Tool-Supported User-Centred Prototyping of Mobile Applications

Karin Leichtenstern, Augsburg University, Germany

Elisabeth André, Augsburg University, Germany

Matthias Rehm, University of Aalborg, Denmark

ABSTRACT

There is evidence that user-centred development increases the user-friendliness of resulting products and thus the distinguishing features compared to products of competitors. However, the user-centred development requires comprehensive software and usability engineering skills to keep the process both cost-effective and time-effective. This paper covers that problem and provides insights in so-called user-centred prototyping (UCP) tools which support the production of prototypes as well as their evaluation with end-users. In particular, UCP tool called MoPeDT (Pervasive Interface Development Toolkit for Mobile Phones) is introduced. It provides assistance to interface developers of applications where mobile phones are used as interaction devices to a user's everyday pervasive environment. Based on found tool features for UCP tools, a feature study is described between related tools and MoPeDT as well as a comparative user study between this tool and a traditional approach. A further focus of the paper is the tool-supported execution of empiric evaluations.

Keywords: *Computer Science, Human-Centred Design, Human-Computer Interaction, Hybrid Simulation, Mobile Phone Application Development, Pervasive Environments, User-Centred Prototyping Tools, User Evaluation of Mobile Applications*

INTRODUCTION

Recent years have brought the tendency to develop mobile applications in human-centred iterations (ISO norm 13407) in order to increase the user-friendliness (Nielsen, 1994; Rogers, Sharp, & Preece, 2002) of the final product. Usually, this process includes a phase to (1) understand and specify the context of use, (2) specify the user and organisational requirements (3) produce design solutions, and (4) evaluate these design solutions against the requirements. With the continuous involvement of end-users in empiric user evaluations, these phases are iteratively executed until the intended application fulfils all requirements of the end-users (ISO norm 13407). All phases of this human-centred process can be very time-consuming and error-prone and thus they can increase the development costs and time. In particular, the

DOI: 10.4018/jhcr.2011070101

third phase requires experienced programming skills since different kinds of prototypes need to be efficiently and effectively designed and implemented. Also, the fourth phase involves expert knowledge in usability engineering due to the fact that user evaluations need to be properly conducted and analysed in order to efficiently and effectively verify whether the requirements are met.

These are causes why Myers (1995) argues for the development and application of tools or toolkits in order to save time and money. Myers points out two main requirements for tools. Firstly, tools need to improve the result of a development process: the quality of the resulting product. Secondly, tools should also enhance the process itself: the ease of use and efficiency to run through the process.

In this paper we present the approach of all-in-one tools which support both the third and fourth phase of the human-centred design process. We call these tools user-centred prototyping (UCP) tools since they support the design and implementation of prototypes (third phase) as well as the prototypes' evaluation and analysis (fourth phase) with end-users. As an example we introduce our UCP tool called MoPeDT (Pervasive Interface Development Toolkit for Mobile Phones) that supports the UCP of mobile applications where mobile phones are used as interaction device in a user's everyday environment. This concept follows Alan Kay's term *Third Paradigm Computing* (http://www.ubiq.com/hypertext/weiser/UbiHome.html) and the concept of *Pervasive* or *Ubiquitous Computing* (Weiser, 1991) where users can either (1) directly interact with real world objects of the user's everyday life or (2) mediated via an interaction device.

In the following, we illustrate typical steps of the process that we call user-centred prototyping – the third and fourth phase of the human-centred design process. These steps base on our practical experience (Rukzio, Leichtenstern, Callaghan, Schmidt, Holleis, & Chin, 2006). Then we describe existing tool-support for the UCP and typical tool features. The main part of the paper addresses MoPeDT and its validation via a feature and user study. Finally, we cover ideas to tool-supported execute empiric evaluations.

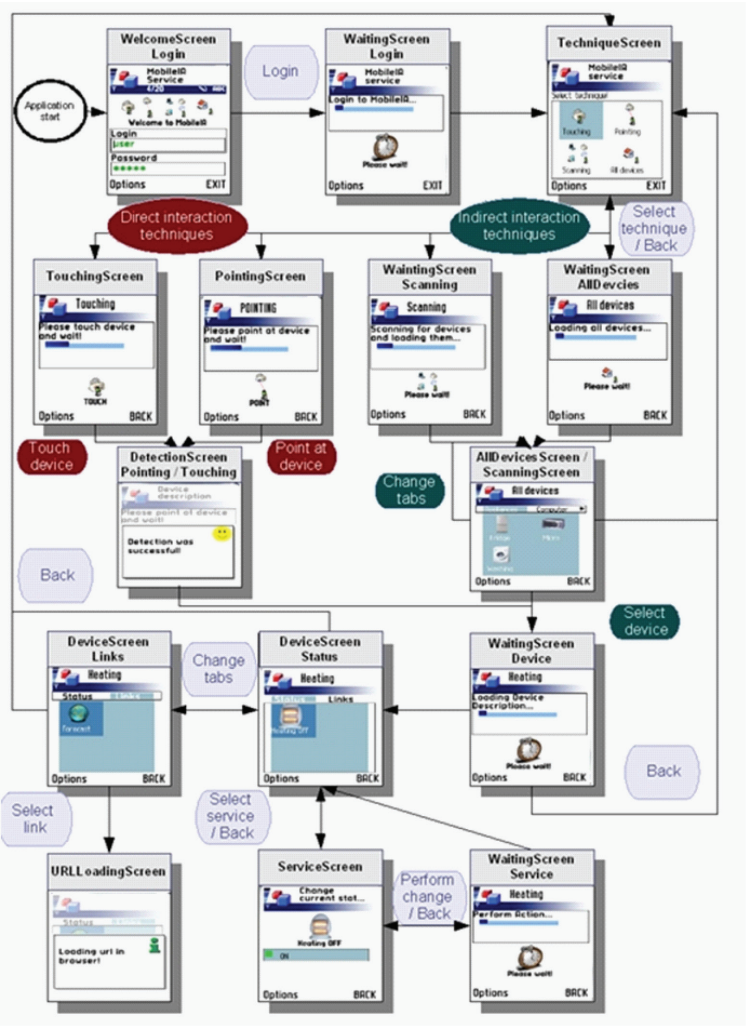
Design Specification of Prototypes

When producing a design solution, the application's appearance and behaviour is typically first designed and later on implemented. For mobile applications, the design is often done by defining a model for the different screens and the application flow which typically resembles a state chart. In this model, 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 since thereby the models can quickly and easily be changed. High-fidelity prototypes are often designed with the support of graphical tools (Rogers, Sharp, & Preece, 2002). Their look and feel is often quite similar to the final product. At early stages of the UCP, the design specification characteristically is less detailed than in later iterations. Figure 1 illustrates a high-fidelity design specification of a mobile application.

Implementation of Prototypes

After having specified the mobile application in the design phase, the prototype can be implemented. In terms of interface design, a prototype represents a partial simulation of a product with respect to its final appearance and behaviour (Houde & Hill, 1997). Prototypes can be classified by their level of detail, range of functions and reusability. Similar to the design specification, the prototypes can be implemented with a low or high level of detail (Nielsen, 1994; Rogers, Sharp, & Preece, 2002). Low-fidelity prototypes normally are prototypes

Figure 1. High-fidelity design specification of the appearance and behaviour of a mobile application



which are implemented with paper (paper prototypes) while high-fidelity prototypes typically are running applications for the intended interaction device (e.g. mobile phones). High-fidelity prototypes for mobile phones 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). Both of these kinds of prototypes are usually restricted by their range of functions. Their supported functions can be limited vertically or horizontally (Nielsen, 1994; Rogers, Sharp, & Preece, 2002). Horizontal prototypes provide an overview of all functionalities but do not provide these functionalities in depth whereas vertical prototypes provide functionality in depth but not for all functionalities. A further prototype classification aims at the prototype's reusability in the following iterations (Davis, 1992). Throwaway prototypes are not reused. For instance, low-fidelity prototypes (e.g. paper prototypes) are often thrown away after a single iteration. 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).

