MoPeDT - Features and Evaluation of a User-Centred Prototyping Tool

Karin Leichtenstern, Elisabeth André
Multimedia Concepts and Applications, Universitätsstr. 6a, 86159 Augsburg
{leichtenstern, andre}@informatik.uni-augsburg.de

ABSTRACT
User-Centred Prototyping (UCP) tools are expected to support interface developers in order to more efficiently, effectively and satisfactorily design, evaluate and analyse user-friendly products by an all-in-one tool solution. We developed such a UCP tool called MoPeDT that supports the user-centred development of interactive evolutionary prototypes for mobile phones in the context of *the Internet of Things*. In this paper we address our tool features for the design, evaluation and analysis phase that mostly base on meaningful features from related tools. Additionally, we cover potential enhancements to former UCP tools in order to support interface developers with a wide-ranging playground to investigate the user's behaviour and preferences. The paper also describes MoPeDT's evaluation with 20 students over one month that investigated the interface developer's efficiency, effectiveness and satisfaction when applying our UCP tool. As an outcome of this study we describe potential benefits and problems that might be of interest to other developers of UCP tools.

Author Keywords
User-centred Prototyping Tool, Evolutionary Prototypes, Pervasive Interface, Internet of Things, Mobile Phones

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Prototyping, User-Centred Design*

General Terms
Human Factors

INTRODUCTION

A product's usability is an important quality criterion that can determine about its success or failure. The user-centred design process is a widely established usability engineering process that can be used in order to obtain a good design [18, 24]. A characteristic feature of this process is the iterative prototyping that includes several iterations of designing interface prototypes along with continuous evaluations and analyses of these prototypes with end-users. The iterations with the involvement of the end-users are required to build the interface as user-friendly as possible. The application of this user-centred design process, however, adds new layers of complexity to the development task since the process requires combined knowledge about software and usability engineering. For instance, the developers of pervasive interfaces for mobile phones require knowledge about network programming and knowledge about how to address different built-in hardware, such as the microphone, camera, GPS interface and NFC reader. Additionally, interface developers need to know how to deal with limited input and output functionalities (e.g. small displays). Finally, they also require knowledge about approved interface guidelines as well as experiences in the conduction and analysis of user studies in order to get meaningful results. A lack of these experiences and skills can decrease the interface's quality and increase the required development time [27] as well as decrease the interface developer's motivation. Consequently, Myers [27] claims the development and application of tools or toolkits for interface development processes, such as the user-centred design process in order to save time and money. He describes two main requirements for a tool that appropriately supports in an interface development process. Firstly, the tool needs to improve the result of the process: (1) the quality of the resulted interfaces and thus the effectiveness of the interface developers. Secondly, the tool should also enhance the process itself: (2) the ease of use and efficiency to run through the process. Consequently, the interface developer's efficiency and satisfaction should be increased when applying the tool for a process. We assume that the so-called user-centred prototyping (UCP) tools can potentially meet Myers' requirements for the user-centred design process. They map the iterative cycle of the user-centred design process: a tool-supported design, evaluation and analysis of interface prototypes. We developed such a UCP tool called MoPeDT (Pervasive Interface Development Toolkit for Mobile Phones) to support interface developers of pervasive interfaces for mobile phones. The main motivation of MoPeDT's development was the knowledge acquisition of practical experience in order to address insufficiently researched questions for developers of UCP tools to finally answer the following questions: What are extensions to available UCP tools? What are features of MoPeDT that might provide an insight to useful UCP tool features? Which findings emerge when applying an all-in-one solution, such as MoPeDT for all three phases of the user-centred design pro-

EICS'10, June 19–23, 2010, Berlin, Germany.

cess that also point to general benefits and problems of UCP tools? The remaining paper is now structured as followed. After describing UCP tools and tool features of MoPeDT, the main part describes a user study that we conducted with MoPeDT.

USER-CENTRED PROTOTYPING TOOLS

The ISO norm 13407 abstractly describes the user-centred design process. This norm contains a cycle of designing, evaluating and analysing prototypical solutions of a product until they meet the specified user's requirements [18, 24]. This cycle is the basis of the user-centred prototyping tools' definition [9, 20]. (1) An all-in-one UCP tool supports interface developers in the tool-based design, evaluation and analysis of interface prototypes. Characteristically, UCP tools have a strong link between the design, evaluation and analysis component, e.g. the evaluation component supports the conduction of user studies with prototypes that were generated during the tool-based design and thus provide logging mechanism for the user studies. The analysis component assists designers with the interpretation of synchronously captured data via the evaluation component. (2) A further feature of UCP tools is the generation of evolutionary prototypes. In terms of interface design, a prototype represents a partially simulation of an interface with respect to its final appearance and behaviour [17]. Evolutionary prototypes are characteristically prototypes that are built in rapid iterative development cycles in order to experimentally find and validate the user's requirements. In each cycle, evolutionary prototypes are modified and tested with experts and end-users until they meet all requirements of the end-users [6, 7]. A UCP tool should generate evolutionary prototypes that provide several implemented functionalities of the final product and directly run on the respective interaction devices, e.g. mobile phones. These prototypes rather enable realistic user studies and prevent the user's misled understanding of the interface in user studies [16] than mock-ups of an interface that simulate several functionalities. (3) A last important aspect of UCP tools is the support to conduct remote usability studies. A remote usability study characteristically means a spatial separation of the subjects and evaluators [1] during a user study in a laboratory or in a field setting (in-situ, i.e. at home or at the office). Typically, in a remote usability study the user interactions (e.g. mouse clicks) are logged while the user is audio-visually captured. Moreover, screen shots or records are saved from the interface (e.g. [10]).

