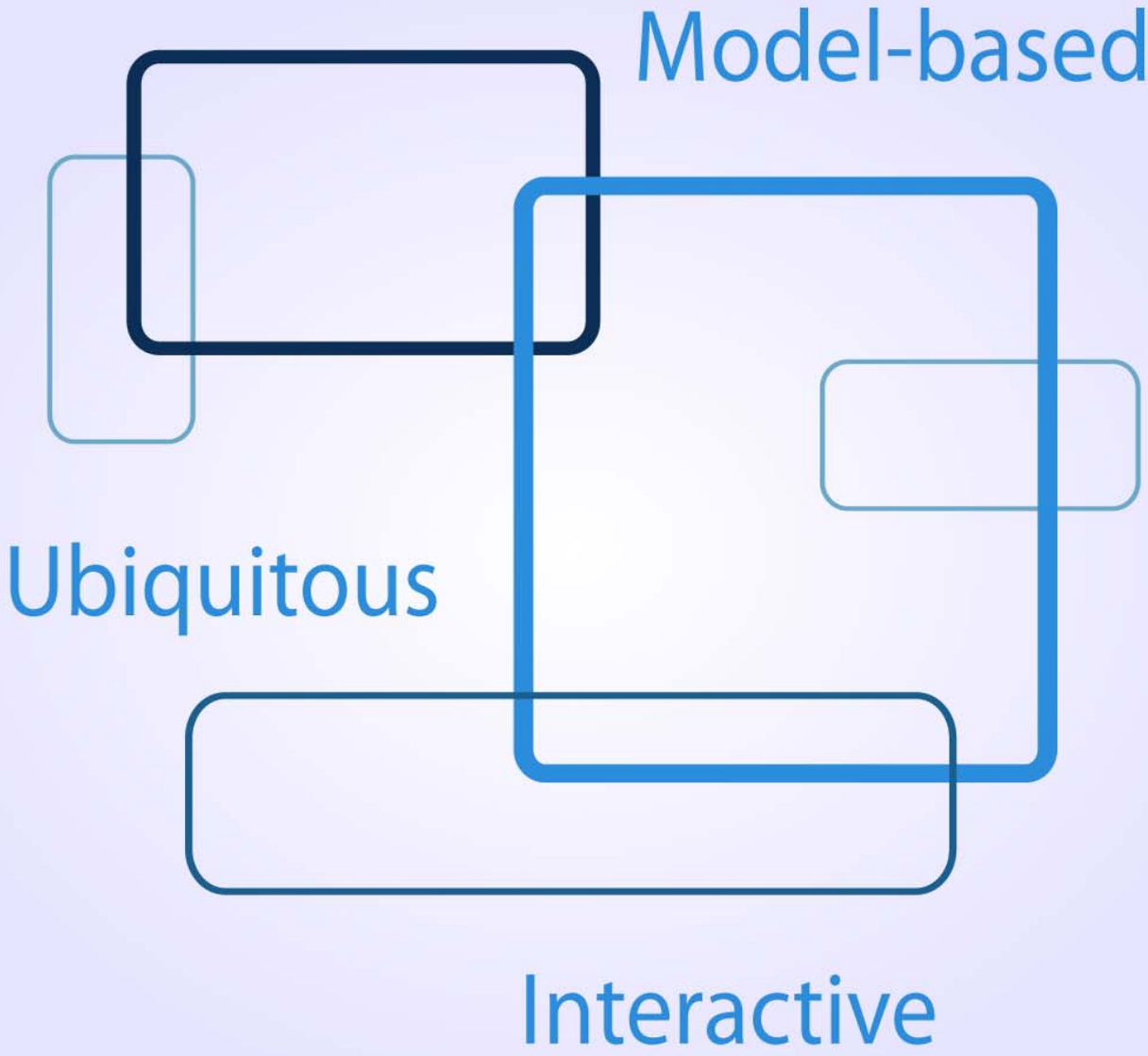

Modiquitous 2011

Proceedings of the 1st International Workshop on Model-based Interactive Ubiquitous Systems

SES

MMT

TECHNISCHE UNIVERSITÄT DRESDEN

Proceedings of Modiquitous Workshop

Publication Online - CEUR Proceedings (CEUR-WS.org)
CEUR-WS Vol. 787
Publication Year 2011

V.i.s.d.P:
Jun.-Prof. Dr. Thomas Schlegel
Junior Professorship for Software Engineering of Ubiquitous Systems
Institute for Multimedia and Software Technology
Technische Universität Dresden
01062 Dresden
Germany

CONTENTS

WORKSHOP ORGANIZERS

Thomas Schlegel

Technische Universität Dresden
01062 Dresden
Germany
thomas.schlegel@tu-dresden.de

Thomas Schlegel is Junior-Professor for Software Engineering of Ubiquitous Systems at the Institute of Software and Multimedia Technology of the Technical University of Dresden. Before he joined the University of Stuttgart as team leader for Interactive Systems, he worked as senior researcher and research project leader at Fraunhofer IAO from 2002, where he served as research cluster leader in the European Network of Excellence I*PROMS and led various national and international research projects. He received his PhD in engineering from the University of Stuttgart. He is author and co-author of 60 scientific publications and serves as reviewer and committee member for diverse international conferences.

Stefan Pietschmann

Technische Universität Dresden
01062 Dresden
Germany
stefan.pietschmann@tu-dresden.de

Stefan Pietschmann is research associate and Ph.D. student at the Institute of Software and Multimedia Technology of the Technical University of Dresden. He has been actively involved in several research projects in the field of collaborative and context-aware web applications. In the project CRUISe he specifically addresses the model-driven development of adaptive interactive applications based on the idea of a universal service composition.

PROGRAMME COMMITTEE

- Uwe Aßmann, Technical University of Dresden, Germany
- Jan van den Bergh, Hasselt University, Belgium
- Birgit Bomsdorf, Hochschule Fulda, Germany
- Raimund Dachselt, Otto von Guericke University of Magdeburg, Germany
- Florian Daniel, University of Trento, Italy
- Alfonso Garcia-Frey, University of Grenoble, France
- Geert-Jan Houben, Technical University of Delft, Netherlands
- Heinrich Hussmann, Ludwig-Maximilian University Munich, Germany
- Sevan Kavaldjian, Vienna University of Technology, Austria
  Programme Committee
- Gerrit Meixner, DFKI,Germany
- Philippe Palanque, University of Toulouse, France
- Fabiò Paterno, CNR-ISTI, Italy
- Michael Raschke, University of Stuttgart, Germany
- Dirk Roscher, Technical University Berlin, Germany
- Enrico Rukzio, University Duisburg-Essen, Germany
- Stefan Sauer, University of Paderborn, Germany
- Thomas Springer, Technical University of Dresden, Germany
- Gerhard Weber, Technical University of Dresden, Germany
- Anette Weisbecker Fraunhofer IAO, Stuttgart, Germany
- Jürgen Ziegler, University Duisburg-Essen, Germany

INTRODUCTION

Ubiquitous systems today are introducing a new quality of interaction both into our lives and into software engineering. Systems become increasingly dynamic making frequent changes to system structures, distribution, and behavior necessary. Also, adaptation to new user needs and contexts as well as new modalities and communication channels make these systems differ strongly from what has been standard in the last decades.

Models and model-based interaction at runtime and design-time form a promising approach for coping with the dynamics and uncertainties inherent to interactive ubiquitous systems (IUS). Hence, this workshop discussed how model-based approaches can be used to cope with these challenges. Therefore, it covers the range from design-time to runtime models and from interaction to software engineering, addressing issues of interaction with and engineering of interactive ubiquitous systems.

The MODIQUITOUS workshop was intended to discuss challenges and possible solutions of the EICS community to design and runtime aspects of interactive ubiquitous systems with a focus on model-based approaches. It aims to bring together researchers and practitioners focused on different problems of IUS.

THEME, GOALS, AND RELEVANCE

Model-based interactive ubiquitous systems form a new promising yet challenging domain within the scope of the Engineering of Interactive Computing Systems (EICS) conference. This workshop is intended to discuss these challenges and possible solutions of the EICS community to design and runtime aspects of interactive ubiquitous systems with a focus on model-based approaches.

The related problem space becomes clear when looking at typical future scenarios: users will not only carry their data but also their applications and profiles with them. This may mean switching from planning a project on a desktop system to a collaborative setting in a meeting and further to a mobile or public display setting where a mobile device is used for creating sketches for the first steps in the project. Consequently, applications will evolve from device-oriented to emergent cyber-physical and ubiquitous software in a broad sense, forming interactive and socio-technical systems. This opens manifold possibilities, but also a number of research problems regarding both the development process and the execution environment for those kinds of applications.

The MODIQUITOUS workshop is intended to provide a basis for discussion the adequate solution space. Therefore, it aims to bring together researchers and practitioners focused on different challenges of IUS, including:

- Model-driven architecture (MDA) in the context of IUS
- Advantages and potential problems of using MDA in the IUS domain
- Meta models for IUS, specifically for IUS-related aspects like interaction, different modalities, dynamic distribution, context-awareness, etc.
- Domain-specific models for IUS
- Model-driven generation of (intelligent) IUS
- Model-to-model and model-to-code transformations for IUS development
- Model-driven development and execution architectures, i.e., runtime systems for IUS
- Tools and frameworks for supporting the model-driven development of IUS
- Concepts for context-awareness and self-adaptation of IUS on model and runtime
- Software Engineering aspects of IUS
- Human Computer Interaction aspects of IUS

All these topics are of high relevance to a big part of the EICS community as their use is not restricted to ubiquitous systems and will show new ways for many kinds of new systems like mobile device settings, pervasive computing and social software.

PROGRAM

The workshop was held in the morning of June 13, 2011 as part for the ACM EICS Symposium.

08:45 **Arrival**

09:00 **Welcome and introductions**

Introductory statements by the organizers and brief introduction by each participant

09:15 **Paper presentations**

- Emmanuel Dubois, Christophe Bortolaso and Guillaume Gauffre
  Models Articulations for Ubiquitous System Design
- Linshu Duan and Heinrich Hussmann
  Model-Based Testing of Automotive Infotainment System HMIs
- Jan Zibuschka, Uwe Laufs and Heiko Roßnagel
  Towards Ubiquitous Emergency Management Systems
- Ivan Gudymenko and Katrin Borcea-Pfitzmann
  A Framework for Transforming an Abstract Privacy Model into Implementable UbiComp System Requirements

10:35 **Coffee break**

11:00 **Paper presentations**

- Florian Haag, Michael Raschke and Thomas Schlegel
  Ubiquitous Alignment
- Marius Feldmann, Thomas Springer and Alexander Schill
  Generating consistent universal controllers for Web-Service-enabled appliances
- Romina Kühn, Christine Keller and Thomas Schlegel
  A Context Taxonomy Supporting Public System Design

12:00 **Discussion and Topic Definition**

Discussion of hot research topics and definition of topics for the groups

12:15 **Group Work**

Discussion of the selected topic, e.g., identification of research roadmap items

12:45 **Discussion**

Plenum discussion and topic integration

13:00 **Lunch**

Models Transformations for Ubiquitous System Design

Emmanuel Dubois, Christophe Bortolaso, Guillaume Gauffre
University of Toulouse - IRIT
118, route de Narbonne
31 062 Toulouse Cedex 9, France
[firstname.lastname]@irit.fr

ABSTRACT
Many different models and tools exist for supporting the design of ubiquitous interactive systems. Each of them deals with a different point of view. As a result designing such systems has to involve a set of models rather than just one. In this paper we first provide an overview on existing models dedicated to Mixed Interactive Systems, one form of ubiquitous systems. Then, to facilitate the elicitation of the most appropriate model, we organize them along the steps of the development process. Finally, to smoothly guide the use of these different design resources along the development process, we provide an overview of different linking mechanisms between design models for ubiquitous systems and highlight their characteristics.

Keywords
Interaction model, software architecture model, model transformation, mixed interactive system

INTRODUCTION
Among the most recent forms of interactive techniques, one aims at taking advantage of physical objects to support the interaction with a computer system: physical artifacts surrounding users during their activity become part of the loop. Users' everyday objects thus constitute an extension of their body to communicate with the system. Such systems are either called tangible UI, mixed or augmented reality, etc.: we hereafter refer to them with the generic term of Mixed Interactive System (MIS). Such systems are emerging in many different domains, ranging from specific application such as surgery [19] to mass market [15]. It also comes together with the emergence of new usages and the combination of advanced and various technologies. Furthermore, new spaces are now opened to interaction since the interaction is simply requiring the presence of everyday physical objects. According to Weiser's definition, it is therefore a form of ubiquitous interactive system because the interaction mechanisms "*weaved themselves into the fabric of everyday life*" [25].

Nevertheless, the growing interest into the development of such interaction forms is undoubtedly linked to the constant exploration of new sensors, modalities and communication channels: as a result these forms of interaction are very different from traditional WIMP based situations. To better understand their differences and precisely highlight their specificities, efforts has been paid to develop descriptive models: such models express considerations related to the interaction [7], the physical properties of the required entities [16], the abilities of the modalities involved [6], etc. We observe from the diversity of approaches that complementary aspects, relevant for the design of MIS are addressed by different models. In the course of the design, the designer thus has to identify, for each step of the development, the most appropriate model, method or tool supporting the design. Developing a MIS thus appears to be a real challenge [21]. An optimistic solution could rely on the use of a unique and universal approach, aggregating all the required dimensions and enabling the design of all kinds of MIS. However, given the low maturity of the domain, the multiple attempts being developed, such an approach is not yet conceivable

Rather than contributing to the creation of such a unique reference model, we propose to compare existing models according to their role and place in the development process. Then, to facilitate their combined use, i.e. to smoothly guide the use of these design resources along the development of MIS, we explore possible linking mechanisms between models. In this paper, we first give a brief overview and characterization of modeling approaches existing in the field of MIS. We then introduce three fundamental design resources on which we have investigated the development of different model transformations and couplings.

EXISTING MODELS IN MIS
Designing Mixed Interactive Systems (MIS) requires considering many specifics facets: the nature of physical artifacts, the links between these physical objects and digital data and the variety of devices and technologies which can be involved. Consequently, adapted design resources have been developed. Hereafter, we review a set of design resources dedicated to MIS.

First, conceptual frameworks [9,12,16,22] provide a high level of abstraction on the MIS field. They raise questions about the generic role of the system and its place in the physical world. They provide a big picture of the MIS field and somehow help to lead the **analysis** of interactive situation.

Taxonomies and models [6,7,23] have also been defined to understand mixed interactive situations, the elements that characterize them and their advantages. This second set of approaches therefore contributes to a better understanding of the **interaction design** of MIS.

Toolkits and frameworks [13,17,21], rapid prototyping environments [5] or runtime platforms [1,18] have been proposed to facilitate the **implementation** of MIS.

Finally, many user experiments results have been published to compare different MIS among them or against WIMP solutions [4]. In addition, **evaluation** methods dedicated to such tests are explored [24] to provide an appropriate form of evaluation.

All these modeling approaches cover different but complementary design considerations. Although their levels of abstraction vary, they offer a clear definition of the development space and constitute a common terminology supporting interdisciplinary communication. As depicted in Figure 1, we also highlight that the different design models and resources, used to develop a MIS, can be organized along the traditional phases of an interactive software design cycle. Existing models dedicated to WIMP can therefore easily be put in parallel with models dedicated to MIS, and either be compared to or used in addition to these dedicated models.

Phase	Ref.	Name	Type
Analysis	9	Fishkin's taxonomy	M
	12	Expected, Sensed, Desired	M
	16	RBI	M
	22	MCPrd	M
A to D	2	MACS	C
	3	KMAD to ASUR	C
Design	6	MIM	M
	7	ASUR	M
	10	ASUR-IL	M
	23	TAC	M
D to P	10	ASUR to ASUR-IL	T
Prototyping	1	DWARF	M
	5	Wcomp	M
	13	Phidgets	M
	17	ARToolkit	M
	18	Papier Mache	M
	21	TUIMS	M
Evaluation	4	RESIM	M, C
	24	design review	M

Figure 1: Existing design Models (M), Couplings (C) and Transformations (T) for MIS development.

The main limits are however that these models and approaches are almost exclusively usable by MIS experts and remain highly compartmentalized. Indeed, even if high-level resources used during the design should guide the implementation, concrete and systematic links have not been clearly expressed yet. To support the design through the four development phases of a MIS and to highlight a chain of design models and tools from the earliest design considerations to the latest in the development process, connections among models are required. We propose in the following section, an overview of different linking mechanisms we have been exploring over the past years. We depict the goal, source and target of the links between two models and describe the resulting overall mechanism (see Figure 7). The presented linking mechanisms are also positioned in Figure 1.

BASIS OF OUR INSTRUMENTED DESIGN PROCESS

The three pillars of our articulating efforts respectively support the abstract description of the user's interaction with a MIS, the software level decomposition required to implement the designed MIS and the concrete component based implementation of the final MIS. We first summarize these models and illustrate them on a case study: the notepad assisted slideshow.

Figure 2: Use of the notepad assisted slideshow

For oral presentations, sequential slideshow systems like PowerPoint are largely used. The prototype we propose is a physical enhancement of a slideshow system: it involves the use of a notepad as "remote control" and feedback source, and associates each page of the notepad to one digital slide (see Figure 2). The speaker can thus write his own comments on the notepad and easily access to the corresponding slide. Potential animation steps of each slide are controlled through user's tap on the notepad. In the next sections, this prototype is used to illustrate the different models used to develop this system and how they have been linked.

Interaction model

Overview

ASUR [7] is a model which provides an abstract view on the user's interaction with a MIS. It describes physical and digital entities, adapters between the two worlds and information channels among them. It is a static representation: it describes a snapshot of the interaction required at a given time to perform a given task. It is also totally independent of the technology since it relies exclusively on an abstract description.

Goal

The goal of this model is to describe the different types of elements and data exchange required to support the interaction with a MIS. Both entities and channels are further characterized by attributes such as the type of entity (real object, adapter bridging the two worlds, etc.), the medium and language of the channels. Additional elements are expressible such as physical constraints among entities, links between a physical and digital entity, etc.

Example

The model of the notepad assisted slideshow is presented in Figure 3. The "user" interacts with the "notepad" for which each page is detected by an adapter ("PageDetector"). Another interaction with a second adapter detects user's tap ("StepDetection"). These two adapters deliver to the

"slideshow" some digital data from which the current slide and step in the animation are identified. Finally, the state of the slideshow is rendered through three different adapters to the "user" and the "attendees".