Evaluation of Prototypes

After having produced a design solution, this solution is typically evaluated with end-users in empiric evaluations (Nielsen, 1994). Afterwards, the results of the evaluation are analysed whether the user's requirements are fulfilled or not. In empiric evaluations, the end-users apply the implemented prototypes to either execute pre-defined tasks or freely use it. During this usage, observation techniques (Rogers, Sharp, & Preece, 2002) are applied in order to record different objective data of the user evaluation, such as user behaviour via cameras and microphones or user interactions via logging mechanisms. In addition to the observation techniques, inquiry techniques (e.g. interviews or questionnaires) are also often applied to gather subjective data (Rogers, Sharp, & Preece, 2002). The subjective data can often help to interpret the gathered objective data, such as why the users preferred a specific functionality of the prototype.

Analysis of Data

The last step is the analysis of the captured objective and subjective data (Rogers, Sharp, & Preece, 2002) in order to find usability problems and whether the user requirements are met. The gathered 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. 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 (e.g. a video file). Now, these quantified data can be analysed statistically in order to find significant differences between prototypes, such as an increased user involvement when using a prototype compared to the others. After the analysis and interpretation of the gathered data, a new iteration is started with a modification of the design specification if the requirements of the users are still not completely fulfilled otherwise the iterations terminate.

TOOL-SUPPORT FOR USER-CENTRED PROTOTYPING

In the remaining paper we call the design and implementation of prototypes as well as their evaluation and analysis as the four main phases that need to be conducted when executing UCP iterations. In our research we aim at all-in-one tools which support all of these steps. Using these UCP tools, however, the implementation phase is not required anymore since evolutionary prototypes with high level of detail are automatically generated as results of the design specification. The generated prototypes provide the designed functionalities and directly run on the respective interaction devices, such as mobile phones. For instance, MoPeDT generates applications that run on mobile phones which support Java2Mircoedition (J2ME). More details about MoPeDT's generated prototypes are given in the next section. Applying functional high-fidelity prototypes enables rather realistic empiric evaluations and gives the user a clearer picture of the prototype (Holmquist, 2005) than prototypes with a low fidelity and a strongly limited range of functions. 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. A last important aspect of UCP tools is the support to execute remote evaluations. A remote evaluation normally means a spatial and / or temporal separation of the subjects and evaluators (Andreasen, Nielsen, Schroder, & Stage, 2007) during the execution of an empiric evaluation in a laboratory or in a field setting (in-situ, i.e. at home).

Tools for User-Centred Prototyping

The only known tools which support all phases of the UCP are d.tools Hartmann et al. (2006) and SUEDE (Klemmer, Sinha, Chen, Landay, Aboobaker, & Wang, 2000). Klemmer et al. (2000) developed SUEDE that assists in the iterative development of speech interfaces whereas Hartmann and colleagues implemented d.tools that supports the design, evaluation and analysis of physical computing applications. SUEDE is used to design dialogue examples, evaluate the examples in a Wizard of Oz setting and later on to analyse the evaluation, such as the user's used dialogue path during the test. SUEDE focuses on the execution of local evaluations in a laboratory setting similar to d.tools. 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. Using d.tools, the interaction devices (e.g. a media player), however, need to be connected to the 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.

Apart from the mentioned UCP tools, there are further tools which only support one or two phases of the UCP. In the remaining paper we concentrate on tools which address the development of mobile applications. Topiary (Li, Hong, & Landay, 2004), MScape (Hull, Clayton, & Melamed, 2004), TERESA (Chesta, Paternò, & Santoro, 2004), OIDE (McGee-Lennon, Ramsay, McGookin, & Gray, 2009), iStuff Mobile (Ballagas, Memon, Reiners, & Borchers, 2007), OmniSCOPE (de Sá & Carriço, 2009) and MakeIT (Holleis & Schmidt, 2008) assist the design of mobile applications. Topiary and MScape support developers of location-based applications for PDAs. The main difference between MScape and Topiary is the fact that Topiary only provides a mock-up of the application whereas MScape provides functional prototypes which do not require a Wizard to manually call GPS events of the application. MScape's prototypes directly run on PDAs. TERESA provides assistance for the tool-based design of functional nomadic interfaces (e.g. websites for mobile phones) and OIDE assists the generation of multimodal input that, for instance, can be used for mobile applications. The result of a design specification via OIDE, however, still requires comprehensive programming skills to generate a functional prototype which is similar to OmniSCOPE, MakeIT and iStuff Mobile. iStuff Mobile supports the design of mobile applications but only with a focus on the sensor-based input. OmniSCOPE and MakeIT similarly support the design specification of a prototype's appearance but MakeIT additionally assists in the specification of the behaviour and the execution of analytic evaluations (Holleis, Otto, Hussmann, & Schmidt, 2007).

For the evaluation and analysis phases, we only found few tools. These tools primarily focus on a support for the evaluation in-situ (field studies): MyExperience (Froehlich, Chen, Consolvo, Harrison, & Landay, 2007), ContextPhone (Raento, Oulasvirta, Petit, & Toivonen, 2005) and Momento (Carter, Mankoff, & Heer, 2007). Via these tools, different user interactions

as well as active and passive user contexts can be logged for later on analyses. For instance, GPS locations of the users (passive user context) can be logged as well as text notes or images (active user context).

Tool Features for the User-Centred Prototyping

All mentioned tools provide ideas for tool features of a UCP tool.

For the design, a graphical user interface is usually provided that enables the modelling of the appearance and behaviour. Characteristically, a state-chart is visualised (e.g. see d.tools) that can be edited by the interface designers in order to specify the screens and user interactions. Since most of the mentioned tools (e.g. d.tools and SUEDE) rather focus on local stand-alone applications for specific devices with static content, the content of their prototypes is known during the development time. This aspect simplifies the design specification because the multimedia content can directly be assigned to a screen. Content, however, is often not known at the specification time but instead just at runtime. For instance, MScape dynamically displays content dependent on the user's outdoor location. To specify such a dynamic appearance and behaviour, interface designers make use of a scripting language during the design.

In the context of the *Third Paradigm,* a tool is additionally expected to support the specification of user interactions which base on novel input channels. Classically, users can interact with an application via a keyboard or mouse. In the context of *the Third Paradigm* users rather apply more natural input channels to interact with an application (e.g. via a camera, microphone, GPS receiver and NFC reader). In order to enable interactions via these input channels, the design tools need to support the specification of novel user interactions. For instance, MScape and OIDE provide assistance for the specification and application of pervasive or ubiquitous techniques. MScape enables the design specification of GPS-based locations which can be applied as user input in a mobile application. For their specification they make use of maps which can be loaded and edited in order to input so-called location-based points of interest, such as places with interesting sightseeing in a town.

Usually, the synchronous recording of all user interactions is supported in the evaluation phase (e.g. MyExperience or Momento). The only analysis of the logged user interactions often does not provide a covering insight to user behaviour and preferences. Instead, the analysis of the recorded user interactions in combination with the audio-visual content can provide valuable data for interpretations. D.tools is an example that supports the synchronised recording of user interactions and audio-visual content but does not consider user context. MyExperience or Momento are examples which enable the logging of active and passive user context, such as passively the user's GPS locations or actively the user's captured images and text notes. Momento additionally enables a remote communication between the evaluator and the subjects during the execution of a user evaluation whereas MyExperience enables the recording of the prototype's appearance since screenshots of the subject's mobile phone can be captured.