Tools that support all three process steps of the user-centred design in single software, the dynamic generation of interactive evolutionary prototypes and the conduction of usability studies are d.tools [9], SUEDE [20] and MoPeDT [21]. Klemmer and colleagues 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 analyse the evaluation, such as the user's used dialogue path during the tests. D.tools can be primarily applied to develop, test and analyse new information appliances in a laboratory, such as new media players or cameras and their buttons and sliders. User studies conducted with both tools point to benefits when supporting interface developers in all three steps of the iterative prototyping. When applying their UCP tools in the design, evaluation and analysis of prototypes, the interface developer's efficiency, effectiveness and satisfaction, however, have not been comprehensively investigated as we do with our tool called MoPeDT (Pervasive Development Toolkit for Mobile Phones) in order to reveal potential benefits and problems of UCP tools that support an all-in-one solution for the user-centred prototyping.

MoPeDT offers a wide-ranging playground to conduct different user studies in pervasive environments. Interface developers can user-centred develop pervasive interfaces for mobile phones, e.g. a pervasive shopping assistant or a pervasive game [22], remotely evaluate the interfaces in the pervasive environment and later on analyse the results. MoPeDT is the only known tool that supports interface developers during the user-centred development of interactive evolutionary prototypes for mobile phones in order to cope with the previously mentioned challenges, such as the comprehensively required programming and interface skills. Using MoPeDT, applications for mobile phones can be generated that support different pervasive interaction techniques [21] for the interaction with physical objects, e.g. products in a shopping store or objects of art in a museum. For instance, the mobile phone and its built-in NFC[1] reader can be applied to select a product via an RFID tag that is attached to it.

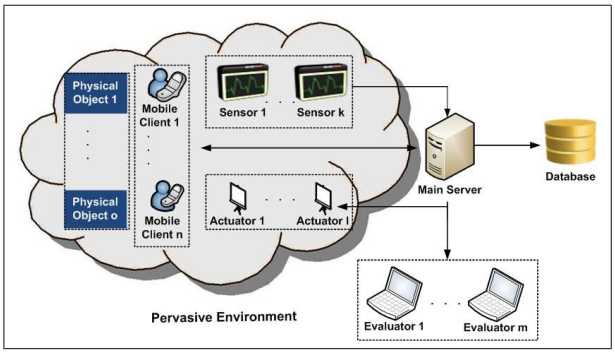

Figure 1. MoPeDT's Plug-and-Play Architecture

After having selected a physical object based on one of the supported interaction techniques, different services and their content are loaded and displayed on the mobile phone, such as a detailed description of the selected object or information about the object's origin. The idea to interact with physical objects and provide services to these objects is called *the Internet of Things* [25, 14]. In the context of *the Internet of Things* another example of an outdoor application is treasure hunt. Generating treasure hunt with MoPeDT, one or more mobile interaction techniques can be utilized that make use of input via the mobile phone's keyboard, NFC reader, GPS module (location), microphone (keyword-based speech

[1] http://www.nfc-forum.org/

recognition), camera (images recognition) or accelerometer (gesture recognition). During the game, the current location of the user can be detected. By this means, the number of the available physical objects can be automatically reduced. Now, at the current location a user can select a physical object via using one of the interaction techniques: by speaking its name, capturing a picture of it, performing a corresponding gesture, touching it with the mobile phone or selecting its GUI representation via the phone's keyboard. Then services and content of the selected object can be loaded, such as information about the next location or a quiz.

In order to support interface developers to user-centred develop and evaluate such prototypes in the term of *the Internet of Things*, MoPeDT employs a plug-and-play architecture and their software modules (see Fig. 1). The architecture contains physical objects, mobile clients, a server, a database as well as sensors, actuators and evaluators.

The idea is that end-users employ a generated evolutionary prototype on their mobile phones that utilize the software module for mobile clients. This module provides different functionalities, such as a network communication to the server or a support of the different mobile interaction techniques. By using the database, the server can answer all requests of these mobile clients about the physical objects, such as a list of all their supported services. Sensors (e.g. a temperature sensor) are also plugged in to the server in order to provide knowledge about the user's environmental context in user studies. These sensors can also be used as a further input channel for the mobile clients. By this mean, a particular context of a sensor (e.g. TEMPERATURE.HOT) might cause an adaptation of the mobile phone's appearance. A further architecture's component called *evaluator* is applied whenever tool-supported local or remote user studies have to be conducted. Several of these *evaluators* can connect to the server and register the interest in other connected components: mobile clients and sensors. Then, the *evaluators* can synchronously log all contexts of the selected mobile clients and sensors for later on analyses. Having all these contexts and user interactions synchronously captured together with videos, interface developers can shed light in the user's behaviour or preference in different contextual situations. The last plug-and-play component of the architecture is the *actuator* that can be used as an additional output channel for multimedia content. For instance, as described in our previous work [22], our architecture can be applied to generate a pervasive game that not only contains multiple users applying different mobile devices with different interaction techniques but also includes a public display presenting video content. After having generated the application based on our architecture, MoPeDT supports in the conduction of a user study and its analysis, such as to answer the question which multi-user setting of a pervasive game best supports collaboration [22]. Concluding, our plug-and-play architecture provides a meaningful basis for MoPeDT since its application easily enables the tool-supported generation and evaluation of prototypes for mobile phones considering a pervasive environment.

Consequently, MoPeDT can be applied in a completely different application domain compared to d.tools and SUEDE - *the Internet of Things* with mobile phones as pervasive interaction devices. Also, d.tools and SUEDE rather focus on local stand-alone applications for specific devices with static content. The content of their prototypes are known during the development time. This aspect simplifies the specification because the content (e.g. text or images) can directly be assigned to an interface element. In addition to their approach, we require network connections and the just described client-server architecture due to the fact that the content of MoPeDT's prototypes is often dynamically loaded and displayed at the runtime. As a consequence for the design component, the UCP tool requires a scripting language to enable static and dynamic specifications of content. A further important aspect is the fact that d.tools and SUEDE mainly apply their evaluation component in order to optimize their speech and physical interfaces during the evaluation phase but we are also interested in the investigation of the user's behaviour and preferences during using the interface. For instance, we are currently using MoPeDT for the investigation of user trust in different pervasive interaction techniques in the context of *the Internet of Things*. For the objectives of d.tools and SUEDE's users, laboratory settings are often adequate but in order to shed light on the user's behaviour and preferences, the user's environment and the user's contextual information additionally need to be considered. Consequently, with MoPeDT, we do not only support typical user tests in laboratory settings but also remote usability studies in-situ (e.g. at home or a museum) due to the plug-and-play architecture. However, there are also limitations when using MoPeDT mainly caused by two aspects. Firstly, interface developers only can develop interfaces in the context of *the Internet of Things* with mobile phones as interaction devices. Secondly, in contrast to SUEDE and d.tools, we use screen templates for the specification of the application that can potentially improve the usability of the interface but also can cause a limitation of the interface developers in their sphere of action.