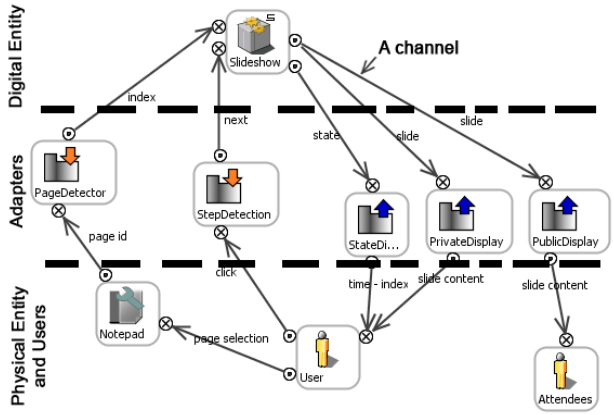

Figure 3: ASUR model for notepad assisted slideshow

Tools

A metamodel of ASUR has been defined [11]. Based on this metamodel a graphical editor has been developed as an Eclipse plug-in within the Eclipse Modelling Project [8].

Software architecture model

Overview

The ASUR-IL model is used to describe the MIS software architecture: it defines the software's skeleton through a components based architecture. As for ASUR it remains sufficiently abstract to be independent of the implementation platform since it relies exclusively on generic descriptions of the components.

Goal

The goal of this second model is to promote the integration of design considerations related to component-based specificities (port, data flow, component) and to software architecture of interactive system (functional core, views and controller). ASUR-IL is thus composed of two types of sub-assemblies:

- Adapters describing the required devices and API to implement the link between physical and digital world
- Entities describing system-dependent components; entities are decomposed according to the MVC pattern. It thus contains a Model, View(s) and Controller(s)

In comparison to ASUR, ASUR-IL adopts a software point of view on the interaction with the MIS. It therefore provides a list and description of the software bricks required, interfaces, ports, data types, etc.

Example

Left side of Figure 4 illustrates the adapter sub-assembly required to implement the "page detector" expressed with ASUR (Figure 3). It involves a digital camera component and a marker detection component. The right side of Figure 4 describes the entity sub-assembly required to implement the "slideshow" expressed in ASUR. This sub-assembly involves four components: two controllers, one model and one view.

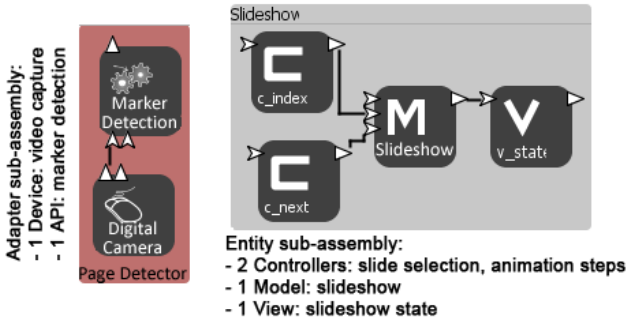

Figure 4: Detailed description of two of the ASUR-IL assemblies involved in the notepad assisted slideshow

The ASUR-IL model which entirely covers the case study is represented in Figure 5.

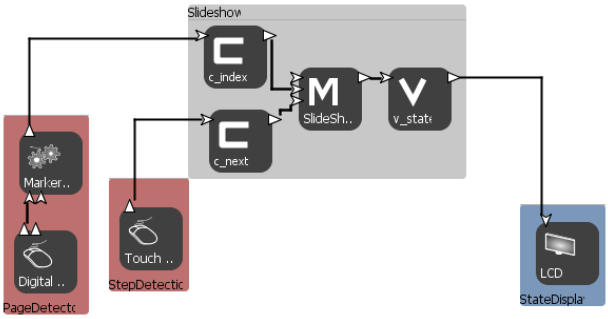

Figure 5: Two of the ASUR-IL assemblies involved in the notepad assisted slideshow

Tools

As for ASUR, a metamodel has been defined, and a graphical editor has been defined as an Eclipse plug-in.

Software component model

Overview

To implement the prototypes that we design with ASUR and ASUR-IL we rely on two existing prototyping platforms: Open Interface (OI) [20] and WComp [5].

Goal

The goal of such platforms is to allow a rapid development of system through manipulation of component assemblies at run-time. In association with each platform, a repository of components is available: it consists in a set of reusable software components ready to be integrated into new assemblies.

Example

The Notepad Assisted Slideshow has been implemented as an assembly of WComp components. This is illustrated in Figure 6. Each element of the assembly corresponds to one of the ASUR-IL element defined in the ASUR-IL model of the system.

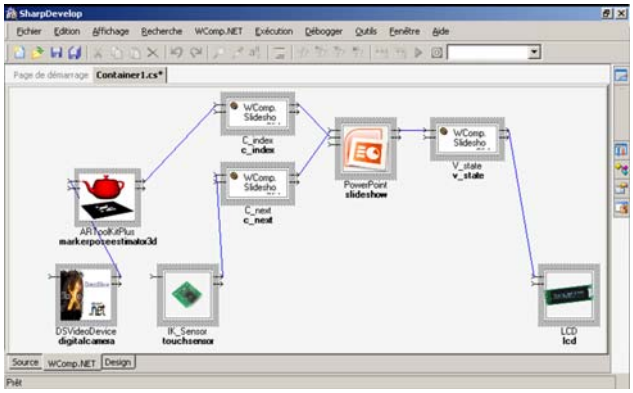

Figure 6: WComp assembly of the notepad assisted slideshow.

Tools

Metamodels of WComp and OI include the three concepts of component based architecture: components, ports and data flows. Each platform also offers a graphical editing environment for creating the appropriate assemblies.

Outcomes and limitations

There are obviously links between concepts expressed in each of these three pillars. But there is no clear constraint that drives their respective use and it is not ensured that the designer will conform to design recommendations made with the other models. For example, so far interaction design decisions expressed with the ASUR model are not constraining the development of the software architecture with ASUR-IL. And yet, these three pillars are required to drive a MIS from its early specification to its final implementation. We therefore complement them with linking mechanisms, such as model transformations or model coupling, in order to support the transitions between several phases of the development process. The goal is to better take advantage of the design choices expressed with the different models.

ARTICULATING DESIGN MODELS

MIS models have been developed to cover different phases of the development process: transformations and couplings are thus required at different places in the process.

From requirement to interaction design

Overview

We explored different linking mechanisms at this level:

- Linking models resulting of a KMAD task analysis to the ASUR interaction model [3]. It involves a unique expert, whose role is to translate the most of a task model into the definition of the contour of the interaction model: concepts expressed in the task tree are mapped to elements of an ASUR model describing an interaction technique supporting the realization of the task.
- Stimulating creativity session with the ASUR model [2]. A Model Assisted Creativity Session (MACS) is based on a scenario and a set of different constraints; it fosters the generation of mixed interaction techniques inside a group of multidisciplinary designers. MACS participants are invited to manipulate elements of an interaction model, in order to augment the potential of variations they might consider to generate ideas.

Goal

The goal of these two linking mechanisms is to ensure that the interaction techniques proposed are really in line with the specified task to achieve (KMAD-ASUR) and with the design problems to solve (MACS). In both cases the result of the linking mechanism is just a partial interaction model.

Tool

KMAD-ASUR and MACS are not so far supported by automatic tools. KMAD-ASUR is based on a set of rules and an algorithm describing the sequence of use: managing the alternatives generated by this transformation is left in charge of the designer. A MACS is composed of a set of steps, guidelines for the facilitator and manual post-treatments for the generated modelled ideas.

From interaction design to software architecture

Overview

This transformation, represented in the left side of Figure 7, converts an abstract specification of the interaction technique into a structured set of required software components: the generation of this software components structure is driven by the type of ASUR entity, attributes and channels involved in the model [10].

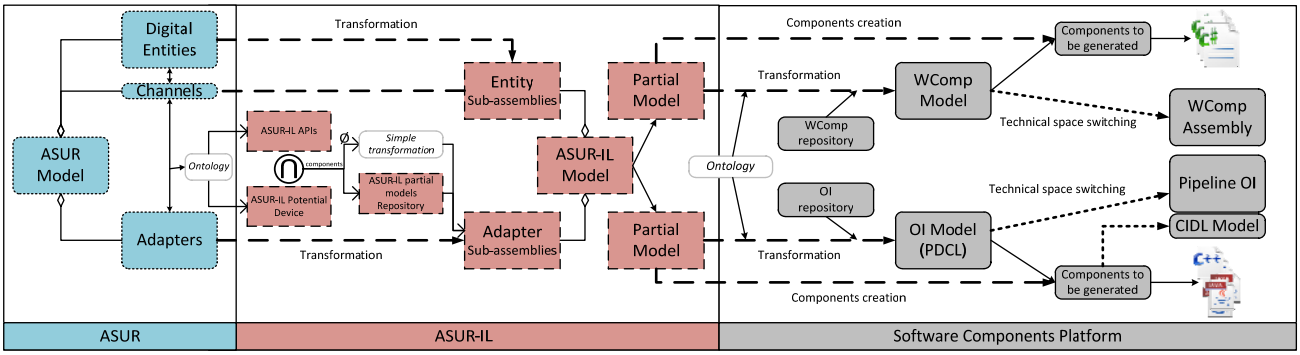

Figure 7: Synthetic summary of the model transformations linking ASUR to an implementation. (available as Eclipse Plugins http://www.irit.fr/recherches/ELIPSE/guideme/)

Goal

This transformation maintains coherence between the interaction specification and the proposed implementation. The software structure produced through it is only partial: indeed information related to the type of data for example, is not expressed in the ASUR model. This refinement of the software architecture design is thus left to the designer.

Tool

To support this transformation, ATL rules automate part of the transformation. It is assisted by the use of an ontology that establishes links between parts of the interaction design and parts of the software architecture definition: the ontology provides additional information to choose among existing components. A repository of already defined and used ASUR-IL components is available. Finally a wizard helps the designer to go through the different steps of the transformation and suggest design options. All these technologies have been packed into Eclipse plug-ins.

From software architecture to implementation

Overview

Translating ASUR-IL model to an implementation produces a running prototype [10]: this transformation is represented in the right side of Figure 7. The prototype is therefore made of an assembly of existing components (either from the OI or WComp platforms) and strictly conforms to the software architecture previously expressed.

Goal

The goal of this final transformation is to concretely instantiate the designed interactive technique. Until this final point, there is no need to pay attention to the soft- and hardware technologies to use. As a result, the running prototype can easily reuse existing bricks, even if they are not all available on the same platform: indeed communication mechanisms among the components have also to be specified.

Tool

To support this transformation, ATL rules, repositories of components, ontology and an interactive wizard have been developed. All these technologies have been packed into Eclipse plug-ins.

From development models to evaluation

Overview

We developed this linking mechanism in order to relate ergonomic recommendations to part of an interaction model describing the MIS [2]. This is based on a formal pattern describing usability recommendations: this pattern involves elements constituting the ASUR interaction model.

Goal

The main objective of this link between evaluation and design model is to facilitate the identification on the model, of part of the solution that is affecting (positively or negatively) the usability of the system. Such links thus potentially reduce the duration of one cycle of the four phases development process.

Tool

So far the navigation through the recommendations is only supported by a multiple criteria query on a web site. Refined tools would be useful so that usability recommendations pop out as soon as one relevant elements of the interaction model is added or selected.

CONCLUSION

In this position paper we highlighted the diversity of design-time and run-time models existing in the field of mixed interactive system, one form of ubiquitous interactive system. This diversity is partly explained by the amount of design considerations to handle when it comes to designing such systems: indeed most models covers only one specific aspect or at the best a limited subset of relevant considerations. However, following our analysis of existing works, we have been able to identify for each of these development resources one of the different phases of a development cycle for which the development resource is dedicated. This is thus classifying these design resources.

To go past the comparison of models through a classification, it is required to chain one model to another. Indeed one model provides one view on the system to design; chaining one to another provides a support for considering different complementary views without leaving one aspect aside. Furthermore, one unique and integrated design platform would be hard to propose because of the multiplicity of options, situations, technologies and usages potentially involved in a MIS. Chaining models to each other allows the definition of different ways in the design process: for example, going from A to B through a transformation in model C (A → C → B), may very flexibly be replaced by a longer transformation chain involving two other models instead of model C (A → D →E →– B). The result is the same, but the specialists of models D and E are no longer enforced to use model C.

Based on the different linking mechanisms between models of different phases of the development process that we have investigated, this paper showed that different forms of transformation exist: they use repositories of partial solutions, graphical representations, manual application of rules, methodological principles or transformation language. Among them, those exploiting Model Driven Engineering (MDE) approach and tools (ASUR to ASUR-IL to WComp/OI) appear to be the most promising: they use a standard language; they are easily supported by tools; they contribute to the definition and diffusion of the metamodels; they support the generation of multiple representations of the same model; they define transformation mechanisms, constitute guides through the design process; finally MDE has already proven its efficiency in classical software engineering. However, using MDE raised new challenges to investigate.

First in terms of properties, what happens to system or interaction properties settled in one model when a transformation is applied to the model? And more generally, are there properties of a transformation that are particularly important for “modiquitous” activities?

Managing retroactive loops in the design process also raises questions: if a model B is generated from a model A, how to ensure that modifications on B are still in line with A? How to send back to A modifications performed on B?

Given that ubiquitous systems are still quickly evolving and adding considerations to new dimensions, technologies, artifacts, etc., how MDE might help integrates these emerging new considerations? What would be the relevant characteristic of an ubiquitous interactive situation that could help identify the most relevant design path among the available models and transformations?

Finally, evaluating ubiquitous systems is a challenge in itself, but “modiquitous” activities are definitely contributing to this challenge through the elicitation of the most relevant design aspect of ubiquitous system, thus emphasizing the need to base the design of ubiquitous system on models.

REFERENCES

1. Bauer, M., Bruegge, B., Klinker, G., et al. Design of a Component-Based Augmented Reality Framework. ACM and IEEE *ISAR’01*, (2001), 45--54.
2. Bortolaso, C., Bach, C., Dubois, E., *A combination of a Formal Mixed Interaction Model with an Informal Creative Session. EICS’11*, 10 pages, 2011 **(in press)**.
3. Charfi, S., Dubois, E., Bastide, R.. *Articulating Interaction and Task Models for the Design of Advanced Interactive Systems. TAMODIA 2007*, Vol. 4849, Springer, LNCS, p. 70-83, 2007.
4. Charfi, S., Dubois, E., Scapin, D.L., *Usability Recommendations in the Design of Mixed Interactive Systems, EICS’09, USA*, ACM, p. 231-236, 2009.
5. Cheung, D., Tigli, J., Lavirotte, S., et Riveill, M. Wcomp: a Multi-Design Approach for Prototyping Applications using Heterogeneous Resources. *IEEE International Workshop on Rapid System Prototyping*, IEEE Computer Society (2006), 119-125
6. Coutrix, C. et Nigay, L. An Integrated Framework for Mixed Systems. in The Engineering of Mixed Reality Systems - Chap 2. Springer-Verlag London, 2010, 9-32.
7. Dubois, E., Gray, P., Nigay, L. ASUR++ : A Design Notation for Mobile Mixed Systems. Interacting with Computers 15, 4 (2003), 497-520.
8. Eclipse Foundation. Eclipse Modeling Project. 2006. http://www.eclipse.org/modeling/.
9. Fishkin, K.P. A taxonomy for and analysis of tangible interfaces. PUC’04, 8, 5 (2004), 347-358.
10. Gauffre, G., Dubois, E., Bastide, R.. *Domain-Specific Methods and Tools for the Design of Advanced Interactive Techniques.* in : *Models in Software Engineering*. Springer, Vol. 5002, LNCS, 2008, 65-76.
11. Gauffre, G., Dubois, E., Taking Advantage of Model-Driven Engineering Foundations for Mixed Interaction Design. Dans / In : Model Driven Development of Advanced User Interfaces, Springer-Verlag, Vol. 340, Studies in Computational Intelligence, 2011, p. 219-240
12. Gaver, B., Boucher, A., Walker, B., and al. Expected, sensed, and desired:A framework for designing sensing-based interaction. ACM TOCHI 12, 1 (2005), 3-30.
13. Greenberg, S. et Fitchett, C. Phidgets: easy development of physical interfaces through physical widgets. *UIST*, ACM (2001), 209-218.
14. GuideMe. Editor of MIS specific models. 2010. http://www.irit.fr/recherches/ELIPSE/guideme/.
15. Hornecker, E., Shaer, O., Tangible User Interfaces: Past, Present and Future Directions, in Foundations and Trends in HCI, Vol. 2 Nr. 1-2, 2010, pp. 1-138
16. Jacob, R.J., Girouard, A., Hirshfield, L.M., et al. Reality-Based Interaction: A Framework for Post-WIMP Interfaces. CHI’08, ACM (2008), 201-210.
17. Kato, H. et Billinghurst, M. Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. IEEE IWAR’99, (1999), 85-95.
18. Klemmer, S.R., Li, J., Lin, J., et Landay, J.A. Papier-Mache: toolkit support for tangible input. CHI’04, ACM (2004), 399-406.
19. Lamata, P., et al., Augmented Reality for Minimally Invasive Surgery: Overview and Some Recent Advances, in Augmented Reality, ISBN 978-953-7619-69-5, (2010), p. 73 – 98.
20. Open Interface. STREP. http://www.oi-project.org/.
21. Shaer, O. and Jacob, R.J. A specification paradigm for the design and implementation of tangible user interfaces. ACM TOCHI, 16, 4 (2009), 1-39.
22. Ullmer, B. and Ishii, H. Emerging frameworks for tangible user interfaces. *IBM Syst. J. 39*, 3-4 (2000), 915-931.
23. Ullmer, B., Ishii, H., et Jacob, R.J.K. Token+constraint systems for tangible interaction with digital information. *ACM TOCHI 12*, 1 (2005), 81-118.
24. Wang, X. Dunston, P.S., Usability Evaluation of a Mixed Reality Collaborative Tool for Design Review, CGIV’06, (2006), p. 448 – 451.
25. Weiser, M., The computer for the 21st century. Scientific American, 3(265):94–104, 1991

Model-Based Testing for the Menu Behavior of Automotive Infotainment System HMIs

Linshu Duan
Ludwig-Maximilians-Universität München
and
AUDI AG
linshu.duan@audi.de

Heinrich Hussmann
Institut für Informatik
Ludwig-Maximilians-Universität München
heinrich.hussmann@ifi.lmu.de

Dieter Niederkorn and Alexander Höfer
Infotainment System Testing
AUDI AG
dieter.niederkorn@audi.de
alexander.hoefer@audi.de

ABSTRACT

Testing the graphical human machine interface (HMI) of automotive infotainment systems has shown to be costly and challenging due to its large function scope, high complexity and multiple variants. To ensure the quality and reduce testing costs we are working on a model-based testing concept for graphical HMIs of infotainment systems. In our work the short form "HMI" is used for the term "graphical HMI". In this paper, we present some preliminary results of our model-based testing research. We firstly introduce the classification and distribution of HMI errors. This statistic shows that errors in the menu flow construct an essential part of HMI errors. In this paper we focus on the detection of this kind of errors. For this UML state machine has to be extended to describe the menu behavior, so that valid tests can be generated from its instances. Common coverage criteria for the state machine can not produce efficient tests for infotainment system HMIs. Therefore, we discuss some defined adequacy criteria for infotainment system HMI tests. At last we briefly introduce how we use software product-line approaches to integrate variability into the model-based HMI testing concept.