The main objective of the analysis phase is to find problems of a prototype. Additionally, the analysis also helps answer questions about user behaviour or preferences in different situations. A software tool is expected to appropriately display all captured data to easily and quickly enable the analysis. A common feature is to synchronously display the audio-visual content as well as the other captured data in a time-line based GUI. By this means, the audio-visual content is pre-annotated. The developers do not need to perform this by hand anymore which is a time-consuming and annoying task. Now, the developer can immediately scroll through the pre-annotated video or jump to intended data. Additionally, the developer can add, edit or remove annotations. D.tools applies the interactive time-line based visualisation of the logged data as well as the audio-visual content.

MOPEDT: A USER-CENTRED PROTOTYPING TOOL FOR MOBILE PHONES

Based on the knowledge of the related tools and their features, we investigated further aspects of the tool-supported UCP, such as additional desirable tool features as well as aspects on user acceptance and problems when using UCP tools. The focus of our research was not to find generic solutions for the mobile application development but instead to develop a UCP tool as a test bed to investigate and improve the tool-supported UCP process.

MoPeDT's application domain bases on Alan Kay's term *Third Paradigm Computing* and the concept of *Pervasive* or *Ubiquitous Computing* (Weiser, 1991) where users can either (1) directly interact with real world objects of the user's everyday life or (2) mediated via an interaction device. MoPeDT (Leichtenstern & André, 2010) focuses on the second aspect where mobile phones are used as interaction devices to a pervasive environment (e.g. a store) and their physical objects (e.g. products in the store). The generated applications of MoPeDT can be used to select a physical object and call its services. Services offer different kinds of contents about the physical object (e.g. a description about the ingredients of a product) but also provide opportunities to generate and provide multimedia content to other users (e.g. user reviews about a product). In contrast to d.tools and SUEDE, MoPeDT generates evolutionary prototypes with a high fidelity that directly run on the intended interaction. These prototypes can support different mobile user interactions (Leichtenstern & André, 2009) to select and use services of physical objects (e.g. via the mobile phone's keyboard, NFC reader, microphone). The generated prototypes run on mobile phones which support Java2Microedition (J2ME) and the corresponding hardware for the intended mobile user interactions (e.g. NFC reader). Since mobile devices of the different operators (e.g. Nokia, Samsung or Sony Ericson) often have different operating systems and implementations of J2ME, we decided to focus on S40 and S60 devices from Nokia. We successfully tested our automatically generated prototypes on Nokia 6131 NFC and Nokia N95 phones. In empiric evaluations, the generated prototypes do not require a Wizard as with SUEDE or a connected desktop PC as with d.tools but instead the subjects and the evaluation can be remotely.

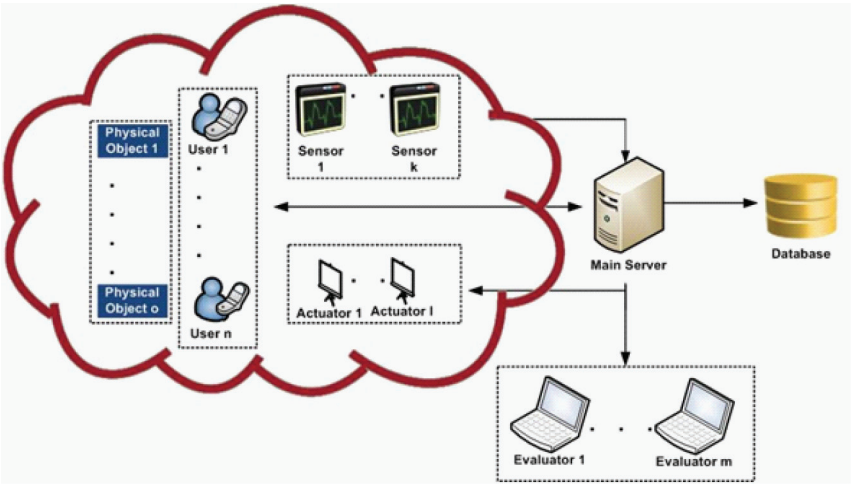
The following requirements for a UCP tool should be fulfilled for the design: (1) Static and dynamic specifications of the appearance and behaviour should be enabled. (2) Also, different mobile user interaction should be supported. (3) High-fidelity prototypes should automatically be generated as results of the design specification. (4) The generated applications should enable remote access to load and display remote content. (5) Approved interface guidelines should automatically be considered to increase the user-friendliness of the resulting prototypes. For the evaluation, the following requirements should be fulfilled: Recordings of (1) user interactions, (2) audio-visual content, (3) live comments and the prototype's appearance, (4) active and passive user context and (5) environmental contexts should be supported. Finally, the tool should support field and laboratory studies (6) locally and (7) remotely. The analysis component should (1) synchronously display all recorded data, (2) automatically annotate the audio-visual content, (3) enable a modification of the annotations and enable the (4) export of the data for statistical analyses. To enable a UCP tool that support these features, an appropriate architecture and several software modules are required.

Architecture and Software Modules

Our UCP tool bases on a plug&play client-server architecture that contains physical objects, mobile clients, a server, a database as well as sensors, actuators and evaluators as components (Figure 2).

Physical objects are real objects in a pervasive environment (e.g. objects of art or products in a store). In order to address physi-

Figure 2. The client-server architecture of MoPeDT



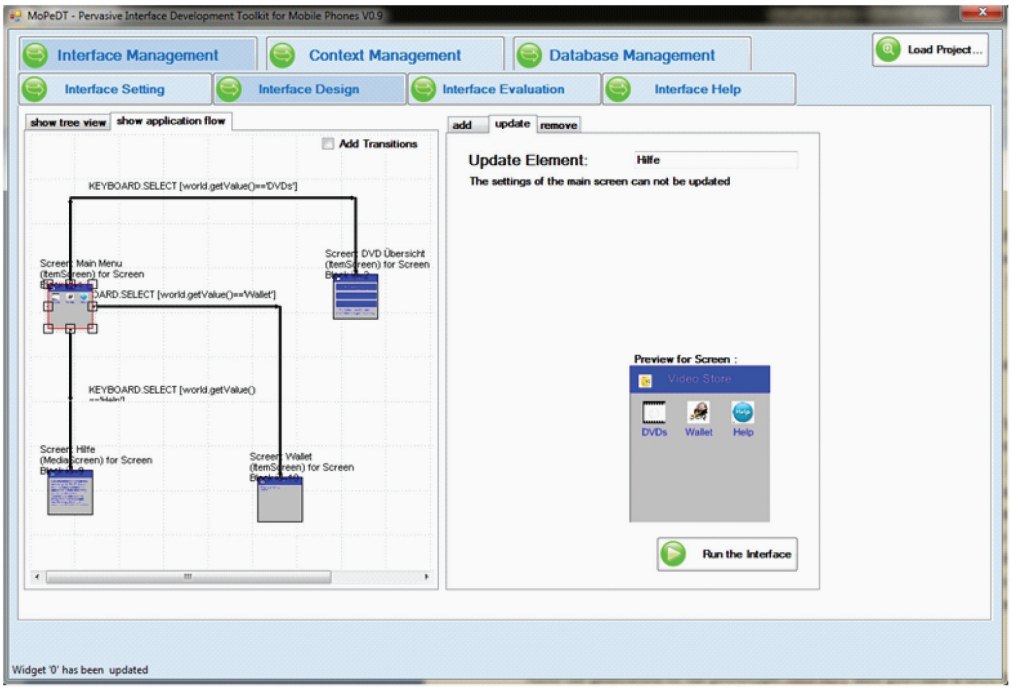
cal objects (e.g. via the mobile phone's keyboard or NFC reader), users can apply their mobile phones with an application that bases on the software module of the *mobile client*. After having selected a physical object, the *mobile client* communicates with the *server* to load services and contents which are stored in the *database*. *Sensors*, a further component, can also be plugged in to the *server* in order to provide knowledge about the user's environmental context (e.g. lighting condition or loudness). This knowledge can help to interpret user behaviour in the analysis phase. Another component called *evaluator* is applied whenever tool-supported evaluations are executed. Several *evaluators* can connect to the *server* and register their interest in other connected components: *mobile clients* and *sensors*. Now, the *evaluators* can synchronously log all contexts of the selected *mobile clients* (user interactions as well as active and passive user context) and *sensors* (environmental context) for a later on analysis. The last plug&play component of the architecture is the *actuator* that can be used as an additional output channel for multimedia content, such as to display video content on a public display (Leichtenstern & André, 2009).

