The Assisted User-Centred Generation and Evaluation of Pervasive Interfaces

Karin Leichtenstern and Elisabeth André

Institute of Computer Science
Multimedia Concepts and Applications
Universittsstr. 6a
D-86159 Augsburg
{Leichtenstern,Andre}@informatik.uni-augsburg.de
https://mm-werkstatt.informatik.uni-augsburg.de

Abstract. The usability of a pervasive interface is crucial because it is a quality criterion which can determine about the success or the failure of a product. The application of the user-centred design is a possibility to reach a good design. However, this process requires combined knowledge of software and usability engineering. A lack of these skills can cause long development times and additional costs as well as badly usable interfaces. We address these problems and support interface developers with a tool-based assistance which reduces the required programming and interface design skills of the developers in order to more efficiently conduct the user-centred design. In this paper we describe this tool-based assistance for the user-centred generation and evaluation of pervasive interfaces for mobile phones as well as its evaluation.

1 Introduction

Ubiquitous Computing and its synonyms *Pervasive Computing* and *Ambient Intelligence* become more and more important in our everyday life ([21],[8]). Their common idea is to make the computer invisible in our everyday life for interactions with everything, everywhere at any time [7]. The first real pervasive interaction devices are mobile phones. Almost everybody owns a mobile phone and takes it around constantly. Recent phones support novel hardware and network facilities which enable different mobile interactions to pervasive environments. Ballagas and colleagues [2] give a comprehensive overview about the different mobile interaction techniques available with todays smart phones. For example, the mobile phone's camera can be used to recognize visual markers or built-in RFID reader can be used to receive information stored on RFID tags. Consequently, interactions with the user's ambient world, e.g. RFID-tagged posters become possible. Compared to the development for desktop settings, developing pervasive interfaces for mobile phones adds new layers of complexity. For instance, the developer has to cope with the limited input and output capabilities of the mobile devices [8]. Ensuring usability in this context is challenge. A user-centered design process ([17], [10]) is one possibility to obtain a good design

for pervasive applications. A characteristic feature of this process is an iterative design which includes several iterations of implementing prototypes along with continuous user evaluations of these prototypes. The iterations are required to build the interface as usable as possible for different contextual conditions in a pervasive environment. This user-centred design requires advanced usability and software engineering skills in order to efficiently and correctly develop a pervasive interface for mobile phones. One option to solve this problem is the usage of software tools which adapt these skills and assist in the user-centred design of pervasive interfaces. Our approach of the tool-based user-centred design supports interface developers in the iterative design with an assisted generation and evaluation of dynamic pervasive interfaces for mobile phones which are compliant with approved mobile phone guidelines. Before our tool-based approach is described more detailed, we first reflect existing tools on the requirements of the assisted user-centred design. At the end of the paper we introduce a preliminary study which shows benefits and problems of the assisted user-centred design with our tool.

2 Related Work

Until now, several tools or frameworks have been implemented which support developers in order to generate pervasive interfaces. Hull [1] classified four different categories of tools: ubiquitous middleware, ubiquitous modules, prototyping tools and content&behaviour tools. Middleware and modules ([18], [19]) provide developers with valuable software components and architectures whereas prototyping tools assist in building up a sketch of a concept [14] or a specification of particular rules, e.g. for context-aware applications [6]. Content&behaviour tools ([1], [4], [16]) provide assistance for the generation of interfaces as well as for the specification of their services and content. The specification of services and content is crucial in order to generate interfaces which content is dynamically loaded and displayed at the runtime. All of these four introduced types of tools only support in the interface's generation but not in their evaluation. To support assistance for the user-centred design process in all stages, approaches are required which also support in the conduction and analysis of user evaluations. Thus, we added two further types of tools to Hull's categories: evaluation tools and user-centred design tools. Evaluation tools ([9], [3]) assist in conducting user evaluations whereas user-centred design (UCD) tools ([13], [5]) support in the generation and evaluation of interface's prototypes. We evaluated these six categories of tools if they assist the user-centred design process in all stages (Generation and Evaluation) in order to quickly and easily develop pervasive interfaces for mobile phones which are compliant with some approved mobile phone guidelines (Guidelines). Additionally, we investigated whether these tools enable the dynamic content presentation on the mobile device (Dynamic Content) as well as the support of different mobile interaction techniques (Pervasiveness). The results of this evaluation are illustrated in table 1[1].