Author Keywords

Model-Based Testing, Test Models, HMI-Testing, Software Product-Line, Statechart with Variability

INTRODUCTION

Infotainment system HMIs of new generations have a very wide function scope and can contain more than 2000 menus and 100 pop-up menus. They usually have many variants caused by different markets, product-lines, individually configurable features and equipments.

To reduce the testing costs and ensure a systematic code coverage, we are working on a model-based and automated testing concept specific for infotainment system HMIs. The concept has been introduced in previous papers [5] [6]. In this paper we present some new results of our ongoing work.

This paper is organized as follows. In order to clarify which kinds of HMI errors can occur in practice, we have evaluated some parts of a past HMI development project. We will present our results of the error classification and a rough distribution.

UML statechart, which graphically represents a state machine, is widely used for the development and specification of infotainment system HMIs. However, the standard UML state machine from the OMG (Object Management Group) [14] is not sufficient for specifying the menu behavior so that valid and automatable tests can be generated from the specification. We firstly introduce the test-oriented HMI specification in which the menu behavior model is located and then required extensions of the state machine for creating a testing-ready menu behavior model.

Test generation based on common coverage criteria can not produce adequate tests for infotainment system HMIs. We firstly introduce the generated tests based on two chosen common coverage criteria and then explain the specific adequacy criteria for infotainment system HMI tests.

Finally, we will introduce the variability of infotainment system HMIs and how variability handling can be integrated into our model-based testing concept.

RELATED WORK

A number of research efforts have addressed the model-based testing of GUI applications [17], [16], [1], [10] and [11].

The NModel framework introduced in [2] supports finite state machine (FSM) models and automatic test generation for GUI-driven applications. In this approach binding user data or data exchange with external components are not considered.

In [17] a model-based software testing method for web applications is presented. This method focuses on testing the functionalities of the front end of web applications, i.e., the linking behavior of the links and forms, which is modeled with statecharts. However, this method has not yet found solutions to the problems of modeling the back-ends of a web application. Without the behavior of the back-end, a gener-

ated test only describes a possible sequence contained in the model and does not consider the conditions. So generated tests are usually infeasible for the test automation. For infotainment systems HMIs the behavior of the back-end is very complex and error-prone. Testing this logic automatically is a very important test purpose.

The concept described by Memon [11] does not separate the menu flow behavior and the physical structure of the HMI, which is inappropriate for specifying the infotainment system HMI. Different variants of infotainment system HMIs are usually developed in one model as one product family. The variants in the family usually have the same or very similar behavior but different physical HMI elements or structure. Separation of them is an important requirement in the HMI specification.

In [10] LTS (labeled transition system) with action-word and key-word technique is used. The concept separates the specification of the business logic and the presentation logic. The action model contains action-words, which are abstract events. They describe the behavior of the system. The refinement model contains both action-words and key-words. Key-words are user events or menu navigation, which are performed by concrete HMI elements. Refinement model describes also how action-words can be refined with key-words.

In [1] the authors extend the state machine with regular expressions to consider not only correct HMI actions but also incorrect transitions.

In [12] and [16] some specific coverage criteria for HMI testing are introduced in addition to common coverage criteria [13] [7].

Works [18] and [9] focus on variability of SW products with product-line approaches in the domain of SW development. In [9] a state machine contains all potential features of all products in the product family. The goal is to choose required sub states from the state machine, resolve the relations and generate code for a certain product. In [18] for each feature of the product family there is a state machine available. The task is to select the required state machines, find the connections of them and generate code for a certain product.

ERROR CLASSIFICATION AND DISTRIBUTION

Past infotainment system projects containing advanced and complex HMIs are chosen for a statistical analysis. HMI error tickets are evaluated which were created during the development phase. Figure 1 presents the classification and distribution of the errors.

About a quarter of HMI errors are in the menu logic, so-called menu behavior errors. They appear in the form of switching to an unexpected menu in response to some inputs from the user or underlying applications. In practice of automotive HMI domain, the menu behavior is usually specified with statechart models.

More than the half of the HMI errors are in the views and contained graphical elements. Views are usually called screens in the automotive domain.

A view usually contains static contents such as a title and

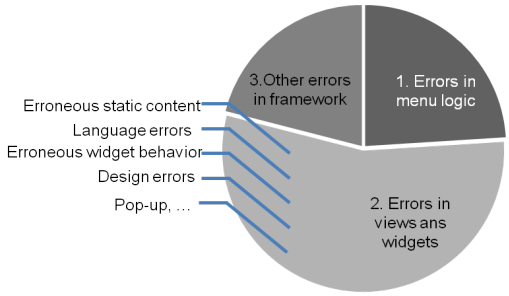

Figure 1. Error classification and distribution

subtitle as in Figure 2, which are displayed at any time and in any context. The error statistic has shown that an essential part of HMI errors arise from erroneous static content such as a missing text. They can be easily found, if menu behavior tests are extended with sub tests verifying the static contents. Required information for testing static contents are specified in the presentation layer of the test-oriented HMI specification, which will be introduced in the next section. Simple image processing methods can be used to get the presented texts from the display [5].

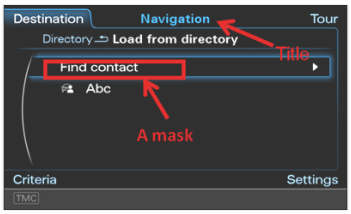

Figure 2. Static content of a menu and a mask

Infotainment system HMIs are usually available for many languages. Language errors are either contained in the localization database or caused by erroneous linking between entries in the localization database and representing widgets. Errors caused by erroneous linking and representing widgets can also be found by static content tests.

Infotainment system HMIs contain usually a lot of advanced widgets with dynamic behavior. Dynamic widgets lead to a lot of HMI errors. The widget behavior can also be defined with statechart models. Widget behavior tests can be generated from these models and extended to menu behavior tests. However our prototype of a widget behavior model has shown that modeling widgets can be very work-intensive and time-consuming. It only makes sense to model especially errors-prone widgets and test them automatically. Therefore it is very important that the menu behavior and the widget behavior are separated in different models in a HMI testing concept.

There are many other errors, which are not in the HMI but directly affect the HMI behavior. They are either due to the HMI framework or underlying applications. For example, the phone application has sent an empty string as a contact name to the HMI or the switching between different

menus have some delay because the bus system is heavily loaded. Modeling the behavior of underlying applications or the whole infotainment system is infeasible for infotainment system HMIs. This category of errors can not be found with the concept.

Preliminary evaluation results provide only a picture of the error classification and contribution. One can define the categories in a very different way and we believe, statistic of other projects can deviate from current results.

TEST-ORIENTED HMI SPECIFICATION

In the last section, we have introduced that menu behavior errors construct an essential part of HMI errors. To detect menu behavior errors, a test model has to be available which specifies the expected menu behavior. In our concept, such a test model is called menu behavior model and is arranged in the behavior layer of the test-oriented HMI specification.

A test-oriented specification [5] [6] is a HMI specification, which contains sufficient information for testing purposes. It's constructed with a layered structure as shown in Figure 3:

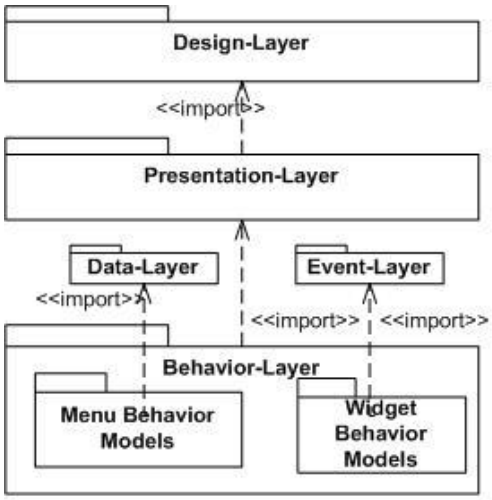

Figure 3. Layers of the test-oriented specification

The presentation layer contains testing-relevant information about screens and their graphical elements thus: potential events which can be triggered by a screen and the abstract content and structure of a screen. Data and event layers contains variables and events used in other layers. The widget behavior models describe the behavior of complex widgets. The design layer which contains design information is optional. It is only required if design tests should be performed. This is not the focus of our work. As shown in Figure 3, menu behavior models are separate to other models or information. This separation provides the possibility to specify the menu behavior and perform menu behavior tests independently.

THE MENU BEHAVIOR MODEL

In this section we focus on the specification of the menu behavior and introduce some required extensions of the UML state machine for specifying the menu behavior with testing purposes. We introduce our extensions based on the state machine definition from the OMG [14].

ViewState

Currently three kinds of states are distinguished in the state machine: simple state, composite state and submachine state. A new kind of state: ViewState has to be extended for describing the HMI menu behavior. A view state is a special kind of simple state signifying that the current state is associated with an abstract screen in the presentation layer. A views state has an attribute of type string, which is the name of the associated view in the presentation layer. When a view state is active, the associated view has to be displayed.

PreStepsCondition

In many situations some user actions are only enabled, if some other actions are previously performed. For example, entering a city name as navigation destination is only enabled, if a country name has been entered. A test which enters a city name and starts the guidance without to enter a country name before is an invalid test. PreStepsCondition is extended into the state machine, which indicates this dependency for the test generator. Transitions which need other transitions as previous steps, have to be labeled with a PreStepsCondition. Transitions fulfilling some PreStepsConditions have to be labeled with actions making these conditions true. A PreStepsCondition contains a function, which evaluates to a boolean value. For each PreStepsCondition, there must be at least one transition labeled with an action which makes the contained function true.

RuntimeCondition

An infotainment system HMI is not a closed application such as a simple calender, which contains the complete behavior logic in itself. During the runtime, an infotainment HMI communicates with the underlying applications almost all the time. The menu behavior is strongly dependent on the runtime data. However at the time of creating models and generating tests, the runtime data and consequently the completion of conditions are unknown. For example, when a user has entered the destination completely and started the route guidance, the screen with the map and calculated results should only be shown after that the calculation is finished. However, the calculation is performed by the underlying application. That means, the transition pointing the view state associated with the map screen is only active if a condition is fulfilled during the runtime by the underlying application. A new type of condition "RuntimeCondition" has to be extended to describe this dependency. A RuntimeCondition contains a function, which evaluates to a boolean value. The function has to be bound with runtime variables, from e.g. the interface between the head unit and the underlying application. The function can only be fulfilled by the runtime variables.

Some new event types

Figure 4 show the event types defined by the OMG for the UML state machine.

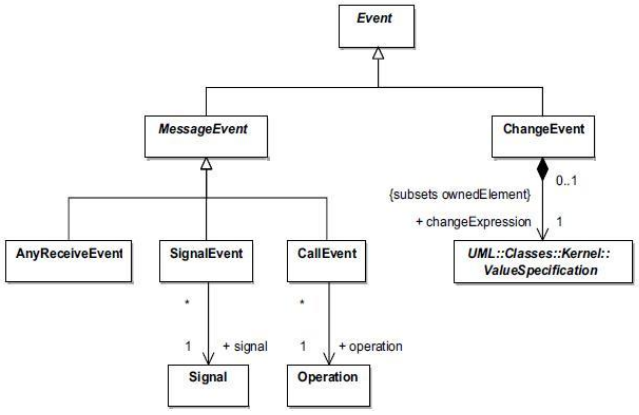

Figure 4. Event types in a UML state machine

We have defined different types of events depending on their sources. For instance, the type ApplicationEvent is defined for events initialized by underlying applications. An application event can be a message event or a change event. In events which are performed by users we distinguish two types: GlobalEvent and ReacionEvent. Global events can be performed via the control unit anytime and they are effective for any HMI states. For example, for switching between different infotainment system functions such as radio and navigation, buttons are provided in the control unit, which trigger global events. In contrast to global events, a reaction event can only be triggered in certain screens. Distinguishing different event types is very important for the test generation and instantiation.

A lot of functions of infotainment systems require user input-data, e.g. a phone number to dial or a destination for the guidance. Representatives for each user input-data equivalent class have to be tested. User input-data for tests should better be separately defined independent from the test model (not a part of the test model). In this way, to change the user input-data for different testing purposes or phases would not lead to some model changes. On the other hand, the separately defined user input-data can be reused for other tests.

To bind the user input-data, "UserInputEvent" has to be extended into the state machine. Currently we have defined equivalent classes for correct user inputs and unexpected user inputs.

COVERAGE CRITERIA FOR INFOTAINMENT SYSTEM HMI TESTS

We have implemented test generation algorithms based on some common coverage criteria in order to evaluate their adequacy for infotainment system menu behavior testing.

As explained, the menu behavior of an infotainment system HMI is strongly dependent on the runtime data. For an infotainment system HMI with 1000 menus, up to 250 conditions states are needed to model the dependency of the menu behavior on runtime data. So we have implemented a generation algorithm based on the branch coverage, which means all outgoing transitions of existing condition states have be tested. Our implementation is based on the depth search and allows currently each cycle for once. The generation results have shown that the generated tests can cover all branches, the number of generated tests is limited and the tests are very short. Generated tests are very unusual user scenarios.

Infotainment systems are very function-oriented. Each function e.g. starting the route guidance is usually accessible on one unique menu. We have implemented an algorithm, which generates all paths to a destination menu in which a certain function is accessible. All-path coverage could produce infinite tests. To limit the number of generated tests we allow each cycle only once. The generated tests cover all possible paths to the destination menu. However, most of the tests are in the same equivalent class, which means, the error could already be found with only one of the tests. Execution of all tests is unnecessary and impossible in the testing life due to very limited testing time and resources. Evaluation of other common coverage criteria e.g. transition coverage and HMI-special criteria as introduced in [12] and [16] is planned.

We firstly discuss criteria of adequate and efficient tests for infotainment system HMIs.

One of the most important requirements in premium HMIs is a faultless textual and graphical representation. So viewing all existing menus for all languages at least once would be the first criterion for infotainment HMI tests. We could derive the ViewState-coverage from this criteria.

Reusability of menus is very common in the implementation of infotainment system HMIs. For example a menu representing the contacts exists only once and is accessible from both the navigation and address book context. The menu shows different color and widgets depending on the accessing context. The reuse of menus could be very error-prone. So tests accessing reused menus from different ways can be very efficient to find errors.

We are still working on the definition of infotainment system HMI-specific coverage criteria based on the found adequacy criteria.
At last we would like to show a small example of a generated test, which firstly enters a destination and then starts the route guidance. We use the syntax [] for an expected menu name and () for a test step:
[navMain]−> (enter on widget "country") −> (wait(5ms)) −> [navCountryList]−> (userInput_selectCountry_correct) −> (enter) −> [navMain]−> (enter on widget "startRG") −> [navRgStarted]

INTEGRATING VARIABILITY INTO MODEL-BASED HMI-TESTING

A software product-line (PL) [3] [8], also called system family, is a set of software systems sharing common features that satisfy the specific need of a particular market segment and that are developed from a common set of core assets in a prescribed way.

Infotainment system HMIs are multi-variant products. The variability results from product series such as different generations, market variants such as for Europe or Asia, configuration variants such as with or without DVD player and system variants such as standard resolution with normal display or higher resolution with larger display. In practice many of these variants are developed in one project due to a large set of commonalities in features, looks and behaviors. That means, an HMI model in such a project describes all features, looks and behaviors potentially required for different variants e.g. for both Europa- and Asia-market. A product, which is created from such a model exactly satisfies one variant e.g., a standard system with navigation feature for the Europa market. For these reason, the model-based HMI testing concept has to be extended to support the variability.

We reuse feature models to extend the test-oriented HMI specification for the variability management. Feature models (FMs) allow us to describe both commonalities and differences of all products of a PL and to describe the relationships between them. A FM configuration (FMConf) is an instance of a FM that describes the properties and functionality of a product. In [4] FM and FMConf are described in details. We extend the in [4] defined FM and FMConf with distinction of two kinds of children: functional features and non-functional features. A functional feature can be e.g. the feature radio or navigation. A non-functional feature can be a variant feature, e.g. Europa variant or Asia variant. Usually, the relationship between functional features is or-relation and the relationship between variant features is alternative-relation. Figure 5 shows a strongly simplified example: f stands for a functional feature and v stands for variant feature.

To extend menu behavior models for variabilities, some

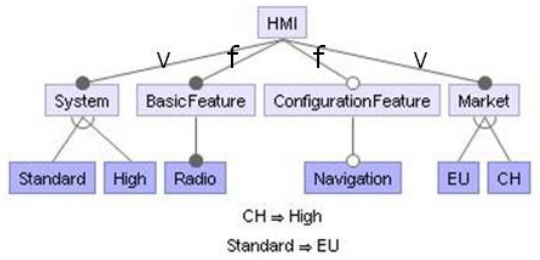

Figure 5. A FM with functional and non-functional features

new elements have been introduced as extensions for UML state machines. A feature composite state is a special kind of composite state in which the behavior of a function feature is described. A feature composite state is always related to a functional feature in the FM. An HMI allows inter-feature activities e.g., from the feature navigation the user can switch to the feature telephone and choose an address of a contact as destination. Therefore, in the menu behavior model there can be transitions between two feature composite states, which are called inter-feature transitions. Furthermore, variation points and junction points which are originally defined in [15] for activity diagrams are extended for the state machine. Each variation point is related to a child node which is marked with v in the FM e.g. "system" in Figure 5. Each outgoing transition is related to one of the contained variant features e.g. "standard" or "high". A variation point can only be used within a pair with a junction point, which merges the distinction of the variant feature behaviors.

