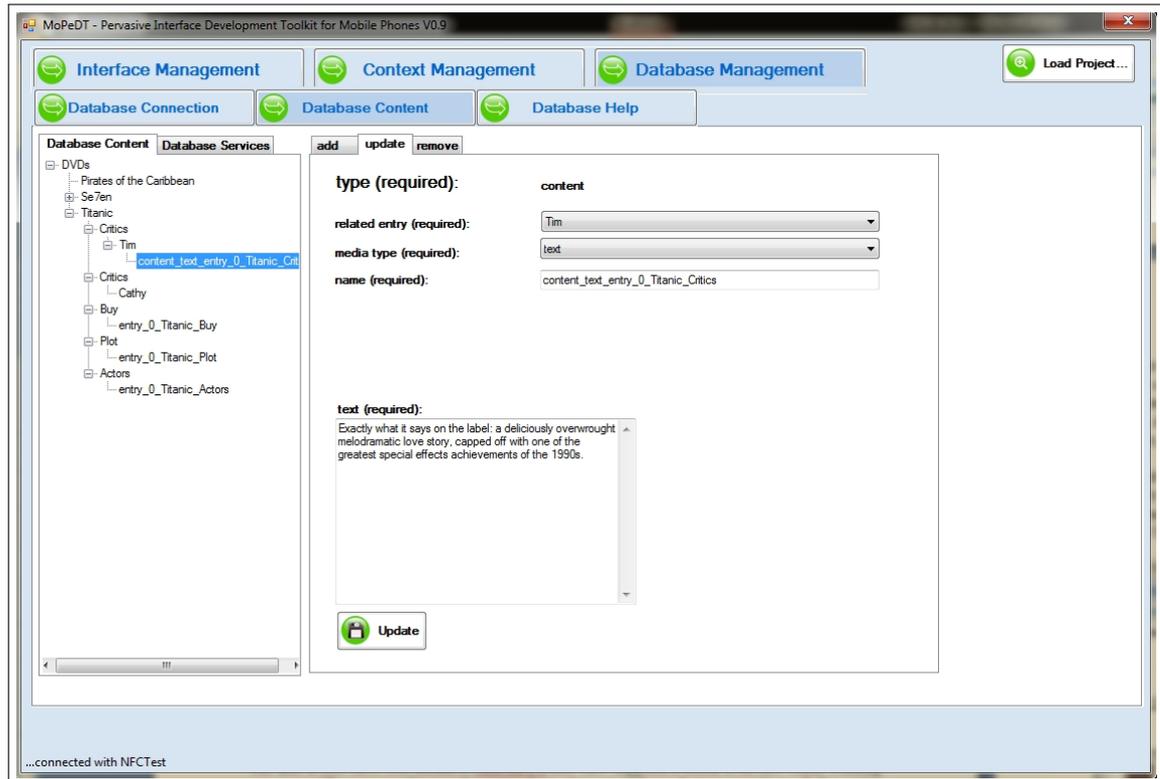


Figure 4.13: The graphical user interface of MoPeDT for the specification of the database. The left part of the component displays the tree view of the database elements - the environments, objects, services and service entries - while the right part of the component provides options to modify - add, update and remove - the content of the corresponding database element.

to generate prototypes with a low level of fidelity whereas at later phases prototypes can be produced with detailed content and a high level of fidelity. Further, after each iteration an existing prototype can be reused and modified, such as its screens and transitions can be added, updated and removed. Thus, the resulting prototypes are evolutionary prototypes.

Since the generated prototypes base on the MOBILE CLIENT'S screen templates, the prototypes are automatically compliant with approved interfaces guidelines (see Requirement [FR-T01.3]) from Nokia. By this means, the resulting prototypes are expected to be more user-friendly compared to prototypes which are generated without this automatic verification since inexperienced interface developers can forget to consider the different guidelines, such as a consistent layout and use. The quality of the resulting prototypes in terms of their usability, however, needs to be validated which is covered in Chapter 5.

4.2.2 The Evaluation of Prototypes

When conducting a user evaluation, the evaluation component supports the execution of local and remote evaluations asynchronously and synchronously during the entire development process (see Requirement [FR-T03]). That is possible due to the used architecture and software modules (see Section 4.1). The mobile users can use their mobile phones and the installed prototypical applications spatially and temporally separated from the evaluator since the communication between them is handled by means of an infrastructure-based network (e.g. UMTS). The evaluator only needs to start the evaluation component in order to log all incoming messages from the network layer. Since several evaluator components can connect to the server, data of several mobile users can be logged at the same time (see Requirement [FR-T03.4]). Examples about the conduction of the different empirical evaluation types are given in Chapter 6.

The conduction of empirical evaluations: A main feature of MoPeDT's evaluation component is the support to synchronously record all explicitly and implicitly executed mobile interactions during the conduction of an empirical evaluation. For this feature, MoPeDT makes use of the architecture's plug&play component called evaluator (see Section 4.1.1). This component does not only support the logging of contexts of the selected mobile user but also contexts of sensors in order to store the environmental context. By logging the user and environmental context, extensive knowledge is provided for the analysis phase. For instance, the environmental context might provide knowledge why the test user preferred a mobile interaction style in contrast to another. In order to display the synchronously logged data as well as to enable the selection of the intended mobile user and sensors, MoPeDT provides a GUI for the evaluation component (see Figure 4.14).

Besides the logging and display of objective data which are rather quantitative (see Requirement [FR-T03.1]), the evaluation component also supports the logging and display of objective data which are rather qualitative (see Requirement [FR-T03.2]). When conducting a local evaluation, the user and her environment can audiovisually be captured and displayed while she is interacting with a prototype, such as to complete a task. For this capturing, the interface designer can select two cameras and microphones.

When conducting a remote evaluation, media input screens (e.g. see the audio or video input screen in Section 4.1.1 as types of a `MediaScreen`) can be used to capture objective data by the mobile user which are rather qualitative. The moment of capturing the data either can freely be decided by the mobile users themselves or it can be defined as part of the prototype's application flow. A media input

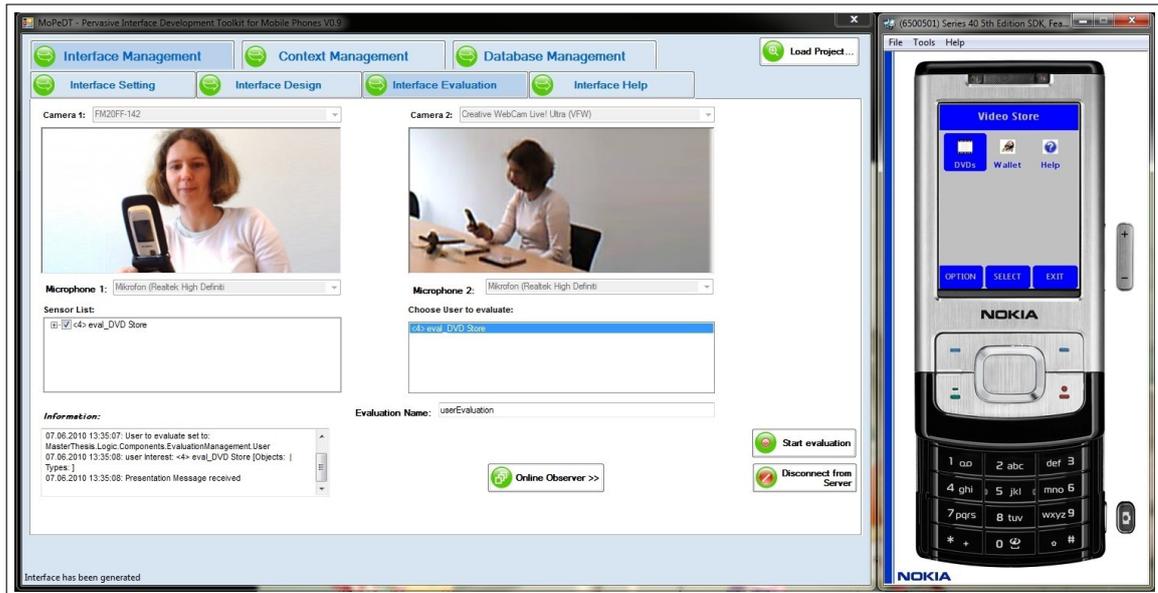


Figure 4.14: The graphical user interface of MoPeDT for the conduction of a user evaluation. The upper part displays the two selected cameras while the lower part shows the selected mobile user and sensors as well as incoming messages from the server. The right part displays the cloned screen view of the selected mobile user.

screen, for instance, can appear whenever a certain context has been detected. This capturing screen also can be activated by the mobile user, such as by selecting a corresponding keyboard command (e.g. FEEDBACK). Since the mobile users or the system do decide on the moment of capturing data and not the evaluator spontaneously, the conduction of remote evaluations and the kind of capturing qualitative data are rather *Cultural Probe studies* than studies which base on the *Experience Sampling Method* (see Section 2.1) when using MoPeDT. But, even though that the evaluator does not actively communicate with the test user in order to initiate the capturing of data, it also makes sense to not only conduct user evaluations with MoPeDT asynchronously but also synchronously. This is important to mention due to the fact that the evaluator can still synchronously handle the logging options.

Whenever an empirical evaluation is executed locally and remotely, MoPeDT displays a cloned screen view of the test user's mobile phone screen on the evaluator's computer (see Figure 4.14 on the right part). To do so, the emulator on the evaluator's computer makes use of the module `MOBILE CLIENT` and connects to the main server. Now, the GUI of the cloned client is always updated whenever the mobile application of the selected mobile user is updated which is independent on the selected mobile user's location. Consequently, the evaluator can always trace all

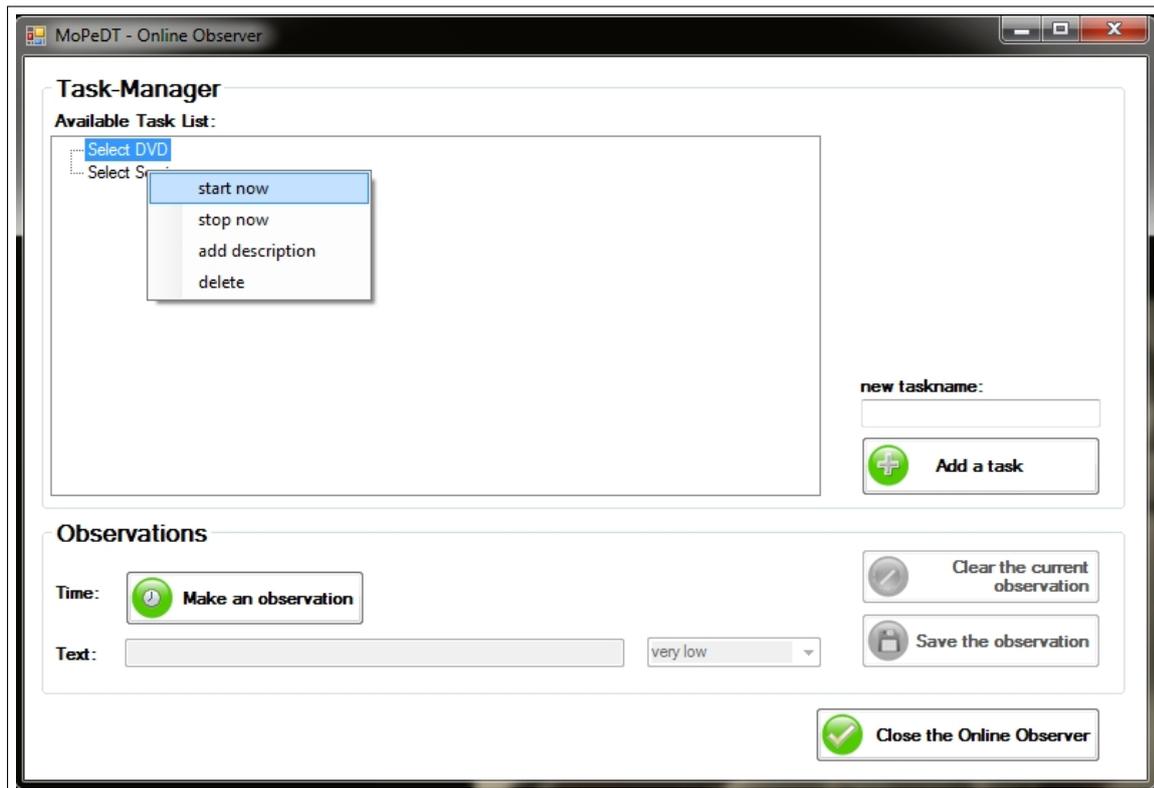


Figure 4.15: The graphical user interface of MoPeDT for the recording of live comments and details about the tasks. The upper part displays the supported tasks while the lower part provides the input of live comments.

interactions of the selected mobile user and the mobile user's current screen view. The appearance of the mobile user's prototype is also synchronously recorded. MoPeDT captures screen shots of the cloned screen view whenever a new screen is displayed.

All in all, by means of MoPeDT's evaluation component the Requirements [FR-T03.1] and [FR-T03.1] - the synchronous logging and display of objective data that are quantitative or qualitative - are fulfilled except that the support of *the Experience Sampling Method* is not provided. A precondition of the synchronous logging, however, is the previous synchronisation of all required components - the desktop computers of the evaluators, the main server and sensors as well as the mobile phones.

Besides objective data, MoPeDT also enables the display and logging of subjective data that are generated by the evaluators or mobile users (see Requirement [FR-T03.2]). Similar as for the logging of objective data by means of the mobile users, media input screens can also be used to capture textual comments or answers of a questionnaire. The moment of capturing subjective data also can either be defined

as part of the prototype's application flow (e.g. the display of a questionnaire always after the selection of a physical object) or it freely can be selected by the mobile user. Not only the mobile users, also the evaluators can log subjective data. Figure 4.15 shows the GUI that can be used to record live comments of the evaluator, such as details about interesting occurrences.

The conduction of analytical evaluations: MoPeDT focuses on the support to execute empirical evaluations with mobile end-users of the system. The conduction of analytical evaluations (see Requirement [FR-T03.5]), however, is also supported by providing assistance for the inspection-based evaluation (see Section 2.1). The support of model-based analytical evaluations is currently not supported.

As previously mentioned, whenever the interface developer has changed the design specification by means of MoPeDT, the prototype of a mobile application is automatically generated. Besides the generation of this functional prototype, screen shots of the different screen states and a screen shot of the prototype's application flow are also automatically generated. Having these screen shots and the running prototype for an emulator or real phone enable the conduction of inspection-based evaluations (see Section 2.1), such as walkthrough-based (e.g. *the Cognitive Walkthrough*) or guideline-based evaluations (e.g. *the Heuristic Evaluation*).

For *the Cognitive Walkthrough*, the interface developer can use the generated prototype or the generated screen shots to step by step execute pre-defined tasks and always answer the following four questions. Will the users try to achieve the right effect? Will the user notice that the correct action is available? Will the user associate the correct action with the effect to be achieved? If the correct action is performed, will the user see that progress is being made towards the solution of the task? If one question will be answered with no, a problem has been detected and need to be solved in a new iteration.

For *the Heuristic Evaluation*, a list of heuristics is used, such as the ten usability heuristics (e.g. the prototype's compliance of *Consistency and Standard*) from [Nielsen, 1993]. For each heuristic, the interface developers verifies the frequency, impact and persistence of violations against the corresponding heuristic and, as a result, rates the heuristic from a zero-to-four scale where zero means no usability problem and four means a usability catastrophe. For instance, for the heuristic *Consistency and Standard*, the developer can use the generated prototype and screen shots to check whether the wording, layout and actions are consistently used and whether standardised guidelines are considered. After having documented violations against the different heuristics, the frequency, impact and persistence of

the found violations lead to the rating of the severity for the category *Consistency and Standard*. In the next iteration of the user-centred prototyping process, the different found usability problems can be fixed dependent on their severities.

4.2.3 The Analysis of Evaluations

In the final step of the user-centred prototyping process, the interface developers have to analyse the captured data of a user evaluation. In order to simplify this task, the analysis component of MoPeDT also provides a GUI (see Figure 4.16) for this process step (see Requirement [FR-T05]).

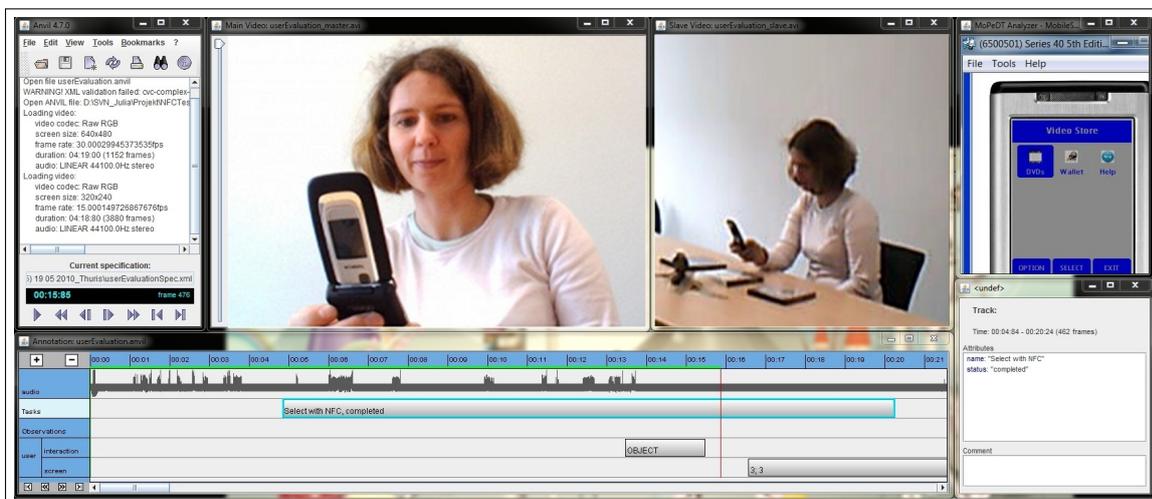


Figure 4.16: The graphical user interface of MoPeDT for the synchronous display of all recorded data. The upper part displays the captured videos and screen shots of the prototype's appearance (right). The lower part provides a time-line-based visualisation of the audio track as well as the labelling - annotation - by means of the logged quantitative data.

MoPeDT's analysis component provides the synchronous time-line-based visualisation of all recorded data. The data, however, of only one captured evaluation session can be loaded and thus Requirement [FR-T05.4] is not fulfilled by MoPeDT - the support to display captured data of several sessions. For the development of MoPeDT's analysis component (e.g. [Leichtenstern et al., 2008, Leichtenstern and André, 2010]), ANVIL (see Section 3.3) was extended. In addition to its support to display audio-visual content and time-line-based tracks, the extended version of ANVIL also displays the captured screen shots of the application - the prototype's appearance. Additionally, the extended version of ANVIL automatically synchronises and annotates the captured qualitative data by means of the recorded quantitative data (see Requirement [FR-T05.1]). Now, the interface developer can scroll

through the pre-annotated video or jump to intended data which are displayed by means of the tracks in order to, for instance, investigate the user's behaviour in different contextual situations. The pre-annotated data can be modified, such as annotations can be added, updated or removed (see Requirement [FR-T05.2]).

Since the determination of significant results is often an important analysis task, a further feature of MoPeDT's analysis component is to support statistical analyses (see Requirement [FR-T05.3]). MoPeDT supports the export of the annotated data in different formats of statistic tools (e.g. SPSS) in order to, for example, investigate the probability of occurrence for an intended context or behaviour. Consequently, the number or the mobile user's errors or the mobile user's required time to complete a task can be statistically analysed with the assistance of MoPeDT.

4.3 Summary

This chapter presented the concept and the development of the user-centred prototyping tool (UCP) tool called MoPeDT that can be used to generate and evaluate prototypes in a user-centred way in the context of *the Third Computing Paradigm*. In the following, the new conceptual and technical approaches of MoPeDT are highlighted. Aspects of this section were also published within this PhD project [Leichtenstern et al., 2011a].

In terms of the automatic prototype generation, a component-based client-server architecture was described as well as software modules of the architecture's components. Several benefits of the network-based architecture and modules exist, such as the possibility to load or change remote content as well as to run a remote empirical evaluation due to the fact that all remotely conducted user interactions can thereby be stored. Since the architecture is component-based, several components can be plugged-in including additional interaction and presentation devices. By this means, not only multiple mobile users can independently or dependently upon each other interact with the system but also these several users can be evaluated simultaneously. To handle the complex interplay between these various users and evaluation components an XML-based protocol was introduced as well. None of the related tools (see Section 3.3) aims at multi-user applications where interactions of several users have to be synchronised.

As a further new approach, MoPeDT considers the compliance of mobile HCI guidelines by making use of screen templates. Some related tools just provide layout managers to consider aspects of usability (e.g. OmniScope) but none of the related tools considers the compliance by means of a concept such as screen

templates. By means of these templates, for instance, a consistent layout and keyboard use can be ensured. Also, reversibility of actions can be addressed as well as the providing of contextual help for each screen. When using screen templates, further new aspects can be realised, such as a generation of prototypes with a low and high level of fidelity.

Also in contrast to the related tools, MoPeDT provides assistance when conducting both analytical and empirical evaluations. Screen shots and the screen flow of the intended application can be generated during the design specification to run a guideline-based analytical evaluation. For the conduction of empirical evaluations, the use of a cloned screen view was presented as a new approach. Due to the component-based architecture, the evaluator component can register the interest in a mobile user and thereby a mobile emulator is able to display a cloned screen view of the intended user to always provide details about the user's current displayed screen. This cloned display is also used to capture the screens of the mobile users for the later on analysis phase.

Finally, MoPeDT provides support for local and remote evaluations in different test settings. Chapter 6 goes into the details about the tool-supported conduction of user evaluations in three different settings: real evaluations in the real world, simulated evaluations in the real world and simulated evaluations in the virtual world. The related tools of Section 3.3 concentrate on one of the test settings. Some tools support during remote evaluations in the field: real world evaluations while other tools focus the conduction of remote or local evaluations in the laboratory: real world simulations. None of the related tools provide details about a support to run virtual world simulations.

Even though the chapter presented new conceptual and technical approaches how to meet the different functional requirements towards the tool and the resulting prototypes by means on the implementation of MoPeDT, the meeting of non-functional requirements still needs to be addressed. Further, the question is whether interface developers accept to use an all-in-one tool for the user-centred prototyping process in all stages and whether it meaningfully supports them during the process to generate a user-friendly product. The next chapter aims at answering these questions.

Chapter 5

The Empirical Tool Validation

After having introduced conceptual and technical approaches about the meeting of the functional requirements, this chapter aims at answering the question whether MoPeDT also meets the non-functional requirements. The resulting prototypes should be highly usable (see Requirement [NFR-P01]), reliable (see Requirement [NFR-P02]) and functional (see Requirement [NFR-P03]). As non-functional requirements towards a UCP tool, the tool should also be highly usable (see Requirement [NFR-T01]) by improving the interface developer's efficiency (see Requirement [NFR-T01.1]), effectiveness (see Requirement [NFR-T01.1]) and satisfaction (see Requirement [NFR-T01.2]) when executing the UCP process. Further, the tool should reduce be easy to learn (see Requirement [NFR-T01.4]) as well as highly intuitive (see Requirement [NFR-T01.5]) and transparent (see Requirement [NFR-T01.6]). Finally, it should be highly reliable (see Requirement [NFR-T02]) and functional (see Requirement [NFR-T03]).

This chapter aims at the execution of an experimental tool study in order to reveal benefits and problems of MoPeDT. The approach of using screen templates, for instance, is validated as well as whether interface developers see benefits in using an all-in-one tool approach. Before MoPeDT's tool study is described, related tool studies are covered in order to find an appropriate experimental setting.