[1] The rating scale ranges from a very weak (- -) to a very strong support (++).

Table 1. Tool Categories and the Assisted User-Centred Design's Requirements

	Generation	Evaluation	Dynamic	Pervasiveness	Guidelines
Middleware	+	- -	++	++	- -
Modules	+	- -	++	++	- -
Prototyping Tools	+	- -	-	- -	- -
Content & Behaviour Tools	++	- -	++	+	- -
Evaluation Tools	- -	++	- -	- -	- -
Todays UCD-Tools	++	++	-	- -	- -
UCD-Tools: MoPeDT	++	++	++	++	++

Only UCD-tools assist in the generation and evaluation of interfaces. For this category we consider a lack of tools as our review only revealed two UCD-tools: SUEDE [13] and d.tools [5]. Both tools neither support in the compliance of approved guidelines nor assist in the development of pervasive interfaces for mobile phones. SUEDE assists in the development of speech interfaces whereas d.tools addresses the development of physical computing applications. In summary, our literature review revealed no known tool which assists interface developers in the user-centred generation and evaluation of pervasive interfaces for mobile phones which follow approved mobile phone guidelines and support the presentation of even dynamic content.

3 Assisted User-Centred Design

Our literature review proved the lack of an appropriate tool-based assistance for pervasive interface developers. We provide such a tool called MoPeDT (Pervasive Interface Development Toolkit for Mobile Phones). Different mobile interaction techniques can be used in order to generate pervasive interfaces for mobile phones which dynamically load and display content of a database. These generated interfaces are compliant with approved guidelines, e.g. a consistent layout. In contrast to other tools, our approach also assists in the conduction and analysis of user studies. For example, the assistance automatically annotates captured videos on different contexts which emerge in the pervasive environment. Thus, our tool meets all requirements of the assisted user-centred design of pervasive interfaces (see Table 1). In this section we describe the three components of MoPeDT more detailed: the architecture, the mobile phone framework and the IDE (Integrated Development Environment).

3.1 MoPeDT - The Architecture

The architecture is the basic component of MoPeDT. It enables the integration of different components which emerge in a pervasive environment. Figure 1 shows this architecture which consists of a main server and a database as well

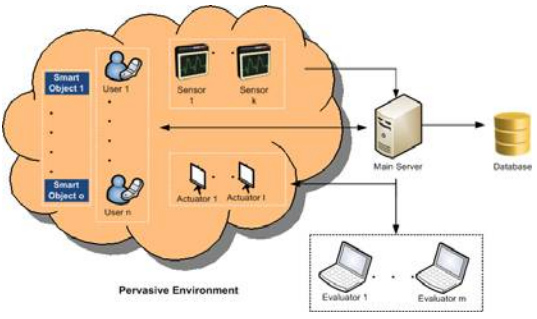

Fig. 1. The Architecture of MoPeDT

as several smart objects, users with mobile phones, sensors, actuators and evaluators. Smart objects, such as pictures or domestic home appliances are tagged objects in a pervasive environment. Users can address these objects by using the pervasive interface of the mobile phone in order to load or store persistent multimedia content about the objects via the main server. Other components of the architecture are the sensors and actuators. They can be used to collect and display additional information about the pervasive environment. Sensors, such as a temperature or a humidity sensor can collect, interpret and broadcast context to the main server. The main server can forward this context to interested users or actuators. Actuators, e.g. a public display can receive and display context or other information, such as video content. The last component of the architecture is called evaluators which are a special kind of actuators. They are required to perform user studies and log user interactions. For all these components software modules exist in Java.

3.2 MoPeDT - The Mobile Phone Framework

A special software module of MoPeDT's architecture is the mobile phone framework. The framework adopts software and usability engineering aspects, such as the implementation of the network communication, the support of different mobile interaction techniques and the support of screen templates which are grounded on approved mobile phone guidelines. Instead of considering and implementing these aspects, developers only need to specify the pervasive interface's interactions and behaviour via XML files.