For all components we provide software modules in J2SE or J2ME (Leichtenstern & André, 2009). One interesting software module is the *mobile client*. This module handles the whole mobile client-server communication and contains the implementation of different mobile user interactions (e.g. based on the keyboard, microphone, NFC reader and GPS receiver) for the mobile phone. For the specification of the intended mobile user interactions and user context as well as the static and dynamic application's appearance and behaviour, XML files are used. The software module of the *mobile client* can interpret these XML specifications and correctly display the application at the runtime. For the specification of the appearance and behaviour, the *mobile client* provides screen templates with different layouts (e.g. ItemScreen, MediaScreen or InfoScreen). These templates base on mobile phone guidelines from Nokia's Design and User Experience Library (http://library.forum.nokia.com). Due to these templates, the generated prototypes fulfil approved interface guidelines. For instance, reversibility is considered as well as a consistent layout, soft key usage and navigation style.

User-Centred Prototyping with MoPeDT

Based on our architecture and the software modules, the UCP tool was built.

Figure 3. The design component of MoPeDT. The left part of the component displays the state chart view of the application that contains the screen states and transitions while the right part of the design component provides a preview of the selected screen and options to modify the design specification.

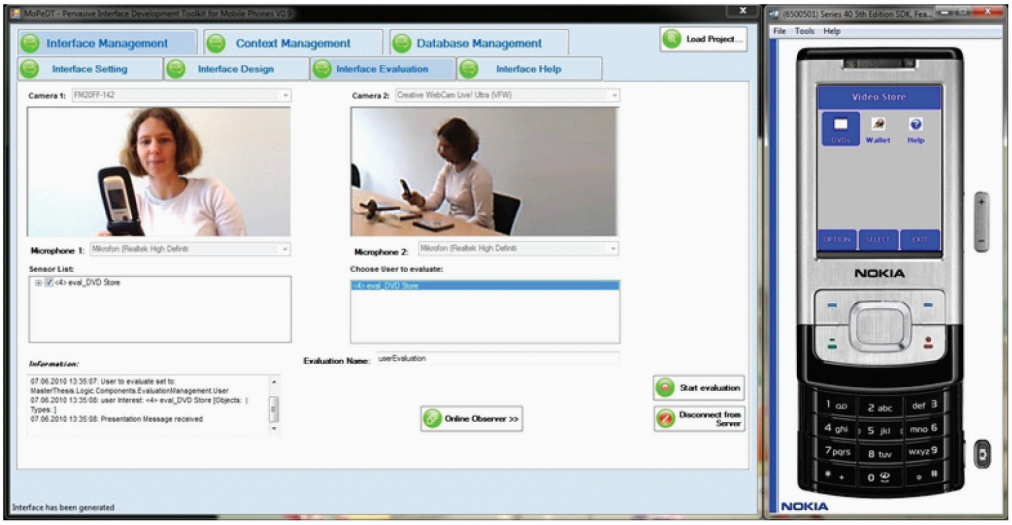


Similar to the related tools, the design component of MoPeDT (Figure 3) supports the GUI-based specification of the appearance and behaviour as well as the specification of different novel user interactions (e.g. based on the keyboard, NFC and microphone). Dynamic appearance and behaviour are specified by making use of a scripting language (Leichtenstern & André, 2009). At the runtime, the dynamic specified screens display local content or remote content from the database. This database content can also be specified via the GUI of MoPeDT. The automatically generated prototypes of the design specification are executable JAR files which directly run on the end-device.

Besides the generation of prototypes, MoPeDT also provides a GUI to capture user interactions and audio-visual content in the evaluation phase (Figure 4). In addition to most other evaluation and analysis tools, MoPeDT assists the synchronous logging of live comments and task descriptions as well as the capturing of the prototype's appearance at the evaluation time. To capture the prototype's appearance, a cloned screen of the subject's mobile phone is displayed on the desktop PC of the evaluator. Now, whenever a new screen is loaded for the subject's device, the cloned screen is updated and captured. A last difference to the former mentioned tools is the possible recording of the already mentioned environmental context via the *sensor* component of the architecture. To execute an evaluation with MoPeDT, all required components of the evaluation, however, require synchronous clocks.

In the final step of the UCP, the developer can analyse captured user evaluations. MoPeDT's analysis component (Figure 5) also provides a time-line based visualisation of the

Figure 4. The evaluation component of MoPeDT. The upper part displays the two selected cameras while the lower part shows the selected mobile client and sensors as well as incoming messages from the server. The right part displays the cloned screen view of the selected mobile client.



recorded data in order to navigate through and interact with them (e.g. to find usability problems or user preferences). For the analysis component we extended ANVIL (Kipp, 2001), which supports the display of audio-visual content as well as the visualisation and modification of annotations at various freely definable time-line based tracks. 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. MoPeDT supports the export of the annotated data in different formats of statistic tools (e.g. SPSS) in order to investigate typical behaviours in particular contexts.

Feature Study with MoPeDT

We downloaded and tested all former mentioned tools which were available for download and reviewed them based on the established tool features. Tables 1, 2, and 3 illustrate the results of this feature study. The three tables show that d.tools, MScape, MyExperience and MoPeDT fulfil several of the identified features but have limitations. D.tools, for example, does not give support for the development of high-fidelity prototypes which directly run on the intended interaction device and thus does not enable remote evaluations which are required to execute studies in the field.

Nevertheless, it provides several useful features, such as the specification of novel user interactions. MScape supports dynamic specifications of the prototype's appearance and behaviour as well as the automatic generation of high-fidelity prototypes but does not consider interface guidelines or a remote access. Additionally, it neither provides assistance for the evaluation nor for the analysis. MyExperience covers several useful features for the execution of remote evaluations, such as the logging of user interactions but does neither provide a synchronised display of the recorded data in the analysis phase nor support for the complete design phase.

User Study with MoPeDT

We did not only wanted to evaluate MoPeDT based on a feature study but also via a user study with potential end-users of MoPeDT: interface developers (Leichtenstern & André, 2010).

Figure 5. The analysis component of MoPeDT. The upper part displays the captured videos and screenshots (right). The lower part provides a time-line based visualisation of the audio track as well as the labelling (annotation) of tasks, user interactions or contexts in the corresponding track.