5.1 Related Studies

[Scriven, 1967] describes two promising evaluation methods: the formative and summative evaluation. Formative evaluations concentrate on the investigation of a process whereas summative evaluations analyse the results of a process. Scriven originally described his two methods for the evaluation of the children's learning

processes - formative - and their effects - summative. These two kinds of evaluations are also applicable for tool studies because the concept of a formative evaluation can be used for the investigation of the process to design, evaluate and analyse a prototype with the support of a software tool whereas the summative evaluation method can be applied to review the resulted prototypes of the user-centred prototyping (UCP) process. Thus, by using both evaluation methods, the meeting of non-functional requirements towards the tool - formative - and the resulting prototypes - summative - can be validated. This section aims at related tool studies and the discussion whether they were executed as formative and summative evaluations.

5.1.1 Related Methods

In the following, the previously mentioned related tools (see Section 3.3) are reflected on their applied tool study method in order to find an appropriate method that combines the formative and summative evaluation method. Most tool studies were conducted as formative evaluations in order to investigate the quality of the tool. The results of the tool-support (e.g. the resulting prototypes) were rarely considered.

Informal usability study: A typical formative tool study is rather informally performed in order to validate a tool's reliability (see Requirement [NFR-T02]) and usability (see Requirement [NFR-T01]) with a focus on the determination of the user's overall acceptance and satisfaction with the tool (see Requirement [NFR-T01.3]). Such a formative evaluation normally starts with the test user's task to design, evaluate and analyse an instructed prototype by using the corresponding tool and ends with the test user's feedback via interviews or questionnaires about found problems and the overall experience with the tool. MakeIT [Holleis and Schmidt, 2008] is an example that was evaluated in that order as well as Momento [Carter et al., 2007]. The UCP tool d.tools [Hartmann et al., 2006] was also similarly evaluated except that the developers of d.tools did not only consider questions about the user's acceptance and satisfaction (see Requirement [NFR-T01.3]) but also questions in terms of intuitiveness (see Requirement [NFR-T01.5]) and efficiency (see Requirement [NFR-T01.1]).

Feature and usability study: Another type of formative evaluation was conducted for the tool Mobile Bristol [Hull et al., 2004]. This tool was evaluated on the test user's used tool features. Two artists were instructed to develop prototypes with the tool and their used features were investigated. Thus, the investigation was rather on the tool's functionality (see Requirement [NFR-T03]) than on

the usability (see Requirement [NFR-T01]) and reliability (see Requirement [NFR-T02]). The evaluation of OIDE [McGee-Lennon et al., 2009] also addressed the determination of the used tool features (see Requirement [NFR-T03]) but also investigated the user's acceptance of the tool and the tool's usability problems (see Requirement [NFR-T01]). In contrast to the other mentioned formative tool evaluations, OIDE's developers used much more evaluation techniques to investigate their tool. They applied and analysed video observations, questionnaires and protocols of the test users and the evaluators as well as screen shots of the tool. This evaluation illustrates the need to combine techniques which provide subjective data with techniques which give access to objective data. Questionnaires rather provide subjective data with information about a test user's attitude while video observations generate objective data with information about a test user's behaviour and performance (see Section 2.1).

Comparative usability study: TERESA [Chesta et al., 2004] is the only tool that was evaluated based on a comparative tool study. Two different settings were utilised. In the first setting test users used traditional approaches to generate websites for a desktop computer and a mobile phone whereas TERESA was used in the second setting. During the development of the websites, the test users documented problems, commented on tool features as well as noted the required time to develop the prototype in both settings. Several non-functional requirements were validated, such as the tool's intuitiveness (see Requirement [NFR-T01.5]) and learnability (see Requirement [NFR-T01.4]). Also, the developer's satisfaction (see Requirement [NFR-T01.3]) and efficiency (see Requirement [NFR-T01.1]) were rated in a questionnaire to also get subjective feedback of the users. Despite the carefully thought out method, TERESA's evaluation method, however, focused on the investigation of the process - the development of a prototype with and without a tool-support - and not on the result of the process - the resulting prototypes.

5.1.2 Discussion

As a conclusion, the literature review shows that most tool developers only conducted informal formative tool studies. TERESA's tool study illustrates an elaborated formative evaluation method that gives valuable input to MoPeDT's tool study. In order to meaningfully reveal benefits and problems of the tool, the test users of the tool study should develop a clearly predefined prototype in a user-centred way in two different settings - with the tool and with the baseline - since comparative studies can provide meaningful knowledge about an added-value

when using the tool compared to the other approach. Observations and protocols can help objectively investigate the development process in terms of the designer's efficiency (see Requirement [NFR-T01.1]) as well as the tool's reliability (see Requirement [NFR-T02]) and functionality (see Requirement [NFR-T03]). Questionnaires can give valuable subjective data about the interface designer's satisfaction and acceptance (see Requirement [NFR-T01.3]). Further, the intuitiveness (see Requirement [NFR-T01.5]) and learnability (see Requirement [NFR-T01.4]) of the tool also can also be validated by means of questionnaires.

In contrast to formative tool evaluations, there are less known comprehensively described summative tool studies which analysed the generated prototypes of the development process. The combined evaluation of a process - formative - and its results - summative - in a comparative evaluation, however, has to be conducted in order to meaningfully determine the meeting of non-functional requirements towards the tool and the resulting prototypes as well as the strengths and weaknesses of a tool. The resulted prototypes can be analysed by means of different evaluation techniques in order to investigate the interface designer's effectiveness (see Requirement [NFR-T01.2]) to develop a highly usable (see Requirement [NFR-P01]), reliable (see Requirement [NFR-P02]) and functional prototype (see Requirement [NFR-P03]). For instance, the resulting prototype's reliability and functionality as well as its usability can be investigated by conducting an analytical evaluation, such as an guideline-based inspection.

5.2 Tool Study with MoPeDT

Inspired by the tool study conducted for TERESA [Chesta et al., 2004], MoPeDT's tool study was planned and executed [Leichtenstern and André, 2010, Leichtenstern et al., 2011a]. In the following, the experimental setting of the utilised evaluation method is covered, then a report on the conduction of the tool study is described and finally the results are illustrated.

5.2.1 The Experimental Setting

The tool study of MoPeDT was conducted as a comparative study that applied analytical and empirical evaluation techniques to collect subjective and objective data in order to reveal the meeting of the non-functional requirements towards the tool and the resulting prototypes. The resulting prototypes should be highly usable (see Requirement [NFR-P01]), reliable (see Requirement [NFR-P02]) and functional (see Requirement [NFR-P03]). As non-functional requirements, the

tool should also be highly usable (see Requirement [NFR-T01]) by improving the interface developer's efficiency (see Requirement [NFR-T01.1]), effectiveness (see Requirement [NFR-T01.1]) and satisfaction (see Requirement [NFR-T01.2]) when executing the UCP process. Further, the tool should reduce the required skills by being easy to learn (see Requirement [NFR-T01.4]) as well as highly intuitive (see Requirement [NFR-T01.5]) and transparent (see Requirement [NFR-T01.6]). Finally, MoPeDT should be highly reliable (see Requirement [NFR-T02]) and functional (see Requirement [NFR-T03]).

Furthermore, the results of the tool study should shed light on potential problems and benefits when using a UCP tool, such as MoPeDT. In particular, the evaluation should provide insights whether interface designers like the all-in-one tool-support approach to execute the design, evaluation and analysis with a single tool. Moreover, novel tool features of MoPeDT should be validated, such as whether the utilisation of screen templates can improve the quality of the resulting prototypes.

Independent variables: To investigate the meeting of the non-functional requirements, the used platform was defined as independent variable with the following two levels. In the first level the interface developers applied MoPeDT and the mentioned features for the user-centred design specification, evaluation and analysis of a prototype in the context of *the Third Computing Paradigm* (see Chapter 4) whereas in the second level the interface developers utilised traditional design, evaluation and analysis platforms (e.g. Eclipse or Netbeans). Thus, in contrast to most mentioned tool studies, a comparative study was conducted that applied well-known and commonly used platforms as baseline for all steps of the user-centred prototyping process.

Dependent variables: Quite contrary to most other methods, for the evaluation of MoPeDT, the used approach gathered subjective and objective data in order to measure the following dependent variables.

- **The interface developer's efficiency** (see Requirement [NFR-T01.1]) to quickly run through the user-centred prototyping process.
- **The interface developer's effectiveness** (see Requirement [NFR-T01.2]) to generate a highly usable (see Requirement [NFR-P01]), reliable (see Requirement [NFR-P02]) and functional (see Requirement [NFR-P03]) prototype.
- **The interface developer's satisfaction** (see Requirement [NFR-T01.3]) by reducing the required skills and being easy to learn (see Requirement [NFR-T01.4]) as well as by providing a highly intuitive (see Requirement [NFR-T01.5]), transparent (see Requirement [NFR-T01.6]), reliable (see Requirement [NFR-T02]) and functional (see Requirement [NFR-T03]) tool.

For the measurement of the three dependent variables, two empirical methods and an analytical method were applied. As a first empirical method, an inquiry method of a questionnaire was utilised in order to acquire subjective data. As a further empirical method, an observation method of a protocol recording was used to collect objective data. Additionally, an analytical method was utilised. An inspection method of a guideline review was applied to collect objective data.

Protocol recording (see Appendix C): The protocol recording was applied in order to conduct a formative evaluation and gather objective data for the investigation of the interface developer's efficiency and satisfaction. The test users used the protocols for the documentation of the required time to complete different subtasks while designing, evaluating and analysing the prototypes in both settings. Moreover, emerged problems had to be noted for each subtask in order to find problems in terms of the tool's usability (see Requirement [NFR-T01]), reliability (see Requirement [NFR-T02]) and functionality (see Requirement [NFR-T03]). In order to keep comparability between the different test users, they used the same wording when documenting the subtasks. During the design, for instance, the protocols provided subtasks to implement *the client-server communication* or *the graphical user interface of the mobile phone*. For the evaluation and analysis, the protocol also listed subtasks, such as *the recording of videos* or *the synchronisation of the recorded events with the captured videos*.

Questionnaire (see Appendix D): The post-task questionnaire first addressed questions of demography. The test users had to answer questions about their age, gender as well as Software and Usability Engineering skills. The main part of the questionnaire asked for ratings of statements which concerned the prototype's design, evaluation and analysis of both levels: with MoPeDT and with the traditional approach. The statements targeted the efficiency (E), effectiveness (Eff), satisfaction (S), learnability (L), transparency (T), intuitiveness (I) and sphere of action (A).

In terms of efficiency, the test users had to estimate a level's influence on their efficiency in the design (Question E1) as well as in the evaluation and analysis (Question E2) phase. Additionally, the levels were measured whether they usefully supported the conduction of the user-centred prototyping process (Question E3) and whether they could provide a gain in time (Question E4). The statement about effectiveness concerned the satisfaction with the generated prototype in both levels (Question Eff1).

The questionnaire also resulted subjective data in terms of satisfaction, learnability, transparency, intuitiveness and sphere of action - user control. First, the test

users had to rate their satisfaction when designing (Question S1) and when evaluating and analysing a prototype (Question S2). Then, the test users had to estimate the ease of learnability for the design (Question L1) and for the evaluation and analysis (Question L2). In this context, the test users also had to rate the required programming (Question L3) and evaluation skills (Question L4). Further statements were about a level's system transparency for the design (Question T1) and for the evaluation and analysis (Question T2). Also, the tool's intuitiveness had to be rated for the design (Question I1) and evaluation and analysis (Question I1). Finally, the test users had to estimate the satisfaction with the sphere of action for the design (Question A1) and the evaluation and analysis (Question A2) in both levels. In this term, the test users also assessed the scope of the supported screen templates (Question A3). All mentioned statements had to be rated on a five point scale from strongly disagree to strongly agree.

In addition to the statements, the questionnaire also contained questions that asked the test users to select preferences. In one question, the test users had to select their preferred level for the design, evaluation and analysis while another question addressed the preferred component of MoPeDT. By means of these questions, the test users' overall acceptance should be validated with a UCP tool, such as MoPeDT as well as their attitudes towards the all-in-one tool-support approach.

Guideline review (see Appendix A): To gather objective data and validate the interface developer's effectiveness, a summative evaluation - a guideline review - was conducted. The resulted prototypes for both levels - with and without MoPeDT - were investigated. An independent usability expert who was not involved in the development or evaluation of MoPeDT used the generated prototypes and investigated their robustness (see Requirement [NFR-P02]) and functional completeness (see Requirement [NFR-P03]) based on the task description (see Paragraph *Task Description*) as well as their violation against the 22 mentioned guidelines (see Requirement [NFR-P01]) which base on *Nokia's Design and User Experience Library*.

For example, the expert controlled a consistent softkey use (see Guideline G1) and layout (see Guideline G2) as well as a correct error handling (see Guideline G3), a support of a contextual help in the different screens (see Guideline G4) and an easy reversibility of actions (see Guideline G5). Further examples are the consideration to minimise the number of screens (see Guideline G12), to display important information with text and icons (see Guideline G15) and to clearly structure screens (see Guideline G16). Some of the guidelines are automatically supported by MoPeDT (e.g. see Guidelines G1, G2, G3, G5 and G16) but other guidelines are not covered, such as Guidelines G12 and G15.

Group design: For the experiment, a within-subject design was used and therefore, all test users participated in both settings of the experiment. To prevent any practice and carryover effects, the half of all test users started with the design, evaluation and analysis of the intended prototype based on the traditional application of the user-centred prototyping process and afterwards used MoPeDT whereas the other test users used MoPeDT first. The test users were randomly assigned to the two groups.

Task description (see Appendix B): In both levels of the user-centred prototyping with and without MoPeDT, the same pervasive shopping assistant for mobile phones had to be designed, evaluated and later on analysed with end-users of the application.

This pervasive shopping assistant helps users to receive information about products in a shopping store (e.g. about the ingredients of products). A very simple task scenario was used for the tool study in order to enable the conduction within one month.

Tasks for the design specification Phase: To keep comparability, the test users received a detailed description about the intended prototype. First, the test users had to generate a database and add content about three products: tomato, cucumber and potato. Then, they had to generate a prototype that network-based replies the content of the database.

The user interface of the prototype was described as followed: A first screen should be provided to enable the selection of the intended product. After the selection of the product, the corresponding services (e.g. origin or ingredients) should be presented and later on the content of the service entry in form of text and images. During the design of the prototype based on MoPeDT, the appearance and behaviour of several screens had to be specified dynamically since most contents were loaded and displayed from the remote database.

The task description also contained the requirement to implement prototypes which enable a keyboard-based and a NFC-based mobile interaction. For the design and implementation of the prototype with the traditional approach, the test users were also instructed to implement a logging mechanism for the prototype in order to enable a recording of the user's performed mobile interactions in the evaluation phase.

Tasks for the evaluation phase: For the evaluation of their generated prototypes, the test users were instructed to audio-visually capture three end-users while they were interacting with the two generated prototypes. Besides the

capturing of audio-visual content, the user's mobile interactions also had to be recorded.

The conduction of the evaluation phase was predefined in order to enable the comparison between the captured evaluations of the two settings. First, the three end-users had to read the information about the tomato's origin. Then, they should request the cucumber's description and finally they had to consider the ingredients of a potato.

Tasks for the analysis phase: After the evaluation phase, the participants of the tool study had to analyse their captured audio-visual content and logged user's mobile interactions in order to find usability problems of the two prototypes. For instance, they validated the logged mobile interactions to find problems of efficiency and effectiveness. Before they could analyse their captured data in the traditional setting, they had to synchronise all recorded data and annotate the audio-visual content by means of the logged user's mobile interactions.

Overall, the task description contained all required steps - tasks and subtasks - to complete the design, evaluation and analysis with and without MoPeDT.

5.2.2 Conducting the Pilot Tool Study

Before the main tool study was executed a short pilot study [Leichtenstern and André, 2009b] was executed with 7 students to investigate the quality of the experimental setting. The participants had intermediate Software Engineering skills - between one and five years - and minor Usability Engineering skills - less than one year.

The same experimental setting was executed in the pilot study. The only difference was that an interview was applied as subjective evaluation method and not the introduced questionnaire (see Appendix D). Based on the findings of the interviews, interesting questions were extracted to finally came up with the final questionnaire for the main tool study. During the pilot study test users had to design, evaluate and analyse the same prototype with and without MoPeDT as in the main tool study. The test users used protocol recordings to document their execution time and found problems. A guideline review was also applied to distinguish the quality of the prototypes which were generated with MoPeDT and with the traditional approach.

The results of the protocol recording and guideline review were promising. On average, the required time in minutes to design, evaluate and analyse a prototype without MoPeDT was significantly higher than when using MoPeDT ($p < 0.05$).

When analysing the guideline review, the prototypes developed without MoPeDT had significantly more usability problems than when using MoPeDT ($p < 0.001$). All in all, the guideline review and the protocol recoding seemed to be useful evaluations techniques.

The questionnaire conducted with the participants of the pilot tool study showed benefits and problems of MoPeDT. The most acute problem was the limitation in the scope of supported operations, such as the number of the supported screen templates. Highlighted benefits of MoPeDT were the less required programming and interface design skills as well as the saved time and improved quality of the prototypes. As result of the pilot tool study, statements about the sphere of action - user control - as well as the scope of the supported screen templates were added to finally come up with the questionnaire as described in Section 5.2.1.

5.2.3 Conducting the Main Tool Study

After having described, the experimental design of the tool study and the conduction of a short pilot study, this section aims at the conduction of the main tool study.

Prior to the study: The test users of the main tool study were students of a three-month course *Usability Engineering*. Before the study was executed at the last third of the course, tutorials were conducted within the course and all participants of the tool study were taught how to use MoPeDT for the interface's design, evaluation and analysis and how to implement and evaluate mobile phone prototypes using Eclipse with EclipseME or Netbeans with the Mobility Pack. Prior to the tool study the test users had to implement a server-client-based chat program for mobile phones in J2ME and J2SE. While implementing this prototype, they acquired knowledge about the socket programming as well as the programming of graphical user interfaces for mobile phones. During the tool study, the test users could reuse parts of their chat program or use other available source code, such as program code from the Internet.

In addition to the software skills, all test users were comprehensively taught about usability in general, the Human-Centred Design process, mobile phone usability and the mobile phone guidelines which the usability expert also used for the guideline review. The test users were reminded to apply these guidelines for both levels when designing, evaluating and analysing the prototypes in the tool study. The test users learnt how to execute an evaluation as well as how to analyse the acquired data. By this means, the annotation tool ANVIL [Kipp, 2001] was introduced and discussed. This traditional tool could have been used for the analysis

of the captured video data. Whilst the training period of MoPeDT and the conduction of the tool study, the test users were not informed that MoPeDT is a software tool that was developed at the Human-Centered Multimedia Lab.

5.2.4 Results of the Tool Study

In this section the results of the tool study are discussed: the potential increased interface developer's **efficiency** to quicker develop prototypes (see Requirement [NFR-T01.1]), the optional improved interface developer's **effectiveness** (see Requirement [NFR-T01.2]) to develop more user-friendly (see Requirement [NFR-P01]), reliable (see Requirement [NFR-P02]) and functional (see Requirement [NFR-P03]) prototypes as well as the potential increased interface developer's **satisfaction** (see Requirement [NFR-T01.3]) when using MoPeDT instead of the traditional approach by reducing the required skills and being easier to learn (see Requirement [NFR-T01.4]) as well as providing a more intuitive (see Requirement [NFR-T01.5]), transparent (see Requirement [NFR-T01.6]), reliable (see Requirement [NFR-T02]) and functional (see Requirement [NFR-T03]) tool.

Additionally, the results answered questions whether the tool users liked the all-in-one tool approach or not and whether the novel tool features of MoPeDT were useful or not.

The demography: 20 computer science students - 16 male and four female students - of the course *Usability Engineering* participated in the one-month tool study. The test users were aged between 22 and 29 ($M = 24.15$, $SD = 1.90$). First, the test users had to rate their programming and user interface design skills on a five point scale from none to expert. On average, most of the participants rated themselves as medium skilled in object-oriented programming languages, e.g. Java and C++ ($M = 3.9$, $SD = 0.64$) and mobile phone programming, e.g. J2ME ($M = 2.3$, $SD = 0.86$) as well as medium skilled in Usability Engineering in general ($M = 3.25$, $SD = 0.86$) and mobile Usability Engineering ($M = 2.95$, $SD = 0.83$). Two test users had previous knowledge in mobile phone programming - about one year.

The efficiency: To answer whether the efficiency of the interface designer can be improved when using MoPeDT instead of the traditional approach (see Requirement [NFR-T01.1]), the test users rated questions about their efficiency and the tool-supported user-centred prototyping process in terms of time. Figure 5.1 shows the overall ratings.

Based on the rating scale from one to five, on average, the test users agreed with the statement about their increased efficiency when using MoPeDT for the design

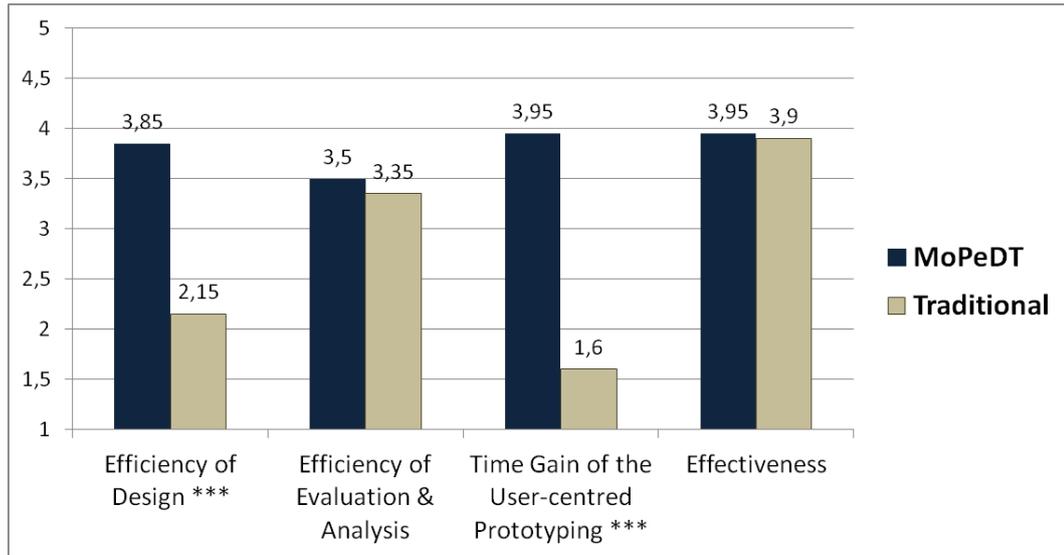


Figure 5.1: User ratings of effectiveness and efficiency. The test users rated the provided statements based on a scale from 1 (strongly disagree) to 5 (strongly agree). In terms of design specification, efficiency and time gain were significantly better rated for MoPeDT compared to the traditional approach (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).

specification (E1: $M = 3.85$, $SD = 1.27$), which was significantly higher than when using the traditional approach (E1: $M = 2.15$, $SD = 0.88$), $t(19) = 4.68$, $p < 0.001$. The efficiency when using MoPeDT for the evaluation and analysis (E2: $M = 3.5$, $SD = 1.32$) was also seen as a bit higher but not significantly (E2: $M = 3.35$, $SD = 0.88$). The test users also found that MoPeDT (E3: $M = 3.5$, $SD = 0.09$) makes the whole user-centred prototyping process more efficient compared to the traditional approach (E3: $M = 2.8$, $SD = 1.01$), $t(19) = 2.05$, $p = 0.054$. Additionally, the time gain with MoPeDT (E4: $M = 3.95$, $SD = 1.0$) was significantly higher rated than the time gain when using the traditional approach (E4: $M = 1.6$, $SD = 0.6$), $t(19) = 7.37$, $p < 0.001$. The qualitative feedback of the test users substantiate the results. Most test users found the use of MoPeDT as *quick and easy* and see a benefit in *the very quick prototyping and evaluation of applications*.