The other related tools do not provide the whole pipeline of the user-centred prototyping (design - evaluation - analysis) and rarely the other two requirements of UCP tools: the generation of evolutionary prototypes that run on the end-user device and the support to conduct remote usability studies. SUPPLE [13], MakeIT [15], OIDE [26], Mobile Bristol [30] and TERESA [5] are examples of tools that support in the design of a prototype. SUPPLE, MakeIT and Mobile Bristol support developers of pervasive or ubiquitous interfaces. TERESA addresses the tool-based design of functional nomadic interfaces and OIDE the generation of multimodal interfaces. The mixed-fidelity prototyping tool from Sá [8] supports in the design and conduction of user studies of low-fidelity, mid-fidelity and high-fidelity prototypes for mobile phones. Compared to these tools, there are fewer tools that assist in the conduction and/or analysis of user studies. Most available evaluation and/or analysis tools support in the logging of web traffic and their visualisation (e.g. [2]). MyExperience [12], DRUM [23], Momento [4] and UMARA [3] are examples of tools that support in the conduction of

user studies and/or their analyses. MyExperience supports in the recording of evaluations for mobile phone applications whereas DRUM, UMARA and Momento are both evaluation and analysis tools that even concentrate on tests for mobile (Momento) or desktop settings (DRUM and UMARA).

MOPEDT'S COMPONENT FEATURES

In this section, we summarize MoPeDT's features for the design, evaluation and analysis components that were applied in our user study. MoPeDT's features mainly base on meaningful features of the just mentioned tools. Thus, the following overview also might provide an insight to feasible tool features for other UCP tools.

The Design Component

For the specification of the appearance and the behaviour of a prototype, SUEDE, d.tools, the mixed-fidelity prototyping tool from Sá, MakeIT and OIDE are examples that provide a graphical user interface (GUI) and visualise a state-chart diagram: screens or speech acts are represented by states whereas the user's interactions are represented by transitions. By these means, at the runtime, a user interaction triggers a specified transition and leads to another state and the display of a specified screen. MoPeDT also applies this approach (see Fig. 2). Since several views on the specification can provide benefits for the developer, MoPeDT does not only support a state-chart diagram view but also a tree view. For instance, MoPeDT's tree view supports a better overview when working on a specific screen whereas the state-chart view provides a better overview of the entire specification of the appearance and behaviour. A special feature of MoPeDT is the support to specify static and dynamic content of the interface's appearance. Most design tools focus a static content specification, such as the mixed-fidelity prototyping tool from Sá, SUEDE and d.tools. SUPPLE, Mobile Bristol as well as MoPeDT [21] facilitates a scripting language to specify dynamic content of interfaces, e.g. for multimedia presentations that are context-adaptively loaded and displayed. As previously described, MoPeDT also features the specification and employment of different interaction techniques that make use of the mobile phone's built-in hardware (e.g. the NFC reader). Mobile Bristol, SUPPLE, OIDE and MakeIT are examples that also provide assistance for the application of pervasive or ubiquitous interaction techniques. For instance, Mobile Bristol enables the specification of GPS-based locations that can be used as user input in a mobile application. Finally, in contrast to most other tools (e.g. d.tools and SUEDE), MoPeDT also assists in the compliance of approved interface guidelines in order to improve the quality of the resulted prototypes. Besides classical IDE's with integrated GUI builder (e.g. Netbeans), the mixed-fidelity prototyping tool from Sá and MoPeDT are the only tools that support in the compliance of interface guidelines. Sá's tool supports assistance to arrange the location and size of screen components for a consistent layout. MoPeDT facilitates the compliance of mobile phone guidelines by supporting interface developers with an expendable set of screen templates that base on Nokia's Design and User Experience Library[2]. For instance, these templates consider a consistent layout, softkey usage and navigation style. Each screen generated with MoPeDT has a heading, content and a softkey part. The left softkey is used for options, the middle key is used for conformations and navigations and the right softkey is used for negative actions (back, cancel or exit). Additionally, each screen contains a help and an option to return to the main menu. Moreover, from each screen the user can return to the previous state automatically. The result of MoPeDT's design component is an executable JAR file that makes use of the previously mentioned plug-and-play component called *mobile client*. The JAR file directly runs on the end-device. We successfully tested the generated prototypes on several Nokia phones (S40 and S60).