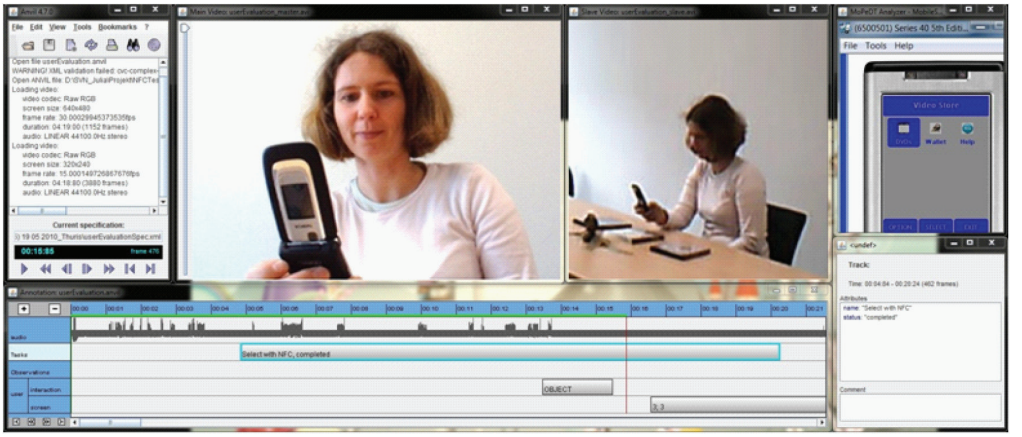


Table 1. Supported tool features for the design of prototypes: (1) Static and (2) Dynamic specifications of the appearance and behaviour; (3) Specifications of Novel User Interactions; (4) Auto-Generation of high-fidelity prototypes as result of the design specification. (5) Remote Access for the generated prototypes. (6) Automatic compliance of approved Interface Guidelines.

Tool	Static	Dynamic	Novel Interactions	Auto-Generation	Remote Access	Interface Guidelines
d.tools	+	-	+	-	-	-
SUEDE	+	-	+	-	-	-
TERESA	+	-	-	+	-	-
OIDE	-	-	+	-	-	-
MakeIT	+	-	+	-	-	-
MScape	+	+	+	+	-	-
Topiary	+	-	+	-	-	-
OmniSCOPE	+	-	-	-	-	-
iStuff Mobile	-	-	+	-	-	-
MoPeDT	+	+	+	+	+	+

With such a user study, we wanted to answer the question whether interface developers can quicker (Efficiency) develop user-friendlier products (Effectiveness) with a higher satisfaction (Satisfaction) compared to traditional approaches (e.g. via an IDE). Additionally, we searched for benefits and problems of UCP tools.

20 subjects participated in our user study and used MoPeDT and the traditional approach (with-in subjects approach) for the design, evaluation and analysis of a pervasive shopping assistant which helps users to receive information about products in a shopping store (e.g. about the ingredients of products). To prevent

Table 2. Supported tool features for the evaluation of prototypes: Recording of (1) User Interactions and (2) Audio-Visual Content; (3) Recordings of Live Comments, Tasks and Appearance of the prototype; (4) Remote Communication of evaluator and subjects; Recording of (5) active and passive User Contexts as well as (6) Environmental Context; Execution of (7) Local and (8) Remote Evaluations

Tool	User Interactions	Audio-Visual Content	Live Comments, Tasks and Appearance	Remote Communi- cation	User Context	Environmental Contex	Local Evaluations	Remote Evaluations
d.tools	+	+	-	-	-	-	+	-
SUEDE	+	-	-	-	-	-	+	-
My-Experience	+	+	+	-	+	-	+	+
Context-Phone	+	-	-	-	+	-	+	+
Momento	+	+	-	+	+	-	+	+
MoPeDT	+	+	+	-	+	+	+	+

Table 3. Supported tool features for the analysis of prototypes: (1) Synchronous Display of all recorded data; (2) Automatic Annotation of audio-visual content; (3) Modification of annotations; (4) Multi-Display of several recorded user evaluations; (5) Export to execute Statistical Analyses

Tool	Synchronous Display	Automatic Annotation	Modifications	Multi-Display	Statistical Analyses
d.tools	+	+	+	+	-
SUEDE	+	+	-	-	-
MyExperience	-	-	-	-	-
MoPeDT	+	+	+	-	+

any positioning effect, ten subjects started with the traditional approach and afterwards used MoPeDT whereas the other ten students used MoPeDT first. The subjects of our user study were students of our three-month course *Usability Engineering* with the age between 22 and 29 (M = 24.15, SD = 1.90). They rated themselves as medium skilled in object-oriented programming and usability engineering.

To keep comparability, the subjects received a detailed description about the intended prototype. For example, the screen contents were pre-defined as well as the user interactions which support must be given: keyboard-based and NFC-based. The subjects were also instructed to implement a logging mechanism when using the traditional approach to enable a recording of the user interactions in the evaluation. For the evaluation, the subjects were instructed to audio-visually capture three end-users and their user interactions while they were executing pre-defined tasks. The logging of user and environmental context were not considered. After the evaluation, the subjects had to analyse their captured data to find usability problems of the two prototypes. For instance, they validated the logged user interactions to find problems of efficiency.

Before we ran the one-month user study, we conducted tutorials within our course and

taught all subjects how to use MoPeDT for the UCP and how to implement and evaluate mobile phone prototypes using the IDEs Eclipse and Netbeans. During this period and the study, the subjects were not informed that we developed MoPeDT. Also, we comprehensively taught all subjects about usability in general, the human-centred design, mobile phone usability and Nokia's mobile phone guidelines. We reminded the subjects to apply these guidelines in our user study.

During the study, we used a post-task questionnaire to acquire subjective data whereas protocol recordings and a guideline review were utilised to collect objective data. In the questionnaire, we asked the subjects to rate statements for both levels in terms of efficiency, effectiveness, satisfaction, learnability, transparency, and user control. By using our protocol, the subjects documented their required time for the UCP's phases. Also, emerged problems had to be noted. Finally, we conducted a guideline review and investigated the resulted prototypes. An independent usability expert who was not involved in the development or evaluation of MoPeDT used the generated prototypes and investigated their violation against the mentioned Nokia guidelines.

After the conduction of the user study, we analysed the data in order to shed light in our main objectives whether MoPeDT can improve the usability for the interface developer and which benefits and problems emerge when using a UCP.

When analysing the protocols with regard to the interface developer's efficiency, on average, the required UCP time in minutes with traditional approaches ($M = 816.60$, $SD = 318.81$) was significantly higher compared to MoPeDT ($M = 266.65$, $SD = 208.14$), $t(19) = 9.2$, $p < 0.001$. During the design, the programming of the GUI and network communication required much more time whereas the annotation task impaired the interface developer's efficiency in the analysis phase. The qualitative and quantitative feedback of the questionnaire substantiates the results (Figure 6). Most subjects found the tool usage *quick and easy* and see a benefit in *the very quick prototyping and evaluation of applications*.

Regarding effectiveness, the qualitative and quantitative feedback (Figure 6) reveals no clear differences between the two levels. While most of the subjects highlighted the generated prototypes of MoPeDT as *beautiful* which *follow design guidelines*, they also claimed the limitation caused by the screen templates. Despite this moderate subjective data, prototypes from MoPeDT had, on average, less violations against the 22 guidelines from Nokia ($M = 0.85$, $SD = 0.93$) than prototypes from the traditional approaches ($M = 4.35$, $SD = 2.52$), $t(19) = 5.48$, $p < 0.001$. Typical problems were the inconsistent usage of the soft keys (17 of 20 subjects).

Despite the positive results in terms of effectiveness and efficiency, the interface developer's satisfaction turned out to be a problem (Figure 7). The weak satisfaction was mainly caused by a lack of an appropriate user control and transparency of MoPeDT. The interface developers *want to see what is going on in the background* and they want to have an increased level of freedom. Nevertheless, the subjects considered benefits for the learnability when using MoPeDT since they realised less required skills compared to traditional approaches.

Finally, we asked for the preferred levels: MoPeDT or traditional approach. Figure 8 shows the distribution. Most subjects either liked MoPeDT or both levels. Finally, we asked for the preferred components of MoPeDT (Figure 9). Most subjects liked all components and thus the all-in-one tool solution. A subject mentioned that *only the combination of all components meaningfully supports the iterative prototyping* whereas another subject mentioned that the prototyping can be improved by *the close interleaving of the three components* in order to prevent *the induction in several programs*.