Also the presentation layer has to be extended for variabilities. Parameterized inheritances are used in the abstract description both of the screen structure and events which can be potentially triggered by the screen. Since presentation layer is not the focus of this paper, it is thus not further discussed.

Each infotainment system test bench conforms to a valid FMConf. For instance, a test bench is a "high" system for Europa with the basic feature radio and configurable feature navigation as shown in Figure 5. In a testing farm test benches for many configurations are available. Since many configurations share a common set of functional or non-functional features, avoiding the redundancy is one of the most important requirements for the test generation and test execution. Therefore the test generation is composed of two steps. Firstly, algorithms traverse the whole test model and generate partial tests for all required functional and non-functional features. Then tests are created from these partial tests for all required configurations. In this way redundant generation of common features are avoided. If changes are only carried out in a sub set of the features, the test generation is able to regenerate partial tests from the affected partial test model and the second step has to be executed for the affected features. Avoiding redundant test execution is especially important in industrial practice due to limited test resources.

CONCLUSION

In this paper, we introduced some preliminary results of our model-based testing research for infotainment system HMIs. Error statistic and some additional elements needed for modeling the menu behavior were introduced. We also discussed which tests are adequate and efficient to detect errors in our area. Furthermore, we have briefly introduced the main ideas how we extend the model-based HMI testing for variabilities.

ADDITIONAL AUTHORS

REFERENCES

1. F. Belli. Finite-state testing and analysis of graphical user interfaces. In *ISSRE '01: Proceedings of the 12th International Symposium on Software Reliability Engineering*, page 34, Washington, DC, USA, 2001. IEEE Computer Society.

2. V. Chinnapongse, I. Lee, O. Sokolsky, S. Wang, and P. L. Jones. Model-based testing of gui-driven applications. In *Software Technologies for Embedded and Ubiquitous Systems*, volume 5860/2009, pages 203–214. Springer-verlag New York Inc, 2009. 7th IFIP WG 10.2 International Workshop, SEUS 2009 Newport Beach, CA, USA, 2009 Proceedings.

3. P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in SE. Addison-Wesley, 2002.

4. K. Czarnecki. Generative programming: Methods, techniques, and applications. In *Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools*, ICSR-7, pages 351–352, London, UK, UK, 2002. Springer-Verlag.

5. L. Duan, A. Hoefer, and H. Hussmann. Model-based testing of automotive hmis based on a test-oriented hmi specification model. In *Proceedings of the the 2nd International Conference on Advances in System Testing and Validation Lifecycle (VALID 2010), August 22-27 2010, Nice, France*. IEEE, Aug. 2010.

6. L. Duan, H. Hussmann, and A. Höfer. A test-oriented hmi specification model for model-based testing of automotive human-machine interfaces. In *GI Jahrestagung (2)*, pages 339–344, 2010.

7. C. Gaston and D. Seifert. Evaluating coverage based testing. In *Model-Based Testing of Reactive Systems*. Springer-Verlag New York, LLC, 2005.

8. H. Gomaa. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. The Addison-Wesley Object Technology Series. Addison-Wesley, 2004.

9. A. Gonzalez and C. Luna. Behavior specification of product lines via feature models and uml statecharts with variabilities. In *2008 International Conference of the Chilean Computer Science Society*, pages 32 – 41. IEEE Computer Society, 2008.

10. A. Kervinen, M. Maunumaa, T. Pkknen, and M. Katara. Model-based testing through a gui. In *In Proceedings of the 5th International Workshop on Formal Approaches to Testing of Software (FATES 2005), number 3997 in Lecture Notes in Computer Science*, pages 16–31. Springer, 2006.

11. M. A. M. An event-flow model of gui-based applications for testing: Research articles. *Softw. Test. Verif. Reliab.*, 17(3):137–157, 2007.

12. A. M. Memon, M. L. Soffa, and M. E. Pollack. Coverage criteria for GUI testing. pages 256–267, 2001.

13. J. Offutt and A. Abdurazik. Generating tests from uml specifications. page 76, 1999.

14. OMG. Omg uml, superstructure. `http://www.omg.org/spec/UML/`.

15. S. Reis and K. Pohl. Wiederverwendung von integrationstestfällen in der software-produktlinienentwicklung. *Inform., Forsch. Entwickl.*, 22(4):267–283, 2008.

16. H. Reza, S. Endapally, and E. Grant. A model-based approach for testing gui using hierarchical predicate transition nets. In *Proceedings of the International Conference on Information Technology*, ITNG '07, pages 366–370, Washington, DC, USA, 2007. IEEE Computer Society.

17. H. Reza, K. Ogaard, and A. Malge. A model based testing technique to test web applications using statecharts. In *ITNG '08: Proceedings of the Fifth International Conference on Information Technology: New Generations*, pages 183–188, Washington, DC, USA, 2008. IEEE Computer Society.

18. N. Szasz and P. Vilanova. Statecharts and variabilities. In P. Heymans, K. C. Kang, A. Metzger, K. Pohl, P. Heymans, K. C. Kang, A. Metzger, and K. Pohl, editors, *VaMoS*, ICB Research Report, pages 131–140, 2008.

Towards Ubiquitous Emergency Management Systems

Jan Zibuschka
Fraunhofer IAO
Nobelstr. 12
70569 Stuttgart
jan.zibuschka@iao.fraunhofer.de

Uwe Laufs
Fraunhofer IAO
Nobelstr. 12
70569 Stuttgart
uwe.laufs@iao.fraunhofer.de

Heiko Roßnagel
Fraunhofer IAO
Nobelstr. 12
70569 Stuttgart
heiko.rossnagel@iao.fraunhofer.de

ABSTRACT
This contribution introduces an emergency management system design based on platform-independent multi-touch technology as an interactive, ubiquitous front-end technology for pervasive sensor and communication components. This combination aims at supporting decision making and analyses during all phases of the emergency management process. From stakeholders providing planning information beforehand to end users communicating via their mobile phones or even distributed sensor networks, the system taps into a wide range of data sources and offers a comprehensive, digestible view on the data using multi-touch surfaces in the operations centre. In order to integrate legacy data sources and to keep the system open for the integration of additional data sources in the future, a model based approach is used. Demonstration versions of the system's components based on current multi-touch frameworks and hardware as well as mobile apps and communities are also presented.

Categories and Subject Descriptors
H.5.2 [**User Interfaces**]: Input Devices and Strategies; D.2.11 [**Software Architectures**]: Domain-specific Architectures;

General Terms
Design, human factors, software engineering.

Keywords
Multi-touch, emergency management, crisis response, mobile.

1. INTRODUCTION
In emergency situations, the stakeholders responsible for the organization and execution of the emergency management have to cope with complex situations and short time frames for reaction. Therefore, they often have to make quick decisions based on the data available to them. Emergency management systems (EMS) provide the capability to provide crucial information support and to enable disaster forces to manage disaster events, including detection and analysis of incidents [10].

According to [35] the lifecycle of a crisis or emergency comprises of several stages:

(1) a pre-event stage allowing the development of strategy and plans;

(2) a stage immediately before or after a crisis or disaster occurs which requires the implementation of strategies to deal with its impacts;

(3) continued implementation of strategies to control or reduce the severity of the crisis/disaster; and,

(4) a long term recovery or resolution phase allowing for evaluation and feedback into future prevention and planning strategies for destinations and businesses.

Emergency management systems that utilize ubiquitous components can provide relevant information during all phases of the emergency lifecycle that can contribute to saving human lives. In this contribution we propose a system design for ubiquitous emergency management that addresses these potentials. We base our approach on pervasive information gathering components that are connected to ubiquitous monitor devices handled by e.g. first responders.

We start with a short review of related work in section 2. We outline our overall approach in section 3 and discuss its implementation in section 4, before we summarize our results.

2. RELATED WORK
There has been a lot of work on supporting crisis control rooms with visualization and interaction on large displays, e.g. [20] [40], including multi-touch solutions [14]. Multi-touch has also been used in related disciplines such as IT security for visualizing the outputs of intrusion detection systems [17], as interface to large scale simulation in the context of astrophysics [16], or as front end for sensor networks [27].

Such systems are often used in this context as visualizing spatial information is one of the technology's strengths [39]. An overview of spatial information and geographic information systems in the context of emergency management is given in [12].

Micire et al. [28] offer a broad set of technologies integrated into a multi-touch interface for disaster response. Similarly, the German SoKNOS project has developed a broad visualization and data integration approach [13] for supporting *"the response and the recovery from natural and socio-technical disasters"* [31]. SoKNOS also provides planning components, and integrates web services and sensors. We envision a system where planning and learning from past events are more central, and which is able to data mine web communities, as well as provide real time information from integrated mobile apps.

Zibuschka et al. [42] present a multi-touch design aimed at supporting the planning stages of emergency management. The presented design integrates information provided by the individual stakeholders to provide a common picture to support planning of large public events. We extend this approach by adding additional support for the after-event stages, and integrate a broader set of data sources, such as mobile apps and web sites such as social networks or news sites.

Nóbrega et al [29] present a system which simulates the propagation of disaster such as floods, and can visualize the scenarios on an interactive display for operative briefings. Simulation is not currently our main focus, but the visualized scenarios are similar.

Generally speaking, none of the systems presented offer support for the full emergency management life cycle. They often focus on the operations during a disaster, neglecting the planning and learning phases before and after the event. This is especially relevant as in case of disaster, it may not be prudent to rely on the availability of a technical system too much; much less its

established ubiquitous components such as mining of external information or collection of information from end-users. Our contribution aims to fill this gap.

3. APPROACH

Our approach is built on three main ideas:

– Disaster management involves a lot of spatial information, and it makes sense to build an overarching information integration framework around this information [18], to improve communication possibilities and awareness of information from other stakeholders or from the field, which are key issues in emergency management [34].

– Multi-touch interfaces are very well suited for visualizing [27] and interacting with [15] spatial information.

– As already pointed out in the related work, we feel that disaster management systems should focus more on the pre-event and resolution phases [35], as infrastructure failures can knock out the system infrastructure for information integration [26].

Realizing a system built on these basic ideas, we try to meet the basic requirements for emergency management systems identified by Scherner et al. [38]. They are:

– system effectiveness

– reliability

– cost efficiency

– smooth service integration

– multilateral user interaction

– availability

– security.

To address these factors, we build on the reference architecture given by Roßnagel et al [37].

3.1 Vision

Our vision is to make emergency management systems truly ubiquitous in nature, combining mobile and/or pervasive sensors (in the broadest sense) and mobile terminals in general with ubiquitous Roomware [32] approaches supporting the staff in the crisis control centre. The components are tied together by a model-driven data integration engine fusing data from various sources (Figure 1).

In the field, sensor networks pervasively collect information. In the current implementation, those are mainly mobile clients handled by end users or first responders, but the future vision includes sensor networks (including "smart dust", appliances for detection of e.g. explosives and similar systems).

In addition, the system is also able to retrieve information from arbitrary web sites, e.g. third party news sites and communities, using a web mining approach. In the future, this component may leverage the ontologies used in the data integration for full-fledged reasoning on the Semantic Web or similar large-scale data integration approaches, but it is useful for monitoring social media and news even as is. In addition, interfaces for relevant information systems at involved organisations can be provided,

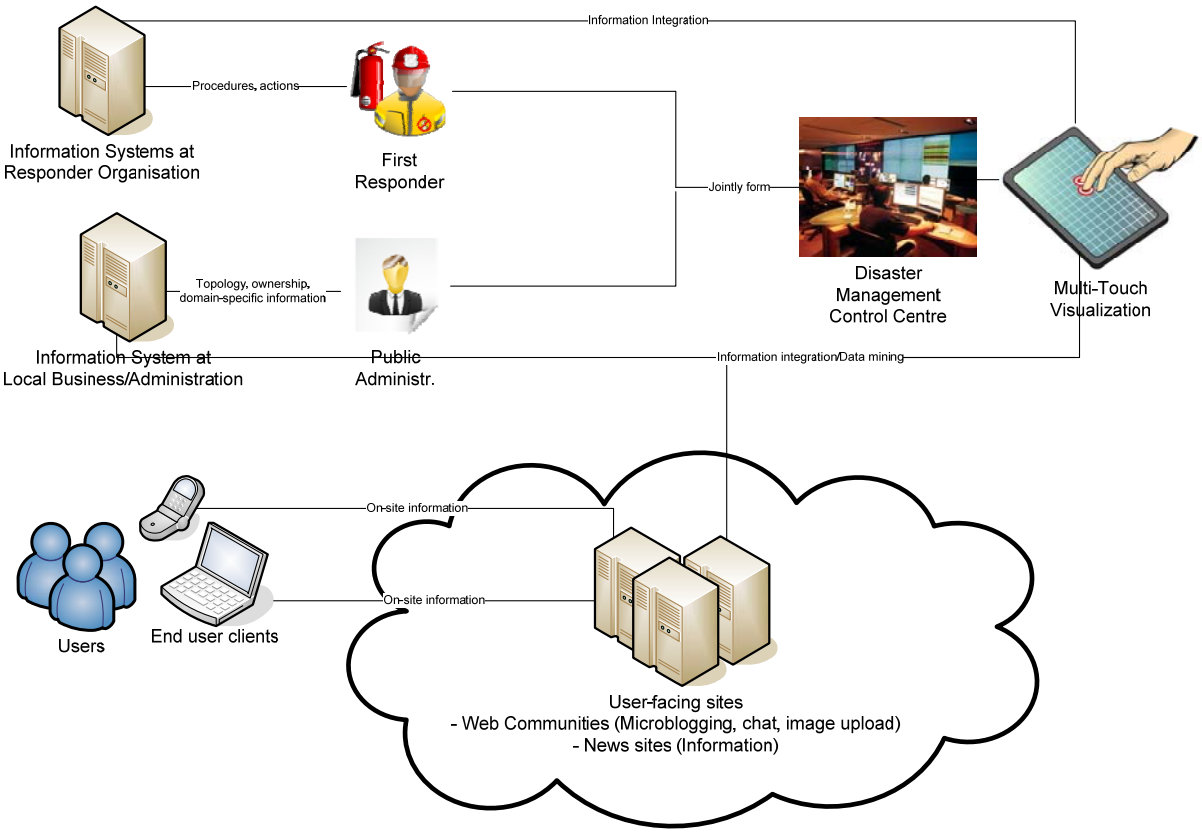

Figure1: Deployment Overview

allowing for integration of data like property and liability information from the local administration, or event reports from first responder organisations. The data collected this way makes the emergency management system immediately useful to its users [7].

The incoming information is digested using a data integration back-end that is built on ontologies, bringing the heterogeneous inputs into a form that can be displayed as spatial data on a map. For cases where this is not sensible, additional UI ontologies can be supplied to configure specific display modes for the information.

The visualization is performed on interactive displays back at the crisis control centre, providing a more complete picture of the situation in the field during operations and later during after-event briefing. The same visualizations may also be used by pervasive screens in e.g. first responder vehicles or mobile clients. For this, it is very beneficial to use a portable multitouch API solution supporting all common operating systems [24], including on mobile terminals.

Personnel at crisis control centres can benefit from collaborative interaction to improve information sharing and communication, especially during the pre-event and event resolution phases of the lifecycle [42].

In large part, this vision is similar to what is presented (purely as a video illustrating the vision) in [33], which is a work by collaborators. However, the system we envision has an additional focus on pre-event and event resolution support [35]. We can also offer a realization of the system based on today's technologies, described in the next sections.

3.2 Current Realization

To realize this vision today, we implement a system using current multi-touch surfaces, such as tables and tablets, as a front end, with a back-end consisting of on the one hand customized apps on mobile phones distributed by the stakeholders in various contexts (e.g. large public events [37]) and on the other hand with broad data integration capabilities, for deployment at e.g. control centres. As we are focussing on the post- and pre-event phases, we are not so much focussed on the communication interface to first responders in the field, instead focussing on interfaces to information systems at the involved organisations, as well as collecting input from users that were/are connected to the internet during the event, or reports acquired using data mining components from e.g. news sites post-event. An overview of this approach is given in Figure 1.

So, the system integrates several of the ubiquity aspects given in the vision. Mobile phones have a market penetration of more than 100% in the EU [30], and people are using them to push information about their surroundings all the time. We leverage this by integrating contemporary mobile apps with an information integration component and a collaborative multi-touch visualization that can also be brought to the small displays of mobile devices using the portable MT4j framework [24], as described in the next section.

4. IMPLEMENTATION

Earlier versions of the multi-touch and mobile components were developed in national project VeRSiert, but integration is still ongoing. In addition, the components are recontextualized in relation to the disaster management life cycle, covering more phases, specifically post-event support.

The mobile communities within the system are based on the design presented by [38] [37], offering both value-added services in the context of tourism/large public events/transportation and emergency services such as emergency notifications.

4.1 Mobile Services

The mobile service platform utilized in our current mobile service infrastructure is presented in [36]. It can be utilized for emergency management and commercial mobile value-adding service as described in the previous section. It offers modular basic services that were identified based on the value-adding event management and emergency services [36]. These basic services include:

- chat platform
- micro-blogging platform
- localization of users
- multicast and broadcast messages
- mobile ticketing
- mobile payment

As the same underlying technologies can be used for both value-added and emergency management services economies of scale significantly reduce the associated costs. In addition, by offering a service platform implementing those building blocks, a quick development of value-adding mobile services can be achieved [36].

We have implemented a system prototype, based on customized Open Source components for the server side, and using Google's Android [1] platform for implementation of the client application [36].

On the server side, we use StatusNet's Laconica micro-blogging service [2] (providing support for persistent, asynchronous, bidirectional communication between stakeholders such as end users and emergency managers). To also offer a synchronous, non-persistent communication channel allowing for group communications, we use the OpenFire [3] server implementing the eXtensible Messaging and Presence Protocol (XMPP) [36].

On the client, components like routing, chat client, micro-blogging connector and friend finder have been implemented using the Android API, in part based on additional online services like Google Maps [4] and components of the mobile phone, e.g. the Global Positioning System (GPS). Further basic services can be implemented as modular components. A rebranding of the client application was integrated, to allow for additional distribution channels, and enabling the system to reap the benefits associated with strong brand names in the context of new product deployment [36] [41].