[2]http://library.forum.nokia.com

The Evaluation Component

A main feature of MoPeDT's evaluation component is the support to synchronously record all user interactions during a user study. For this feature, MoPeDT makes use of the architecture's plug-and-play component called *evaluator*. D.tools, the mixed-fidelity prototyping tool from Sá, MyExperience, DRUM, UMARA and Momento also log user interactions for the later analysis, e.g. in order to measure the occurrence of a specific event. Besides the recording of user interactions, a further feature of MoPeDT's evaluation component is the support to audio-visually record the user and her environment while she is interacting with a prototype, e.g. in order to complete a task (see Fig. 3). Similar to MoPeDT, d.tools and Momento are examples that also support the synchronised recording of user interactions and audio-visual content. By this mean, the interpretation of captured audio-visual data can be eased when having them synchronised with the logged user interactions. Whenever a user study is conducted, MoPeDT displays a cloned screen view of the subject's mobile phone screen on the evaluator's computer. Thus, during the local or remote user study in a laboratory or field setting, the evaluator can always trace all interactions of the subject and the subject's current screen view. This cloned mobile phone screen is also used in order to take screen shots at certain points in time during the user study. For instance, MoPeDT captures screen shots of the cloned screen view whenever a new appearance is displayed. After the study, the captured screen shots can help to analyse the logged interactions, contexts and the captured videos. In addition to the afore mentioned features of the evaluation component, live annotations are a further support that is provided by MoPeDT. During the evaluation, evaluators can use this feature in order to log comments and observations or describe the executed task more detailed. Since the user context (e.g. the user's activity and state) can also give valuable information for later analyses, their recording is another feature of MoPeDT's evaluation component. Apart from the user context, the environmental context (e.g. the temperature and lighting conditions) can also be recorded for the later analysis. In contrast to most other user-centred prototyping or evaluation tools, the MoPeDT architecture enables the add-on and application of components that enable the determination and logging of user and environmental contexts. This feature, however, was not used in our user study in order to reduce the complexity of the evaluation phase for our subjects.

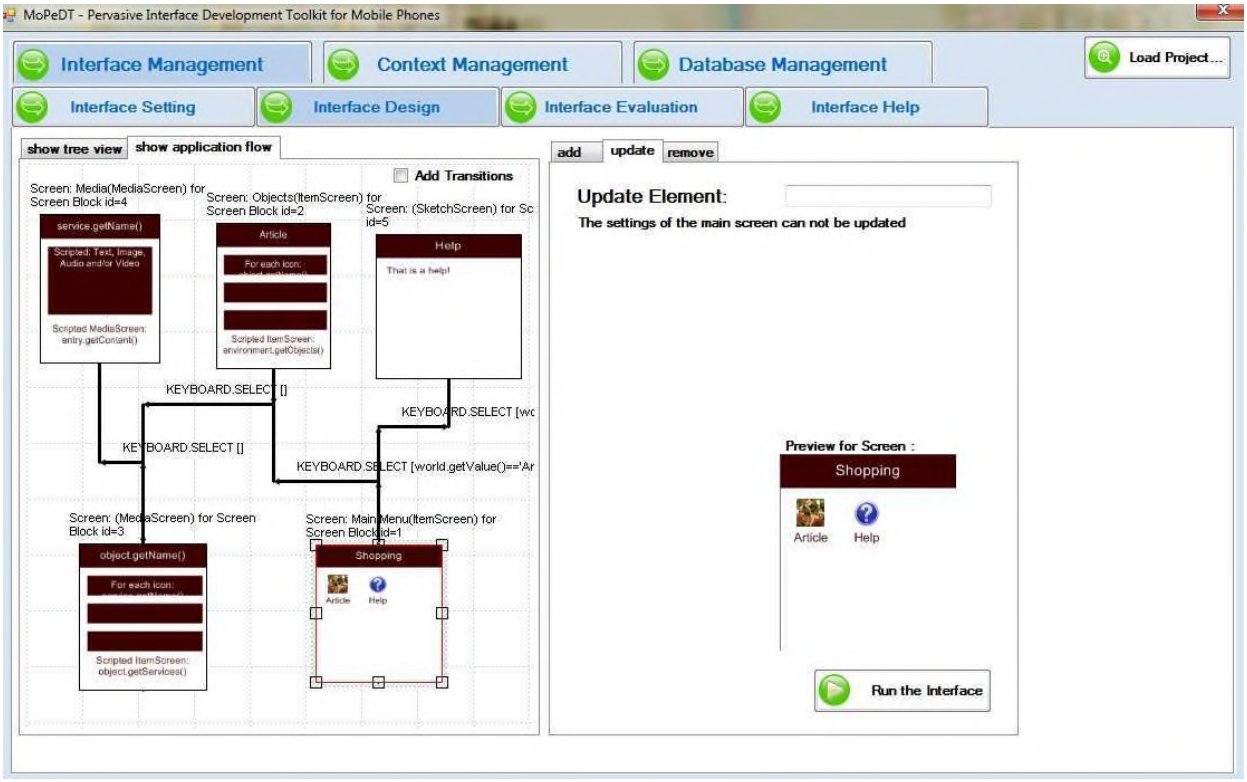

Figure 2. The Design Component of MoPeDT

The Analysis Component

MoPeDT's analysis component provides the time-line based visualisation of the recorded data in order to navigate through them and interact with them (e.g. to find usability problems or user preferences). D.tools and Momento also apply the interactive time-line based visualisation of the logged data as well as the audio-visual content. We extended ANVIL [19] in order to develop MoPeDT's analysis component. ANVIL supports the display of audio-visual content as well as the visualisation and modification of annotations at various freely definable time-line based tracks. The extended version of ANVIL automatically synchronises and annotates the captured audio-visual content with the recorded user interactions and contexts. Now, the interface developer can scroll through the pre-annotated video or jump to intended data that are displayed in the tracks in order to investigate the user's behaviour in different contextual situations. 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 the probability of occurrence for an intended context or behaviour. Thus, the number or the subject's errors or the subject's required time to complete a task can be statistically analysed with the assistance of MoPeDT.

THE EVALUATION OF MOPEDT

The evaluation approach of MoPeDT mainly bases on the evaluation method of TERESA [5]. Few authors have described their evaluations [11] and if so most of them were performed rather informally. From all previously mentioned tools, TERESA is the only tool that was evaluated by a comparative study in two different settings. Comparative studies can provide meaningful knowledge about an added value when using the tool instead of another approach. In the first setting of TERESAS' evaluation - the baseline setting - subjects 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 interfaces in both settings, the subjects documented problems, commented on tool features but also noted the required time to develop the interfaces. Several aspects of the evaluation, such as the developer's satisfaction and efficiency were rated in a questionnaire to also get subjective data of the users. Despite the carefully thought out method, TERESA's evaluation method, however, rather addressed the investigation of the process (formative evaluation) than the resulted prototypes (summative evaluation) [29]. Thus, TERESA's evaluation method can only be applied to verify Myers' second requirement about the ease of use and the efficiency to apply a tool for a process [27].