Based on our results we conclude that MoPeDT improves the interface developer's efficiency by reducing the required time for the UCP as well as effectiveness by reducing the prototypes' non-compliance of interface guide-

Figure 6. User ratings for effectiveness and efficiency. The subjects rated the provided statements based on a scale from 1 (strongly disagree) to 5 (strongly agree). In terms of design, efficiency and time gain were significantly better rated for MoPeDT compared to the traditional approach (p < 0.05, ** p < 0.01, *** p < 0.001)*

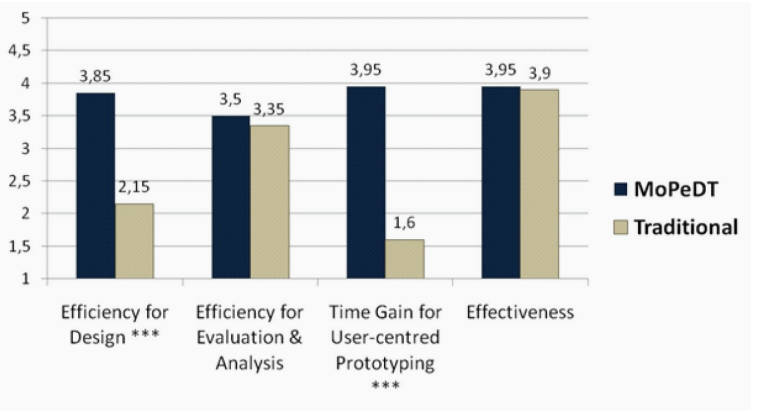
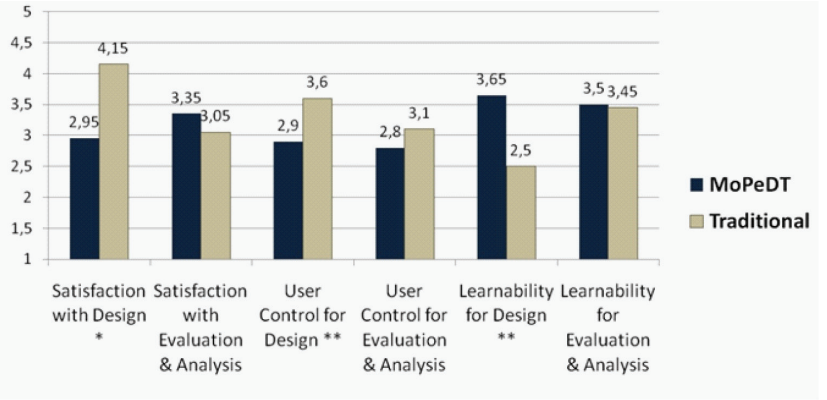


Figure 7. User ratings for satisfaction, user control and learnability. The subjects rated the provided statements based on a scale from 1 (strongly disagree) to 5 (strongly agree). In terms of design, satisfaction and user control were significantly better rated for the traditional approach compared to MoPeDT while learnability was better rated for MoPeDT (p < 0.05, ** p < 0.01, *** p < 0.001)*



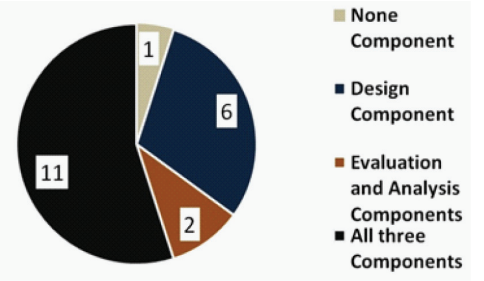
lines. The results additionally indicate that interface developers accept UCP tools for all steps of the UCP and that they give a priority to an all-in-one tool for the UCP instead of separate tools for the different phases. The results of our user study, however, also suggest problems when applying UCP tools. The interface developer's satisfaction is strongly linked to an appropriate user control as well as transparency that need to be indispensability considered.

EXECUTING EMPIRIC EVALUATIONS: OBSERVATION

One interesting aspect of the tool-supported UCP is the assistance to execute empiric evalu-

Figure 8. Preferred approach (MoPeDT or the traditional) for the design (left) as well as evaluation and analysis (right)

Figure 9. Preferred components of MoPeDT



ations. Most tools only support the execution of real world simulations in laboratories or real world evaluations in the field. A new idea is to also support a hybrid simulation with partial simulations in a virtual world. In the following we illustrate the tool-supported real world evaluation and simulation as well as the hybrid simulation.

Real World Evaluations and Simulations

The execution of empiric evaluations in the field (Häkkilä & Mäntyjärvi, 2006) is the most reasonable method. There is evidence that these studies provide realistic and valuable data because they are performed with real contextual constraints. Despite promising benefits, their conduction, however, might also lead to uncontrolled contextual settings rendering the outcome useless. Also, field tests are often time-consuming and expensive. Consequently, the idea is to simulate the real world in a more controlled environment and execute laboratory studies instead. Laboratory studies can be used during minor iterations of the UCP if appropriate methods are applied. Field tests, however, cannot be completely substituted and should be used at least at the end of the development process. There is a complementing effect between field and laboratory studies and thus both settings should be supported by UCP tools.

In the following we present an example of a laboratory study that we executed with MoPeDT. We simulated a DVD store (Figure 10) to investigate user trust in two different mobile user interactions: Keyboard-based and NFC-based (Figure 10). Figure 11 shows screens of the prototype that can be used to select and buy DVDs. If this evaluation should be executed in the field, the prototype needs to be generated in the same way.

After the generation, we asked 20 subjects to participate in our evaluation and perform different tasks (e.g. buy a DVD). During the execution of these different tasks the subjects were introduced to apply *the method of thinking aloud* (Nielsen, 1994) to express their thoughts in the different situations. We used MoPeDT to audio-visually capture the users and their

Figure 10. The user while interacting with the mobile phone's keyboard (left) and NFC reader (right)

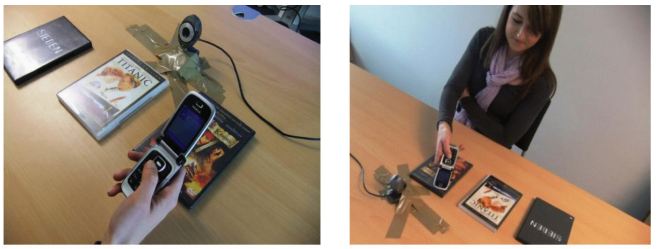


Figure 11. Some screens of the mobile prototype that was used in the user evaluation

user interactions. As a second method we applied a post-task questionnaire with questions in terms of user trust. This evaluation can be conducted in the same way if it is executed in the field but the experimenter needs to be locally present. A remote evaluation can also be executed but in this case the audio-visual recordings need to be dropped.

Later on, we reflected the captured user comments and interactions as well as the questionnaire to find differences of user trust. The procedure to analyse the captured data would have been the same for a field study. The results of the questionnaire revealed no significant differences. The analysis of the video showed that most subjects mentioned the NFC-based interaction as easier and quicker to use but not more trustworthy compared to the keyboard-based interaction.