When analysing the protocol recordings based on a two-sided dependent t-test, on average, the required design, evaluation and analysis time in minutes with traditional approaches, e.g. Eclipse ($M = 816.60$, $SD = 318.81$) was significantly higher than when using MoPeDT ($M = 266.65$, $SD = 208.14$), $t(19) = 9.2$, $p < 0.001$. When not using MoPeDT, the network and GUI programming required much more time in the design specification phase. In the evaluation and analysis phase, the annotation and analysis of the captured videos decelerated the user-centred prototyping process when not using MoPeDT.

The results of the protocol recordings and the questionnaire prove the meeting of the Requirement [NFR-T01.1]. A UCP tool, such as MoPeDT can improve the interface developer's efficiency to quickly run through a user-centred prototyping iteration.

The effectiveness: The second aspect of the tool study aimed at the interface developer's effectiveness (see Requirement [NFR-T01.2]) to develop prototypes with a high user-friendliness (see Requirement [NFR-P01]), reliability (see Requirement [NFR-P02]) and functionality (see Requirement [NFR-P03]) when applying a UCP tool, such as MoPeDT.

Figure 5.1 illustrates the participants' ratings about the quality of the resulting prototypes. The analysis of the subjective data revealed no significant benefit when using a UCP tool, such as MoPeDT. The participants of the tool study similarly rated the quality of prototypes that were generated with traditional approaches (Eff1: $M = 3.9$, $SD = 0.92$) compared to the prototypes that were generated with MoPeDT (Eff1: $M = 3.95$, $SD = 0.76$). The analysis of the qualitative data shed light on the participants' ratings. While most of them highlighted the MoPeDT's generated prototype as *beautiful which is independent on the mobile phone platform and follows design guidelines*, they also claimed the limitation caused by the screen templates. For instance, one test user claimed that *I could not individually design the application because I had to conform to the prefabricated patterns*. This finding can be supported by the moderate ratings regarding the appropriate scope of the supported screen templates (A3: $M = 2.7$, $SD = 1.03$).

In contrast to the subjective data, based on a two-sided dependent t-test, the objective results of the guideline review showed a highly significant difference between the prototypes which were designed, evaluated and analysed with MoPeDT compared to the generated prototypes with the traditional approach. Prototypes generated with MoPeDT had, on average, less violations against the 22 guidelines ($M = 0.85$, $SD = 0.93$) than the prototypes that were developed with traditional approaches ($M = 4.35$, $SD = 2.52$), $t(19) = 5.48$, $p < 0.001$. Most often, the prototypes developed with traditional approaches did not consider a consistent use of the softkeys (G1: 17 of 20 test users) and did not use icons and text for important information (G15: 17 of 20 test users). Another often occurred error was the non-compliance to support contextual help for each screen (G4: 11 of 20 test users). In terms of functionality and reliability, the expert did not find significant differences between the prototypes that were generated by means of MoPeDT or the traditional approach.

Figure 5.2 shows a prototype that was designed, evaluated and analysed by one test user with MoPeDT whereas Figure 5.3 shows the prototype that was devel-

oped by the same test user with the traditional approach.

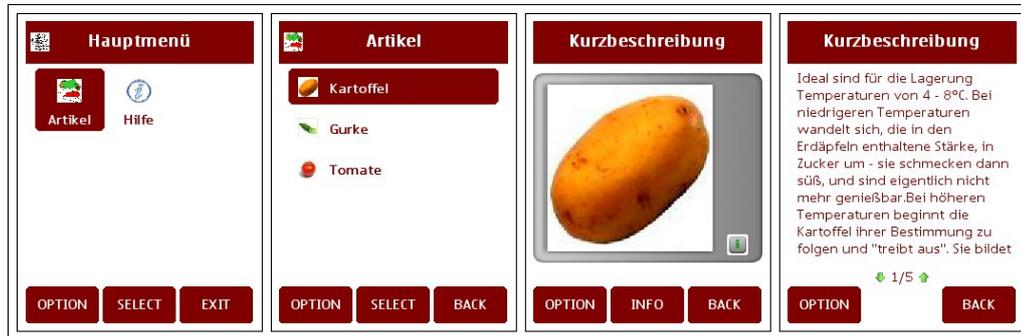


Figure 5.2: Screen shots of a prototype that was developed with MoPeDT.



Figure 5.3: Screen shots of a prototype that was developed with the traditional approach.

Based on the objective data, the increased effectiveness of the developers (see Requirement [NFR-T01.2]) to generate user-friendlier prototypes (see Requirement [NFR-P01]) is suggestively proved due to the fact that the resulted prototypes of MoPeDT provided a better usability based on the guideline review than the prototypes which were generated with traditional approaches. The participants of the study, however, would like to have a wider range of action when designing the prototype's layout with MoPeDT. Consequently, the concept of using screen templates seems to improve the objectively measured quality of prototypes but the interface developer's felt user control should not be limited. In terms of reliability (see Requirement [NFR-P02]) and functionality (see Requirement [NFR-P03]), both generated prototypes - with and without MoPeDT - meet the two corresponding requirements.

The satisfaction, user control user control, transparency, intuitiveness and learnability: A central part of the questionnaire addressed the test user's satisfaction, user control, transparency, intuitiveness and learnability (see Requirement [NFR-T01.2]). Figure 5.4 shows the most interesting corresponding results.

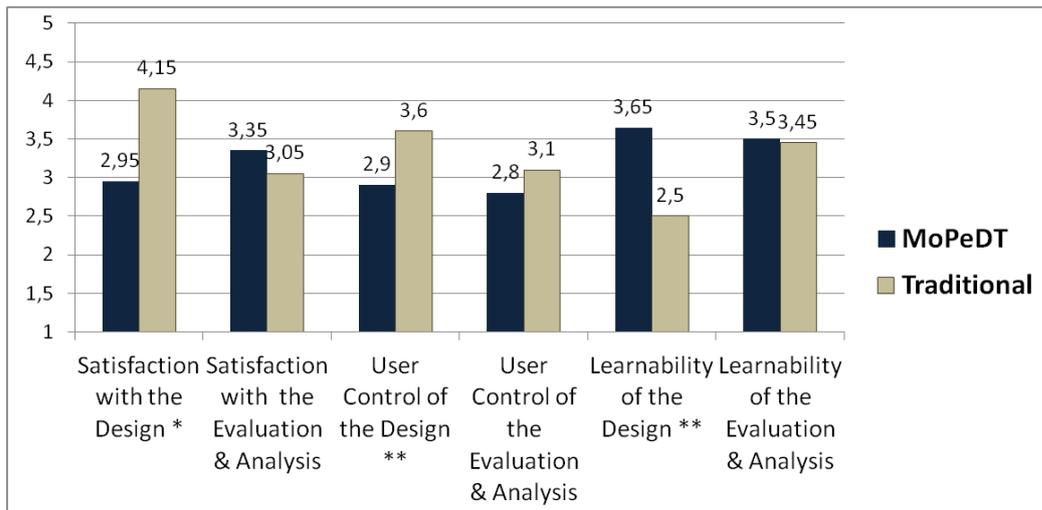


Figure 5.4: User ratings of satisfaction, user control and learnability. The test users rated the provided statements based on a scale from 1 (strongly disagree) to 5 (strongly agree). In terms of design specification, satisfaction and user control were significantly better rated for the traditional approach compared to MoPeDT while learnability was better rated for MoPeDT (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).

The limitations of MoPeDT and its screen templates did not only affect the design specification of the prototype's appearance. In general, the test users reported a significant lack of user control when using MoPeDT (A1: $M = 2.95$, $SD = 1.15$) compared to the sphere of action that was supported by the traditional approach (A1: $M = 4.15$, $SD = 1.09$), $t(19) = 3.21$, $p < 0.01$. The supported sphere of action for the evaluation and analysis with MoPeDT (A2: $M = 3.35$, $SD = 1.09$) was rated a little bit higher than for the traditional approach (A2: $M = 3.05$, $SD = 1.19$). The limited sphere of action mainly caused that the test users were significantly more satisfied when using the traditional approach (S1: $M = 3.6$, $SD = 0.68$) for the design of a prototype compared to MoPeDT (S1: $M = 2.9$, $SD = 1.07$), $t(19) = 2.41$, $p < 0.05$. For the evaluation and analysis, the satisfaction with MoPeDT (S2: $M = 2.8$, $SD = 1.11$) was similar as with the traditional approach (S2: $M = 3.1$, $SD = 0.85$).

Besides the limited sphere of action, the transparency was also pointed out as a problem of MoPeDT. The collected data in terms of transparency for the design specification (T1: $M = 3.05$, $SD = 1.10$) and evaluation and analysis (T2: $M = 3.2$, $SD = 0.89$) indicated a moderate transparency of MoPeDT. The interface designers claimed that they *want to see what is going on in the background*. With regard to intuitiveness, the test users found the design specification with MoPeDT (I1: $M = 2.8$, $SD = 1.28$) a bit more intuitive than with traditional approaches (I1: $M = 2.55$,

SD = 0.89) but not significantly. For the evaluation and analysis, the traditional approach (I2: M = 3.3, SD = 0.92) was rated more intuitive compared to MoPeDT (I2: M = 3.15, SD = 0.93).

Despite these overall negative results, the test users considered some benefits for the learnability. On average, the ease of learnability was significantly higher rated for the design specification with MoPeDT (L1: M = 3.65, SD = 1.27) than the traditional approach (L1: M = 2.5, SD = 1.05), $t(19) = 3.61$, $p < 0.01$ whereas the ease of learnability for the evaluation and analysis with MoPeDT (L2: M = 3.45, SD = 0.89) was similarly rated than when not using MoPeDT (L2: M = 3.50, SD = 0.76). The test users rated less required skills when designing (L3: M = 4.4, SD = 0.60) or evaluating and analysing prototypes (L4: M = 3.1, SD = 0.97) with MoPeDT compared to the traditional approach.

In summary, even though MoPeDT reduces the required skills and simplifies the learnability (see Requirement [NFR-T01.4]), the interface developer's satisfaction (see Requirement [NFR-T01.3]) with MoPeDT is low. This is mainly caused due to the fact that the tool is less transparent (see Requirement [NFR-T01.4]). Further, the functionality (see Requirement [NFR-T03]) of the tool is too limited. The interface developers need to have a larger user control, such as by providing a wider range of screen templates. In terms of reliability (see Requirement [NFR-T02]), the participants reported on occasional system errors that need to be fixed.

The authors of related tools reported on similar results when describing the results of their evaluations. The evaluation of OIDE [McGee-Lennon et al., 2009], for instance, also showed that tool users need to have a wide range of user control in order to increase their creativity and thus their satisfaction with the tool. This is even more relevant to bear in mind if the tool user have advanced programming skills like it was the case with the MoPeDT's tool study. The tool's usability based on aspects on a tool's intuitiveness and transparency also were identified as critical aspects of user satisfaction by several tool developers, such as the developers of OIDE [McGee-Lennon et al., 2009], TERESA [Chesta et al., 2004] and d.tools [Hartmann et al., 2006].

User acceptance: In the questionnaire, the test users were also asked for their overall acceptance of MoPeDT. Thus, they had to decide about their preferred approach - MoPeDT or the traditional approach - for the design specification as well as the evaluation and analysis. Figure 5.5 illustrates the distribution for the design specification while Figure 5.6 shows the distribution for the evaluation and analysis. For the design specification, five participants of the tool study chose the traditional approach whereas seven selected MoPeDT and eight test users mentioned both approaches as useful which is quite similar to the results of the pre-

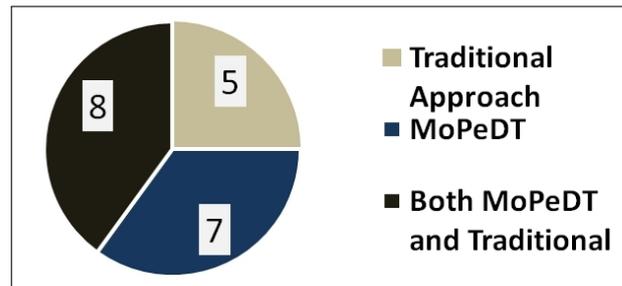


Figure 5.5: Distribution of the preferred approach for the design specification. Most test users liked both approaches (8) and MoPeDT (7) for the design specification phase.

ferred evaluation and analysis approach. Four test users favoured the traditional approach for evaluating and analysing prototypes whereas 11 test users only preferred MoPeDT and five saw benefits in using both approaches. Consequently, despite the negative satisfaction with MoPeDT, the test users tendentially preferred MoPeDT for the design specification, evaluation and analysis compared to the traditional approach. This might be caused since the test users recognised the general benefits of using a UCP tool namely the improved learnability and efficiency of use.

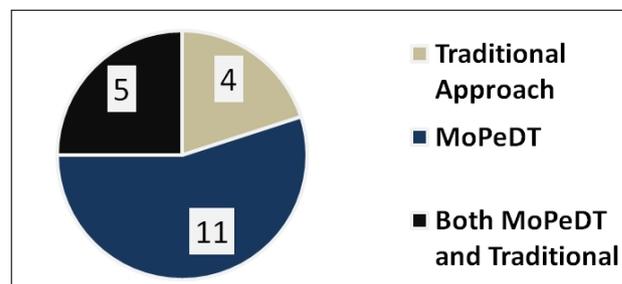


Figure 5.6: Distribution of the preferred approach for the evaluation and analysis. Most test users liked MoPeDT (11) or both approaches (5) for the evaluation and analysis phases.

All-in-one tool-support: Finally, the test users were asked for their preferred components of MoPeDT (see Figure 5.7), one test user did not like a single component of MoPeDT, while six test users only liked the design specification component and two test users only liked the evaluation and analysis component. 11 test users liked all components of MoPeDT, the components to design, evaluate and analyse a prototype. The qualitative feedback reveals that the participants favoured the approach to have an all-in-one solution. For instance, a test user mentioned the benefit to *handle everything in a single program: the database, the design, evaluation and analysis*. Another test user mentioned that *only the combination*

of all components meaningfully supports the iterative prototyping which is similarly to the statement that the quick and easy prototyping can be improved by *the close interleaving of the different components* and *the all-in-one approach* that prevents the induction in several programs. In summary, interface designers seem to accept tools for the user-centred prototyping process in all phases.

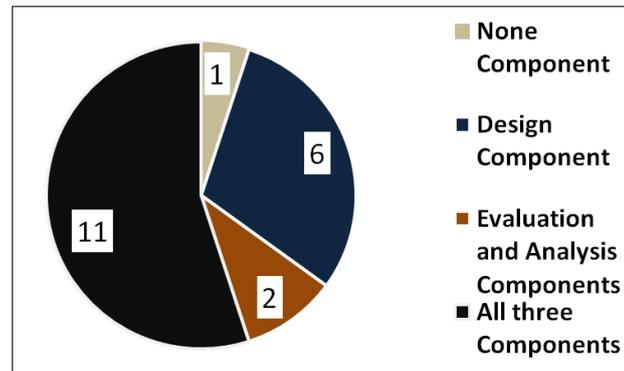


Figure 5.7: Distribution of the preferred component of MoPeDT. Most test users liked all components of MoPeDT: the design, evaluation and analysis component.

5.3 Summary

In this chapter the planing, conduction and the analysis of a tool study with MoPeDT was described.

To meaningfully validate the user-centred prototyping (UCP) tool, an evaluation setting of a comparative tool study was introduced that applied analytical and empirical evaluation techniques to collect subjective and objective data in order to reveal the MoPeDT's meeting of the non-functional requirements of Chapter 3. The results of MoPeDT's tool study also highlighted typical benefits and problems of different approaches introduced in Chapter 4 (e.g. the use of screen templates).

The findings of the tool study indicate that interface developers mostly accept a user-centred prototyping tool, such as MoPeDT for all steps of the user-centred prototyping: the design specification, conduction of an evaluation and its analysis. Also, the results showed that the interface developer's efficiency (see Requirement [NFR-T01.1]) to run through the UCP process can be enhanced when using a tool such as MoPeDT. UCP tools provide the strong benefit of a time gain. Further, the learnability of the UCP process can be improved by using a tool, such as MoPeDT (see Requirement [NFR-T01.4]). The interface developer's effectiveness (see Requirement [NFR-T01.1]) to generate highly usable (see Requirement

[NFR-P01]), reliable (see Requirement [NFR-P02]) and functional (see Requirement [NFR-P03]) prototypes was objectively fulfilled. Subjectively, the tool users did not see significant differences between prototypes that were developed with the MoPeDT compared to the prototypes that were generated by the baseline approach. The interface designer's overall satisfaction with a UCP tool seems to be strongly depending on the satisfaction (see Requirement [NFR-T01.2]) with the tool's functionality (see Requirement [NFR-T03]) and transparency (see Requirement [NFR-T01.6]).

There appears to be a tendency that a tool should make use of approaches to systematically consider well-known interface guidelines in order to increase the tool user's effectiveness to build highly usable, reliable and functional prototypes. The use of screen templates is an appropriate approach to meet that requirement. Screen templates increase the usability of the generated prototypes with regards to a compliance of approved interface guidelines. Moreover, they can improve the learnability, intuitiveness and comfort of use of the tool since reappearing patterns are provided for the design specification. The problem of screen templates, however, is the limitation of the interface designers' user control in order to individually adapt the appearance of the application.

This aspect goes along with the finding to provide a high level of tool functionality, such as by supporting the opportunity to add new types of screen templates or change existing one. By providing a higher system functionality and thus a stronger user control, however, the efficiency of the tool user as well as the learnability, intuitiveness and comfort of use of the tool might be negatively affected due to the fact that a higher tool functionality might also mean a higher complexity. Also, the interface developer's effectiveness to generate usable prototypes might be impaired if tool users have free access to adapt screen templates and thus possibly generate new violations against approved guidelines.

Besides user control and functionality, it was also recognised the strong need to keep a high level of system transparency, such as by providing detailed feedback about system processes. In particular the system process of automatically generating a prototype typically caused a problem in terms of system transparency. Even though tool users saw a strong benefit in the automatic prototype generation, they also saw the problem of losing the link between the design specification and the resulting prototype. More system notifications can help provide a clearer picture about system processes which, however, might impair the comfort of use since continuous notifications also might be annoying in time. At that point tool developers need to find a trade-off to appropriately meet all aspects: a high tool's functionality and transparency as well as an appropriate level of user control on

the one hand and an improved tool user's effectiveness and efficiency as well as a proper tool's learnability, intuitiveness and comfort of use on the other hand.

Chapter 6

Tool-Supported Empirical Evaluations

Besides the automatic generation of software prototypes, one further interesting aspect of the tool-supported user-centred prototyping (UCP) is the software assistance to conduct empirical evaluations. Characteristically for these empirical evaluations, end-users of the application use the generated prototypes in order to either freely use the application or complete pre-defined tasks (see Section 2.1). During the conduction of an empirical evaluation, tools can support evaluators by systematically assisting the capturing of user data, such as data about the user's behaviour and performance.

Most presented tools (see Section 3.3) only either assist the execution of traditional laboratory studies - real world simulations - or field studies - real world evaluations. For instance, d.tools [Hartmann et al., 2006] and SUEDE [Klemmer et al., 2000] concentrate on real world simulations whereas MyExperience [Froehlich et al., 2007], Momento [Carter et al., 2007] and ContextPhone [Raento et al., 2005] focus on real world evaluations in the field.

A new idea is to tool-assisted support different settings: real world evaluations in the field and real world simulations in the laboratory as well as empirical evaluations with partial simulations in a virtual world. These partial simulations potentially provide benefits compared to studies in the laboratory and field. For example, money and time can be saved due to the fact that neither a study has to be conducted in the real world setting of the user which is normally difficult to organise and conduct nor the real setting has to be reproduced in a real world simulation. Also, ideas of an application can virtually be presented and tested even at the very beginning of a development process which is expected to reduce development risk and help address user needs very early. Despite these benefits,

problems of the tool-supported so-called hybrid world simulations certainly exist.

The following chapter goes into the details about the tool-supported conceptions of real world evaluations and simulations as well as hybrid world simulations as a kind of a virtual world simulation. For each of these evaluation settings, typical benefits and problems are discussed.

6.1 A Real World Evaluation

The execution of empirical evaluations in the user's real world (e.g. [Häkkinen and Mäntyjärvi, 2006]) is probably the most reasonable evaluation setting. There is evidence that these so-called field studies provide very realistic and valuable data, for instance, to investigate the real user's behaviour and attitude [Palen and Salzman, 2002a] because the data are generated with real contextual constraints. In particular when developing mobile applications, it is important to execute tests in real world settings since mobile applications are often used in the wild and on the go. As a consequence of this use, application developers never can be completely sure how, when, where and under which circumstances end-users apply a mobile application and which problems thereby emerge. [Zhang and Adipat, 2005], for instance, see the major challenges for developers of mobile applications due to the following aspects: the unknown mobile context, the slow and unstable connectivity, the small screen size, the different display resolutions, the limited processing capabilities and power and finally the problematic data entry methods (e.g. text entry by means of a numerical keyboard or speech input in noisy environments).

Tools for the UCP process should provide assistance for these real world evaluations to get a clear picture about the users and their use of the system. By example of MoPeDT, a tool-supported conduction of a real world evaluation is described in the following section to finally discuss its benefits and problems.

6.1.1 An Example of a Real World Evaluation

The selected example of an application targets a mobile diary. Users shall be able to log different multimedia data - text, images, audio or video files - in order to report on their daily life. By this means, for instance, they can document their life as a university student. Besides just logging multimedia data, the users also should be able to observe their own logged data as well as data of their community, such as their friends. Thus, the community can always track on interesting new occurrences similar as propagated by the social network Facebook.

A possible objective of a real world evaluation: The intended application can be used in different situations. For instance, it can be used indoor (e.g. in a lecture hall) or outdoor (e.g. on the way to the canteen). When using the application alone and on the go under different weather conditions (e.g. strong solar radiation or rain), the users might have different needs towards the system as if they would use the application together with friends while sitting in a room.

A real world evaluation might provide details about the different relevant contexts and their corresponding needs for the users. Besides collecting knowledge in terms of the user's context, test users can also document on classical usability problems when using the application, such as inappropriate wordings of control elements.

A possible implementation: Based on MoPeDT, a version of the intended application was implemented. Figures 6.1, 6.2 and 6.3 show different screen shots of the application. The users have the possibility to configure their profiles, add new multimedia content or load data about their community to check entries of their friends. In order to document their feedback towards the current version of the application, the users can apply the item `FEEDBACK` of the option menu. By this means, a feedback screen is displayed that provides the opportunity to either log rather objective feedback (e.g. video data about their current context) or rather subjective feedback (e.g. textual comments).

To enable the logging of the location context during the conduction of an empirical evaluation in the real world setting, the different points of interest need to be specified (e.g. the location of the canteen). Further, all other aspects have to be specified that need to be logged, such as the different keyboard-based mobile interactions. Section 4.2 goes into the details how to perform this specification task. As a particular new aspect when specifying the described mobile diary application, it needs to be highlighted that the application's pervasive computing environments are not places as with the former mentioned examples but instead people. Thus, the database of the application needs to be specified as followed: a pervasive computing environment per community as well as a physical object per community member.

A possible conduction of a real world evaluation: As a possible conduction of the evaluation in the real world setting, the test users might have the main task to use the application for a certain period of time (e.g. one hour) in order to log interesting occurrences, such as comments on the today's canteen food. As a further secondary task, the test users are also asked to provide feedback about problems and their context when using the application.