In our evaluation approach we also used a comparative method that applies analytic and empiric evaluation techniques to collect subjective and objective data for revealing a UDP tool's benefits and problems as well as the meeting of Myers' mentioned requirements. Nielsen [28] mentioned the need to combine analytical and empirical techniques to gather objective and subjective data. In our method we also combine the

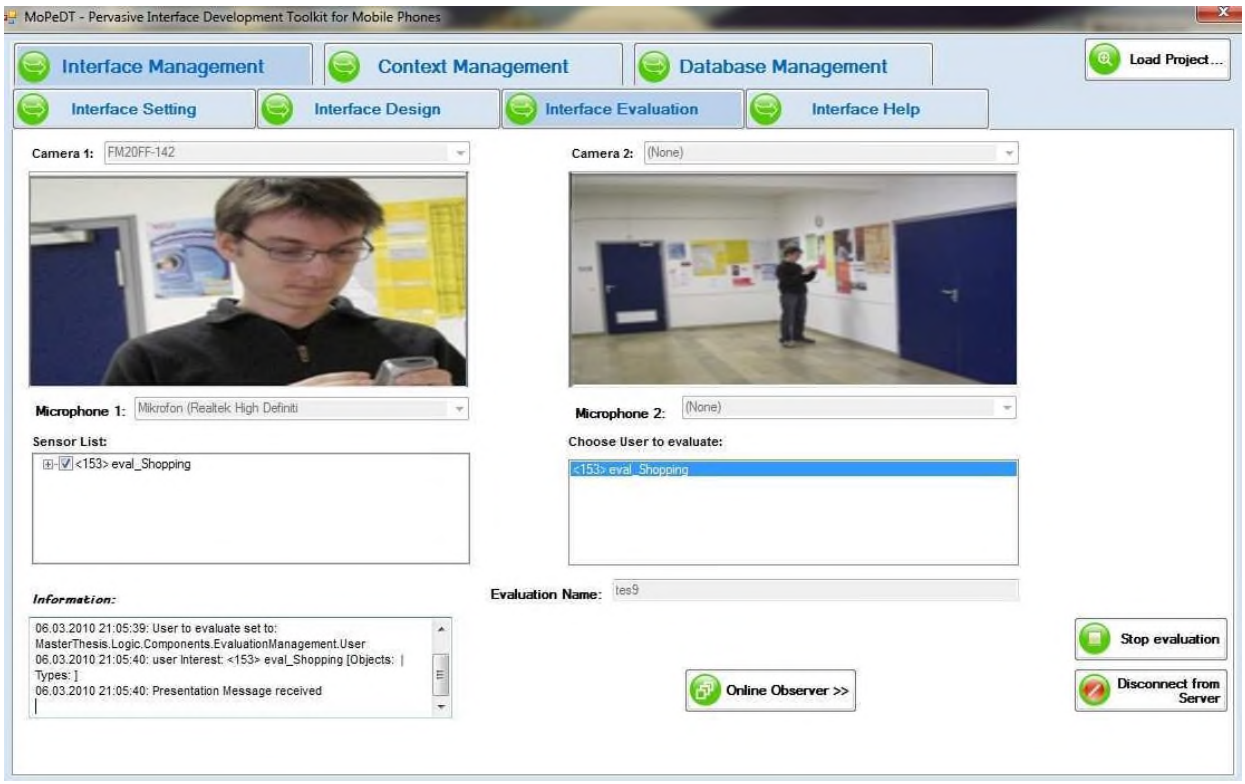

Figure 3. The Evaluation Component of MoPeDT

formative with the summative evaluation for analysing (2) the ease of use, the satisfaction and efficiency when applying the user-centred development process (formative) with and without the tool-support as well as (1) the results of the two processes (summative): the effectiveness of interface developer to generate user-friendly prototypes. By these means, the results of the evaluation can verify whether a tool meets the two requirements defined by Myers [27] or not. Additionally, the results should also shed light in the questions of potential problems and benefits of an all-in-one UCP tool, such as MoPeDT. In the following, we first describe the experimental setting of our evaluation method, we next report on the conduction of the evaluation and finally we illustrate our results.

The Experimental Setting

We formulated the following expectations, falling into three categories based on the definition of usability (ISO 9241 part 11): efficiency, effectiveness and satisfaction. H1: MoPeDT improves the developer's **efficiency** to quickly develop an interface prototype. H2: MoPeDT improves the developer's **effectiveness** to develop a highly user-friendly interface prototype. H3: MoPeDT improves the developer's **satisfaction**. To investigate our three hypotheses, we defined the used platform 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, evaluation and analysis of a pervasive interface whereas in the second level the interface developers used as a baseline traditional design, evaluation and analysis platforms (e.g. Eclipse or Netbeans). Thus, in contrast to most mentioned tool evaluations, we conducted a comparative study with well-known and commonly used tools for all steps of the user-centred design process to investigate our hypotheses. Moreover, quite contrary to the other methods, we gathered subjective and objective data in order to measure our dependent variables: efficiency, effectiveness and satisfaction. For these measurements, we applied a questionnaire to acquire subjective data whereas protocol recordings and a guideline review were utilized in order to collect objective data. Later on, the aggregated data were compared in order to determine the correctness of our hypotheses. For the experiment we used a within-subjects design and therefore, all of our 20 subjects participated in both levels of the experiment. To prevent any position effects, ten subjects started with the design, evaluation and analysis of the prototype based on the traditional application of the user-centred design and afterwards used MoPeDT whereas the other ten students used MoPeDT first.