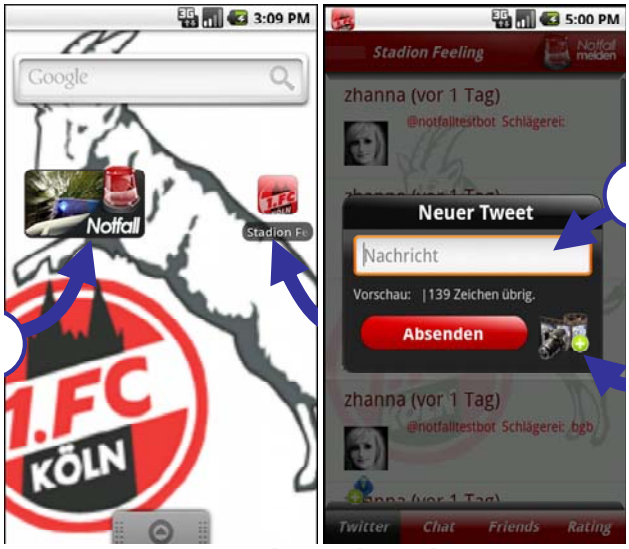

Figure 2:Prototype screenshots (widget and app, running app)

In addition to the app, we implemented a widget component for quick access to the emergency functionality (see Figure 2).

4.2 Pervasive Sensors

Another interesting type of data provider for emergency management is distributed sensor units. These devices can provide information e.g. about fire, toxic gasses or radiation. Furthermore, camera based systems can e.g. be used as a sensor for the detection of unusual large crowds [21]. While at the current state of the art, extensive use of sensors is quite expensive and difficult, the large amount of research activity in this area may provide cheaper and better suited solutions in the future.

At the current state of implementation, we use Sun SPOT devices [5] (Figure 3) as an example of mobile distributed sensor devices. It contains several on-board sensors and it provides breakouts which can be connected to external sensors. The device communicates via radio transmission as well as via a USB interface. While the device is still quite expensive and much bigger than sensors in the vision of smart dust, the device behaves similar regarding the way it communicates and the data that it produces.

Figure 3: Sun SPOTs as mobile sensor devices

4.3 Data Extraction

Within the World Wide Web already a lot of structured or semi-structured information exists which can be queried using standardized interfaces. For example, the Open Search interface [6] provides access to several content management systems, blogging systems and search engines and can be used to retrieve information from many existing sources in the web. Information sources without a dedicated external interface often can also be accessed using bots and parsers with additional effort for interfacing, extraction and additional maintenance efforts in case of changing structures of the information sources.

4.4 Multi-touch User Interface

The multi-touch front-end provides an integrated view of the available emergency management information. This includes information provided by the stakeholders as well as information from the other components of the overall system (sensors, mobile services and data extracted from the WWW). Location-dependent data is visualized on a map. There are several kinds of maps that can be used, e.g. aerial shots or roadmaps.

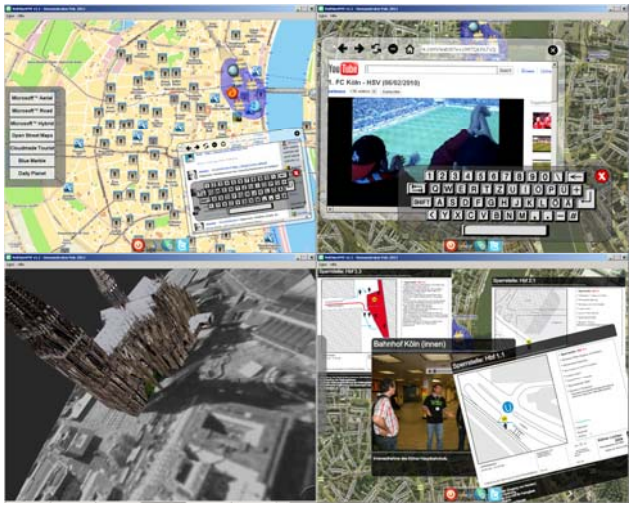

Figure 4: Screenshot prototype (Multi-touch application visualizing different kinds of location based information)

The multi-touch user interface is realized using "Multi-touch for Java" (MT4j), which is an open source framework for rapid development of multi-touch applications on the Java platform [24]. MT4j runs on Microsoft Windows, Linux and Mac OSX. It supports several multi-touch input protocols such as WM_TOUCH [23] on Windows 7 or the platform independent open source protocol standard TUIO [22]. In the meantime, a first alpha version of MT4j exists for Google's Android platform so that future versions of our multi-touch application can also be run on Android Tablets. Currently, we also think about an adaption of the application's user interface to the smaller display dimension of Android smartphones.

For collaborative use during the pre-event and event resolution phases [30], a 42 inch multi-touch terminal is used. The terminal runs on Windows XP. On the terminal, multi-touch motion data is transferred via the TUIO protocol. Stakeholders can also run the application on any windows or Linux PC. If the hardware is not capable of multi-touch, the application can also be controlled using traditional input devices like keyboard and mouse or a track pad. This allows the stakeholders to use the application on their

normal PC and to contribute e.g. planning information directly via the application.

The application manages the underlying data using the relational database system HSQLDB [19]. Multi-media content like images, videos or 3D models are stored as files and referenced from the database. Since all data including the database itself and all multi-media content is stored in one file system folder, it is possible to store the whole data in a single archive file. Since the system allows importing database archives in read-only mode, there is a simple way to merge data from the different stakeholder's systems.

4.5 Model Based Data Integration

Within the overall system, different kinds of information have to be collected and managed. Against the background of proprietary implementations and heterogeneous data structures as well as semantic differences in the data provided by the various data sources, a model based approach is used. We use ontologies to describe the data sources as well as the information provided by the various platform-internal and external data sources. This approach has already proven to be purposeful, especially in heterogeneous environments [8] [9]. For the realization of the description models, we decided to use the web ontology language (OWL) [25]. OWL is a XML-based ontology description language which is built upon the less expressive W3C standards RDF [7] and RDFS [11]. OWL itself offers three variants that contain different subsets of the OWL syntax. While OWL lite is focussed on simple classifications and restrictions OWL DL and OWL full offer much more expressiveness but also increase the complexity. We decided to use OWL DL because OWL lite is not expressive enough and OWL full does allow complex definitions on which no formal decision making is possible.

In order to allow operating on a defined data structure, a unified data model is used. It describes all the information, the system can operate with. In a data source description model, meta-information about the specific data sources is stored. A visualization model for each data source allows the customization of the user interface. It describes which specific information is visualized and how it is presented. This leads to a high configurability of the front-end to end-user requirements while minimizing costs for minor changes.

A data source connector for each data source has to be implemented. It accesses the data using the given mechanism depending on the specific data source. For example, the data provided by the mobile services is stored in a RDBS and queried via SQL by the connector. A data management module provides scheduling functionality for data fetching. It fetches data using the data source connectors. The management module is also responsible for the storage of the fetched data.

5. CONCLUSION

In this contribution, we presented a ubiquitous emergency management system design, based on the integration of mobile and multi-touch components in the front end with sensor fusion and data mining capabilities in the back end. As we derived our system from evaluated designs from literature, we hope to address the requirements given in [38].

6. ACKNOWLEDGMENTS

This work was in part supported by the VeRSiert project; however, it represents the view of the authors only. The authors would also like to acknowledge their colleagues from VeRSiert, who offered invaluable feedback on the earlier versions of the mobile services and multi-touch.

7. REFERENCES

1. Android.com. http://www.android.com/.
2. StatusNet. http://status.net/.
3. Ignite Realtime: Openfire Server. http://www.igniterealtime.org/projects/openfire/.
4. Google Maps. http://maps.google.com/.
5. SunSPOTWorld - Home. http://www.sunspotworld.com/.
6. Home - OpenSearch. http://www.opensearch.org/Home.
7. Ankolekar, A., Krötzsch, M., Tran, T., and Vrandecic, D. The two cultures: Mashing up Web 2.0 and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web 6*, 1 (2008), 70-75.
8. Bügel, U. and Laufs, U. Einsatz innovativer Informations- und Kommunikationstechnologien. In *Fokus Technologie. Chancen erkennen, Leistungen entwickeln*. Hanser, München, 2009.
9. Bullinger, H.-J. *Fokus Technologie*. Carl Hanser Verlag, München, 2009.
10. Carver, L. and Turoff, M. Human-computer interaction: the human and computer as a team in emergency management information systems. *Communications of the ACM 50*, (2007), 33–38.
11. Celino, I., Valle, E., and Cerizza, D. From Research to Business: The Web of Linked Data. In *Business Information Systems Workshops*. 2009, 141-152.
12. Cutter, S.L. GI Science, Disasters, and Emergency Management. *Transactions in GIS 7*, 4 (2003), 439-446.
13. Döweling, S., Probst, F., Ziegert, T., and Manske, K. Soknos — An Interactive Visual Emergency Management Framework. In *GeoSpatial Visual Analytics*. 2009, 251-262.
14. Flentge, F., Weber, S.G., and Ziegert, T. Designing Context-Aware HCI for Collaborative Emergency Management. *Int'l Workshop on HCI for Emergencies in conjunction with CHI*, (2008).
15. Forlines, C. and Shen, C. DTLens: multi-user tabletop spatial data exploration. *Proceedings of the 18th annual ACM symposium on User interface software and technology*, (2005), 119–122.
16. Fu, C.-W., Goh, W.-B., and Ng, J.A. Multi-touch techniques for exploring large-scale 3D astrophysical simulations. *Proceedings of the 28th international conference on Human factors in computing systems*, (2010), 2213–2222.
17. Guenther, J., Volk, F., and Shaneck, M. Proposing a multi-touch interface for intrusion detection environments. *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, (2010), 13–21.
18. Herold, S., Sawada, M., and Wellar, B. Integrating geographic information systems, spatial databases and the internet: a framework for disaster management. *Proceedings of the 98th Annual Canadian Institute of Geomatics Conference*, (2005), 13–15.
19. HyperSQL. HSQLDB. *http://hsqldb.org/*. http://hsqldb.org/.
20. Ijsselmuiden, J., van de Camp, F., Schick, A., Voit, M., and Stiefelhagen, R. Towards a Smart Control Room for Crisis Response Using Visual Perception of Users. *Poster at ISCRAM 2004*, (2004).
21. Junker, O., Strauss, V., Majer, R., and Link, N. Real-time video analysis of pedestrians to support agent simulation of

people behavior. *Fifth International Conference on Pedestrian and Evacuation Dynamics (PED 2010)*, NIST (2010).
22. Kaltenbrunner, M., Bovermann, T., Bencina, R., and Constanza, E. TUIO: A protocol for table-top tangible user interfaces. *6th International Gesture Workshop*, (2005).
23. Kiriaty, Y. MultiTouch Capabilities in Windows 7. *msdn magazine*, 2009, http://msdn.microsoft.com/en-us/magazine/ee336016.aspx.
24. Laufs, U., Ruff, C., and Zibuschka, J. MT4j - A Cross-platform Multi-touch Development Framework. *Engineering Patterns for Multi-Touch Interfaces 2010 - Workshop at the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ACM (2010).
25. McGuinness, D. and van Harmelen, F. OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/.
26. Meissner, A., Luckenbach, T., Risse, T., Kirste, T., and Kirchner, H. Design challenges for an integrated disaster management communication and information system. *The First IEEE Workshop on Disaster Recovery Networks (DIREN 2002)*, (2002).
27. Merrill, D., Kalanithi, J., and Maes, P. Siftables: towards sensor network user interfaces. *Proceedings of the 1st international conference on Tangible and embedded interaction*, (2007), 75–78.
28. Micire, M.J. and Yanco, H.A. Improving disaster response with multi-touch technologies. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, (2007), 2567-2568.
29. Nóbrega, R., Sabino, A., Rodrigues, A., and Correia, N. Flood Emergency Interaction and Visualization System. In *Visual Information Systems. Web-Based Visual Information Search and Management*. 2008, 68-79.
30. Paul Meller. EU report: More mobile phones than citizens - Computerworld. 2007. http://www.computerworld.com/s/article/9014938/EU_report_More_mobile_phones_than_citizens?taxonomyId=15&taxonomyName=mobile_and_wireless.
31. Paulheim, H., Döweling, S., Tso-Sutter, K., Probst, F., and Ziegert, T. Improving usability of integrated emergency response systems: the SoKNOS approach. *Proceedings "Workshop zur IT-Unterstützung von Rettungskräften", LNI Vol. 154*, (2009), 1435–1449.
32. Prante, T., Streitz, N.A., and Tandler, P. Roomware: computers disappear and interaction evolves. *Computer 37*, 12 (2004), 47-54.
33. PrecisionInformation. Precision Information Environments » Blog Archive » VISION VIDEO. http://precisioninformation.org/?p=32.
34. Quarantelli, E.L. Disaster crisis management: A summary of research findings. *Journal of Management Studies 25*, 4 (1988), 373–385.
35. Ritchie, B.W. Chaos, crises and disasters: a strategic approach to crisis management in the tourism industry. *Tourism Management 25*, 6 (2004), 669-683.
36. Roßnagel, H., Zibuschka, J., and Junker, O. EVALUATION OF A MOBILE EMERGENCY MANAGEMENT SYSTEM – A SIMULATION APPROACH. *AMCIS 2010*, (2010).
37. Roßnagel, H., Zibuschka, J., Muntermann, J., and Scherner, T. Design of a mobile service platform for public events - Improving visitor satisfaction and emergency management. *Joint proceedings of ongoing research and projects of IFIP EGOV and ePart 2010*, Trauner (2010), 193-200.
38. Scherner, T., Muntermann, J., and Rossnagel, H. Integrating value-adding mobile services into an emergency management system for tourist destinations. *ECIS 2009 Proceedings*, (2009).
39. Schöning, J. and Krüger, A. Multi-modal navigation through spatial information. *adjunct Proc. of GIScience 2008*, (2008).
40. Stasch, C., Daiber, F., Walkowski, A.C., Schöning, J., and Krüger, A. Multi-Touch- und Multi-User-Interaktion zur Verbesserung des kollaborativen Arbeitens in Katastrophenstäben. *Geoinformatik 2009*, ifgi prints (2009).
41. Tauber, E.M. Brand franchise extension: New product benefits from existing Brand Names. *Business Horizons 24*, 2 (1981), 36-41.
42. Zibuschka, J., Laufs, U., and Engelbach, W. Entwurf eines kollaborativen Multi-Touch-Systems zur Planung und Abwicklung von Großveranstaltungen. *GI Jahrestagung (1)*, GI (2010), 825-830.

A Framework for Transforming Abstract Privacy Models into Implementable UbiComp System Requirements

Ivan Gudymenko
Faculty of Computer Science
Dresden University of Technology
ivan.gudymenko@gmail.com

Katrin Borcea-Pfitzmann
Faculty of Computer Science
Dresden University of Technology
katrin.borcea@tu-dresden.de

ABSTRACT
During the development of UbiComp systems, privacy and security issues often come into play only after the design process is complete. The main development effort is typically concentrated on the direct functionality of the system, which too often results in immaturity of privacy compliance of the end product. This is one of the main burdens on the way to acceptance of such systems among potential users and to commercial success thereof as a consequence.

We claim that ensuring privacy and security in any UbiComp system should be taken into account already at the system design stage and should continue throughout all steps of the development of a UbiComp system. In this paper, we focus on privacy issues of UbiComp, namely we consider a framework which enables for consistent transformation of abstract privacy models into a set of implementable system requirements. A general approach to creating an abstract privacy model, which takes into account social, legal, and functional issues, is outlined. The further transformation of the model into a set of system-specific and platform-independent requirements is described.

Author Keywords
Ubiquitous Computing, Privacy, Privacy Model

ACM Classification Keywords
K.6.5 MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS: Security and Protection—*Privacy*

General Terms
Design

INTRODUCTION

Marc Weiser, one of the pioneers in the area of Ubiquitous Computing (UbiComp), outlined this concept as "The idea of integrating computers seamlessly into the world at large [...]" [21]. Frank Stajano in his book [18] described UbiComp as "[...] a scenario in which computing is *omnipresent*, and particularly in which devices that do not look like computers are endowed with computing capabilities." According to him, UbiComp does not imply "the computer on every desk" but rather embodying the computational power into different parts of the surrounding environment (clothes, household appliances, etc.), that normally are not considered to be equipped with it thus making them "smart".

Whereas UbiComp introduces a set of tangible benefits for the user[1], it also raises serious privacy concerns. The reason for this is that the advances of sensing technology and memory amplification have provided UbiComp systems with qualitatively new opportunities of covert surveillance. Marc Langheinrich claimed in [11] that "ubiquitous devices will per definition be ideally suited for covert operation and illegal surveillance, no matter how much disclosure protocols are being developed".

Privacy concerns of the users can impede the development and especially the deployment of UbiComp systems. To give an example, alongside its intended purpose, Smart Grid systems may pave the way to privacy violation scenarios (see [8, 13] for more details.)

That means, that deploying a UbiComp system in a privacy-preserving manner will increase the likelihood of its acceptance among potential users and broaden the target audience. Moreover, having created the infrastructure of a UbiComp system, it is relatively easy to deliver the end product to customers (to deploy the system, e.g. accompany individuals with respective sensors) since "individual investments pay off immediately" [14]. Due to this fact and because of the higher acceptance among customers, a system with decent privacy management mechanisms is more likely to be commercially successful.

In this paper, we outline how a UbiComp system can be designed in a privacy-preserving way. Namely, a concept, which describes how privacy requirements can be elaborated and how respective privacy mechanisms can be "woven" into UbiComp system's functionality, is considered. This approach enables for privacy to be *inherently* built into the UbiComp system under development, which should facilitate privacy management in the deployed system and make it more efficient.

USED TERMINOLOGY

Privacy is a broad notion and defining it is a difficult task due to the substantial difference of privacy perception among individuals. However, in order to avoid ambiguity and not to confuse the reader, we present our own understanding of this notion.

[1]For example, unobtrusiveness of the devices with respect to their size and operation mode, ability of the user to concentrate on the specific (business) tasks without having to pay much attention to the management of the underlying technical system, etc.