The current version of the mobile phone framework supports four different mobile interaction techniques: Keyboard-based, NFC-based, Speech-based and Location-based. Thus, the framework not only enables the development of location-based applications but also of pervasive applications which require the other interaction techniques. Using the XML file for the interaction specification, developers can specify these mobile interaction techniques and map their emergence to contexts. For instance, in this file the GPS position of a location can be defined and mapped to trigger an action depending on a certain context,

e.g. the context HOME. Now, whenever a user has entered a specified location or performed a defined interaction technique, the framework automatically triggers and processes contexts. These triggered contexts can induce transitions and change the pervasive interface's behaviour on the local device or on other components in the architecture. The contexts can also be used whenever user studies are conducted. They are automatically logged and later on used in order to annotate the captured audio-visual content.

The behaviour of the pervasive interface is specified via state-chart diagrams which is common practice for integrated development environments (e.g. [13], [5]). We use *State Chart XML* [20] to represent the diagram in XML. These XML files can be interpreted by the mobile phone framework to generate the corresponding screen or to handle the corresponding events because each state of the diagram represents a screen of the interface and each transition of a state represents triggerable context. For example, figure 2 shows a very simplified state-chart diagram which consists of two states: the Main Menu screen and the Hello World screen as well as one transition: HOME. In this example, the Hello World screen is only loaded once the context HOME has been triggered by a location-based interaction. Hence, in order to load the Hello World screen, the user has to enter the specified location of the context HOME. Using our approach, transitions can also be specified for contexts triggered by Keyboard-based, NFC-based and Speech-based interactions. In contrast to the developers

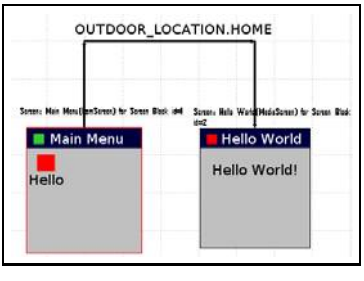

Fig. 2. Behaviour Specification of a Pervasive Interface via a State-Chart Diagram

of d.tools and SUEDE, we apprehend sources of errors when giving developers free options to specify the interface's states and transitions in the state-chart diagram due to several inducible usability problems, e.g. the non-compliance of the consistency. Thus, when specifying states in the state-chart diagram, we support developers with an expandable set of screen templates which follow approved usability guidelines from Nokia [15]. For instance, these templates consider a consistent layout, softkey usage and navigation style. Each screen has a heading, content and a softkey part. In the softkey part the left softkey is used for options whereas the middle key is applied for confirmations and navigations and the right softkey is utilized for negative actions (back, cancel or exit). Additionally, each screen contains a contextual help and an option to return to the main

menu. From each screen the user can return to the previous state automatically. Thus, each screen provides a back, cancel or exit option.

The current version of the framework supports templates for various kinds of menu screens, media input and output screens as well as feedback screens, e.g. a waiting or error screen. Developers can use these screen templates and define their static or dynamic content, e.g. the heading, the items and the options of a menu screen. In contrast to static screens, dynamic screens add new layers of complexity to the specification task due to several reasons. Static screens display unchangeable content whereas dynamic screens are generated and adapted to the content at the runtime. For instance, the number of items in a dynamic menu screen or the media type of a dynamic media output screen are not known at the development time. Thus, screen templates are expected to dynamically adapt their layout, e.g. on the loaded media type (text, image, audio and video). Certainly, these dynamic screens are crucial for pervasive interfaces of mobile phones because several content are context-adaptive loaded. For instance, once a user has selected a smart object, a dynamic screen automatically loads and displays all services and their content for the just selected smart object. Our screen templates enable the dynamic screen definition by means of a simplified scripting language and thereby enable the dynamic presentation of content defined in the database.