Evaluation Techniques

In the following we describe our three used evaluation techniques. We applied an analytic method: an inspection method of a guideline review and two empirical methods: an observation method of a protocol recording and an inquiry method of a questionnaire. To gather objective data for hypothesis H2 (Effectiveness), we conducted a summative evaluation (Guideline Review) and investigated the resulted prototypes for both levels (with and without MoPeDT). An independent usability expert who was not involved in the development or

evaluation of MoPeDT used the generated prototypes and investigated their robustness and completeness based on the task description (see Conducting the Experiment) as well as their violation against the 22 mentioned guidelines that base on Nokia's Design and User Experience Library. For example, the expert controlled a consistent softkey usage (G1) and layout (G2) as well as a correct error handling (G3), a support of a contextual help in the different screens (G4) and an easy reversibility of actions (G5). Further examples are the consideration to minimize the number of screens (G12), to display important information with text and icons (G15) and to clearly structure screens (G16). Some of the guidelines are automatically supported by MoPeDT (e.g. G1, G2, G3, G5 and G16) but other guidelines are not covered, such as G12 and G15.

The protocol recording was applied in order to conduct a formative evaluation and gather objective data for the investigation of hypothesis H1 (Efficiency). The subjects used the protocols for the documentation of the required time to complete different steps while designing, evaluating and analysing the prototypes. Moreover, emerged problems had to be noted. In order to keep comparability between the different subjects, they used the same wording since the protocols provided a predefined list of all required steps for the design, evaluation and analysis (e.g. Implementation of the client-server connection; Programming of the graphical user interface for the mobile phone; Recording of videos; Synchronisation of the recorded events with the captured videos).

In our post-task questionnaire, we first asked for the subject's age, gender and software and usability engineering skills whereas in the main part of the questionnaire we asked the subjects to rate statements about the prototype's design, evaluation and analysis for both levels: with MoPeDT and with the traditional approach. The statements addressed the efficiency (E), effectiveness (Eff), satisfaction (S), learnability (L), transparency (T) and their satisfaction with the interface developer's sphere of action (A). In terms of efficiency, the subjects had to estimate a level's influence on their efficiency in the design (E1) and in the evaluation and analysis (E2). Additionally, the levels were measured whether they usefully support the conduction of the user-centred design process (E3) and whether they could provide a gain in time (E4). The statement about effectiveness concerned the satisfaction with the generated prototype in both levels (Eff1). In order to investigate hypothesis H3 about the user's satisfaction, we collected the subjective data in terms of satisfaction, learnability, transparency and the satisfaction with the interface developer's sphere of action (user control). First, the subjects had to rate their satisfaction when designing (S1) and when evaluating and analysing a prototype (S2). Then, the subjects had to estimate the ease of learnability for the design (L1) and for the evaluation and analysis (L2). In this context, the subjects also had to rate the required programming (L3) and evaluation skills (L4). Further statements were about a level's transparency for the design (T1) and for the evaluation and analysis (T2). Finally, the subjects had to estimate the satisfaction with the sphere of action for the design (A1) and the evaluation and analysis (A2) in both levels. In this term, the subjects also assessed the scope of the supported screen templates (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 subjects to select preferences, such as the preference for a component of MoPeDT.

Conducting the Experiment

In both levels, the user-centred prototyping with and without MoPeDT, the same pervasive shopping assistant for mobile phones had to be (1) designed, (2) evaluated and (3) later on analysed with end-users of the application. This pervasive shopping assistant helps users to receive information about articles in a shopping store (e.g. about the ingredients of articles). We used a very simple task scenario for our user study in order to enable the conduction within the one month of our course. Normally, we also could have done another previously described scenario, such as the game called treasure hunt. (1) To keep comparability, the subjects received a detailed description about the intended prototype. For example, the types of screens and content to display were predefined. Moreover, the description also contained the requirement to implement prototypes that enable a keyboard-based and a touch-based interaction (NFC). The subjects were also instructed to implement a logging mechanism for the prototype that was generated with the traditional approach in order to enable a recording of the user interactions in the evaluation phase. (2) For the evaluation of their generated prototypes, the subjects were instructed to audio-visually capture three end-users while they were interacting with the two generated prototypes. The tasks for these evaluations were also predefined (e.g. the users had to select the potato to read its description) in order to enable the comparison between the captured evaluations of the two settings. (3) After the user evaluation, the subjects had to analyse their captured audio-visual content and logged user interactions in order to find usability problems of the two prototypes, such as wording problems. Overall, the task description contained all required steps to complete the design, evaluation and analysis with and without MoPeDT. Figure 4 shows two prototypes that were designed, evaluated and analysed from one subject in the levels with and without MoPeDT.

The subjects of our studies were students of our three-month course 'Usability Engineering'. Before we ran the study at the last third of the course, we conducted tutorials within the course and taught all subjects 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 and Netbeans with the Mobility Pack. During this training period and the user study, we did not tell the subjects that MoPeDT is a software tool that was developed at our lab. In addition to the software skills, we comprehensively taught all subjects about usability in general, the user-centred design, mobile phone usability and the mobile phone guidelines that the usability expert also used for the guideline review. We reminded the subjects to apply these guidelines for both levels when designing, evaluating and analysing the prototypes in our user study. During the de-

sign, evaluation and analysis of the user study, the subjects were ordered to fill in the protocols for all completed steps. After having developed prototypes in both levels of the independent variable, the subjects had to fill in our questionnaire. 20 computer science students (16 male and four female students) of our course 'Usability Engineering' participated in our one-month user study. The subjects were aged between 22 and 29 (M = 24.15, SD = 1.90). First, our subjects had to rate their programming and usability skills on a 5 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 subjects had previous knowledge in mobile phone programming - about one year.

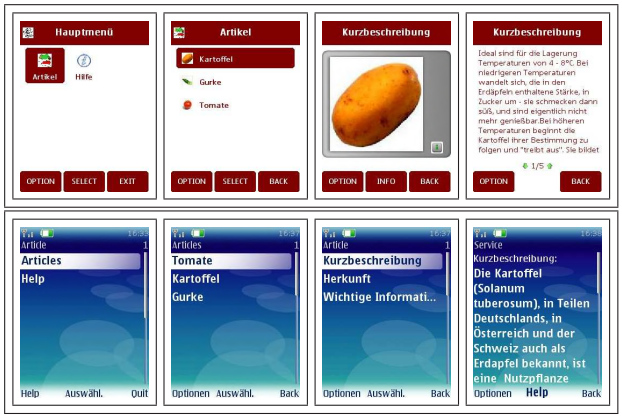

Figure 4. Screen shots of the interfaces that were developed with MoPeDT (first row) and with the traditional approach (second row).