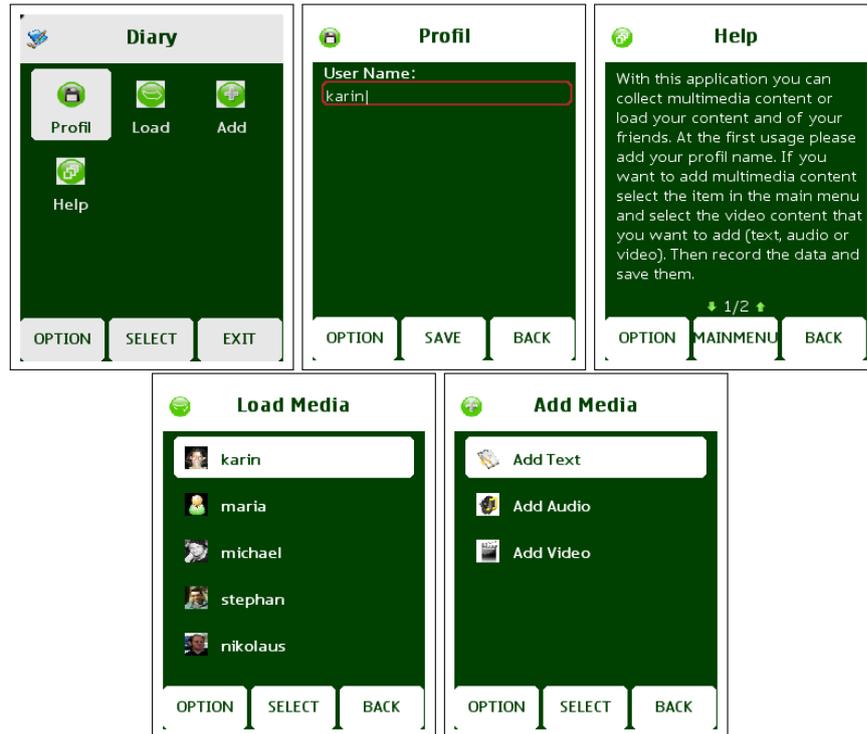


Figure 6.1: Screens of the prototype that was developed with MoPeDT. The first screen is the main menu followed by the profile and help screen. The last two screens can either be used to load the stored content of a user and his community or to add new multimedia content.

While performing these tasks, MoPeDT logs all of the user's generated data as well as all the performed mobile interactions, the appearance of the mobile application - the screens of the application - and the entrance at point of interests. The evaluator can actively observe the conduction of the evaluation by getting just-in time knowledge about the occurrence of an event. Further, he or she can directly log comments, such as the hint to check the sequence later on during the analysis phase. An additional audio-visual capturing during the conduction of the evaluation in a real world setting is difficult and typically not applied since stationary hardware cannot be used due to the mobility of the user. The use of mobile hardware (e.g. a backpack with camera and microphone) is theoretically applicable but practically often too distracting for the users from their actual tasks as well as physically too overloading.

A possible analysis of the data: After having conducted the evaluation in the real world setting, the evaluator can analyse the logged data. By means of MoPeDT's analysis component, the time-line based visualisation of the recorded data can help find interesting occurrences, such as the loggings of user feedback. Having knowledge about these moments can help review the corresponding se-

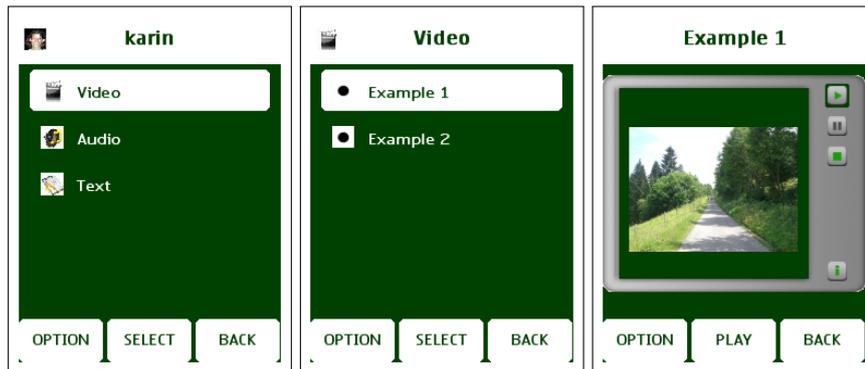


Figure 6.2: Screens of the prototype to load multimedia content of a specific user. In this case a video content is loaded.



Figure 6.3: Screens of the prototype to store new multimedia content. In this case a new video content is stored.

quences and thereby get knowledge about usability problems and the context of use.

6.1.2 A Discussion about Real World Evaluations

All in all, the tool-supported conduction of real world evaluations in the field has different benefits and problems. As already mentioned, the main benefit is the ability to provide a comprehensive overview about a realistic use of the application. Due to the automatic logging of events and their synchronisation by means of the tool, the test users and the evaluators can concentrate on the goal of the evaluation itself, such as the completion of the instructed tasks for the users and the observation of the evaluation for the evaluators.

The problems of tool-supported real world evaluations mainly emerge due to the spatial separation of the evaluator and the users when conducting remote evaluations. As one consequence, an audio-visual capturing of the users and their environments is difficult to perform as described earlier. A solution is to ask the

user for the documentation of problems and the context of use. This solution is event-driven by the user since the user decides on the right moment to report on a situation (e.g. see *the Cultural Probes Method* in Section 2.1). During these user reports, a tool can provide assistance by supporting options to automatically log all user data for the later on analysis. A problem of using user protocols is that the users might forget the capturing of data. Then, important knowledge can be lost for the analysis phase.

That lost of knowledge also can happen when using a protocol recording method that is not event-driven by the user but by the evaluator, such as *the Experience Sampling Method* (see Section 2.1). Even then, interesting data might be missed because moments to log data might be inappropriate, such as in terms of time and location. As a solution for that problem, not the users or the evaluators do decide on moments to log data but instead the system as part of the application flow by automatically logging data at certain times, such as whenever the user is interacting with the system. But even by doing this, the system still might miss capturing interesting data due to the fact that, for instance, the camera of the phone might provide a wrong perspective which does not correctly describe the context of use.

In some evaluations a continuous audio-visual monitoring, however, would be indispensable which requires the conduction of a local evaluation with the presence of the evaluators in the field and their direct observation of the users. These direct observations in the field, however, can lead to disturbances of the users and thus changed behaviour if the evaluators, for example, always have to follow the users in order to completely track them. Also due to the fact of the presence of the evaluator in the field, the evaluations can become very time-consuming to conduct and complex to organise. Finally, long-term studies are difficult to conduct if the evaluator always has to be present in the field. The developers of tools have to take these aspects under consideration. The tool should make sure to provide both local and remote evaluations for direct and indirect observations in the field. For these different settings but also for fulfilling the needs of the test users and the application domain, there is a need to find appropriate tool-supported observation methods [Zhang and Adipat, 2005], such as protocol recordings performed by the user or automatically by system.

Besides the mentioned problems, there is a further general problem when just conducting evaluations in the field. This kind of evaluation might lead to uncontrolled contextual settings rendering the outcome useless (e.g. [Zhang and Adipat, 2005]). Thus, experimental evaluations (e.g. to compare two different interaction styles) rather should be conducted in controlled settings than in the field. The idea is to simulate the real setting in a more controlled environment and execute laboratory studies instead.

6.2 A Real World Simulation

A review of [Kjeldskov and Graham, 2003] showed that 71% of the empirical evaluations of mobile applications are conducted as real world simulations in laboratory settings. The question, however, is whether laboratory studies can substitute real world evaluation in field settings.

Several comparisons of real world evaluations in the field and their simulations in the laboratory have aimed at answering this question (e.g. [Kaikkonen et al., 2005, Duh et al., 2006, Kjeldskov and Stage, 2004, Kjeldskov et al., 2005]). For instance, [Kjeldskov and Stage, 2004] investigated empirical evaluations for mobile applications in the field and in the laboratory. They found some differences (e.g. the social comfort of use) but pointed out most basic usability problems as similar. Other works such as described by [Kaikkonen et al., 2005] validated these results. They revealed laboratory studies as sufficient in some cases since field studies do often not provide an added value.

Nevertheless, research also showed that real world evaluations cannot be completely substituted by real world simulations (e.g. [Zhang and Adipat, 2005]) and should be used at least at the end of the development process to investigate specific user behaviour in different contextual settings. Real world simulations in laboratory settings, however, can be used during minor iterations of the user-centred prototyping process if appropriate evaluation methods and techniques are applied.

6.2.1 An Example of a Real World Simulation

To illustrate a tool-supported real world simulation by means of MoPeDT, an exemplary mobile application should be implemented that can be used for mobile payment. Concretely, the mobile application should first provide details about products in a store (e.g. about DVDs) and later on the mobile application should be usable as a kind of wallet.

A possible objective of the real world simulation: For such an application, the main objective of a user evaluation in a laboratory setting might be to reveal whether different mobile interaction styles (e.g. the keyboard-based or the NFC-based) can increase the initial trust [McKnight et al., 1998] of the user in the mobile application or not. Concretely, based on expectations of the results from [Rukzio et al., 2006], the following assumption can be validated by means of an evaluation in a setting of a real world simulation: the NFC-based mobile interaction style is more trustworthy compared to the keyboard-based mobile interaction style.



Figure 6.4: Screens of the prototype that was developed with MoPeDT. The user can apply the application in order to get knowledge about a DVD and later on pay it.

A possible implementation: MoPeDT was used to design and generate a prototype for the Nokia 6131 NFC that supports the NFC-based and keyboard-based mobile interaction styles for mobile payment. Figure 6.4 shows some screens of the generated prototype.

When using the keyboard-based mobile interaction style, the user first selects the list of all DVDs and then the intended DVD (see Figure 6.5). Now, a list of all available services is loaded and displayed, such as the service to display information about the plot, the actors or critics. After the selection of a service, the corresponding content is loaded and displayed. By this means, the user can also request the price of the DVD and afterwards pay the DVD by using the mobile phone as a wallet.

For the NFC-based mobile interaction style, the user selects a DVD by simply touching its RFID tag (see Figure 6.5). Now, the services of the intended DVD are automatically loaded and displayed. In contrast to the keyboard-based mobile interaction style, the user does not need to navigate through the first screens and directly reaches the service screen.

The specification of such an application is described in Section 4.2. The wallet functionality was simulated by adding a simple constant that provided details



Figure 6.5: A user while interacting with the keyboard-based (left) or NFC-based (right) mobile interaction style.



Figure 6.6: The setting of the real world simulation. The test user sits in front of the DVDs (left) while the evaluator controls MoPeDT's evaluation component (right).

about the content of the wallet. This value also could be changed at the runtime, such as when the user has decided to buy a DVD.

A possible conduction of the real world simulation: Figure 6.6 shows the setting of the exemplary real world simulation. The test users can sit in front of a table. On this table, three DVDs can be placed which are augmented with RFID tags. A camera can capture the frontal view of the user while a second camera can record the user and the table with the DVDs.

During the user evaluation, test users can be instructed to execute predefined tasks. First, they might have the task to select a DVD by using the NFC-based or the keyboard-based mobile interaction style. Then, they might need to complete the task to read the plot and finally buy the DVD by making use of the corresponding service.

While completing the tasks, the cameras and microphones can be used to audiovisually capture the evaluation. If the user's thoughts and ideas should be cap-

tured as well, the test users have to verbally express their thoughts by applying *the Method of Thinking Aloud* (e.g. [Nielsen, 1993]). Besides the audio-visual capturing, the mobile interactions and the prototype's appearance can also be recorded similar to an evaluation in the field. Further, a task's start and end time can be logged as well as comments of the evaluator.

A possible analysis of the data: During the analysis phase, the captured data can be reviewed whether the user have a different behaviour when using a mobile interaction style compared to the other. The users, for instance, might look more sceptical when using a system if it is meant to be untrustworthy. Besides this aspect, the user's verbal expression also might be analysed, such as whether there are any comments that provide insights to the trustworthiness of a mobile interaction style.

The prototypical application and the conduction and analysis of the evaluation were respectively generated and conducted in the described order with the addition of the use of a questionnaire [Leichtenstern et al., 2011a]. The results did not show any significant differences neither for the results of the questionnaire nor for the analysis of the captured data. After the evaluation, two test users mentioned that only the appearance and the usability of the application would change their feelings of trustworthiness in the application but not the used mobile interaction style.

6.2.2 A Discussion about Real World Simulations

The main benefit of the tool-supported conduction of an empirical evaluation in the laboratory is the opportunity to control context in order to run a controlled experiment as with the introduced example. Having stronger control about the evaluation process also enables the conduction of evaluations with prototypical applications which strongly lack in level of detail and functionality since the evaluator can guide through running aspects of the application. In this regards, laboratory settings are of special interest when evaluating prototypes in order to improve the task performance even though other aspects of the application are still missing. This is hardly possible in real world evaluations where the evaluator has few control especially when being spatially separated to the test users.

In addition to a tool-support for a real world evaluation in the field, a UCP tool in a laboratory setting can also support options to log data regarding the tasks (e.g. their start time and end time). Also, audio-visual data can be recorded without a break and a lost of knowledge during the conduction of a real world simulation in the laboratory.

A main problem of real world simulations is that, as mentioned, they only can partly provide insights to a realistic use of the application. They therefore cannot replace the conduction of evaluations in the field. A further problem of real world simulations is the need to physically rebuild the real setting which can be time-consuming and expensive. Further, it might be difficult to quickly and easily change aspects of the setting. Finally, due to the fact that the laboratory is commonly not located close to the test users but instead to the evaluator, it is often difficult to recruit test users for a real world simulation in a laboratory. The question is whether there is a further option that can compensate some of the problems of real world simulations.

6.3 A Virtual World Simulation

One idea is to apply virtual world simulations of a real world during the conduction of the user-centred prototyping process. Simulations with virtual worlds can improve the development process [Lee et al., 2004] because they can mediate ideas of new products and support first user evaluations. In the context of *the Third Computing Paradigm*, a literature review showed a tendency to apply virtual world simulations in order to investigate the pervasive computing environment itself and its performance [Sato, 2001, Reynolds et al., 2006].

The use of virtual world simulations for the investigation of the users and their system use have not been focused so far. The projects that do address this aspect (e.g. [Jo et al., 2007, Manninen, 2000, Sato, 2001, Barton and Vijayaraghavan, 2002]) mostly used virtual world simulations which directly included the use of the interaction device into the virtual world. Mobile devices are not longer physical available for interactions anymore. Instead, they are just virtually represented and have to be controlled by means of a keyboard or mouse which leads to a disruption of the real use of the device. For example, [Manninen, 2000] used virtual representations of interaction devices in his setting. His main objective was to easily develop and test different virtual worlds and their input devices.

[Barton and Vijayaraghavan, 2002] were also interested in similar objectives. They developed UbiWise that is a simulation tool for *Ubiquitous Computing* applications. This simulation tool helps investigate applications that use cameras or mobile phones as interaction devices. As the devices are only represented in a 3D virtual world, the user has to interact with the traditional input methods (e.g. the mouse) which means a marginal mapping to the real use of the device and the application. Certainly, there is a need to consider the level of immersion that can often not be met by an only virtual world simulation approach.

To solve the problem of the insufficient mapping of a virtual world simulation, another approach of a simulation - the hybrid world simulation - was researched [Leichtenstern et al., 2010, Leichtenstern et al., 2011a] which involves at least parts of the real world.

6.3.1 The Concept of Hybrid World Simulations

[Morla and Davies, 2004] give a first impression of a hybrid world simulation. A hybrid world simulation means an integration and combination of the real and the virtual world. Morla and Davies used this kind of simulation to virtually test the performance of real sensors which were attached to a wearable medical monitoring system. A similar approach called dual reality is introduced by [Lifton et al., 2009]. They used Second Life¹ as a virtual visualisation tool of streams which were generated from real world sensors (e.g. temperature sensors). Driving simulators² also aim at the idea of the hybrid world simulation. The users interact with a real steering wheel and dashboard while they are driving through a virtually presented test route.

When using hybrid world simulations for user-centred prototyping tools, the virtual world is not applied as a visualisation platform for the performance of real devices as described by Morla and Davies or Lifton et al. but instead the virtual world is applied as a platform to execute user evaluations similarly as used for driving simulators. For the application domain of the *Third Computing Paradigm*, real mobile phones are utilised as interaction and presentation devices to a virtualised simulation of the pervasive computing environment.

A similar idea is also introduced by [Haesen et al., 2009]. They used a virtual simulation of a museum to execute a user evaluation where real mobile phones were applied to interact with the virtualised museum. Haesen et al. consider the concept of the hybrid world simulation as a promising new evaluation method in early stages of the user-centred prototyping process. However, they do not provide insights to potential benefits and problems of the hybrid world simulation, in particular when using the approach as evaluation method for user-centred prototyping tools. Additionally, Haesen et al. did not use a well-known virtual platform for the virtual simulation of the pervasive computing environment such as Second Life in order to attract a large number of users remotely.

¹<http://secondlife.com/>

²<http://www.carrsq.qut.edu.au/simulator/>

6.3.2 Tool-supported Hybrid World Simulations

MoPeDT can be used to generate prototypical mobile applications that should be evaluated with a hybrid world simulation but some adaptations are required for the architecture (see Figure 6.7). The main difference to the general architecture of MoPeDT is the partly shift of the pervasive computing environment from the real world to the virtual world by using a platform for a virtual world simulation. This simulation contains the virtual representation of all physical objects. All information about the physical objects is still stored in the database and can be accessed by the users with their real mobile phones. Thus, the user still makes use of a real mobile phone as an interface to the pervasive computing environment even though the physical objects are not longer physically present.

Another difference to the general setting is the need for a representation of the user in the virtual world. With this avatar the user can interact by means of the PC's keyboard within the virtualised pervasive computing environment, such as by moving around to get closer to physical objects. Sensors and actuators can be set up virtually and/or really.

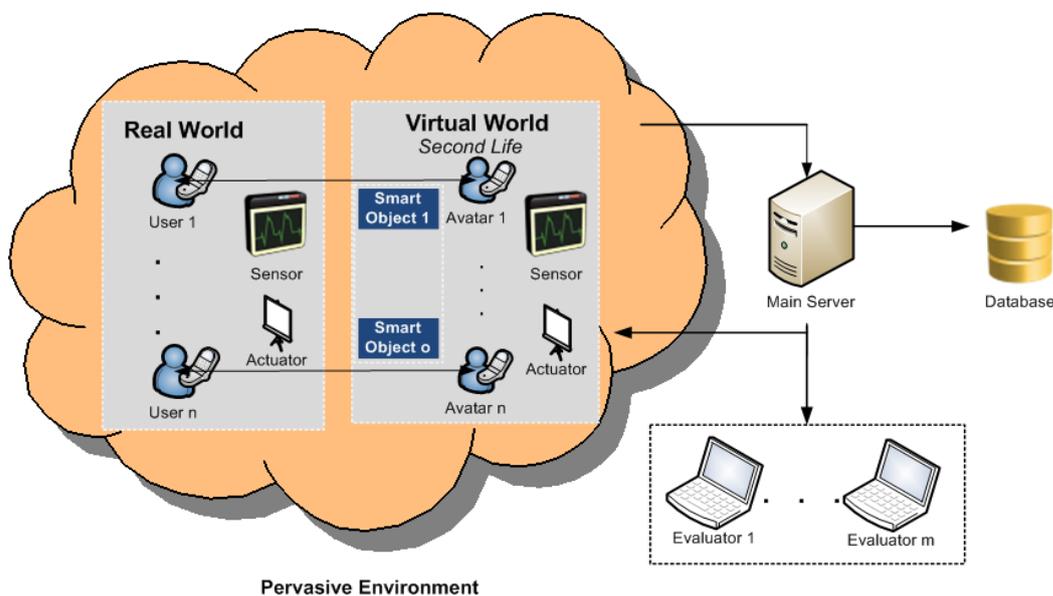


Figure 6.7: The extended software architecture of MoPeDT consists of a main server, a database as well as evaluators alike the general architecture of MoPeDT. The mobile users have a virtual representation: the avatars. The physical objects are virtualised. Sensors and actuators can be virtually and/or really.

The virtual simulation of the real world: To simulate the pervasive computing environment, an open source version of Second Life can be used which is called

Open Simulator³. Open Simulator allows setting up one's own virtual world simulation that behaves exactly like Second Life and can be accessed with the same viewers. Thus, in the remainder of this dissertation Second Life and Open Simulator are used as synonyms.

Second Life represents one of the first massive 3D multi-player platforms which is not primarily concerned with gaming but aims at establishing a general virtual meeting place. Thus, every conceivable type of interaction is in principle possible, be it buying or selling virtual or real goods, be it playing out as a real DJ in a virtual club or be it mobile interactions in a pervasive computing environment. Central feature of Second Life is the use of avatars which represent the real user in the virtual environment. [Pallay et al., 2009] showed that Second Life can serve as an virtual evaluation platform for multi-agent systems involving the user in her natural environment. Therefore, it is proposed to employ Second Life to simulate a real world setting which, however, needs to be augmented for context dependent interactions.

Apart from setting up the simulation server, three steps are necessary for simulating a pervasive computing environment in a hybrid world simulation. The environment itself has to be modelled, it has to be equipped with physical objects and sensors and it has to allow for communicating with the outside world such as the real mobile phone.

An example of setting up a hybrid world simulation: The most basic requirement is a virtual representation of the real world in which the evaluation should take place. To this end, standard modelling tools can be employed making it necessary to import the resulting models in Second Life or in-world modelling tools can be used that supply basic functionalities.

Realising a virtual pervasive computing environment: The challenge of a hybrid world simulation is to realise the complex interplay between sensors, physical objects and the mobile phone which can be seen as the inherent characteristic of a pervasive computing environment. The general idea is to use the real mobile phone for the interaction with the virtual world. This is not always possible.

In the following example of a hybrid world simulation, virtual physical objects were equipped with virtual RFID tags to allow NFC-based mobile interactions with the mobile phone. Creating a virtual RFID tag is no challenge but of course this tag cannot be read out by the real mobile phone. Thus, it is necessary to create a virtual representation of the mobile phone for some of the mobile interaction

³<http://opensimulator.org>

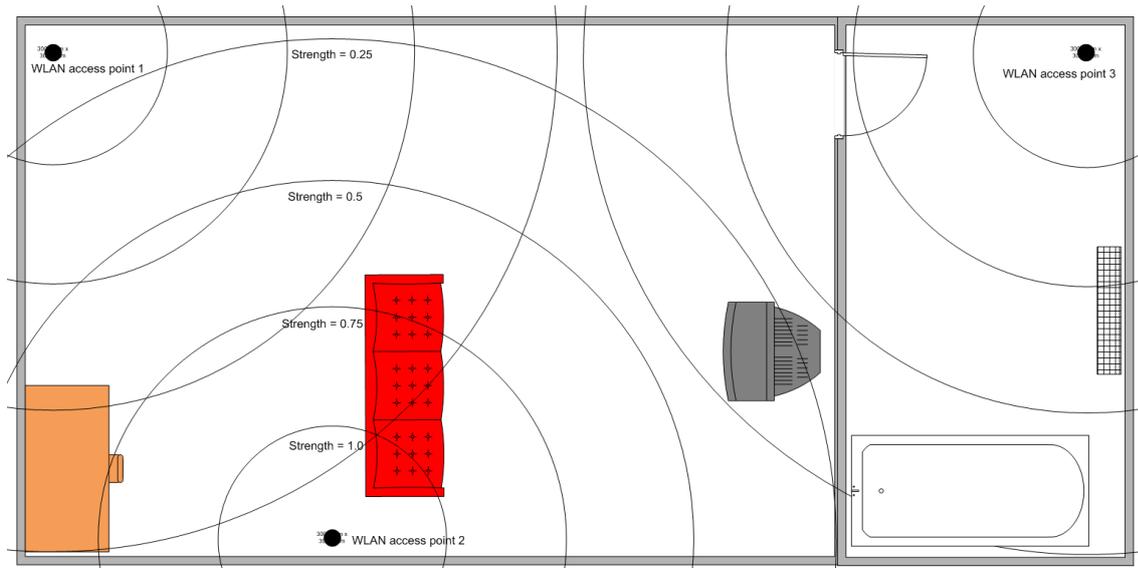


Figure 6.8: Degression model for signal strength of Wifi access points. Example for access point 2.

styles. The avatars and their virtual mobile phones are used for registering virtually performed mobile interactions which are provided by the simulated world. Later on, the real mobile phone handles the output of the virtually triggered mobile interactions.