I. Gudymenko, K. Borcea-Pfitzmann: A Framework for Transforming Abstract Privacy Models into Implementable UbiComp System Requirements. Proc. of 1st International Workshop on Model-based Interactive Ubiquitous System 2011, Pisa, Italy, June 13, 2011, http://ceur-ws.org/Vol-787

The widespread ways of understanding privacy are ”the right to be let alone” [20] and also ”the right to be forgotten” [19, 5]. One of the common delusions is that the ”more” privacy the individual has, the better his identity is protected. However, privacy does not have a monotonic behavior. The optimum is situated in the vicinity of the ”golden middle” because individuals live in society and therefore experience the need for social interaction. This implies exchanging of certain pieces of private information between communicating entities. We claim, however, that individuals, without fully realizing it, need *adequate* and *appropriate* privacy. Managing privacy implies constant processes of negotiation between the parties involved and also with the person[2] concerned of which personal information of that individual is given out in which situation and enforcing that his/her privacy policy is being followed. Thus, in order to take the aforementioned issues into account, we adhere to the following definition of privacy, elaborated in [3]:

DEFINITION. *Privacy of an entity is the result of negotiating and enforcing when, how, to what extent, and in which context which data of this entity is disclosed to whom.*

This definition takes into account the communication partner, the context, in which the communication takes place, and the negotiation processes, which are needed to flexibly manage privacy. This is necessary to reason which personal information an individual is willing to disclose to get which kind of service and to solve possible conflicts, which might arise due to the contradiction of privacy goals of different individuals. The concept of *multilateral security* [1, 15] provides for a flexible and effective way of negotiating such conflicts in a privacy-respecting environment. Moreover, which personal data is disclosed, its granularity, and the enforcement of an individual's privacy requirements are also considered in the definition above.

MAKING PRIVACY INHERENTLY BUILT INTO THE UBICOMP SYSTEM'S FUNCTIONALITY

In order to provide for a privacy-respecting and secure UbiComp system, the process of ensuring privacy and security should begin already at the system design stage, the concept of which is known as ”privacy by design”, and it should continue throughout all the other steps of system development.

It is clearly impossible to predict the security and privacy requirements of all potential users and also their variations in response to future context changes during the system design stage. In order to provide for flexibility and extensibility, a concept of special extension/variation points (so-called hooks) for unforeseeable extensions/variations of privacy and security requirements can be utilized.

Thus, the process of ”weaving” privacy and security mechanisms into the UbiComp system's functionality can be divided into the following steps, depicted in Figure 1:

1. During the system design stage, generic (i.e. foreseeable) privacy and security requirements are considered. In order to provide for flexibility in future, a concept of extension/variation hooks with respect to privacy and security requirements is used.
2. At initialization time, an instantiation of generic requirements considered during the first step is carried out. Also, the so-called *binding*[3] of extension/variation hooks is performed.
3. At run-time, the previously implemented privacy and security management mechanisms are used. In order to provide for *dynamic* adaptation (e.g in response to context changes), the concept of dynamic extension/variation hooks may be exploited.

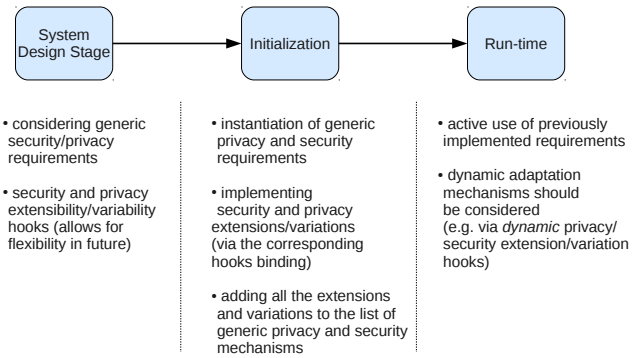

Figure 1. The process of making privacy requirements inherently built into the UbiComp system's functionality.

PRIVACY IN UBICOMP: PECULIARITIES

In order to provide for effective privacy management in UbiComp systems, it is sensible to explore which peculiarities privacy issues have in this domain.

The pervasive nature of UbiComp may impose certain constraints on the users of the system in that it might be hard for them to actually refuse to use it. This problem raises privacy concerns and is called ”the disability to opt-out” [7], where the following example was stated: it would be extremely difficult if not impossible to refuse to use the Ubiquitous RFID system in case ”such devices [RFID tags] get affixed to bank notes, ID cards, and every item that one can buy in a store”. If opt-out is nevertheless made possible, the following problems might arise:

- much inconvenience caused by opt-out (e.g. postal mail of a check instead of a credit card payment);
- opt-out can look suspicious (a denial to give away certain data in particular situations may look suspicious, e.g. switching the location sensor off during the time when a crime was committed, etc.)

Another privacy problem specific to UbiComp is a constantly rising likelihood that intimate conversations might become

[2]In each situation an individual is constantly performing reasoning about what he/she is willing to disclose to get which kind of service.

[3]The term is adopted from programing. It basically means that the corresponding hooks are being directly used, i.e. extension/variation has taken place via the hook.

publicly available. The authors of [7] call this problem ”the loss of ephemeral communication”. Similarly, Schneider states: ”The moral is clear: If you type it and send it, prepare to explain it in public later”. In this case, the problem of *violation of contextual integrity* arises. It was described in [4] as ”falsifying the context in which information has been communicated” by ”putting it into a wrong context”. For instance, consider an example of a debating club: a person receives a topic ”Should foreigners be allowed to work in Germany?” and should state arguments against it. If his speech is put into another context later on (e.g. shown at the TV) *without specifying the original context*, the speaker’s reputation might be dramatically spoiled (i.e. the ”decontextualization of communicated information” has turned ”innocuous” information into the ”mortifying” one [4]).

In [14], it was outlined that the privacy of an individual in UbiComp could be enhanced by changing the main direction of information flow to ”infrastructure → user” and applying filtering in order to avoid overload or annoyance of the user. This change of information flow ”enables a quantum leap in privacy by avoiding the possibility to gather huge amounts of personal data”. In this case, the infrastructure might also broadcast security and privacy advices (e.g. possible options, etc.) to the user if it appears to be of mutual interest to both, the provider(s) of the infrastructure and the users.

Thus, in order to provide for a privacy-respecting UbiComp system, the following issues have to be taken into account:

1. Provide for support of opt-in/opt-out according to the individual’s choice. At the same time, mechanisms against irresponsible behavior should be taken into account (i.e. non-repudiation of performed actions)[4].
2. Anonymization and encryption techniques for resource-constrained devices should be carefully considered in order to mitigate the problem of disclosure of the content of intimate conversations to public.
3. Mechanisms for protecting contextual integrity of data should be provided (especially in case of voice/video recording services, personal communication services, etc.) For example, attaching a special protected tag to data, which will specify the original context and protect the information from decontextualization, should be considered. The tag itself can be authenticated by the individual who owns the information or by the group of individuals to whom the data is relevant (using multi-party authentication, for instance).
4. It is also highly advisable to design a UbiComp system adhering to the concept of reverse information flow (”infrastructure → user”) where possible.

[4]Consider an example of an ”Ambient Coffee Machine” service in the organization, where users are able to drink coffee without being obliged to pay for it at the spot but required to do so at the end of the month. An irresponsible user might want to be using such a service for several weeks and then decide to opt-out ”due to privacy reasons” without paying. In this case, authentication and legal enforcement, for instance, can be used to prevent such case from happening.

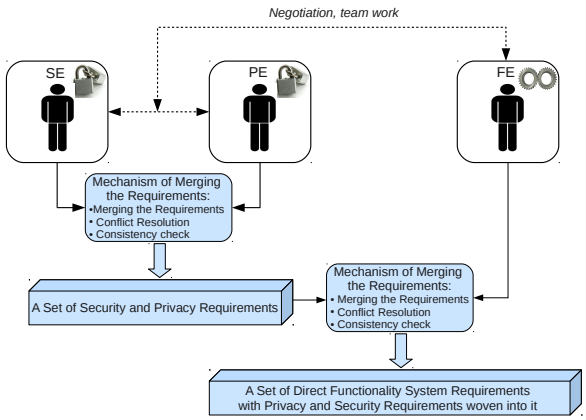

Figure 2. A process of joint development of privacy and security requirements for a UbiComp system.
SE = Security Engineer.
PE = Privacy Engineer.
FE = Direct Functionality System Engineer.

DESIGNING PRIVACY AND SECURITY REQUIREMENTS IN A JOINT FASHION

Privacy and security are closely connected to each other. Important is to understand that neither of them is a byproduct of the other one. Only if having considered both, privacy and security, can the developed UbiComp system be regarded as privacy-respecting and secure.

For this reason, we suggest that privacy and security requirements are elaborated in a joint fashion by two cooperative entities: the Privacy Engineer (PE) and the Security Engineer (SE) (see Figure 2). These entities are responsible for the whole design process of privacy and security policies respectively as well as for administrating and managing privacy and security in the deployed system. The process of designing policies for a privacy-respecting and secure UbiComp system should be carried out in the presence of collaboration between the PE and the SE. Further negotiation with the Functionality Engineer (FE), who is responsible for the design of the direct functionality of the system, should be considered as well. The reason for this is that it is expected that the requirements elaborated by the PE and the SE along with the ones of the FE may not be free of conflicts. That is why conflict resolution mechanisms should be considered during the process of merging the requirements. In order to ensure that the requirements are consolidated in a consistent way (i.e. specific requirements of each area after the merging conform to the ones before the merging), consistency checks should also be performed after the merging.

PRIVACY MODELING

In order to provide for privacy requirements, which are going to enable for efficient privacy management in the deployed system, we suggest that a corresponding model of privacy for the target domain of UbiComp is created. The respective requirements can be inferred from the model later on. Here, with the term ”abstract privacy model” we refer

to a high-level model, which takes into account social, legal, and functional issues and enables the developer to perform a combination of privacy issues from different fields in an interdisciplinary manner. Having an abstract privacy model in the first step will facilitate the process of taking various and often illusive privacy issues and considerations of the UbiComp area into account and make the approximation to the real world scenario more accurate.

Modeling privacy is not a trivial task. Existing privacy models are often abstract and difficult to transform into a set of system requirements. For instance, the model introduced in [12] deals with the concept of ”crossing personal borders”, i.e. privacy violation occurs when ”personal borders” of an individual are crossed. The author provides for a classification of privacy-violation scenarios, analyzes the privacy concerns of the individuals and also considers the impact of technological advance on privacy. However, the model is described in a loose and nontechnical way, which might impede its adoption for the process of inference of privacy requirements.

Another model was introduced in [17], which focuses on the activities that invade privacy: information collection, information processing, information dissemination, and invasion. The model consists of the data subject (the individual) and the data holders (who collect, process and disseminate private information). Similarly to the above mentioned model, it provides for a rather notional description of privacy issues and does not specify how the respective requirements can be inferred and further implemented.

Moreover, new approaches to modeling of privacy should also be considered because of the rapid evolution of technology. For instance, Shapiro in [16] gives an example of Fair Information Practices that have been commonly used for understanding informational privacy. However, he claims that ”As more things become digitized, informational privacy increasingly covers areas for which Fair Information Practices were never envisioned” (e.g. genetics, biometrics, etc.).

UbiComp definitely introduces a serious challenge regarding privacy modeling, translating a model into a set of system requirements and implementing it. It is of little help just having a good model of privacy if it can not be adopted into technical schemes of privacy regulation and thus be used within a UbiComp system. Provided that a decent and *implementable* model of privacy is available, respective privacy mechanisms should be woven into the UbiComp system functionality at the system design stage to allow for designing *inherently* privacy-respecting systems.

Therefore, it would be helpful to consider a framework which will enable for consistent transformation of an abstract privacy model into functional requirements of a UbiComp system, which in turn can be implemented.

Privacy Modeling Framework

The concept of the Privacy Modeling Framework is similar to the meta-modeling approach used in programming (e.g. meta-metamodel $\rightarrow$ metamodel $\rightarrow$ model, see [2] for more details). The task of providing for a consistent privacy model and transforming it into a set of implementable system requirements is within the competence of the Privacy Engineer (cf. Figure 2).

This approach implies several steps, which are depicted in Figure 3.

1. The Privacy Engineer entity (that might be a group of privacy experts in practice) creates an abstract privacy model. This implies the following steps:
   - investigating the privacy area of the future UbiComp system deployment, i.e. determining individuals' privacy concerns, possible privacy threats, taking into account various cultural differences in perception of privacy, etc.;
   - reviewing the current status of legal basis in the area of interest (i.e. finding out which privacy-related laws apply to the future UbiComp system deployment, how the situation is legally regulated and determining the weak sides of it);
   - creating the joint picture of privacy-related issues in the field;
   - on the aforementioned basis, an abstract privacy model is created (system- and platform-independent).
2. Next, a consistent transformation of the abstract privacy model created during the first step into a set of system-specific requirements is carried out. If some of the model preferences can not be transformed, a possible refinement of the abstract model should be considered. The result of the second step is a set of *implementable* system requirements.
3. The last step is the actual implementation (”weaving” of privacy mechanisms into the UbiComp system's functionality).

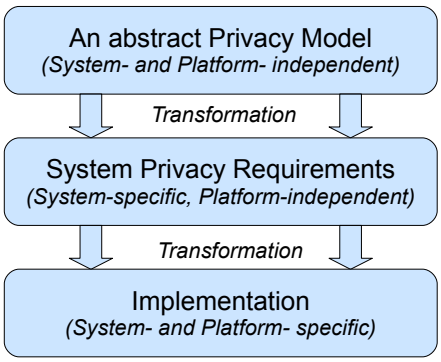

Figure 3. A general structure of a framework for transforming abstract privacy models into implementable requirements.

The above mentioned approach introduces a set of challenges:

1. Merging the individual privacy requirements with legal issues in the area of interest (step one) is a difficult task.

The reason for this is that the former is elusive and not easy to specify. The latter is well specified but coarse-grained and hence inflexible. For example, suppose it is written in the privacy law that location of the individual is private information and any exposure of this information to a third party is subject to law violation. The situation when the individual is *willing* for his location to be known to some of his friends at certain times, is not considered, however. Moreover, the legal part strongly depends on the region, which raises the question of international interoperability and aggravates the outsourcing problem, i.e. the privacy-sensitive data that is governed by law in one country, might be under threat of violation in the other one. This happens due to the absence of a unified international law protection system of privacy-sensitive data.

Having managed to specify privacy requirements of the individual and taken the legal prospective into account, the consistency of the *joint* abstract model should be considered.

2. The second step (transformation of abstract privacy model into a set of requirements) implies the existence (or creation) of a mature language that will enable to express the abstract model in a standardized, ready-to-implement format. To the best of our knowledge, only a few efforts have been made in this direction by now. The authors of [9] described their privacy model using a privacy control language that "includes user consent, obligations, and distributed authorization". In [10], a privacy-specific access control language was used to manage privacy in the environment of so-called "Platform for Enterprise Privacy Practices (E-P3P)", which defines technology for privacy-enabled management and exchange of customer data. The authors in [6] showed how a privacy policy can "be specified and implemented according to the Generalized Framework for Access Control (GFAC)-approach". In order to successfully complete the second step, it should be decided by which means the abstract model should be specified in the most comprehensive and consistent way (e.g. which language to choose or even to introduce a new one).

3. Along with privacy-specific questions, general framework-related issues arise:

   - The framework is described in an abstract way. That is why the ways of its implementation should be outlined. Moreover, it should also be considered, which degree of *automation* of the transformation process can be achieved.
   - Next, the *consistency* of the performed transformation should be carefully considered. Surely, certain trade-offs are going to arise. Their impact on the accuracy of the implemented privacy model should be assessed.

CONCLUSION AND FUTURE WORK

The paper has presented an approach to designing an *inherently* privacy-respecting UbiComp system. We claim that it is not possible to provide for a full-fledged support of privacy management, having considered this issue after designing the direct functionality of a UbiComp system, i.e. building privacy on top of the system. That is why the process of ensuring privacy and security has to begin at the *system design stage* and it should continue throughout all the other steps of system development.

Thus, an approach to making privacy inherently built into the UbiComp system's functionality was considered. In order to provide for *dynamic privacy management* (e.g. to enable the consideration of unforeseeable extensions towards privacy requirements), a concept of special extension/variation points can be utilized while designing a system.

Providing for *efficient privacy management* requires the exploration of the peculiarities of privacy in the target domain. Also, respective recommendations for developing appropriate privacy policies should be formulated. Having considered this issue, we presented our concept of designing privacy and security requirements in a joint fashion. The reason for this is that privacy and security are closely connected and mutually affect each other. According to this concept, privacy and security requirements should be considered by two cooperative entities: the Privacy Engineer (PE) and the Security Engineer (SE). Moreover, further negotiation with the designer of the direct functionality of the system (Functionality Engineer) is considered along with conflict resolution mechanisms.

The creation of an abstract privacy model was suggested to enable *effective development of privacy requirements*, which take various privacy issues and considerations of the UbiComp area into account and provide for a better approximation to the real world scenario. Respective privacy requirements can be further inferred from the model. This can be done within our Privacy Modeling Framework, which considers the creation of an abstract, domain-specific privacy model by the PE entity, further inferring respective requirements from it, and, lastly, implementing them into the UbiComp system's functionality.

Having described our conceptual view on ensuring privacy in a UbiComp system, more concrete ways of creating an abstract privacy model, means of specifying the requirements and necessary recommendations towards their implementation are to be elaborated. Finally, applying the concept to a particular real use case scenario is to be realized.

ACKNOWLEDGEMENT

The authors would like to express their gratitude to a great researcher and friend Andreas Pfitzmann who passed away in September 2010. He was not only a highly qualified professional but also a very kind and responsive person who inspired the people around him on their way to scientific excellence.

This paper is to a large extent influenced by discussions with Andreas and is written in commemoration of him.