Results and Discussion

In this section we describe the results of our user study and discuss their consequences on our three hypotheses: the increased interface developer's **efficiency** to quicker develop prototypes, the improved interface developer's **effectiveness** to develop more user-friendly and robust interfaces as well as the increased interface developer's **satisfaction**, when using the all-in-one solution MoPeDT instead the traditional approach.

The analysis of the objective and subjective data proved our assumption about an increased efficiency when using a UCP tool, such as MoPeDT. When analysing the protocols 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 prototype design phase. In the evaluation and analysis phase, the annotation and analysis of the captured videos decelerated the user-centred design process when not using MoPeDT. These objective data were also reflected when analysing our subjective data of the questionnaire. Based on our rating scale from one to five, on average, the subjects agreed with the statement about their increased efficiency for the design when using MoPeDT (E1: M = 3.85, SD = 1.27), which was significantly higher than when using the traditional approach to run the user-centred design process (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 higher but not significantly (E2: M = 3.35, SD = 0.88). The subjects also found that MoPeDT (E3: M = 3.5, SD = 0.09) makes the whole user-centred design 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 questionnaire substantiates the results. Most subjects found the tool usage "quick and easy" and see a benefit in "the very quick prototyping and evaluation of applications". The results of the protocol recording and the questionnaire prove our first hypothesis and the requirement of Myers that a tool has to increase the interface developer's efficiency.

The analysis of the subjective data revealed no significant benefit for the developer's effectiveness when using a UCP tool, such as MoPeDT. The subjects 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' rating. While most of them highlighted the prototype's design 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 subjects 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 ratings regarding the scope of the supported screen templates (A3: M = 2.7, SD = 1.03). Nevertheless, based on a two-sided dependent t-test, the results of our guideline review showed a highly significant difference between the interfaces that were designed, evaluated and analysed with MoPeDT compared to the generated interfaces with the traditional approach. Interfaces generated with MoPeDT had, on average, less violations against the 22 guidelines (M = 0.85, SD = 0.93) than the interfaces that were developed with traditional approaches (M = 4.35, SD = 2.52), $t(19) = 5.48$, $p < 0.001$. Most often, the interfaces developed with traditional approaches did not consider a consistent usage of the softkeys (G1: 17 of 20 subjects) and did not use icons and text for important information (G15: 17 of 20 subjects) (see Fig. 4). Another often occurred error was the non-compliance to support contextual help for each screen (G4: 11 of 20 subjects). Based on our objective data, the second hypothesis and Myers' requirement about the increased effectiveness of the developers to generate user-friendlier interfaces is suggestively proved due to the fact, that the resulted interfaces of MoPeDT provided a better usability based on the guideline review than interfaces that were gen-

erated with traditional approaches. The subjects, however, would like to have a wider range of action when designing the layout with MoPeDT. The limitations of MoPeDT did not only affect the layout design. In general, the subjects considered a significant lack of their sphere of action 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 a little bit higher rated than for the traditional approach (A2: M = 3.05, SD = 1.19). The limited sphere of action mainly causes that the subjects 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). Beside 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 (T1: M = 3.05, SD = 1.10) and evaluation and analysis (T2: M = 3.2, SD = 0.89) indicated an undistinguished transparency of MoPeDT. The interface designers "want to see what is going on in the background". Despite these overall negative results in terms of satisfaction, the subjects considered some benefits for the learnability. On average, the ease of learnability was significantly higher rated for the design 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 subjects 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 our questionnaire, we also asked the subjects about their overall acceptance of MoPeDT. Thus, we asked them to decide about their preferred approach (MoPeDT or the traditional approach) for the design as well as the evaluation and analysis. For the design, five subjects chose the traditional approach whereas seven selected MoPeDT and eight subjects mentioned both approaches as useful which is quite similar to the results of the preferred evaluation and analysis approach. Four subjects favoured the traditional approach for evaluating and analysing prototypes whereas 11 subjects preferred only MoPeDT and five saw benefits in using both approaches. Thus, despite the negative satisfaction with MoPeDT, the subjects tendentially preferred MoPeDT for the design, evaluation and analysis compared to the traditional approach. We also asked the subject's preferred components of MoPeDT, one subject did not like a single component of MoPeDT, while six subjects only liked the design component and two subjects only liked the evaluation and analysis component. 11 subjects 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 subject mentioned the benefit "to handle everything in a single program: the database, the design, evaluation and analysis". Another subject 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 three components" and "the all-in-one approach" that prevents "the induction in several programs". Overall, we could not prove the hypothesis about the increased interface developer's satisfaction when using MoPeDT because there are some problems due to the insufficiency of user control and transparency. The overall preference of an all-in-one tools solution appears in outlines.

CONCLUSION

In this paper we covered the idea of UCP tools as an all-in-one solution for interface developers. Characteristically, these tools support the user-centred prototyping with three closely linked components: a component to design, evaluate and analyse an interface prototype. As a contribution, we introduced our UCP called MoPeDT that provides possible enhancements compared to other UCP tools, such as a plug-and-play architecture in order to enable more comprehensive user studies. Additionally, we provided an overview of MoPeDT's components that might provide an insight to other tool developers. This paper also illustrated the comparative tool evaluation of MoPeDT. As a result of this study we can summarise the following knowledge. MoPeDT as an all-in-one UCP tool can decrease the required time to design, evaluate and analyse an evolutionary prototype (Efficiency) as well as reduce the prototypes' non-compliance of interface guidelines (Effectiveness). Additionally, our results also indicate that interface developers mostly accept a UCP tool, such as MoPeDT for all steps of the user-centred prototyping as well as that the user feedback seems to imply the preference of a single tool solution which integrates all components instead of separated tools for each single step of the user-centred prototyping. The results of our user study, however, also might indicate typical problems when applying UCP tools, such as MoPeDT. The developers' overall satisfaction (Satisfaction) of a UCP tool seems to be strongly depending on the satisfaction with the user control as well as the tool's transparency. In particular, screen templates increase the usability of the generated prototype in terms of the compliances of interface guidelines but they limit the interface designers' user control in order to individually adapt the layout.