Apart from the virtual representation of the mobile phone, virtual physical objects as well as virtual sensors also have to be realised. Virtualised physical objects trigger contexts of mobile interactions whereas virtualised sensors trigger contexts caused by changes in the virtual world. To create virtualised physical objects for the user evaluation that is described in Section 6.3.3, home appliances (e.g. a TV and DVD player) were augmented with virtual RFID tags allowing for NFC and with virtual IR sensors to register remote activation.

Not only physical objects can be realised in the virtual world but also sensors. A virtual temperature sensor can be realised to register the effects of manipulating the heater. To this end, a model of the heater and its radiation can be realised. To allow for indoor localisation of the user, Wifi access points can be installed in the virtual world as well. By measuring the signal strength, the user's location can be approximated. Again, a model for the radiation needs to be integrated. In both cases, a simple linear degression can be chosen as a suitable model (see Figure 6.8).

Realising the communication between the real and virtual world: Two types of communication take place in the virtual world. Virtualised sensors di-

rectly communicate with the main server of the MoPeDT architecture. They can connect to the main server similar to real sensors and send context messages - XML messages - about the triggered contexts (e.g. `TEMPERATURE.HOT`). See Section 4.1.1 for more details about context messages from the sensors.

In contrast to virtual sensors, virtualised physical objects communicate with the virtual representation of the mobile phone which, in turn, communicates with the main server (see Figure 6.7). The main server forwards the incoming contexts to the real mobile phone. The real mobile phone only needs to register the interest in context of the virtual mobile phone. See Section 4.1.2 for more information about the registration of interest in other real or virtual mobile users - the avatars.

In order to trigger virtual contexts, the home appliances were augmented with RFID tags and IR sensors. To read out an RFID tag, the user has to move her avatar towards the virtually presented physical object. The avatar is holding the virtual representation of the mobile phone. The phone serves as a virtual RFID reader simulating NFC. Thus, if the RFID tag is in a certain range - less than 30 cm - of the mobile phone, its ID is registered by the virtual mobile phone which sends a socket request to the main server containing its own ID and the physical object's ID. The IR sensor allows remote activation of an object's context. To achieve this goal, the user has to point the virtual mobile phone in the direction of the physical object (see Figure 6.12). Having been activated, the sensor sends the object's ID by means of virtual Bluetooth to the virtual mobile phone which, in turn, sends it to the main server with a socket request.

6.3.3 Studying the Concept of Hybrid World Simulation

[Morla and Davies, 2004] describe several requirements of a hybrid world simulation (e.g. the application of the device's intrinsic code) which is considered as fulfilled when combining the features of MoPeDT with Second Life.

(1) Using MoPeDT the intrinsic code of the mobile phone application can be uploaded to the physical device which (2) enables live user interactions on real mobile phones. At the same time, (3) contexts can be generated by means of the users and their real and virtual mobile phones as well as grounded on the connected real or virtual sensors. (4) All conductions of hybrid world simulations can be logged and reproduced with MoPeDT's architecture and the supported evaluation component.

The design of a hybrid world evaluation: Even though meeting the requirements of Morla and Davies, the results of a user evaluation performed in a traditional laboratory setting - a real world simulation - should be compared with the

results achieved by using the hybrid world simulation. This comparison should help get more insights to benefits and problems of using the hybrid world simulation for early user evaluations. To this end, the scenario of the referenced real world simulation was implemented as a hybrid world simulation.

The referenced real world simulation: The referenced evaluation was executed as a traditional real world simulation of a smart environment in a laboratory setting [Rukzio et al., 2006]. The main objective of this evaluation was finding out whether users apply different mobile interaction styles dependent on contextual conditions in a smart environment.

In the setting, the smart environment contained several physical objects (e.g. a TV or a heater) which could be addressed and controlled with a mobile phone. For example, the mobile phone could be applied as a remote control to change the status of the heater by switching it on or off or by changing its temperature. In the context of the referenced real world simulation, the use of the following mobile interaction styles should be investigated: NFC-based, laser beam-based and keyboard-based. When using the NFC-based or laser beam-based mobile interaction style the user had to physically touch or point at the intended physical object with the mobile phone in order to address it. With the keyboard-based mobile interaction style, the user could address a physical object by using the mobile phone's graphical user interface and select the intended physical object out of the detected and graphically listed physical objects.

The referenced real world simulation was performed with 20 people in a living room of a smart environment. All participants were sitting on a couch while they had to solve four different tasks in order to call a service of the intended physical object under different context conditions. (1) First, the user had line of sight to the physical object. The distance to the physical object was about three meters. (2) For the second task, the users were in front of the physical object. The distance to the physical object was about ten centimetres. (3) For the third task, the user did not have line of sight to the physical object. The physical object was located in another room and the distance was about 20 meters. (4) Finally, the user did not have a line of sight to the physical object. The physical object was located in the same room. The distance to the physical object was about four meters. To get a line of sight to the physical object, the user had to move about one meter. To cover most casual activities, users provided information afterwards about their behaviour and preferences when lying or standing.

The results of the referenced real world simulation led to the following three findings. (1) Users tend to switch to a specific mobile interaction style dependent on location, activity and motivation. (2) The current location of the user is the most

important criterion for the selection of a mobile interaction style. (3) The user's motivation to make any physical effort is generally low.

The experimental design towards the hybrid world simulation: The main objective of the experiment was finding out whether hybrid world simulations of pervasive computing environments can potentially be used as an evaluation setting in the tool-supported user-centred prototyping process. To get a first indicator, the findings from the referenced real world simulation were used to formulate the following hypotheses.

- **H-1:** Similar to the referenced real world simulation, the users also tend to switch their mobile interaction styles based on their contextual situations when conducting a hybrid world simulation.
- **H-2:** Similar to the referenced real world simulation, location is also the most important contextual criteria for selecting a mobile interaction style when conducting a hybrid world simulation.
- **H-3:** Similar to the referenced real world simulation, the user's motivation to make any physical effort is also generally low when conducting a hybrid world simulation.

In order to investigate the hypotheses, independent and dependent variables were defined. The independent variables are location and activity with the different levels of the referenced real world simulation. As dependent variables, the user's preference for a mobile interaction style were analysed in the different settings of the independent variables. The same questionnaire as for the referenced real world simulation was used to ask the test users for the preference in the different settings. In summary, the experimental design is an exact replication of the referenced real world simulation.

The implementation of the test setting: The referenced real world simulation constitutes the benchmark for performing a similar test, this time making use of a hybrid world simulation. Thus, the living room and the bathroom were modelled as well as the required physical objects. Figure 6.9 shows the perspective of the avatar when sitting on the couch. In front of the avatar is the DVD player within line of sight. To the left of the avatar is the radio within touching distance.

The idea is to select the different physical objects by using one of the three mobile interaction styles. Once the user has selected one of the physical objects, the respective services are displayed on the real mobile phone and the user can select one of them by using the mobile phone's graphical user interface (see Figure



Figure 6.9: The virtual world simulation of the living room in Second Life.

6.10). In the following, the implementation of the three different mobile interaction styles is described.

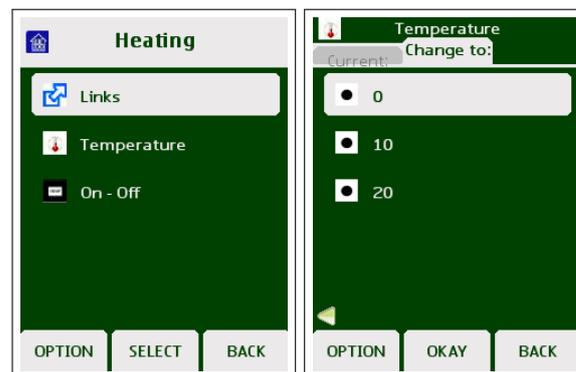


Figure 6.10: Once the user has selected the physical object by one of the mobile interaction styles, the service entries for the selected object are displayed that then can be requested and changed.

MoPeDT was applied to generate the application for the real mobile phone that supports the keyboard-based mobile interaction style as well as the processing of incoming context from the virtual mobile phone. The keyboard-based mobile interaction style is completely realised on the real mobile phone (see Figure 6.11) and therefore no adaptation in the virtual world is required. When using the keyboard-based mobile interaction style, the user navigates through different screens and finally selects the intended physical object in order to use a service for this object. Thus, the keyboard-based mobile interaction style is quite similar to the referenced real world simulation.

In contrast to the keyboard-based mobile interaction style, the laser beam-based



Figure 6.11: Screens which are displayed when performing the keyboard-based mobile interaction style. The user first selects the environment and then the physical object.

mobile interaction style requires a direct interaction with the physical objects. Figure 6.12 shows the use of the laser beam-based mobile interaction style in a hybrid world simulation. The user applies the virtual mobile phone to point at a virtualised physical object in order to perform the selection. The virtual phone can be positioned by pressing the four navigation keys. By hitting the 'PgUp' key, an IR beam is emitted that is registered by the IR sensor of the virtual physical object. The information about the object's ID is then transmitted to the main server of MoPeDT's architecture that forwards this context to the real mobile phone and the application running on it. Now, the real mobile phone loads the services of the selected object and displays them on the real phone (see Figure 6.10). Section 4.2 describes how to specify the mobile application including the mobile interaction styles by means of MoPeDT. Since MoPeDT does not support the real laser beam-based mobile interaction style, a tool user can specify the NFC-based interaction style for the real phone instead. The virtual world only has to send the corresponding context messages - NFC context - to enable the communication between the real and virtual mobile phones.

The NFC-based mobile interaction style is realised by getting very close to the physical object in the virtual world and touching it with the virtual mobile phone (see Figure 6.13). Once the user has touched the physical object, a script sends the identifier to MoPeDT's main server and the real mobile phone as described for the laser beam-based mobile interaction style.

The conduction of the hybrid world evaluation: Before the experiment was started, each participant of the hybrid world simulation was introduced to the correct use of the mobile phone and the three mobile interaction styles. Moreover, the Second Life simulation was introduced. Prior to the conduction of the evaluation, the test users could freely exercise each mobile interaction style and



Figure 6.12: Performing the laser beam-based mobile interaction style. The user applies the avatar and the virtual mobile phone to point at the virtual physical object in order to select it.

the use of the Second Life environment.

The evaluation was executed with 20 test users aged 23 to 32 with an average age of 27.25. The Second Life environment ran on an ordinary computer that required no special hardware capabilities. The participants of the evaluation could navigate through the virtual world simulation using the avatar to trigger the different contexts of the laser beam-based and NFC-based mobile interaction styles. The mobile application ran on a Nokia 6131 NFC that could be used to perform the keyboard-based mobile interaction style as well as the interactions with the service entries.

After the explanation of the mobile interaction styles and the virtual test setting, the avatar was sat on the couch in Second Life. This was always the starting position for each task. Now, the participants of the evaluation had to complete the four tasks described earlier. After each task, the test users were asked about their attitude if the avatar would staying beside the couch or lying on the couch instead of sitting. Therefore, the test users had to fill out a questionnaire that addressed the different situations.

The comparative analysis of the hybrid and real world simulation: For the hybrid world simulation and the referenced real world simulation, location could be identified as the crucial contextual factor for the decision of a mobile interaction style. An ANOVA test revealed these differences in location to be highly significant for the referenced real world simulation (NFC-based: $F = 19.225$, $p < 0.01$, laser beam-based: $F = 123.36$, $p < 0.01$, keyboard-based: $F = 10.769$, $p < 0.01$).

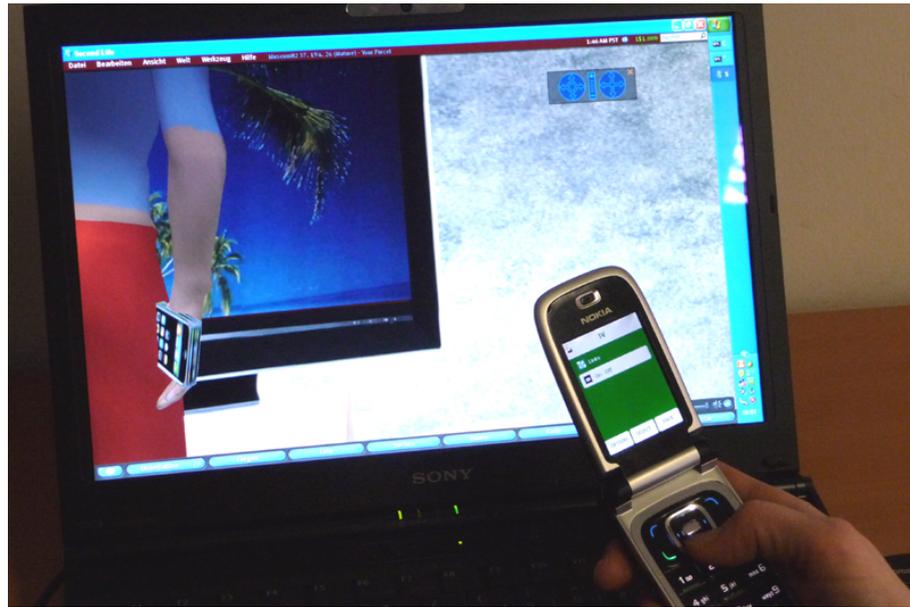


Figure 6.13: Performing the NFC-based mobile interaction style. The user applies the avatar and the virtual mobile phone to get very close to the virtual physical object in order to touch and select it.

A similar result was obtained with the hybrid world simulation. Again, location was the crucial contextual factor that dominated the choice of the mobile interaction style with significant differences depending on the location (NFC-based: $F = 12.013$, $p < 0.01$, laser beam-based: $F = 39.2$, $p < 0.01$, keyboard-based: $F = 9.604$, $p < 0.01$). Table 6.1 summarises the statistical results for the mobile interaction styles and the contextual factors location and activity.

No effect was found for the activity. It did not matter if the user was sitting, standing or lying. A post-hoc test revealed the dependencies between the locations and the mobile interaction styles. The NFC-based mobile interaction style was preferred in scenario 2, where the user was close to the desired object. The laser beam-based mobile interaction style was preferred in scenario 1, where the user was around 3 meters from the object but the object was in her line of sight. The keyboard-based mobile interaction style, at last, was clearly preferred if the object was in another room. There was a tendency to also prefer the keyboard-based mobile interaction style if the object was in the same room but not in the line of sight (scenario 4), but this preference was not significant. Scenario 4 revealed that the chosen mobile interaction style might be dependent on the activity but the results were not conclusive in either evaluation.

Discussing the results of the hybrid world evaluation: The findings from the hybrid world simulation are comparable to the referenced real world simulation. They provided evidence for the first hypothesis: *Similar to the referenced real world*

		NFC-based	Laser beam-based	Keyboard-based
Real	Location	19.225**	123.360**	10.769**
Simulation	Activity	0.141	0.024	0.235
Hybrid	Location	12.013**	39.200**	9.604**
Simulation	Activity	0.683	0.001	0.603

Table 6.1: Results of the referenced real world simulation (first row) and the hybrid world simulation (second row) (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).

simulation, the users also tend to switch their mobile interaction style based on their contextual situations when conducting a hybrid world simulation. None of the participants used the same mobile interaction style in each task. Each of the participants assessed the situation and balanced reasons which mobile interaction style would fit best to which context.

The second hypotheses can also be corroborated: *Similar to the referenced real world simulation, location is also the most important contextual criteria for selecting a mobile interaction style when conducting a hybrid world simulation.* The execution of the user test with the hybrid world simulation led to the result that location is the most important context factor which influences the decision for a mobile interaction style. In all four tasks the users tended to use the mobile interaction style dependent on the location of the avatar and the physical objects. If the NFC-based or the laser beam-based mobile interaction style was possible, they preferred these mobile interaction styles. If there was no line of sight, the test users tended to switch to the keyboard-based mobile interaction style in order to prevent movements of the avatar.

The third hypotheses could also be partly proved: *Similar to the referenced real world simulation, the user's motivation to make any physical effort is also generally low when conducting a hybrid world simulation.* The test user's motivation to spend physical effort was almost as low as in the referenced real world simulation. But, in the hybrid world simulation more test users were willing to move the avatar in Second Life for performing the NFC-based or the laser beam-based mobile interaction style, however, this difference is not statistically significant. A higher test user's motivation to spend physical effort is not completely surprising when using the hybrid world simulation because the test users did not have to actually move themselves but just navigate their avatar through the environment which is not comparable in effort to the real setting.

6.3.4 A Discussion about Hybrid World Simulations

The comparison of results from a real world simulation and a hybrid world simulation showed that very similar knowledge can be gained about the user's behaviour. Consequently, a first indicator points to the assumption that the hybrid world simulation seems to be an appropriate test setting for early stages of the user-centred prototyping process. Detailed benefits and problems of the setting, however, must also be addressed in order to get a deeper insight whether the hybrid world simulation is really meaningful. The now described advantages and disadvantages mainly base on the gained practical experience by the execution of a user evaluation as a hybrid world simulation. In some points the hybrid world simulation benefits compared to a traditional laboratory setting and an only virtual simulation approach.

(1) Compared to a traditionally laboratory setting, there is no need to physically rebuild the user's real world in a laboratory anymore. Thus, the designers can save money and time. (2) Relying on the hybrid world simulation, even initial ideas of pervasive interfaces for mobile phones can easily and efficiently be mediated and investigated because the mobile application can be tried out and demonstrated in the corresponding simulated pervasive computing environment. (3) Another benefit is the ease of changing the setting. Different models of physical objects can rapidly be generated, modified and deleted. Thus, different settings of a pervasive computing environment can be arranged and tested in user evaluations.

Using Second Life as virtual world platform adds further advantages. (4) Due to its widespread use, it is known to a great number of users who do not have to be introduced to the specifics of using the virtual world simulation. (5) A further advantage is the mobility of the test setting. Because the application realises a multi-player platform over the Internet, it can be accessed anywhere anytime. Thus, user evaluations can be run outside the laboratory in the user's familiar surroundings. Consequently, Second Life enables the execution of remote user evaluations. (6) Executing remote evaluations at the test user's home can also reduce the organisational effort of test user recruiting since the users do not need to be physical present anymore in the laboratory which is similar to the execution of online surveys. Consequently, empirical evaluations can quickly and easily be conducted with a large number of participants. Second Life has attracted a large number of users. These are potential test users for investigating interaction concepts. Of course some restrictions apply like the necessity of compatible mobile phones. (7) Finally, in contrast to an only virtual world simulation approach, the hybrid world simulation also arise the benefits that the evaluation can be performed more similar to the real world setting. The users can directly interact with

the real mobile phone which can increase the level of immersion.

Despite these promising benefits, there are also problems. Of course, there inevitably is an offset between a real world setting and a hybrid world simulation. (1) The user moves virtually instead of physically which means a break because the user requires few motivation and few physical effort to move and explore the virtual setting. (2) A further problem of the hybrid world simulation is the level of immersion of the mobile interaction styles. The keyboard-based mobile interaction style is easy to evaluate with a hybrid world simulation because it is completely realised on the mobile phone and therefore quite similar to the real use but other mobile interaction styles, such as the laser beam-based and the NFC-based mobile interaction styles lead to a further break because they inevitably require interactions with the pervasive computing environment. The implementation of the laser beam-based mobile interaction style is fully realised in the virtual world but instead should preferably be realised in the real world to reduce breaks to the real use of the phone. The idea came up to replace the current implementation of the laser beam-based mobile interaction style and instead use the accelerometer of the mobile phone to point towards the screen of the virtual world simulation for selections of the objects, such as the DVD player.

(3) Having too many mobile interaction styles in the virtual simulation also leads to a problem in terms of usability. Sometimes test users had problems to perform the laser beam-based mobile interaction style in the setting of the example evaluation because it required the knowledge of key sequences. (4) A last problem is to generate the pervasive computing environment and thus the virtual world as realistic as the real world. Developers require appropriate skills to virtually model the pervasive computing environment and set up the whole system to run a hybrid world simulation.

6.4 Summary

This chapter presented and discussed different settings for the tool-supported conduction of empirical evaluations. Evaluations in the real world are inevitable to get knowledge about the real context of use. These field studies, however, are difficult to perform particularly at early phases of the user-centred prototyping process if the application is limited in level of detail and functionality. A user-centred prototyping tool (UCP) tool should therefore not only support empirical evaluations in the field but also real world simulations in laboratory settings. Then, rather controlled experiments can be conducted by means of a UCP tool even with strongly limited prototypical applications.

Besides these two types of tool-supported evaluations, a further test setting - the hybrid world simulation - was presented and discussed in the chapter as well. It is a special type of a virtual world simulation since only the pervasive computing environment and its physical objects are virtually. The interaction devices are still physically available. Thus, the user can more realistically interact with the real device and its typical constraints.

The chapter presented the execution of a hybrid world simulation with the support of MoPeDT (see Chapter 4). This exemplary hybrid world simulation was conducted similar to a referenced real world simulation. The comparison of the results of these two evaluation settings show similar acquired knowledge for an interface designer. Thus, a first indicator tends to the assumption that a hybrid world simulation is a further important test setting that should be supported by UCP tools.

Tool users, however, have to consider the different mentioned benefits and problems of the corresponding test setting. All in all, it seems to be feasible to use hybrid world simulations at the very beginning of the UCP process if ideas of an application should be mediated and evaluated. Later on, the realism of the test setting needs to increase and therefore a real world simulation should be used instead of a hybrid world simulation. Finally, at the end of the development process, a real world evaluation in the field is inevitable in order to find typical user requirements towards the application that only can be revealed under real contextual constraints.

With the support of MoPeDT (see Chapter 4), these different empirical evaluation settings are possible to implement due to MoPeDT's flexible plug & play architecture (see Section 4.1.1). The virtual world simulation, for instance, can easily be attached in order to enable an communication between the real and virtual world. Further, different recording methods can be used dependent on the requirements of the evaluation setting. In contrast to the presented literature, MoPeDT is the first tool that considers the support of all three test settings. In particular, the additional support of a hybrid world simulation is a new approach.

Chapter 7

Final Conclusion and Outlook

This dissertation aimed at the investigation of the tool-supported user-centred prototyping process of mobile applications in the context of *the Third Computing Paradigm*. Different research questions were asked in Chapter 1 which were answered in the Chapters 4, 5 and 6. This chapter provides details about the research contributions of the dissertation. Also, an outlook is given about how one could proceed in the future.

7.1 Research Contribution

The overall goal of the dissertation was to cover the user-centred prototyping of network-based mobile applications for multi-users in the context of *the Third Computing Paradigm*. By means of the user-centred prototyping (UCP) tool called MoPeDT - Pervasive Interface Development Toolkit for Mobile Phones - such evolutionary prototypical applications can automatically be generated with different levels of fidelity. These prototypes are compliant with approved interface guidelines in the mobile computing context. Also, the prototypes support different mobile interaction styles (e.g. keyboard-based and speech-based). The tool cannot only be used to generate prototypes, it also can be applied in order to conduct evaluations in different test settings for several users and later on to analyse the captured data. Thereby, MoPeDT addresses the conduction of empirical and analytical evaluations. An interesting new empirical test setting is, in particular, the tool-supported conduction of hybrid world simulations.

1. **The investigation and implementation of new conceptual and technical approaches to software-assisted support the generation and evaluation of mobile applications in the context of *the Third Computing Paradigm*.**

In order to enable the tool-supported user-centred prototyping process, new technical and conceptual approaches were introduced, implemented by means of MoPeDT and finally discussed in this dissertation. Several new concepts addressed the automatic generation of prototypes [Leichtenstern and André, 2009b, Leichtenstern and André, 2011, Leichtenstern et al., 2011a] while other concepts concerned the GUI-supported user-centred prototyping process [Leichtenstern and André, 2010, Leichtenstern and André, 2011, Leichtenstern et al., 2011a]. Also, the hybrid world simulation was introduced as a new test setting to tool-assisted conduct an empirical evaluation [Leichtenstern et al., 2010, Leichtenstern et al., 2011a].