3.3 MoPeDT - The Integrated Development Environment (IDE)

Although MoPeDT's architecture and mobile phone framework support the generation of dynamic pervasive interfaces and provide a platform to conduct user evaluations, MoPeDT also supplies an integrated development environment (IDE). This IDE simplifies the interaction and behaviour specification of pervasive interfaces as well as the conduction of user studies due to its provided graphical user interfaces (GUIs) for all stages of the user-centred design process. Figure 3 and 4 illustrate the GUIs for the generation and evaluation of a pervasive interface.

In the generation phase of an interface, the user of our IDE is assisted with a GUI in order to define the database content and the interface's interaction and behaviour files for the mobile phone framework. At the end of the specification, a high-fidelity prototype is automatically generated which runs on emulators or real mobile phones. So far, we have tested generated prototypes on the Nokia N95 and the Nokia 6131 NFC.

After the generation, a user study can be conducted, e.g. to investigate the usability or the user's behaviour. For instance, a user study can reveal whether the wording of items and options is comprehensible or not and which mobile interaction technique fits best to which situation. Therefore, videos have to be captured and later on reviewed and labeled on corresponding features, such as the user's interactions. Without tool-based assistance this task called annotation is very time-consuming and error-prone. With MoPeDT we release interface developers from performing this annoying task. During the user study, audio-visual data, live annotations and task descriptions can be stored while relevant contexts are

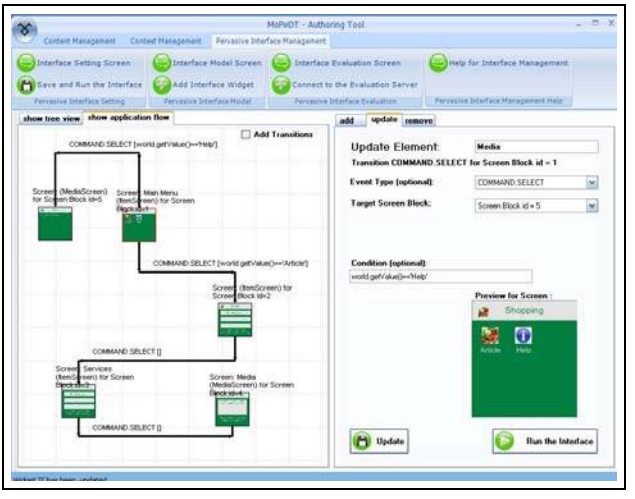

Fig. 3. Prototype Specification: The GUI for the specification of the prototype with the state chart view and options to specifiy the screen templates

logged automatically. After having conducted an evaluation, the captured data can be analyzed by means of an extended version of ANVIL [12] which automatically annotates the captured auto-visual data on the logged contexts. ANVIL is a tool which has been widely used for the annotation of audio-visual material containing human-human dialogue or human-computer interactions. It supports annotation at various freely definable tracks which makes it highly application-independent. The extended version of ANVIL displays the captured videos, a time-line with the logged contexts as well as the screen shots of the pervasive interface. This assistance helps developers to review the already pre-annotated data. For instance, interface developers can scroll through the video or jump to intended contexts of the time-line in order to investigate the user's behaviour in different contextual situations, e.g. to find usability problems or user preferences. Additionally, the extended version of ANVIL supports the export of the annotated data in different formats of statistic tools, such as SPSS in order to investigate the probability of occurrence for an intended context or behaviour.

4 Evaluation of MoPeDT

To investigate if our approach of the tool-based development of pervasive interfaces improves the conduction of the user-centred design process, we accomplished a user study with seven subjects having intermediate software engineering skills (between one and five years) and minor usability engineering skills (less than one year). We were interested whether using MoPeDT reduces the generation and evaluation time of a pervasive interface as well as the number of usability problems compared to traditional approaches. We were also interested