ACKNOWLEDGMENT

This research is partly sponsored by *OC-Trust* (FOR 1085) of the German research foundation (DFG).

REFERENCES

1. M. S. Andreasen, H. V. Nielsen, S. O. Schrøder, and J. Stage. What happened to remote usability testing?: an empirical study of three methods. In *CHI '07: SIGCHI conference on Human factors in computing systems*, pages 1405–1414. ACM, 2007.
2. E. Arroyo, T. Selker, and W. Wei. Usability tool for analysis of web designs using mouse tracks. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 484–489. ACM, 2006.

3. S. Bateman, C. Gutwin, N. Osgood, and G. McCalla. Interactive usability instrumentation. In *EICS '09: 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 45–54. ACM, 2009.

4. S. Carter, J. Mankoff, and J. Heer. Momento: support for situated ubicomp experimentation. In *CHI '07: SIGCHI conference on Human factors in computing systems*, pages 125–134. ACM, 2007.

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

6. A. M. Davis. Operational prototyping: A new development approach. *IEEE Softw.*, 9(5):70–78, 1992.

7. A. M. Davis, H. Bersoff, and E. R. Comer. A strategy for comparing alternative software development life cycle models. *IEEE Trans. Softw. Eng.*, 14(10):1453–1461, 1988.

8. M. de Sá, L. Carriço, L. Duarte, and T. Reis. A mixed-fidelity prototyping tool for mobile devices. In *AVI '08: Conference on Advanced visual interfaces*, pages 225–232. ACM, 2008.

9. B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In *UIST '06: 19th annual ACM symposium on User interface software and technology*, pages 299–308, ACM, 2006.

10. S. Dow, J. Lee, C. Oezbek, B. Maclntyre, J. D. Bolter, and M. Gandy. Exploring spatial narratives and mixed reality experiences in oakland cemetery. In *ACE '05: ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 51–60. ACM, 2005.

11. G. Ellis and A. Dix. An explorative analysis of user evaluation studies in information visualisation. In *BELIV '06: AVI workshop on BEyond time and errors*, pages 1–7. ACM, 2006.

12. J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay. Myexperience: a system for in situ tracing and capturing of user feedback on mobile phones. In *MobiSys '07: 5th international conference on Mobile systems, applications and services*, pages 57–70. ACM, 2007.

13. K. Gajos, D. B. Christianson, R. Hoffmann, T. Shaked, K. Henning, J. J. Long, and D. S. Weld. Fast and robust interface generation for ubiquitous applications. In *Ubicomp*, pages 37–55, 2005.

14. G. Gopal, T. Kindberg, T. Kindberg, and e. a. John Barton. People, places, things: web presence for the real world. In *WMCSA2000*, pages 365–376, 2000.

15. P. Holleis and A. Schmidt. Makeit: Integrate user interaction times in the design process of mobile applications. In *Pervasive*, pages 56–74, 2008.

16. L. E. Holmquist. Prototyping: generating ideas or cargo cult designs? *interactions*, 12(2):48–54, 2005.

17. S. Houde and C. Hill. What do prototypes prototype? *Handbook of Human-Computer Interaction*.

18. E. Kangas and T. Kinnunen. Applying user-centered design to mobile application development. *Commun. ACM*, 48(7):55–59, 2005.

19. M. Kipp. Anvil - a generic annotation tool for multimodal dialogue. In *7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1367–1370, 2001.

20. S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang. Suede: a wizard of oz prototyping tool for speech user interfaces. In *UIST '00: 13th annual ACM symposium on User interface software and technology*, pages 1–10. ACM, 2000.

21. K. Leichtenstern and E. André. The assisted user-centred generation and evaluation of pervasive interfaces. In *AmI '09: European Conference on Ambient Intelligence*, pages 245–255. Springer, 2009.

22. K. Leichtenstern and E. André. Studying multi-user settings for pervasive games. In *11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI)*, pages 190–199. ACM, 2009.

23. M. Macleod and R. Rengger. The development of drum: A software tool for video-assisted usability evaluation. In *In HCI'93*, pages 293–309. Cambridge University Press, 1993.

24. J.-Y. Mao, K. Vredenburg, P. W. Smith, and T. Carey. The state of user-centered design practice. *Commun. ACM*, 48(3):105–109, 2005.

25. F. Mattern and C. Floerkemeier. Vom internet der computer zum internet der dinge. *Informatik-Spektrum*, 33(2), 2010.

26. M. R. McGee-Lennon, A. Ramsay, D. McGookin, and P. Gray. User evaluation of oide: a rapid prototyping platform for multimodal interaction. In *EICS '09: 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 237–242. ACM, 2009.

27. B. A. Myers. User interface software tools. *ACM Trans. Comput.-Hum. Interact.*, 2(1):64–103, 1995.

28. J. Nielsen. Usability engineering. In *The Computer Science and Engineering Handbook*. 1997.

29. M. Scriven. The methodology of evaluation. In R. G. R. Tyler and M. Scriven, editors, *Perspectives on curriculum evaluation*, AERA Monograph Series - Curriculum Evaluation. Rand McNally & Co., 1967.

30. R. Hull, B. Clayton, and T. Melamed. Rapid authoring of mediascapes. In N. Davies, E. D. Mynatt, and I. Siio, editors, *Ubicomp*, volume 3205 of *Lecture Notes in Computer Science*, pages 125–142. Springer, 2004.