REFERENCES

1. Andreas Pfitzmann. *Multilateral Security in Communications*. Addison-Wesley-Longman, 1999, ch. Technologies for Multilateral Security, 85–91.
2. Assmann, U., Zschaler, S., and Wagner, G. Ontologies, Meta-models, and the Model-Driven Paradigm. *Ontologies for Software Engineering and Software Technology* (2006), 249–273.
3. Berg, M., and Borcea-Pfitzmann, K. Implementability of the Identity Management Part in Pfitzmann/Hansen's Terminology for a Complex Digital World. In *Proceedings of PrimeLife / IFIP Summerschool on Privacy and Identity Management for Life*, S. Fischer-Hübner, M. Hansen, P. Duquenoy, and R. Leenes, Eds., IFIP Advances in Information and Communication Technology, Springer (2011).
4. Borcea-Pfitzmann, K., Pfitzmann, A., and Berg, M. Privacy 3.0 : = Data Minimization + User Control + Contextual Integrity (Privatheit 3.0 : = Datenminimierung + Nutzerkontrolle + Kontextuelle Integrität). *IT - Information Technology 53*, 1 (2011), 34–40.
5. Dou, E. EU proposes online right 'to be forgotten', Nov. 2010. Accessed online on 05.04.2011. Reuters. http://www.reuters.com/article/2011/03/17/us-eu-internet-privacy-idUSTRE72G48Z20110317.
6. Fischer-Hübner, S., and Ott, A. From a formal privacy model to its implementation. In *National Infromation Systems Security Conference* (Oct. 1998).
7. Henrici, D. *RFID Security and Privacy*. Springer, 2008.
8. Herold, R. SmartGrid Privacy Concerns, Sept. 2009. Accessed online on 03.04.2011. http://www.privacyguidance.com/files/SmartGridPrivacyConcernsTableHeroldSept_2009.pdf.
9. Karjoth, G., and Schunter, M. A privacy policy model for enterprises. In *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE* (2002), 271–281.
10. Karjoth, G., Schunter, M., and Waidner, M. Platform for enterprise privacy practices: Privacy-enabled management of customer data. Springer (2002), 69–84.
11. Langheinrich, M. Privacy by Design – Principles of Privacy-Aware Ubiquitous Systems. In *Ubicomp 2001: Ubiquitous Computing*, G. Abowd, B. Brumitt, and S. Shafer, Eds., vol. 2201 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2001, 273–291.
12. Marx, G. T. Murky conceptual waters: The public and the private. *Ethics and Inf. Technol. 3* (Sept. 2001), 157–169.
13. Pentland, W. Why Smart People Are Suspicious of Smart Meters, Dec. 2010. Accessed online on 01.04.2011. Forbes. http://blogs.forbes.com/williampentland/2010/12/10/why-smart-people-are-suspicious-of-smart-meters.
14. Pfitzmann, A. Accompanying Ambient Intelligence (AAmI) – Why You should take your Sensors with You. A Sketch on the Future of privacy-aware, secure Ambient Intelligence., Apr. 2010.
15. Rannenberg, K. Multilateral security: A concept and examples for balanced security, 2000.
16. Shapiro, S. S. Privacy by design: moving from art to practice. *Commun. ACM 53* (June 2009), 27–29.
17. Solove, D. J. A taxonomy of privacy. *University of Pennsylvania Law Review 154*, 3 (Jan. 2006), 477 pp. GWU Law School Public Law Research Paper No. 129.
18. Stajano, F. *Security for Ubiquitous Computing*. John Wiley & Sons, LTD, 2002.
19. Warman, M. EU proposes online right 'to be forgotten', Nov. 2010. Accessed online on 05.04.2011. The Telegraph. http://www.telegraph.co.uk/technology/internet/8112702/EU-proposes-online-right-to-be-forgotten.html.
20. Warren, S. D., and Brandeis, L. D. The right to privacy. *Harward Law Review 4*, 5 (Dec. 1890), 193–220.
21. Weiser, M. The Computer for the 21st Century. *Scientific American* (Feb. 1991).

Ubiquitous Alignment

Florian Haag, Michael Raschke
Visualization and Interactive Systems Institute (VIS)
University of Stuttgart
Universitätsstraße 38
70569 Stuttgart, Germany
{haag, raschke}@vis.uni-stuttgart.de

Thomas Schlegel
Institute of Software and Multimedia Technology
Technische Universität Dresden
Nöthnitzer Straße 46
01062 Dresden, Germany
Thomas.Schlegel@tu-dresden.de

ABSTRACT
Ubiquitous thinking means designing human-computer interaction in a fundamentally new way. The perceived distance between ubiquitous systems and their users decreases while the heterogeneity of modalities increases compared to classical human-computer interaction. The initiation of work phases becomes less formalized. Instead of explicitly declaring the start of an interaction by activating a computing device, the interaction starts gradually and sometimes implicitly based on an estimation of the user's needs. For bridging the gap of initiation, in this article we present the ubiquitous interaction concept "Ubiquitous Alignment". It comprises of the three steps recognition, sparking interest and start of collaboration. The Ubiquitous Alignment concept is based on a comparison between traditional human-computer and human-human interaction. Finally, two examples show the applicability of the Ubiquitous Alignment concept.

Keywords
Ubiquitous systems, interaction concepts, mixed initiative, interaction initiation, Ubiquitous Alignment

INTRODUCTION
Traditionally, the boundaries of human-computer interaction are clearly defined. The interface to the computer is defined by peripheral input and output devices and does not extend beyond them. Work with a computing device starts after it has been switched on by the user. When the work is done, the user switches the device off again. This applies to desktop computer systems just as much as any other technical device. From a more semantic point of view, the interaction starts when the user turns toward the terminal or concentrates on it in any way, and it stops when the user leaves the workplace or concentrates on something else.

The next step in the development of computing systems is ubiquitous computing – an environment where computing devices are, often seamlessly, integrated into everyday objects and activities in such a way that users do not need to be aware of them in order to interact [23]. There are two important aspects of how to approach that objective.

One is the integration of computational capabilities in objects that are usually used for non-computing purposes. This can refer to both fixed and portable objects. In the case of fixed objects, the computing equipment can, for example, be built into buildings or parts thereof. Whole buildings may be equipped with linked computers and sensors for specific goals, such as minimizing energy consumption [19], or for general-purpose support in a variety of tasks performed by the people within the building [11]. Likewise, parts of buildings, such as the floor [14] or doorplates [20], can be enhanced with ubiquitous computing technology. Computing and sensor devices in portable objects can refer to so-called smart furniture [12] or wearable computing [18], amongst others. They can be used for similar tasks or even linked with devices embedded in fixed objects using wireless networks.

The other important aspect to consider in ubiquitous interfaces is how the interaction begins. The necessity to use specific computing devices should be avoided, as happened by integrating computer equipment into everyday objects. Still, computer-specific tasks such as activating a device or looking at (after possibly walking to) the display to gather some information pose an obstacle for a natural interaction with the systems [13]. Particularly, the devices should act proactively in certain situations while still appearing unobtrusive. For this purpose, we introduce a concept of how interaction between a human user and a system in a ubiquitous computing environment can be initiated. This refers to both the first contact with the ubiquitous technology as well as later, single interaction sessions. The goal of this concept is a description of how the system gradually approaches the user and gets his or her attention. System and user align themselves to each other in order to communicate and collaborate without any obstacles. Therefore, we refer to this concept as Ubiquitous Alignment.

After discussing related work, we will first analyze how interaction between humans and other humans (human-human interaction) as well as between humans and computers (human-computer interaction) usually starts, then highlight differences between the two situations. Subsequently, we will describe our concept of Ubiquitous Alignment, explain where it differs from human-computer

interaction and identify parallels with human-human interaction. Finally, we will present two example scenarios in which the concept can be applied.

RELATED WORK

There has been a large amount of work to provide computing devices with additional sensors to perceive an arbitrary range of signals from the outside world. To name only a few, sensor systems to detect human means of expression such as pointing at something with the hand [8] or showing different facial expressions [24] are being developed. Sensors for other contextual parameters, such as the room temperature [17], are also being integrated into computer systems. In order to further improve the evaluation of data received from sensors, some research tries to recognize or model human emotions, which may give systems a better understanding of the intentions of users [4] [16]. On a wider scale, Pantic and Rothkrantz present their ideas of how a great number of sensors can enable multiple modalities of input [15]. Similarly, the concept of Perceptual User Interfaces aims for a natural interaction between users and devices [21]. The EasyLiving project focuses on the technical side on coupling a variety of sensors and other devices to form a complete system [2].

In a general notion of ubiquitous computing, Rhodes points out some design objectives for wearable computing in his article [18] that are also useful in other types of ubiquitous systems. Works such as the classroom-related scenarios described by Bravo et al. assume that the system is already there and do not take into account a phase during which users get acquainted and used to it [1].

The behavior of user interfaces that sometimes act proactively and sometimes leave the control to the user is called mixed initiative. There has been much research on this topic over the course of the past few decades. It focuses on ways to achieve and employ mixed initiative [10] [22] [9]. With its close resemblance to human-human interaction, mixed initiative systems sometimes aim at generating a verbal dialogue between user and system which works the same way as a conversation between people [6]. Chu-Carroll and Brown distinguish dialogue and task initiative, which allows for a more accurate dialogue model as it distinguishes which interaction partner is guiding the current interaction and which one is deciding what will be done [3].

TRADITIONAL INTERACTION STYLES

This section describes two examples of starting an interaction between humans and other humans and between humans and today's computers, respectively. Both examples are chosen in a way that the participants of the interaction do not have any prior knowledge about each other or are not yet collaborating. Thus, the situations are analogous to the interaction between a user and ubiquitous devices as described further below.

Human-Human Interaction

As an example of human-human interaction, we have chosen a customer in a self-service store and a sales assistant. This scenario was selected because it closely resembles a situation where ubiquitous technology might come into play. Basically, the customer could manage well without any additional help. Generally, for a comfortable shopping experience, the sales assistant should not behave in a pushy way by insisting on helping the customer against his wish. The assistant may however indicate that she is available and ready to help if help is required, and the customer may decide on his own to start a more thorough interaction.

Initially, the customer is examining the items in a shelf, reading the information given on the labels and the price tags. The sales assistant is waiting nearby. In order to not appear obtrusive, she should not wait right next to the customer or in front of the shelf, as this might make the customer feel controlled. Still, it is important not to express disinterest or lack of attention. This can be achieved by displaying an initial sign of responsiveness, such as greeting the customer when he enters the store, or by explicitly offering help when the customer has been browsing the products for a while.

At the least, when the customer has picked up some items and placed them in his shopping cart, the sales assistant may carefully indicate that she is willing to start a conversation. This can be expressed by a single casual remark about one of the products or by pointing out an alternative. At this point, it is important to note that the information given is not among that which the customer has likely already seen. Instead, he may be pointed to a feature that is not evident from the labels or to an item that is not currently located on the same shelf. In this way, the customer will perceive the assistant as helpful rather than merely reiterating known facts.

If the customer desires to receive more information afterward, he will ask the assistant. The assistant can inform the customer about what information she is able to provide, while the customer gets a feeling of how reliable the information received from that assistant is.

Human-Computer Interaction

Due to the lack of proactivity in most of today's software, the gradual start of an interaction as seen in the example of human-human interaction cannot be customarily found in human-computer interaction. Assuming that the computer is already switched on and the user has logged into her account, she starts for the first time the new application she would like to use.

The application displays a default set of options and commands. Guides to the most important features can be provided. An example location would be a welcome screen. Nevertheless, the user has to start exploring the interface right away. After taking a few steps in the user interface, the application tries to estimate what the user is trying to do and displays hints accordingly. The application can only evaluate the user input and does not possess any additional sensors. Hence, it cannot take into consideration any contextual information about the user and her environment. The estimation of the user's intentions is accordingly imprecise; therefore, the displayed hints occasionally fail to

be of any help, which in turn makes the user dissatisfied with the application.

Once the user has gathered some experience with the application, she will actively customize the user interface and create templates and macros for repeating tasks. Due to bad experience with the automatic input analysis, she might eventually choose to completely disable the automatic adaptation of application behavior. Even though this means some additional effort for the user in that some settings have to be done manually, she values the absence of distracting hints that do not provide any helpful information higher than saving some time by allowing the software to automatically adapt itself.

Comparison

Despite being basically equivalent scenarios of starting an interaction with a previously unknown partner, these two descriptions of human-human and human-computer interaction sport substantial differences. First of all, in human-computer interaction the user has to know and launch the application she wants to use. The application is not just there and ready by default, as it is the case with the sales assistant. By launching that application, the computer user also explicitly declares the start of the interaction, as opposed to the gradual process found in the interaction between the customer and the sales assistant.

As mentioned above, the lack of variety of input channels available to the system results in a lack of knowledge about the overall behavior and context of the user. Thus, any estimation about the current intentions of the user can at most be a rough guess. Accordingly, helpful clues can only be given based on the experience with average users or by trying to find repeating patterns in the behavior of the current user. The same applies to input interfaces such as menus: Even though some software manufacturers have attempted to automatically restructure menus, a user study suggests that any such change is likely to confuse the user rather than support her [5]. That lacking additional information about the user is, however, available to the assistant concerning her customer, as she can see and consider where the customer is located and what he is doing. Thus, she is also capable of quite reliably assessing the customer's current intentions and wishes. This enables her to take over or give away the initiative in the interaction process at the right time. The computer application is not able to provide this degree of mixed initiative in the described scenario.

UBIQUITOUS ALIGNMENT

In order to make human-computer interaction more like human-human interaction, one can take advantage of the special capabilities of ubiquitous computing technology. The additional data gathered by the sensors in a ubiquitous computing environment allows for a more natural initiation of collaboration between users and systems [2].

The Ubiquitous Alignment concept assumes that a user is going to perform a particular task in an environment equipped with ubiquitous computing devices. The user does not yet have sufficient knowledge about those devices to explicitly trigger any operations. He may or may not be aware that his environment is equipped with ubiquitous computing systems at all. In order to achieve collaboration, the three steps recognition, sparking interest and start of collaboration are performed (cf. Fig. 1).

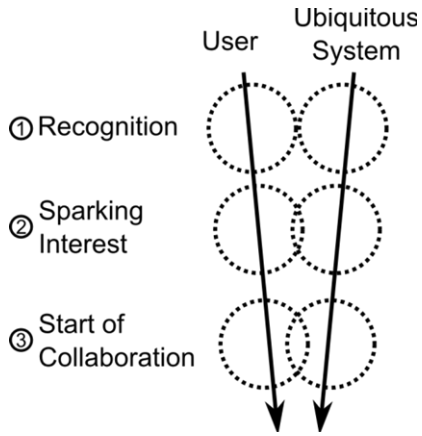

Fig. 1. User and ubiquitous system gradually intensify their interaction in three steps of the Ubiquitous Alignment concept: At first, they barely know of each other, then they start to interact and intensify that interaction further on.

Step 1: Recognition

The user recognizes the system in a pleasant rather than a pushy way. A pleasant ubiquitous system remains in the background until the user wishes to start an interaction. In this phase, the system is still largely ignorant of what the user intends to do. This matches the behavior of the assistant from the human-human interaction example, who remains passive. Any other behavior might annoy or scare away the other person or the user, respectively. The ubiquitous system visibly exhibits a certain level of proactivity only when it is absolutely certain that an intervention is desired by the user. Otherwise it remains largely invisible, except for some unobtrusive hints that it is there and ready.

Any other operations the ubiquitous system performs go unnoticed by the user, who gets the impression that he is just using everyday objects. Automatic locks that secure lids of boxes unless the user actually attempts to open them, the adaptation of fridge power to cool down newly inserted warm items or the temporary dimming of lights while the user is not in the room do not require any active input. That is also why no new interface concepts need to be considered at this point. The ubiquitous devices do not have any new input controls. They can be used just like their non-ubiquitous counterparts.

Step 2: Sparking Interest

User and system begin to communicate with each other. As the user finds out what the system is or is not capable of, the system output at this point must particularly strive for a high reliability. This concerns both the information provided and the estimations made. This step corresponds with the customer becoming acquainted with the sales assistant and vice-versa.

In order to not appear overzealous at communicating with the user where no communication is desired, a good strategy is to continue giving small hints of the presence and features of the system, just as the sales assistant will try to be supportive without flooding the customer with information. In particular, those hints should spark the interest of the user/customer and motivate him or her to find out what kind of support can be obtained. At the same time, the ubiquitous devices may be able to catch some clues as to how the user behaves or reacts and what kind of output inspires him to further interact with the system, just as the sales assistant will adapt his behavior to suit the customer's preferences to a certain degree.

Step 3: Begin of Collaboration

After the computer system has been recognized by the user and he has indicated that he is willing to collaborate with the system, the system can become more active. As the user has become interested in the system, he is likely to try and explore further capabilities of the ubiquitous devices. This behavior can be encouraged by facilitating the exploration process. Amongst others, options related to the current operation that have not yet been employed by the user can be recommended to him. Likewise, any means of discovering and learning about unknown system features must be easy to find. A human sales assistant will too, in a comparable situation, express what information he is able to provide to the customer.

While interacting with the user, a system that follows the Ubiquitous Alignment approach has gathered and is still gathering an increasing amount of information about the user and his behavior. This allows for better estimates of the current intentions of the user and thus provides the system with the means to support the user in an optimized way.

How Ubiquitous Alignment helps improve HCI

Ubiquitous Alignment reflects the gradual process used to establish contact between two humans with the goal of collaboration. These parallels hold true both for situations where the actors do not have any prior knowledge about each other as well as for cases where they do. In the former case, the steps explained serve for the initial contact between two strangers, just as for the initial contact between a future user and a network of ubiquitous devices. In the latter case, the participating persons already know each other, so the objective is collaboration on a given task. The actors do not yet know whether the collaboration will actually turn out to be beneficial, which is why they use the same approach of gradually initiating their interaction. Likewise, a user might already know some parts of a ubiquitous system, but he is not sure yet whether the system is helpful with a new kind of task. At the same time, the system should not behave in a paternalistic way and insist on collaboration in this particular new task just because the user makes frequent use of the system on other occasions.

To sum up, the main advantages of ubiquitous computing systems over traditional computing systems at approximating human-human interaction are their greater variety of input channels and their integration into everyday objects. The additional input channels in the form of a variety of sensors allow for a more accurate and complete perception and evaluation of the user, his behavior and his context. By integrating system parts into appliances previously known to the user, the handling of the ubiquitous system does not have to be learned right from the start on; instead, some features can be used by manipulating appliances the usual way, so the prospective user can gradually extend his knowledge to encompass the additional system features that require any special input.

POSSIBLE APPLICATIONS OF THE UBIQUITOUS ALIGNMENT CONCEPT

To underline that the Ubiquitous Alignment concept can indeed be used in ubiquitous computing scenarios, we describe two example scenarios in which our Ubiquitous Alignment concept is applied.