Hybrid Simulations

Besides the traditional evaluations, MoPeDT can also be used to run hybrid simulations. Morla and Davies (2004) give a first impression of a hybrid simulation. Hybrid simulation means an integration and combination of the real and the virtual world. Morla and Davies (2004) used a virtual world to test real sensors which were attached to a wearable medical monitoring system. A similar approach called *dual reality* is introduced by Lifton and Paradiso (2009). They used SecondLife as a virtual visualisation tool of streams which are generated from real world sensors (e.g. the temperature in the real building or sound and lighting conditions). Driving simulators (http://www.carrsq.qut.edu.au/simulator) also aims at the idea of the hybrid simulation. Users interact with a real

steering wheel and dashboard while they are driving through a virtual presented test route. In our work, we do not use the virtual world as a visualisation platform for the performance of real devices as Davies or Lifton. We apply the virtual world as a platform for evaluations similarly as used for driving simulators but we use real mobile phones as interaction devices and a virtual simulation of a pervasive environment. The user still interacts with real mobile phones similar as in a real simulation but the pervasive environment and the physical objects are only virtually represented.

MoPeDT can be used to generate prototypes for mobile phones which are evaluated via the hybrid simulation but some adaptations are required (Figure 12). There is a need to shift the pervasive environment from the real to the virtual world using a platform that supports virtual simulations. The simulation contains the virtual representation of the physical objects but their services are still stored in the database and can be accessed with real mobile phones. Another difference to the former setting is the need for a representation of the user in the virtual world. With this avatar the user can interact via the PC's keyboard within the virtualised pervasive environment.

To simulate the pervasive environment, we make use of an open source version of SecondLife: Open Simulator (http://opensimulator.org). It allows setting up a virtual world that behaves exactly like SecondLife and can be accessed with the same viewers. Hence, in the remainder of this paper we will use SecondLife and Open Simulator as synonyms. SecondLife represents a multi-player platform that is not primarily concerned with gaming but aims at establishing a general virtual meeting place (e.g. buying or selling virtual or real goods). Central feature of SecondLife is the use of avatars which represent the real user in the virtual environment.

We propose to employ SecondLife to simulate a real environment which has been augmented for context dependent interactions. Apart from setting up the simulation server, three steps are necessary for simulating a pervasive environment in a hybrid 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 device. Standard modelling tools or in-world modelling tools can be used to model the virtual world. Figure 13 shows a snapshot from a pervasive environment. The challenge of a hybrid simulation is to realise the complex interplay between sensors, physical objects, and the mobile device, which can be seen as the inherent characteristic of a pervasive environment. The general idea is to use the real mobile phone for the interaction with the virtual world. This is not always possible. For NFC-based interactions, objects are equipped with RFID tags to allow NFC with the mobile phone (Figure 13). Creating a virtual RFID tag is no challenge but of course this tag cannot be read out by the real mobile device. Thus, it is necessary to create a virtual representation of the mobile device for some of the contextual input. In the current version, a virtual mobile device is used for registering the contextual input that is provided by the simulated environment. Then, the virtual phone communicates with MoPeDT's *main server* (Figure 12) in order to transmit events which lead to an adaptation of the real phone.

As a proof-of-concept study of the hybrid simulation, we replicated a former evaluation that was conducted in a laboratory (Rukzio, Leichtenstern, Callaghan, Schmidt, Holleis, & Chin, 2006) this time we made use of a hybrid simulation (Leichtenstern, André, & Rehm, 2010). In the original and in the replicated evaluation we compared three different mobile user interactions in different contextual situations, such as different locations of users and physical objects. The results of the hybrid simulation provided similar insights in user preferences and behaviour as the laboratory setting: users tended to switch the mobile user interactions dependent on the respective context with location as the most important context. Based on this proof-of-concept study, we see a first indicator that hybrid simulations might be a useful evaluation setting.

Figure 12. Modifications of the architecture to execute a hybrid simulation

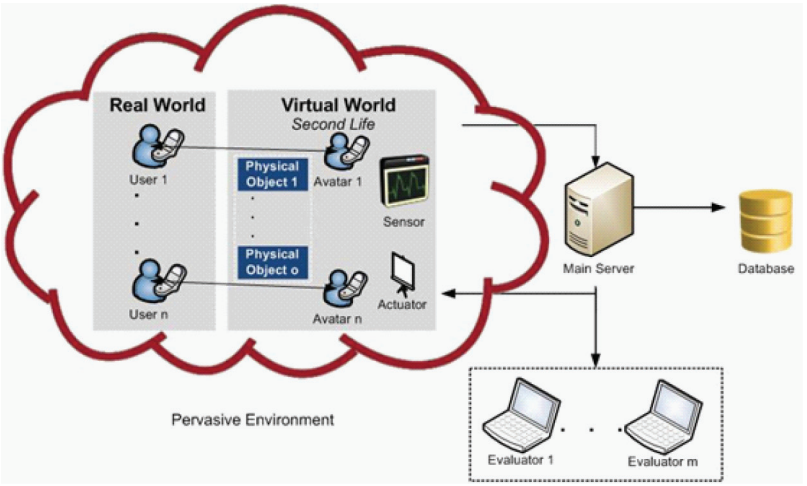


Figure 13. Snapshot of the virtualized environment and the user while interacting via NFC with the virtualized physical object

Overall, in some points the hybrid simulation benefits compared to a laboratory setting. (1) There is no need to physically rebuild the environment in a laboratory which can save money and time. (2) Relying on the hybrid simulation, even initial ideas can easily and efficiently be mediated and investigated because the real mobile application can be tried out and demonstrated in the corresponding simulated environment. (3) Another benefit is the ease of changing the environment. Different models of physical objects can rapidly be generated, modified and deleted. Using SecondLife as virtual world 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 environment. (5) Because the application realises a multi-player platform over the internet, it can be accessed anywhere anytime. (6) This can also reduce the organisational effort of subject recruiting since the subjects do not need to be physical present in a laboratory. Of course some restrictions apply like the necessity of compatible mobile devices. (7) Finally, in contrast to a virtual simulation alone approach, the hybrid simulation can be performed more realistic.

Despite these promising benefits, there are also problems. Of course, an offset inevitably emerges between a real world and a hybrid

simulation. (1) The user requires less motivation and less physical effort to move and explore the virtual setting. (2) Also, user interactions might be different to its real usage (e.g. NFC-based). (3) Having interactions in the virtual simulation can also lead to usability problems. (4) Finally, the virtual world needs to be modelled as realistic as possible to reduce side effects.

We consider the need that UCP should also support hybrid simulations. Supporting evaluations in the field, laboratory or via a hybrid simulation can meet several objectives. At the beginning of the UCP the tool can help to execute hybrid simulations to virtually simulate first ideas of applications. Later on, real world simulations can be performed to increase the realism for the users. At the end of the process, the tool can be used to execute evaluations in the field.

CONCLUSION

In this paper we covered the idea to support interface developers with an all-in-one software solution to more efficiently and effectively execute the third and fourth phases of the human-centred design process (produce design solutions and evaluate them against user requirements) which we call user-centred prototyping (UCP) process: the design, evaluation and analysis of prototypes with the involvement of end-users. The paper provides a review of existing and new tool features for UCP tools as well as insights to their typical benefits and problems.

Based on our proof-of-concept tool called MoPeDT, we extended ideas of other UCP tools. For instance, we introduced ideas of an architecture to enable the generation of network-based mobile applications. By this means remote content of a database can be loaded and displayed but this also enables the conduction of remote user evaluations. Additionally, we described the usage of screen templates to consider the compliance of approved interface guidelines. Finally, we described concepts how UCP tools can be extended to support field and laboratory studies as well as hybrid simulations. Our new ideas for UCP tools, however, also revealed some problems. For instance, if a tool bases on screen templates the interface developers felt a strong limitation of their user control which in turn impaired their satisfaction with the tool.

ACKNOWLEDGMENTS

This research is partly sponsored by OC-Trust (FOR 1085) of the German research foundation (DFG). A special thank to Sebastian Thomas and Dennis Erdmann for their work on MoPeDT and to Julia Karcher and Michael Goj for their support during the conduction of some of the introduced studies.