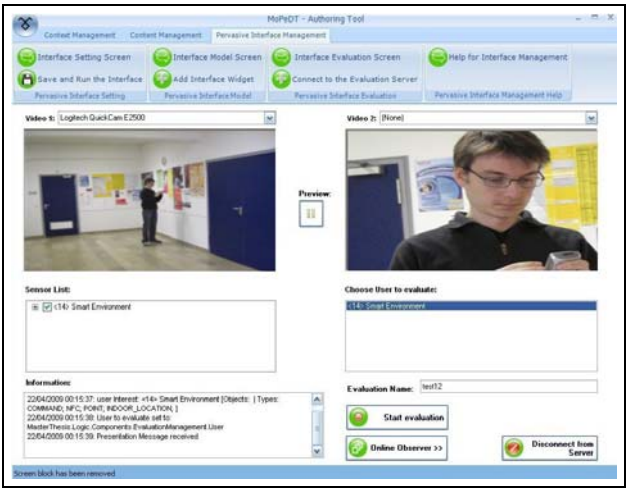

Fig. 4. Prototype Evaluation: The GUI shows the cameras and the supported options, e.g. an option to store live annotations

in the comments and wishes our participants had for our tool-based approach. Thus, after the test we interviewed the participants and asked them for their mind about our tool-based assistance. In order to evaluate the potential benefits of MoPeDT, the subjects had to create and evaluate a particular pervasive interface in two different settings. In the first setting the subjects used classical development and evaluation platforms, such as Eclipse or Netbeans to generate and evaluate a pervasive interface whereas in the second setting the subjects had to use MoPeDT for the same task. In both settings a shopping assistant had to be implemented and evaluated which helps users receive information about articles in a shopping store, e.g. about the ingredients of articles. In the evaluation videos,users had to be captured and later on analyzed in order to find usability problems, e.g. wording problems. For the generation and the evaluation of the interfaces we determined the required generation and evaluation time and counted the number of violation against the Nokia guidelines [15]. All subjects were previously skilled in these guidelines and reminded to deploy them.

The results of our tests proved our assumptions. On average, the required generation and evaluation time in minutes without MoPeDT ($M = 900.71$, $SE = 245.16$), was significantly higher than when using MoPeDT ($M = 184.28$, $SE = 37.71$),$t(6) = 3.41$, $p < 0.05$, $r = 0.94$. When not using MoPeDT, the network and GUI programming required much more time in the prototype generation phase. In the evaluation phase, the annotation and analysis of the captured videos decelerated the user-centred design process when not using MoPeDT because the developers had to manually synchronize the logged events with the captured videos which is done automatically when using MoPeDT. Beside the reduced generation and evaluation time, the number of usability problems and violation

against the Nokia guidelines could also be reduced. The interfaces developed without MoPeDT had significantly more usability problems (M = 4.43, SE = 0.20) than when using MoPeDT (M = 0.71, SE = 0.29),$t(6) = 10.33$, $p < 0.001$, $r = -0.06$. Figure 5 shows screen shots of interfaces generated without and with MoPeDT which illustrate two usability problems. When not using MoPeDT, a common problem is the non-observance to display items with both icons and text. A further usability problem is using scrollbars instead of side-to-side scrolling which causes a user's higher cognitive load. The interviews conducted with the participants of our study showed benefits and problems of MoPeDT. The most acute problem is the limitation in the scope of supported operations, e.g. the number of the supported mobile interaction techniques and screen templates. Highlighted benefits of MoPeDT are the less required programming and interface design skills as well as the saved time and improved quality of the interfaces.

Fig. 5. Screens shots of the interfaces developed without (left screens) and with MoPeDT (right screens)

5 Conclusion and Discussion

In this paper we introduced requirements concerning user-centred design tools and illustrated MoPeDT as an example software which meets these requirements. MoPeDT is the first known UCD-tool which maps the user-centred design process in order to assist interface developer in the generation and evaluation of pervasive interfaces for mobile phones. Our study approved MoPeDT as a promising tool-based assistance for pervasive interface developers because it saves time and improves the interface's quality. Consequently, this saved time can be applied in order to concentrate on the concept development and content management of the application as well as the investigation of the user's behaviour in user studies, e.g. to investigate different multi-user settings of pervasive games [11].