An approach for the automatic generation of prototypes: For the automatic generation of prototypes, the use of a component-based client-server architecture was introduced (see Section 4.1.1). This architecture provides several generic benefits. First of all, it enables the network-based remote communication which enables the loading and displaying of remote content of a database as well as its changing. Further, the architecture also supports the conduction of both local and remote user evaluations as real world evaluations in the field as well as real world and hybrid world simulations in the laboratory since all relevant data can be transmitted by means of the network, such as the occurrence of mobile interactions. The architecture does not only support a remote communication, it also enables the plug&play of several components, such as several mobile users, evaluators, sensors or actors which is a precondition for a multi-user application that requires additional interaction and presentation devices. Further, the architecture supports the conduction of an empirical evaluation for several users simultaneously.

For the components of the architecture, a mechanism was additionally introduced that enables the synchronous communication between them (see Section 4.1.2). Components can register their interest in context of other components. Grounded on this registration, the server forwards incoming context messages to other components, such as the server forwards messages of a mobile user to a connected evaluator component. By means of this mechanism, each component can easily be linked to other components.

A further important approach of this dissertation is the use of screen templates. As highlighted in Section 3.3.4, few tools consider the compliance of approved mobile HCI guidelines when generating prototypes. Typically, interface designers can freely design their application, such as arranging control elements (e.g. buttons) and the screen content (e.g. images). If interface developers are less skilled with interface guidelines, the resulting

prototypes probably will contain several usability problems. To reduce the usability problems of the resulting prototype, the use of screen templates was described in this dissertation (see Section 4.1.2). By this means, for instance, a consistency is ensured. Further, based on the different types of screen templates, different levels of fidelity can be defined for the prototype.

An approach for the GUI-based user-centred prototyping process: Besides aspects of the automatic generation of prototypes, the dissertation also provided an approach about the GUI-supported user-centred prototyping process: the tool-support for the design specification of prototypes as well as the conduction and analysis of evaluation with the generated prototypes.

The need for different views was recognised during the design specification task (see Section 4.2.1). Thus, besides the state-chart view, an approach was introduced to provide a tree view in order to define details about a screen's appearance and behaviour.

For the conduction of an empirical evaluation, the use of a cloned screen view was presented (see Section 4.2.2). Based on the feature to register interest in another component of the architecture (see Section 4.1.1), a cloned emulator view was used to always update and display the current screen of the remote test user on the evaluator's desktop PC. This screen provides an evaluator with the opportunity to always track the user and her interactions. By this means, the evaluator can detect usability problems even at the runtime of the evaluation.

Finally, for the conduction of analytical evaluations, an approach was introduced to support the guideline-based analytical evaluation by always providing the evaluator with a running version of the application as well as screen shots about the specified screen states and the screen flow (see Section 4.2.2). By means of this support, the evaluator, for instance, can execute *the Cognitive Walkthrough* or *the Heuristic Evaluation* (see Section 3.1).

Tool-supported empirical evaluations by means of a hybrid world simulation: As a new approach and a further contribution of the dissertation, the tool-supported hybrid world simulation was introduced in Section 6.3. It is a special kind of a virtual world simulation since it means an integration and combination of the real and the virtual world. The pervasive computing environment and their physical objects are located in a virtual world simulation while the mobile user still makes use of a real phone in order to interact with the virtually simulated objects.

Section 6.3 provided details how a hybrid world simulation can be set up by means of MoPeDT and its component-based client-server architecture.

Thereby, three steps were described: the modelling of the virtualised pervasive computing environment, the generation of virtual physical objects and sensors as well as the communication of the virtual world with the outside world to enable seamless transitions for the mobile user.

2. The introduction of an evaluation method to validate a user-centred prototyping (UCP) tool's quality.

In order to validate the concept of the tool-supported user-centred prototyping as well as whether the different non-functional requirements are met, an evaluation method was also introduced within this dissertation (see Chapter 5).

The method [Leichtenstern and André, 2010, Leichtenstern et al., 2011a] describes a comparative tool study that applies analytical and empirical evaluation techniques to collect subjective and objective data in order to reveal the meeting of the non-functional requirements towards the tool and the resulting prototypes.

The idea is that the participants of the tool study are asked to conduct the tool-supported user-centred prototyping process in order to develop a pre-defined application. To simplify the interpretation of the results of the tool study, a baseline approach should be used as well, such as the additional rather traditional execution of the user-centred prototyping process. During the process to user-centred develop the prototype - formative evaluation - protocol recordings can be used in order to collect objective data towards the tool's usability, reliability and functionality (e.g. the required time to conduct an iteration or the found usability problems).

Subjective data can be collected by making use of a questionnaire that provides statements with regards to different aspects (e.g. learnability or satisfaction). To not only investigate the tool-supported process - formative evaluation - but also the result of the process - summative evaluation -, an analytical technique (e.g. *the Heuristic Evaluation*) can be used to validate the meeting of the non-functional requirements towards the resulting prototypes: their high level of usability, reliability and functionality.

By means of a tool study with MoPeDT and a traditional baseline approach, the method was illustrated and finally discussed in Section 5.3. This section also provided details about other useful evaluation techniques that might be used as well. If the tool study is not conducted as a mid-term but a short-time study, for instance, a video observation can be used instead of a protocol recording to collect objective data about the user's and the tool's performance while executing the user-centred prototyping process.

3. The collection of knowledge and experiences about the development process and the use of user-centred prototyping tools.

A further contribution of this dissertation [Leichtenstern and André, 2010, Leichtenstern et al., 2011a] is to provide knowledge and experience about the development process and the use of user-centred prototyping tools (see Section 4.3 and Section 5.3). This knowledge might be useful for other tool developers and tool users.

Empirically validated knowledge towards the development of a user-centred prototyping tool: The empirically validated knowledge grounds on the tool study that was illustrated in Section 5.2. The results showed that a UCP tool such as MoPeDT can increase the efficiency of tool users to quickly conduct a process iteration. Apart from that, the tool users' effectiveness to generate highly functional, reliable and usable prototypes can also be improved. Further, the need for Software and Usability Engineering skills can be reduced by making use of a tool such as MoPeDT. Thus, the learnability of the UCP process can also be improved. A problem emerged in terms of the limited tool's transparency and the lacked tool user's control. Consequently, a problem occurred regarding the tool user's satisfaction.

Developers of UCP tools have to consider the quick and easy use of the tool by addressing its high intuitiveness, learnability and comfort of use. Help functionalities and tutorials need to be provided as well as a direct mapping between a tool user's interaction and the consequence on the prototype. Also, screen templates should be used to improve the result of the tool-assisted process. The consequence of using screen templates on the tool user's feeling of control, however, needs to be considered. Screen templates can limit the freedom of interaction. To tackle that problem, a wide range of different kinds of screen templates has to be provided as well as options to enable the adaptation and individualisation of the screen templates.

But not only the screen templates, also the tool itself needs to provide a wide range of functionalities to not limit the tool users. However, this can have a negative affect on the tool's intuitiveness and comfort of use. A negative affect on the intuitiveness and comfort of use can also be caused by a higher level of a tool's transparency, such as by providing a continuous feedback about system processes. Tool developers need to find a trade-off between an appropriate level of learnability, intuitiveness and comfort of use on the one hand and user control and transparency on the other hand in order to increase the satisfaction of the tool users.

Nevertheless, the tool study also indicated the tendency to provide an all-

in-one tool for all steps of the user-centred prototyping process instead of separate tools for each step to reduce training periods and improve the learnability.

Experience towards the tool-supported conduction of user evaluations in different test settings: Based on a user evaluation that was conducted as a hybrid world simulation (see Section 6.3), the following gained experiences [Leichtenstern et al., 2010, Leichtenstern et al., 2011a] can also be summed up.

Tool users have to consider the different mentioned benefits and problems of the corresponding test setting. The real world evaluation in the field, for instance, provides very realistic data since it is conducted in the real environment of the user with real contextual constraints. A tool-supported conduction of a real world evaluation in the field can help log all mobile interactions or context data (e.g. the user's location). The problems of tool-supported real world evaluations mainly emerge due to the spatial separation of the evaluator and the users when conducting remote evaluations. As one consequence, an audio-visual capturing of the users and their environments is difficult to perform.

Further problems are discussed in Section 6.1, such as the need to have a highly functional and reliable application when running a real world evaluation. Tool-supported real world evaluations in the field should be conducted at least at the end of the user-centred prototyping process when the prototypes are highly functional, reliable and usable to get very detailed and realistic knowledge about the user behaviour and preferences. At the very beginning and at the middle of the user-centred prototyping process, hybrid world simulations and respectively real world simulations should be used since these test settings can reduce money and time.

Hybrid world simulations shall be used in addition to real world simulations due to the fact that they help quickly communicate very first ideas about a new product as well as that they enable the first conductions of user evaluations mainly with regards to the user acceptance. They do not require building up a real simulation of the real world which can save further money and time. At the same time, they seem to provide similar knowledge about the users as a real world simulation in a laboratory (see Section 6.3).

Also, by making use of a widely used virtual platform (e.g. Second Life) for the hybrid world simulation, several test users can remotely be reached which potentially provide a first comprehensive feedback towards an application idea. Later on during the development process, however, the realism

of the test setting needs to increase and a real world simulation in the laboratory (see Section 6.2) should be used instead of a hybrid world simulation. This is needed because an only hybrid world simulation approach cannot provide the same level of immersion as a real world simulation approach which might cause drawbacks and incorrect results in some situations.

Due to the gained experience, a UCP tool should not only support classical real world evaluations in the field and real world simulations in the laboratory but also hybrid world simulations.

7.2 Future Work

This dissertation focused on the tool-support for the development of mobile application in the context of *the Third Computing Paradigm*. From this point it would be straight forward to investigate further current trends in mobile computing which have not been addressed by this dissertation. There is a need, for instance, to investigate the tool-supported generation and evaluation of prototypes which support other mobile interaction styles than MoPeDT's supported interaction styles. The touch-based or gesture-based input separately of each other or fused might be of interest for further research (e.g. see [Wasinger, 2006]).

Results of this dissertation showed that tool developers have to make sure a tool's high intuitiveness, learnability and comfort of use on the one hand and a tool's high transparency and a user's comprehensive feeling of control when using the tool on the other hand. A high transparency by providing continuous system feedback or a high user control by supporting a wide range of control elements, for instance, can impair the tool user's efficiency as well as the learnability of the tool use. Consequently, these aspects can be competitive and need further research to find approaches how to solve this problem. Of course, one solution would be to develop a tool that can be highly configured to address the different needs of the corresponding tool users. The tool could either be configured to meet a preference for intuitiveness, learnability and comfort of use or a preference for transparency and user control. Interesting, however, would be if there is a solution to meet all aspects at the same time, such as by providing an unobtrusive user feedback to increase the transparency that does not harm the intuitiveness and comfort of use.

In this dissertation the use of screen templates was introduced as an approach to consider the compliance with approved mobile interface guidelines. Interesting would be to investigate further approaches and their limitations and benefits compared to the template-based approach. The use of the template-based

approach, for instance, caused a limited user control when using MoPeDT. By means of other approaches, limitations of MoPeDT might be solved. Constraint-based approaches (e.g. see [Feuerstack et al., 2008]), for instance, might increase the feeling of user control but the resulting usability of the prototype might be impaired.

Finally, the approach of the hybrid world simulation requires future research to find solutions to reduce the mentioned problems of Section 6.3. The level of immersion needs to be increased as well as the realism of the setting. Also, further research has to be conducted to compare the differences between a real and a virtual world simulation in more detail. The study presented in this dissertation (see Section 6.3) provided a first indicator that the novel approach of a tool-based hybrid world simulation can meaningfully support the user-centred prototyping process at least at the beginning of the process. More and deeper research, however, has to be conducted to finally provide evidence for that assumption.

Appendix A

Interface Guidelines

No.	Guideline (based on Nokia http://library.forum.nokia.com)	Yes	No
G1	Soft key usage and navigation style are consistent.		
G2	The layout is consistent.		
G3	Error notes are informative and clear without technical terminology.		
G4	Application contains contextual help with brief descriptions.		
G5	User can cancel actions easily.		
G6	Errors are handled correctly.		
G7	Important actions are mentioned first.		
G8	User is always allowed to navigate backward with the Back key.		
G9	The user always recognises where s/he is.		
G10	Application menus do not contain multiple instances of same command.		
G11	The navigation structure is simple and straight-forward.		
G12	The number of screens is minimal.		
G13	Feedback on user actions is provided immediately.		
G14	Right soft key is used for backward, negative actions and closing the application.		
G15	No essential information is displayed with icons only but also with text.		
G16	The screens are clearly structured to reduce complexity.		
G17	A user-centred speech is used.		
G18	It is clearly indicated to users what s/he can do.		
G19	It is always clear which action has to be performed next to complete the task.		
G20	There are no unnecessary actions.		
G21	Actions are natural and comprehensible.		
G22	Recognition rather than recall. User only has to input data if necessary.		

Appendix B

Task Description

B.1 Task Description for the traditional Approach

Generierung eines Prototyp: Es soll eine kleine pervasive Schnittstelle für ein Mobiltelefon entwickelt werden, um Zusatzinformationen zu Artikeln in einem Supermarkt zu erhalten.

In der Gemüseabteilung befinden sich drei Artikel: Tomaten, Gurken und Kartoffeln. Zu jedem der Artikel kann man folgende drei Dienste abrufen.

- Kurzbeschreibung des Gemüses
- Infos zur Herkunft
- Wichtige Zusatzinfos

Eure Aufgabe ist es jetzt eine kleine Anwendung für ein Mobiltelefon zu implementieren, um die Auswahl des Artikels und der jeweiligen Dienste zu ermöglichen. Zur Interaktion stehen dem Nutzer genau zwei Techniken zur Verfügung. Der Nutzer kann die Selektion über die Tastatur oder über RFID-Tags durchführen. Achtet bei der Realisierung darauf, dass ihr die Richtlinien von Nokia einhaltet. Insbesondere soll auf Konsistenz geachtet werden. Im folgenden wird der genaue Ablauf der Anwendung vorgegeben.

1. Hauptmenü

Sobald der Nutzer die Anwendung startet erscheint ein Hauptmenü. Das Hauptmenü ist eine statische Liste, die einen Item Artikel und einen Item Hilfe darstellt. Selektiert der Nutzer Artikel, dann wechselt er zum Screen

2. Wählt der Nutzer Hilfe, dann wechselt er zum Screen 3. Berührt der Nutzer den RFID-Tag des Artikels mit seinem Handy, dann erreicht er direkt die Dienstübersicht von Screen 2.1.

2. Menü - Artikel

An dieser Stelle wird eine dynamische Liste generiert. Es werden aus der Datenbank alle Artikel geladen und auf dem Screen dargestellt. Wählt der Nutzer einen aufgelisteten Artikel, dann gelangt er zu Screen 2.1. Auch hier kann der Nutzer durch das Berühren eines Artikels zum Screen 2.1 gelangen.

(a) Menü - Dienste

Auch an dieser Stelle wird eine dynamische Liste generiert. Zu dem gewählten Artikel werden alle Dienste aus der Datenbank geladen und aufgelistet. Wählt der Nutzer einen Dienst aus, dann gelangt er zum Screen 2.2. Falls der Nutzer einen anderen Artikel berührt, werden die Dienste zu dem neuen Artikel auf dem Screen 2.1 dargestellt.

(b) Media - Inhalte zu den Diensten

Dieser Screen ladet automatisch die Inhalte zu dem selektierten Dienst und stellt sie dem Nutzer zur Verfügung. An dieser Stelle soll für die Kurzbeschreibung ein Text und falls machbar ein Bild angezeigt werden. Für die Infos zur Herkunft und die Zusatzinfos sollen nur Textinformationen dargestellt werden. Auch von diesem Screen soll man bei einer RFID-Selektion zum Screen 2.1 gelangen.

3. Media - Hilfe

An dieser Stelle wird ein Textbildschirm mit Informationen zur Anwendung zur Verfügung gestellt. Auch von diesem Screen soll man bei einer RFID-Selektion zum Screen 2.1 gelangen.

Geht bei der Realisierung des Prototypen folgendermassen vor:

1. Anlegen der Datenbank

Ihr habt eine Datenbank erhalten, die ihr nutzen könnt, um die Artikel, Dienste und Inhalte abzulegen. Legt im ersten Schritt alle nötigen Tabellen in der Datenbank an und füllt diese mit den zur Verfügung gestellten Inhalten.
<http://nfc.informatik.uni-augsburg.de/USA2009/index.php>

2. Anbindung der Datenbank

Im nächsten Schritt bindet die Datenbank an euren Server an um Datenbankabfragen zu erlauben. Es gibt genau drei Anfragemöglichkeiten:

- Anfrage aller Artikel
 - Anfrage der Dienste zu einem Artikel
 - Anfrage zu den Inhalten eines Dienstes
3. Kommunikation zwischen dem Client und dem Server
Ändert euer vorhandenes (Chat-) Programm so ab, dass DB-Anfragen vom Client an den Server geschickt werden können und als Antwort die gewünschten Informationen erhalten werden.
 4. Aufbereitung der Daten auf der Handyseite
Wandelt auf der Client-Seite die Antwort vom Server in ein OO-Modell um.
 5. Erstellung einer graphischen Benutzeroberfläche
Realisiert eine graphische Benutzeroberfläche auf dem Handy, um die jeweiligen Screens darstellen zu können
 6. Realisierung der tasten-basierten Interaktion
Die GUI kann durch Tastenselektion gesteuert werden.
 7. Realisierung der NFC-basierten Interaktion
Die GUI kann durch Berühren von RFID-Tags gesteuert werden.
 8. Realisierung des Event-Loggings
Alle Interaktionen des Nutzers sollen zum Server übertragen werden und in eine Datei geloggt werden. Als Interaktionen zählen dabei alle Tasteninteraktionen sowie RFID-Selektionen, die der Nutzer durchführt.

Bei jedem der Schritte der Generierung füllt bitte das zur Verfügung gestellte Protokoll für die Implementierung aus.

Evaluation eines Prototyp: Die Aufgabe bei Evaluation besteht darin drei Testpersonen aufzuzeichnen, wenn sie eure Anwendung nutzen, um die drei folgenden Tasks auszuführen.

1. Informieren Sie sich über die Herkunft der Tomaten.
2. Lesen Sie die Zusatzinformationen zu den Gurken.
3. Betrachten Sie die Inhaltsstoffe von Kartoffeln.

Bei der Abarbeitung der Tasks sollen die Testpersonen die Methode des Lauten Denkens anwenden. Hinweis: Die Testpersonen sollen während der Nutzerevaluation den Emulator auf eurem Rechner benutzen!

Geht bei der Evaluation des Prototypen folgendermassen vor:

1. Durchführung des Tests

Zeichnet den Nutzer während des Tests auf Video auf. Loggt alle Aktionen des Nutzers in eure Logfiles

2. Auswertung des Tests

Wertet eure Videos und Logfiles aus. Dabei müsst ihr unter anderem die Logfiles den Videos passend zuweisen. Betrachtet dann die Ergebnisse eurer Auswertung und findet eventuelle Usability-Probleme. Dokumentiert diese Probleme und gebt diese ebenfalls mit ab.

Während der Evaluation denkt bitte daran das Protokoll zur Evaluation auszufüllen.

B.2 Task Description for MoPeDT

Generierung eines Prototyp: Der zu entwickelnde Prototyp soll genau der Beschreibung vom Aufgabenblatt 8 entsprechen. Im Gegensatz zum Übungsblatt 8 werdet ihr den Prototyp allerdings anhand des Tools MoPeDT erstellen.

Geht bei der Realisierung des Prototypen folgendermassen vor:

1. Anlegen der Datenbank

Nutzt das Content Management-Tool von MoPeDT, um eure pervasive Umgebung anzulegen. Auch hier sollen drei Artikel sowie jeweils drei Dienste (siehe Blatt 8) angelegt werden.

2. Realisierung der Kontexte

Nutzt das Context Management-Tool, um die tasten-basierte Interaktion und die NFC-basierte Interaktion für euren Prototyp festzulegen.

3. Erstellung der Präsentation

Nutzt das Pervasive Interface Management-Tool >Interface Model Screen, um die Screenblöcke (die Präsentation) zu spezifizieren. (Benötigte Skriptbefehle sind <object, service, entry >.getName(),

<object, service, entry >.getIcon() sowie environment.getObjects(), object.getServices() und entry.getContent())

4. Erstellung der Transitionen

Nutzt das Pervasive Interface Management-Tool >Interface Model Screen, um die Transitionen zu spezifizieren. (Benötigte Skriptbefehle sind world.getValue()=='Artikel' bzw. world.getValue()=='Hilfe')

5. Spezifikation der generellen Einstellungen

Nutzt das Pervasive Interface Management-Tool >Interface Setting Screen, um generelle Einstellungen für euren Prototypen festzulegen und generiert euren Prototyp. Achtet darauf, dass euer Server läuft!

Bei jedem der Schritte der Generierung füllt bitte das zur Verfügung gestellte Protokoll für die Implementierung aus.

Evaluation eines Prototyp: Die Aufgabe bei Evaluation besteht darin drei Testpersonen aufzuzeichnen, wenn sie eure Anwendung nutzen, um die drei folgenden Tasks auszuführen.

1. Informieren Sie sich über die Herkunft der Tomaten.
2. Lesen Sie die Zusatzinformationen zu den Gurken.
3. Betrachten Sie die Inhaltsstoffe von Kartoffeln.

Bei der Abarbeitung der Tasks sollen die Testpersonen die Methode des Lauten Denkens anwenden. Hinweis: Die Testpersonen sollen während der Nutzerevaluation den Emulator auf eurem Rechner benutzen!

Geht bei der Evaluation des Prototypen folgendermassen vor:

1. Durchführung des Tests

Nutzt das Pervasive Interface Management-Tool >Interface Evaluation Screen, um eine Evaluation für euren Prototypen durchzuführen. Verbindet euch zum Server und startet die Aufzeichnung. Am Ende eines Tests exportiert eure Evaluation in ANVIL

2. Auswertung des Tests

Betrachtet mit ANVIL eure Evaluation und findet eventuelle Usability-Probleme. Dokumentiert diese Probleme und gebt diese ebenfalls mit ab.

Während der Evaluation denkt bitte daran das Protokoll zur Evaluation auszufüllen.

Appendix C

Protocol

Protokoll zur Entwicklung und Evaluation der Prototypen

Hinweis: Bitte protokollieren Sie **jeden** Arbeitsschritt bei der Erstellung bzw. Evaluation der Prototypen. Wählen Sie dazu zunächst die jeweilige Kategorie aus und geben die benötigte Zeit an. Bitte beschreiben Sie außerdem kurz welche Tätigkeit genau zu erledigen war, welchen Schwierigkeitsgrad Sie dieser geben würden und welche Probleme Sie bei deren Bearbeitung hatten.

Liste der Kategorien für die Arbeitsschritt bei der Entwicklung:

1. Backend-Programmierung
 - a. Datenbank füllen
 - b. Anbindung Server an die Datenbank
 - c. Anbindung Handy an den Server (Übertragung der Daten)
 - d. Aufbereitung der Daten auf Handyseite(z.B. Objektmodell erstellen oder XML-Daten parsen)
 - e. Logging der Events zum Server
2. UI-Programmierung
 - a. GUI-Programmierung auf dem Mobiltelefon
 - b. Interaktivität – NFC
 - c. Interaktivität – Tastatur

Liste der Kategorien für die Arbeitsschritt bei der Evaluation:

1. Aufzeichnung
 - a. Aufzeichnung der Videos
 - b. Logging der Events
 - c. Synchronisation der Events
2. Auswertung
 - a. Synchronisation der Events
 - b. Analyse der Videos
 - c. Analyse der Events
 - d. Statistische Auswertung der Daten

Formular für das Protokoll:

a) Welchen Arbeitsschritt haben Sie erledigt (Kategorie)?

b) Wie lange haben Sie für diesen Arbeitsschritt benötigt (in Minuten)?

c) Kurze Beschreibung der Tätigkeit

d) Welchem Schwierigkeitsgrad würden Sie dieser Tätigkeit geben?