REFERENCES

Andreasen, M. S., Nielsen, H. V., Schroder, S. O., & Stage, J. (2007). What happened to remote usability testing?: An empirical study of three methods. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1405-1414). New York, NY: ACM Press.

Ballagas, R., Memon, F., Reiners, R., & Borchers, J. (2007). iStuff mobile: Rapidly prototyping new mobile phone interfaces for ubiquitous computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1107-1116). New York, NY: ACM Press.

Carter, S., Mankoff, J., & Heer, J. (2007). Momento: Support for situated ubicomp experimentation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 125-134). New York, NY: ACM Press.

Chesta, C., Paternò, F., & Santoro, C. (2004). Methods and tools for designing and developing usable multi-platform interactive applications. *PsychNology Journal*, *2*(1), 123–139.

Davis, A. M. (1992). Operational prototyping: A new development approach. *IEEE Software*, *9*, 70–78. doi:10.1109/52.156899

de Sá, M., & Carriço, L. (2009). Mobile support for personalized therapies - OmniSCOPE: Richer artefacts and data collection. In *Proceedings of the 3rd International Conference on Pervasive Computing Technologies for Healthcare* (pp. 1-8). New York, NY: ACM Press.

Froehlich, J., Chen, M. Y., Consolvo, S., Harrison, B., & Landay, J. A. (2007). MyExperience: A system for in situ tracing and capturing of user feedback on mobile phones. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services* (pp. 57-70). New York, NY: ACM Press.

Häkkilä, J., & Mäntyjärvi, J. (2006). Developing design guidelines for context-aware mobile applications. In *Proceedings of the 3rd International Conference on Mobile Technology, Applications and Systems* (p. 24). New York, NY: ACM Press.

Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., et al. (2006). Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology* (pp. 299-308). New York, NY: ACM Press.

Holleis, P., Otto, F., Hussmann, H., & Schmidt, A. (2007). Keystroke-level model for advanced mobile phone interaction. In *Proceedings of the Conference on Human Factors in Computing Systems* (pp. 1505-1514). New York, NY: ACM Press.

Holleis, P., & Schmidt, A. (2008). MakeIt: Integrate user interaction times in the design process. In *Proceedings of the 6th International Conference on Pervasive Computing* (pp. 56-74).

Holmquist, L. E. (2005). Prototyping: Generating ideas or cargo cult designs? *Interaction*, *12*(2), 48–54. doi:10.1145/1052438.1052465

Houde, S., & Hill, C. (1997). What do prototypes prototype? In Helander, M., Landauer, T., & Prabhu, P. (Eds.), *Handbook of human-computer interaction*. Cambridge, MA: Elsevier Science.

Hull, R., Clayton, B., & Melamed, T. (2004). Rapid authoring of mediascapes. In *Proceedings of the 6th International Conference of Ubiquitous Computing* (pp. 125-142).

Kipp, M. (2001). Anvil - a generic annotation tool for multimodal dialogue. In *Proceedings of the 7th European Conference on Speech Communication and Technology* (pp. 1367-1370). New York, NY: ACM Press.

Klemmer, S. R., Sinha, A. K., Chen, J., Landay, J. A., Aboobaker, N., & Wang, A. (2000). Suede: A Wizard of Oz prototyping tool for speech user interfaces. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology* (pp. 1-10). New York, NY: ACM Press.

Leichtenstern, K., & André, E. (2009). Studying multi-user settings for pervasive games. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (pp. 190-199). New York, NY: ACM Press.

Leichtenstern, K., & André, E. (2009). The assisted user-centred generation and evaluation of pervasive. In *Proceedings of the Third European Conference on Ambient Intelligence* (pp. 245-255).

Leichtenstern, K., & André, E. (2010). MoPeDT - features and evaluation of a user-centred prototyping tool. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (pp. 93-102). New York, NY: ACM Press.

Leichtenstern, K., André, E., & Rehm, M. (2010). Using the hybrid simulation for early user evaluations. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction* (pp. 315-324). New York, NY: ACM Press.

Li, Y., Hong, J. I., & Landay, J. A. (2004). Topiary: A tool for prototyping location-enhanced applications. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (pp. 217-226). New York, NY: ACM Press.

Lifton, J., & Paradiso, J. A. (2009). Dual reality: Merging the real and virtual. In *Proceedings of the First International ICST Conference on Facets of Virtual Environments* (pp. 12-18).

McGee-Lennon, M. R., Ramsay, A., McGookin, D., & Gray, P. (2009). User evaluation of OIDE: A rapid prototyping platform for multimodal interaction. In *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (pp. 237-242). New York, NY: ACM Press.

Morla, R., & Davies, N. (2004). Evaluating a location-based application: A hybrid test and simulation environment. *IEEE Pervasive Computing / IEEE Computer Society [and] IEEE Communications Society*, *3*(3), 48–56. doi:10.1109/MPRV.2004.1321028

Myers, B. A. (1995). User interface software tools. *ACM Transactions on Computer-Human Interaction*, *2*(1), 64–103. doi:10.1145/200968.200971

Nielsen, J. (1994). *Usability engineering*. San Francisco, CA: Morgan Kaufmann.

Raento, M., Oulasvirta, A., Petit, R., & Toivonen, H. (2005). ContextPhone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing / IEEE Computer Society [and] IEEE Communications Society*, *4*(2), 51–59. doi:10.1109/MPRV.2005.29

Rogers, Y., Sharp, H., & Preece, J. (2002). *Interaction design: Beyond human-computer interaction*. New York, NY: John Wiley & Sons.

Rukzio, E., Leichtenstern, K., Callaghan, V., Schmidt, A., Holleis, P., & Chin, J. (2006). An experimental comparison of physical mobile interaction techniques: Touching, pointing and scanning. In *Proceedings of the Eighth International Conference on Ubiquitous Computing* (pp. 87-104).

Weiser, M. (1991). The computer for the 21st century. *Scientific American.*

Karin Leichtenstern is a PhD student and research assistant at Augsburg University. Before that, she graduated in Media Informatics at the University of Munich in 2006 and stayed a half of a year at the Intelligent Inhabited Environments Group at the University of Essex, UK. Her main research interests include HCI-related aspects of Pervasive and Mobile Computing as well as Usability Engineering. Karin's PhD aims at tool-support for interface designers of mobile applications when running through the human-centred design process.

Elisabeth André is a full professor of Computer Science at Augsburg University, Germany, and Chair of the Laboratory for Human-Centered Multimedia. Her current work focuses on multimodal interfaces, social signal processing and embodied conversational agents. Elisabeth André is on the Editorial Board of Cognitive Processing (International Quarterly of Cognitive Science), Universal Access to the Information Society: An Interdisciplinary Journal, AI Communications (AICOM), Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS), Journal on Multimodal Interfaces, Editorial Board of IEEE Transactions on Affective Computing (TAC), ACM Transactions on Intelligent Interactive Systems (TIIS) *and* International Journal of Synthetic Emotions (IJSE). *In 2007, Elisabeth André was appointed an Alcatel Research Fellow at Internationales Zentrum für Kultur- und Technikforschung of Stuttgart University (IZKT). In 2010, she was elected to be a member of the German Academy of Sciences Leopoldina and Academia Europaea.*

Matthias Rehm is an associate professor at CREATE, the Department of Architecture, Design and Media Technology at Aalborg University. He received his doctoral degree from Bielefeld University for a thesis on multimodal learning in virtual agents and has since worked in the area of social interactions focusing on embodied agents, multimodal and mobile interaction as well as cultural aspects of computing. He has been involved in a number of international research projects on a European level and beyond in the area of multimodal interactive systems and has over 50 publications in peer-reviewed journals and conferences.