References

1. Hull, R., Clayton, B., Melamed, T.: Rapid authoring of mediascapes. In: Davies, N., Mynatt, E.D., Siio, I. (eds.) UbiComp 2004. LNCS, vol. 3205, pp. 125–142. Springer, Heidelberg (2004)
2. Ballagas, R., Borchers, J., Rohs, M., Sheridan, J.G.: The smart phone: a ubiquitous input device. Pervasive Computing, IEEE 5(1), 70–77 (2006)

3. Carter, S., Mankoff, J., Heer, J.: Momento: support for situated ubicomp experimentation. In: CHI 2007: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 125–134. ACM, New York (2007)
4. Correia, N., Alves, L., Correia, H., Romero, L., Morgado, C., Soares, L., Cunha, J.C., Romao, T., Dias, A.E., Jorge, J.A.: Instory: a system for mobile information access, storytelling and gaming activities in physical spaces. In: ACE 2005: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology, pp. 102–109. ACM, New York (2005)
5. Design, T.I., Hartmann, B., Klemmer, S.R.: Reflective physical prototyping. In: Proceedings of UIST 2006 Symposium on User Interface Software and Technology, pp. 299–308. ACM, New York (2006)
6. Dey, A.K., Hamid, R., Beckmann, C., Li, I., Hsu, D.: a cappella: Programming by demonstration of context-aware applications. In: Proceedings of CHI 2004, pp. 33–40. ACM, New York (2004)
7. Gopal, G., Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., Spasojevic, M.: People, places, things: web presence for the real world. In: Proceedings WMCSA 2000, pp. 365–376 (2000), `http://www.cooltown.hp.com/papers/webpres/webpresence.htm`
8. Gorlenko, L., Merrick, R.: No wires attached: Usability challenges in the connected mobile world. IBM Syst. J. 42(4), 639–651 (2003)
9. Howarth, J., Smith-Jackson, T., Hartson, R.: Supporting novice usability practitioners with usability engineering tools. Int. J. Hum.-Comput. Stud. 67(6), 533–549 (2009)
10. Kangas, E., Kinnunen, T.: Applying user-centered design to mobile application development. Commun. ACM 48(7), 55–59 (2005)
11. Leichtenstern, K., André, E.: Studying Multi-User Settings for Pervasive Games. In: 11th International Conference on Human-Computer Interaction with Mobile Devices and Services, Mobile HCI, pp. 190–199 (2009)
12. Kipp, M.: Anvil - a generic annotation tool for multimodal dialogue. In: Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech), Aalborg, September 2001, pp. 1367–1370 (2001)
13. Klemmer, S.R., Sinha, A.K., Chen, J., Landay, J.A., Aboobaker, N., Wang, A.: Suede: a wizard of oz prototyping tool for speech user interfaces. In: UIST 2000: Proceedings of the 13th annual ACM symposium on User interface software and technology, pp. 1–10. ACM Press, New York (2000)
14. Li, Y., Hong, J.I., Landay, J.A.: Topiary: a tool for prototyping location-enhanced applications. In: UIST 2004: Proceedings of the 17th annual ACM symposium on User interface software and technology, pp. 217–226. ACM Press, New York (2004)
15. Nokia. Design and user experience library (2009), `http://library.forum.nokia.com/`
16. Pan, P., Kastner, C., Crow, D., Davenport, G.: M-studio: an authoring application for context-aware multimedia. In: MULTIMEDIA 2002: Proceedings of the tenth ACM international conference on Multimedia, pp. 351–354. ACM, New York (2002)
17. Rosenbaum, S., Rohn, J.A., Humburg, J.: A toolkit for strategic usability: results from workshops, panels, and surveys. In: CHI 2000: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 337–344. ACM, New York (2000)
18. Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: aiding the development of context-enabled applications. In: CHI 1999: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 434–441. ACM, New York (1999)

19. Serrano, M., Nigay, L., Demumieux, R., Descos, J., Losquin, P.: Multimodal interaction on mobile phones: development and evaluation using acicare. In: MobileHCI 2006: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services, pp. 129–136. ACM, New York (2006)
20. W3C. State chart xml (2009), http://www.w3.org/TR/2009/WD-scxml-20090507/
21. Weiser, M.: The computer for the 21st century. Scientific American (February 1991)