Sehr leicht	Leicht	Mittel	Schwer	Sehr Schwer
<input type="radio"/>				

e) Welche Probleme traten bei der Ausführung auf?

Appendix D

Questionnaire

Allgemeines:

1. Alter:

2. Geschlecht:

3. Programmierkenntnisse:

a. Objektorientierung z.B. Java, C++, C# etc.

Überhaupt keine	Gering	Mittel	Gut	Sehr Gut
<input type="radio"/>				

Welche objektorientierte Sprache beherrschen Sie am besten?

Wie lange haben Sie Programmiererfahrung mit objektorientierten Sprachen (in Monaten oder Jahren)?

4. Kenntnisse zur UI-Programmierung

- a. Wie schätzen sie ihre Kenntnisse zur GUI-Programmierung für Desktop-Anwendungen ein?

Überhaupt keine	Gering	Mittel	Gut	Sehr Gut
<input type="radio"/>				

Wie lange haben sie Erfahrung mit der GUI-Programmierung für Desktop-Anwendungen(in Monaten oder Jahren)?

- b. Wie schätzen sie ihre Kenntnisse zur Mobiltelefonprogrammierung ein?

Überhaupt keine	Gering	Mittel	Gut	Sehr Gut
<input type="radio"/>				

Wie lange haben sie Erfahrung mit der Mobiltelefonprogrammierung (in Monaten oder Jahren)?

5. Kenntnisse zum Usability Engineering

- a. Wie schätzen sie ihre Kenntnisse zum Usability Engineering für Desktop-Anwendungen ein?

Überhaupt keine	Gering	Mittel	Gut	Sehr Gut
<input type="radio"/>				

Wie lange haben sie Erfahrung mit dem Usability Engineering für Desktop-Anwendungen (in Monaten oder Jahren)?

- b. Wie schätzen sie ihre Kenntnisse zum Usability Engineering für Mobilgeräte ein?

Überhaupt keine	Gering	Mittel	Gut	Sehr Gut
<input type="radio"/>				

Wie lange haben sie Erfahrung mit dem Usability Engineering für Mobilgeräte (in Monaten oder Jahren)?

Prototypen-Generierung

Die folgenden Fragen beziehen sich auf die **Prototypen-Generierung** von pervasiven Schnittstellen für Mobiltelefone. Bei den Fragen wird ein Vergleich zwischen der klassischen Prototypen-Generierung und der Generierung mit MoPeDT gezogen. Mit klassischen Werkzeugen sind Entwicklungsumgebungen wie z.B. Eclipse oder Netbeans gemeint.

6. Welche **Prototypen-Generierung** finden Sie besser (Eine Antwort)?

Keines geeignet	Klassisch besser	Mit Tool (wie MoPeDT) besser	Beide gleich gut geeignet
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Begründung:

7. Wie treffen die folgenden Aussagen auf die **Prototypen-Generierung** mit MoPeDT zu?

- a) Die Generierung von Prototypen mit MoPeDT ist schneller als mit klassisch erstellten Werkzeugen:

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

- b) Die Qualität (Usability) der mit MoPeDT generierten Prototypen ist besser als Prototypen, die mit klassischen Werkzeugen generiert wurden:

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

- c) Man benötigt weniger Programmierkenntnisse, wenn man Prototypen mit MoPeDT generiert anstatt mit klassischen Werkzeugen:

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

- d) Man benötigt weniger Einarbeitungszeit, wenn man Prototypen mit MoPeDT generiert anstatt mit klassischen Werkzeugen:

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

e) Der Umfang der unterstützten Screen-Templates von MoPeDT reicht aus.

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

f) Bei der Prototypen-Generierung bietet MoPeDT eine ausreichende Transparenz.

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

8. Generelle Kritik/Meinung zu dem Thema – **Prototypen- Generierung**:

<hr/> <hr/>

Prototypen-Evaluation

Die folgenden Fragen beziehen sich auf die Prototypen-Evaluation von pervasiven Schnittstellen für Mobiltelefone. Bei den Fragen wird ein Vergleich zwischen der klassischen Prototypen-Generierung und der Generierung mit MoPeDT gezogen. Mit klassischen Werkzeugen sind z.B. Videoaufzeichnungstools gemeint.

9. Welche **Prototypen - Evaluation** finden Sie besser (Eine Antwort)?

Keines geeignet	Klassisch besser	Mit Tool (wie MoPeDT) besser	Beide gleich gut geeignet
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Begründung:

<hr style="border: 0; border-top: 1px solid black; margin-bottom: 5px;"/> <hr style="border: 0; border-top: 1px solid black;"/>

10. Wie treffen die folgenden Aussagen auf die **Prototypen - Evaluation** mit MoPeDT zu?

a) Die Evaluation von Prototypen mit MoPeDT ist schneller als mit klassischen Werkzeugen:

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

b) Man benötigt weniger Evaluationskenntnisse, wenn man Prototypen mit MoPeDT evaluiert anstatt mit klassischen Werkzeugen:

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

c) Man benötigt weniger Einarbeitungszeit, wenn man Prototypen mit MoPeDT evaluiert anstatt mit traditionellen Werkzeugen:

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

d) Der Umfang der unterstützten Evaluationsfunktionen von MoPeDT reicht aus.

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

e) Bei der Evaluation ist die Transparenz ausreichend, wenn man Prototypen mit MoPeDT evaluiert.

<input type="radio"/> Stimmt überhaupt nicht	<input type="radio"/> Stimmt nicht	<input type="radio"/> Neutral	<input type="radio"/> Stimmt	<input type="radio"/> Stimmt vollkommen
--	---------------------------------------	----------------------------------	---------------------------------	---

11. Generelle Kritik/Meinung zu dem Thema – **Prototypen - Evaluation**:

<hr style="border: 0; border-top: 1px solid black; margin-bottom: 5px;"/> <hr style="border: 0; border-top: 1px solid black;"/>

Prototyping

Folgende Fragen beziehen sich auf das Prototyping mit MoPeDT oder klassisch z.B. nur mit dem Einsatz einer Entwicklungsumgebung wie Eclipse. Mit Prototyping ist dabei eine Iteration zur Erstellung eines High-Fidelity-Prototyp gemeint. Prototyping beinhaltet also die Generierung eines Prototyps und deren anschließende Evaluation und Auswertung.

12. Welche Bestandteile an **MoPeDT** finden sie gut (Eine Antwort)?

Keines	Prototypen-Generierung	Prototypen-Evaluation	Beides - Die Kombination von Generierung und Evaluation
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Begründung:

13. Was trifft auf die Benutzung von **MoPeDT** zu?

a) Das Prototyping mit MoPeDT ist intuitiv zu nutzen.

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

b) Das Prototyping mit MoPeDT ist leicht zu erlernen und umzusetzen.

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

c) Die angebotenen Funktionen von MoPeDT sind umfassend.

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

d) Ich bin mit MoPeDT zufrieden.

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

e) Ich bin mit dem benötigten Zeitaufwand bei der Benutzung von MoPeDT zufrieden.

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

f) Ich bin mit dem generierten Prototypen zufrieden, der mit Hilfe von MoPeDT entstanden ist.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

g) MoPeDT bildet das iterative Design des nutzerzentrierten Design-Prozesses angemessen ab.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

h) MoPeDT hilft mir den nutzerzentrierten Design-Prozess effizienter durchzuführen.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

i) MoPeDT ist eine sinnvolle Unterstützung bei der Durchführung des nutzerzentrierten Design-Prozesses.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

j) Der Einsatz von MoPeDT ermutigt mich verschiedene Designideen auszuprobieren.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

k) Der Einsatz von MoPeDT ermutigt mich mehr Design-Iterationen während des Entwicklungsprozesses durchzuführen.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

l) Der Einsatz von MoPeDT verhilft mir zu einem Zeitgewinn, den ich für mehr Nutzertests einsetzen kann.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

14. Was trifft auf die Nutzung von klassischen Werkzeugen (z.B. Eclipse) (ohne MoPeDT) zu?

a) Das klassische Prototyping (ohne MoPeDT) ist intuitiv zu nutzen (inklusive der Nutzung von J2ME).

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

b) Das klassische Prototyping (ohne MoPeDT) ist leicht zu erlernen und umzusetzen (inklusive dem Erlernen von J2ME).

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

c) Der angebotene Funktionsumfang des klassischen Prototypings ist umfassend (Umfang von J2ME und der genutzten Tools)

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

d) Ich bin mit dem klassischen Prototyping (ohne MoPeDT) zufrieden.

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

- e) Ich bin mit dem benötigten Zeitaufwand bei der Benutzung von mit dem klassischen Prototyping (ohne MoPeDT) zufrieden.

Prototypen-Generierung	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prototypen-Evaluation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

- f) Ich bin mit dem generierten Prototypen zufrieden, der mit Hilfe der klassischen Tools entstanden ist.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

- g) Klassische Tools reichen aus, um das iterative Design des nutzerzentrierten Design-Prozesses angemessen abzuarbeiten.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

- h) Klassische Tool reichen aus, um den nutzerzentrierten Design-Prozess effizient durchzuführen.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

- i) Klassische Tools ermutigen mich verschiedene Designideen auszuprobieren.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

- j) Klassische Tools ermutigen mich mehr Design-Iterationen während des Entwicklungsprozesses durchzuführen.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

- k) Klassische Tools verhelfen mir zu einem Zeitgewinn, den ich für mehr Nutzertests einsetzen kann.

<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stimmt überhaupt nicht	Stimmt nicht	Neutral	Stimmt	Stimmt vollkommen

Danke!

Bibliography

- [Aarts et al., 2001] Aarts, E., Harwig, R., and Schuurmans, M. (2001). Ambient Intelligence. In Denning, J., editor, *The invisible future: the seamless integration of technology into everyday life*, pages 235–250. McGraw-Hill.
- [Andreasen et al., 2007] Andreasen, M. S., Nielsen, H. V., Schröder, S. O., and Stage, J. (2007). What happened to remote usability testing?: an empirical study of three methods. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1405–1414. ACM.
- [Arroyo et al., 2006] Arroyo, E., Selker, T., and Wei, W. (2006). Usability tool for analysis of web designs using mouse tracks. In *CHI '06: Extended abstracts on Human factors in computing systems*, pages 484–489. ACM.
- [Ballagas et al., 2006] Ballagas, R., Borchers, J., Rohs, M., and Sheridan, J. G. (2006). The smart phone: a ubiquitous input device. *IEEE Pervasive Computing*, 5(1):70–77.
- [Ballagas et al., 2007] Ballagas, R., Memon, F., Reiners, R., and Borchers, J. (2007). iStuff Mobile: Rapidly Prototyping New Mobile Phone Interfaces for Ubiquitous Computing. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1107–1116. ACM Press.
- [Barton and Vijayaraghavan, 2002] Barton, J. J. and Vijayaraghavan, V. (2002). Ubiwise: A ubiquitous wireless infrastructure simulation environment. *Technical Report HPL2002-303*.
- [Benford et al., 2004] Benford, S., Rowland, D., Flintham, M., Hull, R., Reid, J., Morrison, J., Facer, K., and Clayton, B. (2004). Savannah: Designing a location-based game simulating lion behaviour. In *ACE '04: Proceedings of the 2004 International Conference on Advances in computer entertainment technology*. ACM.
- [Beyer and Holtzblatt, 1997] Beyer, H. and Holtzblatt, K. (1997). *Contextual Design: Defining Customer-Centered Systems (Interactive Technologies)*. Morgan Kaufmann, 1st edition.

- [Broll and Hausen, 2010] Broll, G. and Hausen, D. (2010). Mobile and physical user interfaces for NFC-based mobile interaction with multiple tags. In *Mobile-HCI'10: Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 133–142. ACM.
- [Carter et al., 2007] Carter, S., Mankoff, J., and Heer, J. (2007). Momento: support for situated ubicomp experimentation. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 125–134. ACM.
- [Chesta et al., 2004] Chesta, C., Patern, F., and Santoro, C. (2004). Methods and tools for designing and developing usable multi-platform interactive applications. *PsychNology Journal*, 2(1):123–139.
- [Chung et al., 1999] Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. (1999). *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers Group, 1st edition.
- [Consolvo and Walker, 2003] Consolvo, S. and Walker, M. (2003). Using the experience sampling method to evaluate ubicomp applications. *IEEE Pervasive Computing*, 2(2):24–31.
- [Cooper et al., 2007] Cooper, A., Reimann, R., and Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design*. Wiley, 3rd edition.
- [Dahlbäck et al., 1993] Dahlbäck, N., Jönsson, A., and Ahrenberg, L. (1993). Wizard of Oz studies—why and how. *Knowledge-Based Systems*, 6(4):258–266.
- [Davis, 1992] Davis, A. M. (1992). Operational prototyping: A new development approach. *IEEE Software*, 9(5):70–78.
- [Davis et al., 1988] Davis, A. M., Bersoff, H., and Comer, E. R. (1988). A strategy for comparing alternative software development life cycle models. *IEEE Transactions on Software Engineering*, 14(10):1453–1461.
- [de Sá and Carriço, 2009] de Sá, M. and Carriço, L. (2009). Mobile support for personalized therapies - omniscopes: Richer artefacts and data collection. In *PervasiveHealth '09: Proceedings of the 3rd International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–8. IEEE.
- [Dey, 2001] Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7.
- [Duh et al., 2006] Duh, H. B., Tan, G. C. B., and Chen, V. H. (2006). Usability evaluation for mobile device: A comparison of laboratory and field tests. In

- MobileHCI '06: Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 181–186. ACM.
- [Faulkner, 2000] Faulkner, X. (2000). *Usability Engineering*. Macmillan.
- [Feuerstack et al., 2008] Feuerstack, S., Blumendorf, M., Schwartz, V., and Albayrak, S. (2008). Model-based layout generation. In *AVI '08: Proceedings of the working conference on Advanced visual interfaces*, pages 217–224. ACM.
- [Föckler et al., 2005] Föckler, P., Zeidler, T., Brombach, B., Bruns, E., and Bimber, O. (2005). Phoneguide: museum guidance supported by on-device object recognition on mobile phones. In *MUM '05: Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, pages 3–10. ACM.
- [Froehlich et al., 2007] Froehlich, J., Chen, M. Y., Consolvo, S., Harrison, B., and Landay, J. A. (2007). MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 57–70. ACM.
- [Gaver et al., 1999] Gaver, B., Dunne, T., and Pacenti, E. (1999). Design: Cultural probes. *Interactions*, 6(1):21–29.
- [Gellersen et al., 2002] Gellersen, H. W., Schmidt, A., and Beigl, M. (2002). Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, 7(5):341–351.
- [Gershenfeld et al., 2004] Gershenfeld, N., Krikorian, R., and Cohen, D. (2004). The Internet of Things. *Scientific American*, 291(4):76–81.
- [Greenhalgh, 2002] Greenhalgh, C. (2002). Equip: a software platform for distributed interactive systems. Technical report, Equator IRC Technical Report Equator-02-002.
- [Haesen et al., 2009] Haesen, M., De Boeck, J., Coninx, K., and Raymaekers, C. (2009). An interactive coal mine museum visit: prototyping the user experience. In *HSI '09: Proceedings of the 2nd conference on Human System Interactions*, pages 546–553. IEEE.
- [Häkkinen and Mäntyjärvi, 2006] Häkkinen, J. and Mäntyjärvi, J. (2006). Developing design guidelines for context-aware mobile applications. In *Mobility '06: Proceedings of the 3rd international conference on Mobile technology, applications & systems*, pages 24–30. ACM.

- [Hammer et al., 2010] Hammer, S., Leichtenstern, K., and André, E. (2010). Using the mobile application EDDY for gathering user information in the requirement analysis. In *EICS'10: Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, pages 279–284. ACM.
- [Hanington, 2003] Hanington, B. (2003). Methods in the making: A perspective on the state of human research in design. *Design Issues*, 19(4):9–18.
- [Hartmann et al., 2006] Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., and Gee, J. (2006). Reflective physical prototyping through integrated design, test, and analysis. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 299–308. ACM.
- [Holleis et al., 2007] Holleis, P., Otto, F., Hussmann, H., and Schmidt, A. (2007). Keystroke-level model for advanced mobile phone interaction. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1505–1514. ACM.
- [Holleis and Schmidt, 2008] Holleis, P. and Schmidt, A. (2008). Makeit: Integrate user interaction times in the design process of mobile applications. In *Pervasive '08: Proceedings of the 6th International Conference on Pervasive Computing*, pages 56–74. Springer.
- [Holtzblatt and Beyer, 1996] Holtzblatt, K. and Beyer, H. (1996). *Contextual design: principles and practice*, pages 301–333. John Wiley & Sons, Inc.
- [Houde and Hill, 1997] Houde, S. and Hill, C. (1997). What do prototypes prototype? In Helander, M., Landauer, T., and Prabhu, P., editors, *Handbook of Human-Computer Interaction*, pages 367–381. Elsevier.
- [Hull et al., 2004] Hull, R., Clayton, B., and Melamed, T. (2004). Rapid authoring of mediascapes. In Davies, N., Mynatt, E. D., and Siio, I., editors, *UbiComp '04: Proceedings of the 6th International Conference on Ubiquitous Computing*, pages 125–142. Springer.
- [Iso and Yamazaki, 2006] Iso, T. and Yamazaki, K. (2006). Gait analyzer based on a cell phone with a single three-axis accelerometer. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 141–144. ACM.
- [Jo et al., 2007] Jo, D., Yang, U., and Son, W. (2007). Design evaluation using virtual reality based prototypes: towards realistic visualization and operations. In

- MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pages 246–258. ACM.
- [John and Kieras, 1996] John, B. E. and Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3(4):320–351.
- [Kaasinen, 2003] Kaasinen, E. (2003). User needs for location-aware mobile services. *Personal and Ubiquitous Computing*, 7(1):70–79.
- [Kaikkonen et al., 2005] Kaikkonen, A., Kekäläinen, A., Cankar, M., Kallio, T., and Kankainen, A. (2005). Usability testing of mobile applications: A comparison between laboratory and field testing. *Journal of Usability Studies*, 1(1):4–16.
- [Kela et al., 2006] Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., and Marca, S. (2006). Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299.
- [Kellar et al., 2005] Kellar, M., Reilly, D., Hawkey, K., Rodgers, M., MacKay, B., Dearman, D., Ha, V., MacInnes, W. J., Nunes, M., Parker, K., Whalen, T., and Inkpen, K. M. (2005). It's a jungle out there: practical considerations for evaluation in the city. In *CHI '05: Extended abstracts on Human factors in computing systems*, pages 1533–1536. ACM.
- [Kensing and Blomberg, 1998] Kensing, F. and Blomberg, J. (1998). Participatory design: Issues and concerns. *Computer Supported Cooperative Work*, 7:167–185.
- [Kindberg et al., 2002] Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., and Spasojevic, M. (2002). People, places, things: Web presence for the real world. *Mobile Networks and Applications*, 7(5):365–376.
- [Kipp, 2001] Kipp, M. (2001). Anvil - A Generic Annotation Tool for Multimodal Dialogue. In *Eurospeech '01: Proceedings of the 7th European Conference on Speech Communication and Technology*, pages 1367–1370.
- [Kjeldskov and Graham, 2003] Kjeldskov, J. and Graham, C. (2003). A review of mobile hci research methods. In *MobileHCI' 03: Proceedings of the 5th international conference on Human computer interaction with mobile devices and services*, pages 317–335. Springer.
- [Kjeldskov et al., 2005] Kjeldskov, J., Graham, C., Pedell, S., Vetere, F., Howard, S., Balbo, R., and Davies, J. (2005). Evaluating the usability of a mobile guide: The influence of location, participants and resources. *Behaviour and Information Technology*, 24(1):51–65.

- [Kjeldskov and Stage, 2004] Kjeldskov, J. and Stage, J. (2004). New techniques for usability evaluation of mobile systems. *International Journal of Human-Computer Studies*, 60(5-6):599–620.
- [Klemmer et al., 2000] Klemmer, S. R., Sinha, A. K., Chen, J., Landay, J. A., Aboobaker, N., and Wang, A. (2000). Suede: a wizard of oz prototyping tool for speech user interfaces. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 1–10. ACM.
- [Landay and Myers, 1996] Landay, J. A. and Myers, B. A. (1996). Sketching storyboards to illustrate interface behaviors. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 193–194. ACM.
- [Lee et al., 2004] Lee, S., Chen, T., Kim, J., Han, S., Pan, Z.-g., and Kim, G. J. (2004). Affective property evaluation of virtual product designs. In *VR '04: Proceedings of the IEEE Virtual Reality Conference*, pages 207–292. IEEE.
- [Leichtenstern and André, 2009a] Leichtenstern, K. and André, E. (2009a). Studying multi-user settings for pervasive games. In *MobileHCI '09: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 190–199. ACM.
- [Leichtenstern and André, 2009b] Leichtenstern, K. and André, E. (2009b). The Assisted User-Centred Generation and Evaluation of Pervasive Interfaces. In *AmI '09: Proceedings of the European Conference on Ambient Intelligence*, pages 245–255. Springer.
- [Leichtenstern and André, 2010] Leichtenstern, K. and André, E. (2010). MoPeDT: features and evaluation of a user-centred prototyping tool. In *EICS '10: Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, pages 93–102. ACM.
- [Leichtenstern and André, 2011] Leichtenstern, K. and André, E. (2011). Software Support for the User-Centered Prototyping of Mobile Applications that apply Near Field Communications. In Ahson, S. A. and Ilyas, M., editors, *Near Field Communications Handbook*, pages 119–152. CRC Press.
- [Leichtenstern et al., 2010] Leichtenstern, K., André, E., and Rehm, M. (2010). Using the Hybrid Simulation for Early User Evaluations of Pervasive Interactions. In *NordiCHI '10: Proceedings of the 6th Nordic Conference on Human-Computer Interaction*, pages 315–324. ACM.

- [Leichtenstern et al., 2011a] Leichtenstern, K., André, E., and Rehm, M. (2011a). Tool-Supported User-Centred Prototyping of Mobile Applications. *International Journal of Handheld Computing Research*, 3(2):1–21.
- [Leichtenstern et al., 2007] Leichtenstern, K., André, E., and Vogt, T. (2007). Role Assignment Via Physical Mobile Interaction Techniques in Mobile Multi-user Applications for Children. In *AmI '07: Proceedings of the European Conference on Ambient Intelligence*, volume 4794, pages 38–54. Springer.
- [Leichtenstern et al., 2011b] Leichtenstern, K., Bee, N., André, E., Berkmüller, U., and Wagner, J. (2011b). Physiological Measurement of Trust-Related Behavior in Trust-Neutral and Trust-Critical Situations. In Wakeman, I., Gudes, E., Jensen, C., and Crampton, J., editors, *Trust Management V*, volume 358 of *IFIP Advances in Information and Communication Technology*, pages 165–172. Springer.
- [Leichtenstern et al., 2008] Leichtenstern, K., Erdmann, D., and André, E. (2008). Eval - an evaluation component for mobile interfaces. In *MobileHCI '08: Proceedings of the 10th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 483–484. ACM.
- [Leichtenstern et al., 2006] Leichtenstern, K., Rukzio, E., Chin, J., Callaghan, V., and Schmidt, A. (2006). Mobile interaction in smart environments. In *Pervasive '06: Adjunct Proceedings of the 4th International Conference on Pervasive Computing*. Springer.
- [Li et al., 2004] Li, Y., Hong, J. I., and Landay, J. A. (2004). Topiary: a tool for prototyping location-enhanced applications. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 217–226. ACM.
- [Lifton et al., 2009] Lifton, J., Laibowitz, M., Harry, D., Gong, N.-W., Mittal, M., and Paradiso, J. A. (2009). Metaphor and Manifestation - Cross-Reality with Ubiquitous Sensor/Actuator Networks. *IEEE Pervasive Computing*, 8(3):24–33.
- [Lim et al., 2011] Lim, M., Leichtenstern, K., Kriegel, M., Aylett, R., Enz, S., Vannini, N., Hall, L., and Rizzo, P. (2011). Technology-enhanced role-play for social and emotional learning context - intercultural empathy. *Special Issue on Journal of Entertainment Computing*, 2(4):223–231.
- [Lim et al., 2009] Lim, M. Y., Kriegel, M., Aylett, R., Enz, S., Vannini, N., Hall, L., Rizzo, P., and Leichtenstern, K. (2009). Technology-enhanced role-play for intercultural learning contexts. In *ICEC '09: Proceedings of the 8th International Conference on Entertainment Computing*, pages 73–84. Springer.

- [Macleod and Rengger, 1993] Macleod, M. and Rengger, R. (1993). The development of drum: A software tool for video-assisted usability evaluation. In *HCI '93: Proceedings of the 8th Conference of the British Computer Society Human Computer Interaction Specialist Group - People and Computers VIII*, pages 293–309. Cambridge University.
- [Maguire, 2001] Maguire, M. (2001). Methods to support human-centred design. *International Journal of Human-Computer Studies*, 55(4):587–634.
- [Mäkelä et al., 2007] Mäkelä, K., Belt, S., Greenblatt, D., and Häkkinen, J. (2007). Mobile interaction with visual and rfid tags: a field study on user perceptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '07*, pages 991–994. ACM.
- [Manninen, 2000] Manninen, T. (2000). Multimedia game engine as distributed conceptualisation and prototyping tool contextual virtual prototyping. In *IMSA '00: Proceedings of the 4th IASTED International Conference Internet and Multimedia Systems and Applications*, pages 99–104. IASTED/ACTA Press.
- [Mäntyjärvi et al., 2004] Mäntyjärvi, J., Kela, J., Korpiä, P., and Kallio, S. (2004). Enabling fast and effortless customisation in accelerometer based gesture interaction. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 25–31. ACM.
- [Mantjarvi et al., 2006] Mantjarvi, J., Paternò, F., Salvador, Z., and Santoro, C. (2006). Scan and tilt: towards natural interaction for mobile museum guides. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 191–194. ACM.
- [Mattern and Flörkemeier, 2010] Mattern, F. and Flörkemeier, C. (2010). Vom Internet der Computer zum Internet der Dinge. *Informatik-Spektrum*, 33(2):107–121.
- [McDonald and Schvaneveldt, 1988] McDonald, J. and Schvaneveldt, R. (1988). The application of user knowledge to interface design. In Guindon, R., editor, *Cognitive science and its applications for human-computer interaction*, pages 289–338. Lawrence Erlbaum.
- [McGee-Lennon et al., 2009] McGee-Lennon, M. R., Ramsay, A., McGookin, D., and Gray, P. (2009). User evaluation of oide: a rapid prototyping platform for multimodal interaction. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 237–242. ACM.

- [McKnight et al., 1998] McKnight, D., Cummings, L., and Chervany, N. (1998). Initial trust formation in new organizational relationships. *The Academy of Management Review*, 23(3):473–490.
- [Morla and Davies, 2004] Morla, R. and Davies, N. (2004). Evaluating a location-based application: A hybrid test and simulation environment. *IEEE Pervasive Computing*, 3(3):48–56.
- [Muller et al., 1993] Muller, M. J., Wildman, D. M., and White, E. A. (1993). Taxonomy of pd practices: A brief practitioner’s guide. *Communications of the ACM*, 36(6):26–28.
- [Myers, 1993] Myers, B. A. (1993). Why are human-computer interfaces difficult to design and implement? Technical report, Carnegie Mellon University.
- [Myers, 1995] Myers, B. A. (1995). User interface software tools. *ACM Transactions on Computer-Human Interaction*, 2(1):64–103.
- [Nielsen, 1993] Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.
- [Norman, 2002] Norman, D. A. (2002). *The Design of Everyday Things*, volume 16. Basic Books.
- [Nuseibeh and Easterbrook, 2000] Nuseibeh, B. and Easterbrook, S. (2000). Requirements engineering: a roadmap. In *ICSE ’00: Proceedings of the Conference on The Future of Software Engineering*, pages 35–46. ACM.
- [Palen and Salzman, 2002a] Palen, L. and Salzman, M. (2002a). Beyond the handset: designing for wireless communications usability. *ACM Transactions on Computer-Human Interaction*, 9(2):125–151.
- [Palen and Salzman, 2002b] Palen, L. and Salzman, M. (2002b). Voice-mail diary studies for naturalistic data capture under mobile conditions. In *CSCW ’02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 87–95. ACM.
- [Pallay et al., 2009] Pallay, C., Rehm, M., and Kurdyukova, E. (2009). Getting acquainted in second life: human agent interactions in virtual environments. In *ACE ’09: Proceedings of the International Conference on Advances in Computer Entertainment Technology*, pages 36–43. ACM.
- [Raento et al., 2005] Raento, M., Oulasvirta, A., Petit, R., and Toivonen, H. (2005). ContextPhone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4(2):51–59.

- [Rehm and Leichtenstern, 2012] Rehm, M. and Leichtenstern, K. (2012). Gesture-based mobile training of intercultural behavior. *Special Issue on the ACM/Springer Multimedia Systems Journal*, 18(1):33–51.
- [Rehm et al., 2010] Rehm, M., Leichtenstern, K., Plomer, J., and Wiedemann, C. (2010). Gesture activated mobile edutainment (game): intercultural training of nonverbal behavior with mobile phones. In *MUM '10: Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, pages 11–20. ACM.
- [Rettig, 1994] Rettig, M. (1994). Prototyping for tiny fingers. *Communications of the ACM*, 37(4):21–27.
- [Reynolds et al., 2006] Reynolds, V., Cahill, V., and Senart, A. (2006). Requirements for an ubiquitous computing simulation and emulation environment. In *InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, pages 1–9. ACM.
- [Rogers et al., 2002] Rogers, Y., Sharp, H., and Preece, J. (2002). *Interaction Design: Beyond Human-Computer Interaction*. John Wiley and Sons Ltd.
- [Rohs and Zweifel, 2005] Rohs, M. and Zweifel, P. (2005). A conceptual framework for camera phone-based interaction techniques. In Gellersen, H., Want, R., and Schmidt, A., editors, *Pervasive Computing*, volume 3468 of *Lecture Notes in Computer Science*, pages 313–322. Springer.
- [Roque, 2007] Roque, R. V. (2007). *OpenBlocks : an extendable framework for graphical block programming systems*. PhD thesis, Massachusetts Institute of Technology.
- [Rukzio et al., 2007] Rukzio, E., Broll, G., Leichtenstern, K., and Schmidt, A. (2007). Mobile interaction with the real world: an evaluation and comparison of physical mobile interaction techniques. In *AmI '07: Proceedings of the 2007 European conference on Ambient intelligence*, pages 1–18. Springer.
- [Rukzio et al., 2006] Rukzio, E., Leichtenstern, K., Callaghan, V., Holleis, P., Schmidt, A., and Chin, J. (2006). An experimental comparison of physical mobile interaction techniques: Touching, pointing and scanning. In *UbiComp '06: Proceedings of the 8th International Conference on Ubiquitous Computing*, pages 87–104. Springer.
- [Salber et al., 1999] Salber, D., Dey, A. K., and Abowd, G. D. (1999). The context toolkit: aiding the development of context-enabled applications. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 434–441. ACM.

- [Sato, 2001] Sato, I. (2001). Flying emulator: Rapid building and testing of networked applications for mobile computers. In Picco, G., editor, *Mobile Agents*, volume 2240 of *Lecture Notes in Computer Science*, pages 103–118. Springer.
- [Satyanarayanan, 2001] Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8(4):10–17.
- [Schmidt, 2000] Schmidt, A. (2000). Implicit human computer interaction through context. *Personal and Ubiquitous Computing*, 4(2):191–199.
- [Schmitt, 2008] Schmitt, K. (2008). Gastronomyguide - Entwicklung und Evaluation einer nutzerorientierten und marker-basierten Mobiltelefonapplikation. Bachelor's thesis, Augsburg University, Germany.
- [Scriven, 1967] Scriven, M. (1967). The methodology of evaluation. In Tyler, R. G. R. and Scriven, M., editors, *Perspectives on curriculum evaluation*, AERA Monograph Series - Curriculum Evaluation, pages 39–83. Rand McNally & Co.
- [Sears, 1997] Sears, A. (1997). Heuristic walkthroughs: Finding the problems without the noise. *International Journal of Human-Computer Interaction*, 9(3):213–234.
- [Somervell and McCrickard, 2004] Somervell, J. and McCrickard, D. S. (2004). Comparing generic vs. specific heuristics: Illustrating a new uem comparison technique. In *HFES '04: Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pages 2480–2484. Human Factors and Ergonomics Society.
- [Thayer et al., 1997] Thayer, R. H., Bailin, S. C., and Dorfman, M. (1997). *Software Requirements Engineerings*. IEEE Computer Society Press, 2nd edition.
- [Vääätäjä and Roto, 2010] Vääätäjä, H. and Roto, V. (2010). Mobile questionnaires for user experience evaluation. In *CHI '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, pages 3361–3366. ACM.
- [Wagner et al., 2009] Wagner, J., André, E., and Jung, F. (2009). Smart sensor integration: A framework for multimodal emotion recognition in real-time. In *ACII '09: Proceedings of the 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–8. IEEE.
- [Wagner et al., 2010] Wagner, J., André, E., Kugler, M., and Leberle, D. (2010). Ssi/modelui - a tool for the acquisition and annotation of human generated signals. In *DAGA '10: Proceedings of the 6th German Annual Conference on Acoustics*.

- [Wasinger, 2006] Wasinger, R., editor (2006). *Multimodal Interaction with Mobile Devices: Fusing a Broad Spectrum of Modality Combinations*. Akademische Verlagsgesellschaft, Berlin, Germany.
- [Weiser, 1991] Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.
- [Wellner, 1989] Wellner, P. D. (1989). Statemaster: A uims based on statechart for prototyping and target implementation. In *CHI '89: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 177–182. ACM.
- [Wigdor and Balakrishnan, 2003] Wigdor, D. and Balakrishnan, R. (2003). Tilt-text: using tilt for text input to mobile phones. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 81–90. ACM.
- [Zhang and Adipat, 2005] Zhang, D. and Adipat, B. (2005). Challenges, methodologies, and issues in the usability testing of mobile applications. *International Journal of Human-Computer Interaction*, 18(3):293–308.

Lebenslauf

Karin Bee
Diplom-Medieninformatikerin

Berufliche Tätigkeiten

- seit 06/2012 **UX/UI-Developer**, BMW Group AG München: Konzeption und AKB/HMI-Spezifikation von mobile Apps und iDrive Apps.
- 01/2012-05/2012 **HMI-Entwicklungsingenieurin (mobile Software)**, Bertrandt GmbH Ingolstadt. Einsatz als Usability/UX Experte bei der Audi Electronics Venture GmbH (AEV): HMI-Spezifikation und Koordination der Umsetzung einer iPhone und Android-App zur Mobilitätsplanung von E-Fahrzeugen.
- 05/2006-12/2011 **Wiss. Mitarbeiterin**, Lehrstuhl für *Human-Centered Multimedia*, Universität Augsburg: Promotion, Betreuungs- und Lehrtätigkeit sowie Projektarbeit und -akquise im Bereich mobile Human Maschine Interaction (HMI) sowie Usability/UX Engineering.
- 01/2005-07/2005 **Werkstudentin**, BMW Group Forschungs- und Innovationszentrum, München, Thema: *Erweiterung eines SVG-Frameworks zur Beschreibung von MMI-Oberflächen (iDrive)*.
- 09/1994-08/1999 **Ausbildung zur Bankkauffrau und anschließende Berufsausübung**, Sparkasse Fürstenfeldbruck.
-

Auslandsaufenthalt

- 10/2005-04/2006 **Intelligent Inhabited Environments Group, University of Essex**, Colchester, Großbritannien.
Diplomarbeitsthema: *Mobile Interaktion in intelligenten Umgebungen* (Note: 1,0), Gemeinschaftliche Betreuung der LMU München und University of Essex: Prototypenentwicklung und Studien zur kontextadaptiven Nutzung von unterschiedlichen mobilen Interaktionstechniken (z.B. Near Field Communication) in intelligenten Umgebungen.
-

Ausbildung

- 05/2006-06/2012 **Universität Augsburg:** Promotion in der Informatik, Lehrstuhl für *Human-Centered Multimedia*, Universität Augsburg.
Dissertationsthema: *Nutzerzentriertes Prototyping von mobilen Anwendungen im Kontext des dritten Computerparadigmas (Ubiquitous Computing)*
- Untersuchung der werkzeugunterstützten mobilen Entwicklung

- Studien zu unterschiedlichen Phasen des Human-Centered Design Prozesses

10/2001-04/2006 **Ludwig-Maximilians-Universität München (LMU):** Studium der Medieninformatik (Schwerpunkte: Mensch-Computer Interaktion, System Engineering und Kommunikationswissenschaften) (Abschlussnote: 1,0 - mit Auszeichnung)

09/1999-07/2001 **Berufsoberschule Wirtschaft in München:** Allgemeine Hochschulreife (Abschlussnote: 1,7)

Hauptinteressen

Human Maschine Interaction (HMI), Usability/UX Engineering, Ubiquitous Computing, die nutzerzentrierte Entwicklung von Anzeige-Bedien-Konzepten für mobile Applikationen im Bereich Automotive.

Zusätzliches

Sprachen: Deutsch (Muttersprache), Englisch (fließend in Wort und Schrift).

Programmier- und Werkzeugkenntnisse: Java (J2SE, J2EE, J2ME, Android SDK), C#, Flash mit ActionScript, HTML5 mit JavaScript und CSS, XML-Technologien (SVG, XML Schema, XPath, XSLT), Gimp, Balsamiq Mockups, Eclipse, Visual Studio.

Kenntnisse und Erfahrungen zu Usability und User Experience:

- *iOS Human Interface Guidelines* von Apple sowie *Interaction and Visual Design Guidelines* von Google Android
- Iteratives Prototyping mit der Hilfe von Wireframes und Mockups sowie Software-Prototypen
- Nutzer- und Anforderungsanalysen sowie deren Spezifikationen
- Nutzer- und Expertenstudien (z.B. Eye-Tracking Studien)

Mitgliedschaft: Berufsverband der Usability und User Experience Professionals (German UPA).

Lehre&Forschung: 42 Abschlussarbeiten, 8 praktische Kurse (darunter z.B. Praktikum HCI), 6 Vorlesungen und Seminare (z.B. Usability Engineering und User Interface Design), 3 Forschungsprojekte (EU und DFG) sowie 17 Erstautorpapiere (siehe Publikationsliste).

Unterschleißheim, den 02.07.2012

- 2012** [Book Chapter] K. Bee, S. Hammer, C. Pratsch, E. André, *The automatic Trust Management of self-adaptive Multi-Display Environments*, in Trustworthy Ubiquitous Computing, I. Khalil and T. Montoro, Eds., Atlantis Press, 2012.
- [Book Chapter] E. Kurdyukova, E. André and K. Leichtenstern, *Trust Management of Ubiquitous Multi-Display Environments*, in Ubiquitous Display Environments, T. Kuflik and A. Krüger, Eds., Springer, 2012.
- [Journal] M. Rehm and K. Leichtenstern, *Gesture-Based Mobile Training of Intercultural Behavior*, in Special Issue of the ACM/Springer Multimedia Systems Journal, Springer, vol. 18, no. 1, pp. 33-51, 2012.
- 2011** [Book Chapter] K. Leichtenstern and E. André, *Software Support for the User-Centered Prototyping of Mobile Applications that apply Near Field Communications*, in Near Field Communications Handbook, S. A. Ahson and M. Ilyas, Eds., CRC Press, 2011, pp. 119-152.
- [Journal] K. Leichtenstern, E. André and M. Rehm, *Tool-Supported User-Centred Prototyping of Mobile Applications*, in International Journal of Handheld Computing Research (IJHCR), IGI Global, vol. 2, no. 3, pp. 1-21, 2011.
- [Journal] M. Y. Lim, K. Leichtenstern, M. Kriegel, R. Aylett, S. Enz, N. Vannini, L. H. Hall and P. Rizzo, *Technology-Enhanced Role-Play for Social and Emotional Learning Context - Intercultural Empathy*, in Special Issue of Journal of Entertainment Computing, Elsevier, vol. 2, no. 4, pp. 223-231, 2011.
- [Conference (full)] K. Leichtenstern, N. Bee, E. André, U. Berkmüller and J. Wagner, *Physiological Measurement of Trust-Related Behavior in Trust-Neutral and Trust-Critical Situations*, in Trust Management V, volume 358 of IFIP Advances in Information and Communication Technology, I. Wakeman, E. Gudes, C. Jensen and J. Crampton, Eds., Springer, 2011.
- [Conference (full)] E. Kurdyukova, E. André and K. Bee, *Friend or Foe? Relationship-based Adaptation on Public Displays*, in Proceedings of the International Joint Conference on Ambient Intelligence (Aml 2011), Springer, 2011.
- 2010** [Workshop Organisation] I. Aslan, K. Leichtenstern, P. Holleis, R. Wasinger and C. Stahl, *Tool-support for mobile and pervasive application development - issues and challenges*, in Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (Mobile HCI 2010), ACM, 2010.
- [Journal] E. Kurdyukova, E. André and K. Leichtenstern, *A Decision-Theoretic Approach to maintain Trust in Ubiquitous Display Environments*, in International Journal of Computing, vol. 9, no. 3, pp. 220-227, 2010.
- [Conference (full)] K. Leichtenstern, E. André and M. Rehm, *Using the Hybrid Simulation for Early User Evaluations of Pervasive Interactions*,

in Proceedings of the 6th Nordic Conference on Human-Computer Interaction (NordiCHI 2010), ACM, 2010.

[Conference (full)] K. Leichtenstern and E. André, *MoPeDT - Features and Evaluation of a User-Centred Prototyping Tool*, in Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2010), ACM, 2010.

[Conference (full)] M. Rehm, K. Leichtenstern, J. Plomer and C. Wiedemann, *Gesture Activated Mobile Edutainment (GAME) - Bringing Experience-Based Role-Plays with Virtual Characters to the Mobile Phone for Cultural Training of Nonverbal Behavior*, in Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia (MUM), ACM, 2010.

[Conference (full)] J. Steghöfer, R. Kiefhaber, K. Leichtenstern, Y. Bernard, L. Klejnowski, W. Reif, T. Ungerer, E. André, J. Hähner and C. Müller-Schloer, *Trustworthy Organic Computing Systems: Challenges and Perspectives*, in Proceedings of the 7th International Conference on Autonomic and Trusted Computing (ATC 2010), Springer, 2010.

[Poster] S. Hammer, K. Leichtenstern and E. André, *Using the Mobile Application EDDY for Gathering User Information in the Requirement Analysis*, in Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2010), ACM, 2010.

[Workshop] E. Kurdyukova, E. André and K. Leichtenstern, *Trust-centered Design for Multi-Display Applications*, in Proceedings of the 8th International Conference on Advances in Mobile Computing & Multimedia (MoMM 2010), ACM, 2010.

[Workshop] K. Leichtenstern, E. André and E. Kurdyukova, *Managing User Trust for self-adaptive Ubiquitous Computing Systems*, in Proceedings of the 8th International Conference on Advances in Mobile Computing & Multimedia (MoMM 2010), ACM, 2010.

[Workshop] K. Leichtenstern and E. André, *Tool-Support for the User-Centred Prototyping of Pervasive Games*, in Proceedings of the 1st Workshop on Tool-Support for Mobile and Pervasive Application Development (TSMPAD 2010), 2010.

[Workshop] M. Kriegel, M. Y. Lim, R. Aylett, K. Leichtenstern, L. Hall and P. Rizzo, *A Case Study In Multimodal Interaction Design For Autonomous Game Characters*, in Proceeding of the 3rd Workshop on Multimodal Output Generation (MOG 2010), 2010.

2009

[Conference (full)] K. Leichtenstern and E. André, *Studying Multi-User Settings for Pervasive Games*, in Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2009), ACM, 2009.

[Conference (full)] K. Leichtenstern and E. André, *The Assisted User-Centred Generation and Evaluation of Pervasive Interfaces*, in Proceedings of the 3rd European Conference on Ambient Intelligence (Aml 2009), Springer, 2009.

[Conference (full)] M. Y. Lim, M. Kriegel, R. Aylett, S. Enz, N. Vanni, L. Hall, P. Rizzo and K. Leichtenstern, *Technology-Enhanced Role-Play for Intercultural Learning Contexts*, in Proceedings of the 8th International Conference on Entertainment Computing (ICEC 2009), Springer, 2009.

[Conference (short)] E. Kurdyukova, E. André and K. Leichtenstern, *Introducing Multiple Interaction Devices to Interactive Storytelling: Experiences from Practice*, in Proceedings of the 2nd International Conference on Interactive Digital Storytelling (ICIDS 2009), Springer, 2009.

[Demo] M. Kriegel, M. Y. Lim, J. Dias, K. Leichtenstern, W. C. Ho and P. Rizzo, *ORIENT: Interactive Agents for Stage-based Role-play*, in Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), ACM, 2009.

[Workshop] G. Kahl, K. Leichtenstern, J. Schöning, L. Spassova and A. Krüger, *A Contextual Learning Game for Toddlers Installed on an Interactive Display Attached to a Shopping Cart*, in Proceedings of the 2nd Workshop on Pervasive Computing Education (PerEd 2009), 2009.

2008 [Conference (full)] K. Leichtenstern and E. André, *User-Centred Development of Mobile Interfaces to a Pervasive Computing Environment*, in Proceedings of the 1st International Conference on Advances in Computer-Human Interaction (ACHI 2008), IEEE, 2008.

[Poster] K. Leichtenstern, D. Erdmann and E. André, *EVAL - An Evaluation Component for Mobile Interfaces*, in Proceedings of the 10th International Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2008), ACM, 2008.

2007 [Conference (full)] K. Leichtenstern, E. André and T. Vogt, *Role Assignment via Physical Mobile Interaction Techniques in Mobile Multi-User Applications for Children*, in Proceedings of the 1st European Conference on Ambient Intelligence (Aml 2007), Springer, 2007.

[Conference (full)] E. Rukzio, G. Broll, K. Leichtenstern and A. Schmidt, *Mobile Interaction with the Real World: An Evaluation and Comparison of Physical Mobile Interaction Techniques*, in Proceedings of the 1st European Conference on Ambient Intelligence (Aml 2007), Springer, 2007.

[Conference (short)] K. Leichtenstern, M. Kranz, P. Holleis, E. Lössch and E. André, *A Tangible User Interface as Interaction and Presentation Device to a Social Learning Software*, in Proceedings of the 4th International Conference on Networked Sensing Systems (INSS 2007), IEEE, 2007.

2006

[Conference (full)] E. Rukzio, K. Leichtenstern, V. Callaghan, A. Schmidt, P. Holleis and J. Chin, *An Experimental Comparison of Physical Mobile Interaction Techniques: Touching, Pointing and Scanning*, in Proceedings of the 8th International Conference on Ubiquitous Computing (UbiComp 2006), Springer, 2006.

[Poster] K. Leichtenstern, E. Rukzio, V. Callaghan and A. Schmidt, *Mobile Interaction in Smart Environments*, in Adjunct Proceedings of the 4th International Conference on Pervasive Computing (Pervasive 2006), 2006.

[Workshop] K. Leichtenstern and E. André, *Social Mobile Interaction using Tangible User Interfaces and Mobile Phones*, in Proceedings of the 36. Jahrestagung der Gesellschaft für Informatik (Informatik 2006), 2006.

2005

[Workshop] K. Leichtenstern, A. De Luca and E. Rukzio, *Analysis of Built-in Mobile Phone Sensors for Supporting Interactions with the Real World*, in Proceedings of the 1st Workshop on Pervasive Mobile Interaction Devices (PERMID 2005), 2005.