One example of a ubiquitous system that uses mobile computing devices is the ActiveClass system described by Griswold et al. [7]. ActiveClass is a system which allows students' mobile devices to connect to a central component while in a lecture hall. Using the ActiveClass system, students can publicly and anonymously ask questions. Without the ActiveClass system, both the size of the lecture hall and the lack of anonymity may pose obstacles to actually ask questions. When applying the Ubiquitous Alignment approach to a situation where a student does not yet know the ActiveClass system, the first step might present unobtrusive hints about the system. For example, the student's mobile device might display an access icon of the ActiveClass system in the main menu while the student is attending a lecture. In the second step, ActiveClass might display a button for posting a question whenever the student starts searching for an explanation about something which is being discussed by the lecturer right then. In the third step of Ubiquitous Alignment, which starts once the student has begun to actively use ActiveClass, the system provides access to its options menu. There, the other features such as polls, class feedback and votes can be found.

In the second example, we consider a table that is aware of what objects are placed on the tabletop (cf. Fig. 2). This awareness can be achieved through a variety of means, such as load detection, image analysis or tracking of object locations (assuming that each object is tagged in some way), or a combination thereof. In addition, some means of tracking what the user is doing is available. The knowledge about object positions can be used to guide a user by indicating where on the table to find a particular item. This can be used for workbenches or interactive cookbooks, to

name only two examples. In the first step of the Ubiquitous Alignment approach, the user may be using the table just as a table, placing objects on top of it. The system performs its minimum default function, inserting positional indicators such as “on the left” or “next to ...” in the instructions for the user. Only when the user keeps searching for something for a longer time, does the system clarify its output, providing more information in an additional message. If the user responds by locating objects faster based on those hints, step two of the Ubiquitous Alignment concept has the system highlight any references to objects in the displayed instructions (or, in the case of voice output, make clear what is highlighted in text in some other way), pointing the user to the possibility of finding out more about the respective items. Eventually, in the third step the system may display additional information right away as the user requires it, and offer some options to modify how much and what kinds of information the user wants to be displayed about objects referred to in the work instructions.

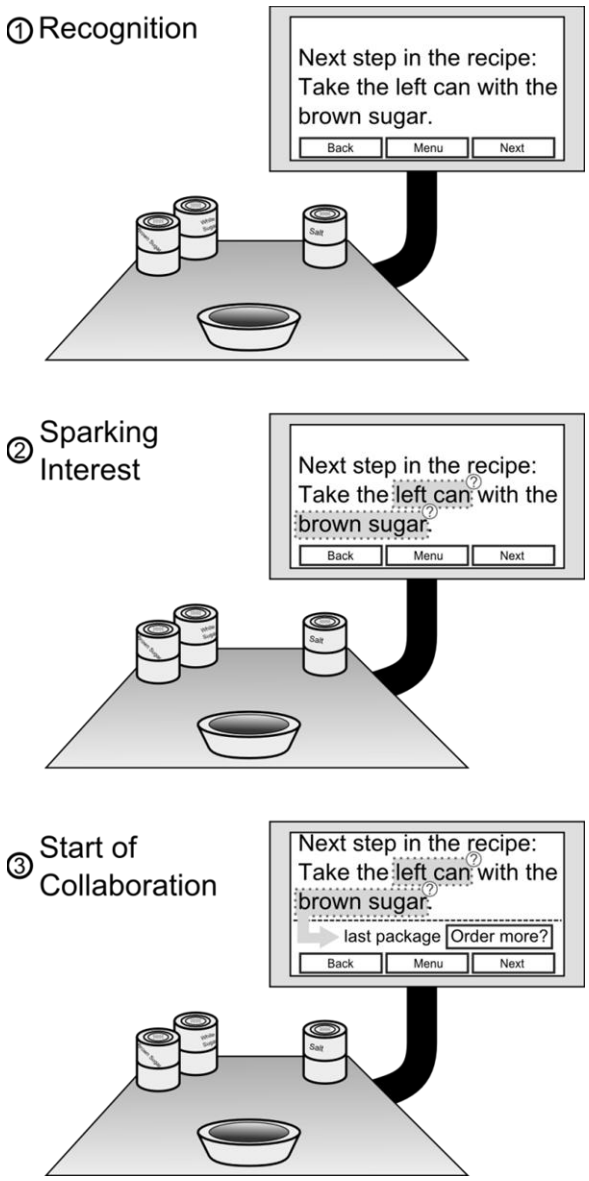

Fig. 2. Ubiquitous devices embedded into a kitchen allow for displaying the current instruction from a recipe in relation to the current state and location of ingredients on the table where the food is being prepared. Depending on the Ubiquitous Alignment phase in which the interaction is taking place, additional information is displayed: hints about the position in step 1, an explicit offer to provide more information in step 2, and further suggestions in step 3.

These examples show how the concept of Ubiquitous Alignment can be applied to scenarios where a user starts getting to know a ubiquitous system or one of its features. In all described scenarios and examples, the user had had a certain resistance to using the system, or at least he or she was not assumed to spend a lot of initial effort to learn how to use the system. This is where Ubiquitous Alignment is particularly beneficial. Users who take the time to read a manual first do not require the same degree of gradual initiation of interaction. Nonetheless, striving for a display of reliability towards that kind of users and not annoying them with frequent messages or other possibly undesired output retain their importance.

CONCLUSION AND FUTURE WORK

In this work, we have examined some exemplary situations of human-human and human-computer interaction. In an effort to make human-computer interaction more alike to human-human interaction, we have described the Ubiquitous Alignment concept. It defines how collaboration between a human user and a computer system can be initiated in a way that closely resembles the interaction between humans, taking advantage of the possibilities found in ubiquitous computing devices. As seen in the comparisons of the Ubiquitous Alignment approach with the previous examples of human-human and human-computer interaction, our approach has a strong resemblance to the former. The main reasons for the differences were found to be the additional sensor input and, similarly, the additional input modalities which can totally match the normal manipulation of everyday objects, as opposed to handling specialized devices such as mice or keyboards to provide input to traditional computers.

As this work presents a concept of how a ubiquitous system should behave, a future goal is the implementation of this concept. Thereby, we hope to show how the Ubiquitous Alignment concept works in practice and how it can be implemented in detail. Also, we expect this to be a starting point for defining processes for the development of ubiquitous software components and for gathering a better understanding of the user’s behavior. A model system will not need to incorporate all of the described attributes. With the incorporation of additional sensors, the system could gradually come closer to the ideal form of the Ubiquitous Alignment concept.

ACKNOWLEDGMENTS

This research was funded through the IP-KOM-ÖV project (German Ministry of Economy and Technology (BMWi) grant number 19P10003N). Also, we would like to thank our students Thomas Bach, Steffen Bold and David Kruzic.

REFERENCES

1 Bravo, J., Hervás, R., and Chavira, G. Ubiquitous Computing in the Classroom: An Approach through Identification Process. *j-jucs*, 11, 9 (2005), 1494-1504.

2 Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S. *Lecture Notes in Computer Science - EasyLiving: Technologies for Intelligent Environments*. Springer Berlin/Heidelberg, 2000.

3 Chu-Carroll, J. and Brown, M. K. An Evidential Model for Tracking Initiative in Collaborative Dialogue Interactions. *User Modeling and User-Adapted Interaction*, 8, 3 (1998), 215-254.

4 Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., and Taylor, J. G. Emotion recognition in human-computer interaction. *Signal Processing Magazine, IEEE*, 18, 1 (Jan 2001), 32-80.

5 Findlater, L. and McGrenere, J. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (Vienna, Austria 2004), ACM, 89-96.

6 Graesser, A. C., Chipman, P., Haynes, B. C., and Olney, A. AutoTutor: an intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 48, 4 (Nov 2005), 612-618.

7 Griswold, W. G., Shanahan, P., Brown, S. W., Boyer, R., Ratto, M., Shapiro, R. B., and Truong, T. M. ActiveCampus: experiments in community-oriented ubiquitous computing. *Computer*, 37, 10 (Oct 2004), 73-81.

8 Guan, Y. and Zheng, M. Real-time 3D pointing gesture recognition for natural HCI. In *7th World Congress on Intelligent Control and Automation* (2008), 2433 -2436.

9 Hearst, M. A., Allen, J. F., Guinn, C. I., and Horvitz, E. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications*, 14, 5 (Sept/Oct 1999), 14-23.

10 Horvitz, E. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit* (Pittsburgh, Pennsylvania, USA 1999), ACM, 159-166.

11 Intille, S. S. Designing a Home of the Future. *IEEE Pervasive Computing*, April-June (2002), 76-82.

12 Ito, M., Iwaya, A., Saito, M. et al. Smart Furniture: Improvising Ubiquitous Hot-Spot Environment. *International Conference on Distributed Computing Systems Workshops*, 0 (2003), 248.

13 Ley, D. Ubiquitous Computing. *Emerging Technologies for Learning*, 2 (2007).

14 Orr, R. J. and Abowd, G. D. The smart floor: a mechanism for natural user identification and tracking. In *CHI '00 extended abstracts on Human factors in computing systems* (The Hague, Netherlands 2000), ACM, 275-276.

15 Pantic, M. and Rothkrantz, L. J. M. Toward an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 91, 9 (Sept 2003), 1370-1390.

16 Picard, R. W. *Affective Computing*. 1997.

17 Ranganathan, A. and Campbell, R. H. A middleware for context-aware agents in ubiquitous computing environments. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware* (Rio de Janeiro, Brazil 2003), 143-161.

18 Rhodes, B. J. The wearable remembrance agent: A system for augmented memory. *Personal and Ubiquitous Computing*, 1, 4 (1997), 218-224.

19 Schor, L., Sommer, P., and Wattenhofer, R. Towards a zero-configuration wireless sensor network architecture for smart buildings. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings* (Berkeley, California, USA 2009), ACM, 31-36.

20 Trumler, W., Bagci, F., Petzold, J., and Ungerer, T. Smart doorplate. *Personal Ubiquitous Comput.*, 7, 3-4 (July 2003), 221-226.

21 Turk, M. and Robertson, G. Perceptual user interfaces (introduction). *Commun. ACM*, 43, 3 (March 2000), 32-34.

22 Walker, M. and Whittaker, S. Mixed initiative in dialogue: an investigation into discourse segmentation. In *Proceedings of the 28th annual meeting on Association for Computational Linguistics* (Pittsburgh, Pennsylvania, USA 1990), ACL, 70-78.

23 Weiser, M. Hot topics-ubiquitous computing. *Computer*, 26, 10 (Oct 1993), 71-72.

24 Zhang, Z., Lyons, M., Schuster, M., and Akamatsu, S. Comparison between geometry-based and Gabor-wavelets-based facial expression recognition using multi-layer perceptron. In *Third IEEE International Conference on Automatic Face and Gesture Recognition* (1998), 454 -459.

Generating consistent universal controllers for Web-Service-enabled appliances

Marius Feldmann, Thomas Springer, Alexander Schill
Technische Universität Dresden
Fakultät Informatik
Professur Rechnernetze
Germany
{marius.feldmann, thomas.springer, alexander.schill}@tu-dresden.de

ABSTRACT
Today, an increasing number of home and office appliances that we interact with contain embedded Web Servers. Making available Web Services for remote access to their functionality is just a small step. In this paper, we present a multi-platform, model-based generation approach enabling efficient and low-cost development of interactive applications for accessing Web-Service-enabled appliances. The results are not stand-alone, monolithic elements of software but are capable of being integrated into a consistent host application ad-hoc during runtime. Thus, a heterogeneous and dynamic infrastructure of appliances from different manufacturers can be accessed via single control applications. The approach has been proven to be applicable for generating interactive applications for various target platforms including mobile devices.

Keywords
Web Services, Model-based User Interface Generation, Dynamic Service Infrastructures

INTRODUCTION AND MOTIVATION
Web Services are software components offering their functionalities via a well-defined interface. This interface is described by a functional interface description language such as the Web Service Description Language (WSDL). Nowadays, Web Services are an accepted and widely used means for offering remote functionalities. Various tools are available for managing the whole lifecycle of Web Services starting from creating, via testing, to deploying and managing them. However, the field of developing user interfaces for Web Services has not been covered in a satisfactory manner yet. Though some approaches have been specified during the last years enabling the development of user interfaces for static Web Service infrastructures, no approach exists that makes it possible to extend an interactive application fully automatically depending on the associated set of Web Services the application interacts with. The necessity for such an interactive application becomes obvious in the case of the selected use case: Within an automated home, a set of appliances can be controlled via a universal control application available on different platforms. In the sketched scenario in Figure 1, three appliances (Light Control System, Music Control System and a Router) can be accessed via their offered Web Services by using either a universal control application for a mobile device or a Web-based universal control application. If a new device (DVB-T device) is added to the device infrastructure, a user interface has to be made available dynamically within the two universal control applications. Developing these user interfaces for various target platforms usually tends to constitute a time consuming and expensive task.

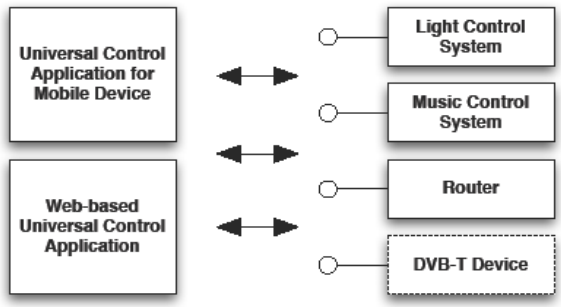

Figure 1. Sketch of the home appliance scenario

In order to solve these problems, this paper presents an approach to generate the user interfaces of the Web-Service-enabled appliances fully automatically from functional interface descriptions. Due to the fact that without further input, the generation result would be of low quality, so called Service annotations [8] are applied as platform independent modeling approach for defining user interfaces.

The results of the generation chain named **interactive components** can be embedded in the universal control applications dynamically during runtime. Without stopping this application, they appear in their navigation menu and can directly be used to remote control the appropriate device. Due to the possibility to parameterize the generation of interactive components by using layout and style descriptions, they can be visually adapted to different appearances of the embedding universal control applications. Thus, a consistent visual representation is achieved.

The remainder of this paper is structured as follows. In the second section an overview of the state of the art in the addressed domain is provided. The third section defines the concept of interactive components and points out their central characteristics. In the fourth section the assumed development approach is described. The fifth section

M. Feldmann, T. Springer, A. Schill: Generating consistent universal controllers for Web-Service-enabled appliances. Proc. of 1st International Workshop on Model-based Interactive Ubiquitous System 2011, Pisa, Italy, June 13, 2011, http://ceur-ws.org/Vol-787

introduces the steps for generating interactive components from the results of the development approach. The sixth section describes the technical realization of the mentioned appliance control scenario and an end user study applied using the prototypical implementation. The paper is concluded by a summary and outlook on future work.

STATE OF THE ART

Creating user interfaces for home appliances is a topic covered in the HCI community already for years (e.g. [3]). As well the specific goal to create consistent user interfaces for controlling different devices has been addressed [4]. However, these approaches do not focus in any way on Web-Service-enabled appliances. Thus, the possibility to generate user interfaces from functional interface descriptions is not discussed. The only approach exploring Web Services in the field of universal appliance control applications exploits a task-driven development approach to create consistent user interfaces [5]. Though the efficiency of this approach may be increased by using Service annotations [6], it still takes various manual steps to develop a user interface for different target platforms. Furthermore, the resulting user interfaces cannot be embedded automatically into a host application during runtime. Extending the focused research and development field to ad-hoc UI generation approaches for Web-Service-based interactive applications, we discovered a set of ten approaches. They usually take either a functional interface description or additionally Service annotations as input and create stand-alone markup-based UIs as output. An example for this category of approaches is described in [7]. The results have a low quality and do not address the generation of consistent UIs enabling the interaction with various underlying Web Services as it is demanded within the home appliance scenario.

To summarize our efforts, we realized an intensive state-of-the-art analysis searching for approaches that make it possible to generate automatically user interfaces for Service-based interactive applications and that create results which can be included ad-hoc into a running host application. To the best of our knowledge, no comparable approach like the one presented in this paper exists.

INTERACTIVE COMPONENTS

Interactive components are a domain-specific solution for user interfaces of Service-based interactive applications. Due to their characteristics, they solve the identified problems of dynamic integration of ad-hoc generated user interfaces into a hosting interactive application.

Interactive components contain all information necessary for registration purposes within a host application. They offer a user interface for providing input values and displaying output values of Web Service operations and contain all information needed for enabling the interaction of a human user with a remote Web Service. After an interactive component has been generated, it is transmitted to a host application (such as the universal appliance control application) via an application specific directory Service.

As soon as an interactive component is physically available on a target device, it runs through the following steps:

1. **Registration:** The interactive component communicates the information needed for its execution to the host application. The registration information covers:
   a. **Identifier**: In order to identify a component and to replace old versions on a target platform, a unique identifier is assigned to every interactive component.
   b. **Component Dependencies**: An interactive component may demand functionality that has to be provided by the host application. This includes for example functionality to determine the current geolocation or specific data storage facilities.
   c. **Input-/Output-Data**: An interactive component may interact with a host application via exchanging data. To identify the data needed as input and returned as output, references to data described within the associated functional interface description is used.
   d. **Context information**: Every interactive component may be generated for a concrete usage context such as a specific language, a specific target platform or a selected geographical location. A context description is expressed as a set of context type and value information.
   e. **Information for visualization purposes**: An interactive component is included dynamically into the navigation menu of a host application so that a user can activate it. For visualizing the component, data such as a label text or an icon is communicated to the host application.

   If a host application accepts the registration information and fulfills the requirements (thus it can offer the necessary components (b) and the appropriate input data (c)), the registration is successful. If the interactive component is not registered successfully, it is removed from the host application.
2. **Activation**: After an interactive component has been registered successfully and the current application context conforms to the context the component is intended for, the component can be activated by a user. The host application hands over the input parameters demanded by the interactive component during the registration procedure.
3. **Usage:** After a successful activation of the interactive component, a user can input data via the user interface and invoke remote Web Service operations. The operation results are being visualized in the user interface.
4. **Deactivation:** The interactive component can be deactivated at any time by the host application or the user. After the deactivation has been triggered, its user interface is not visible anymore. During deactivation of

an interactive component, it returns the output data specified during the registration process to the host application.

We have specified a Meta-model named Service-based Interactive Component Model (SBIC) for representing interactive components during the generation process. The structure of the model is sketched in Figure 2. The SBIC has been developed as a model representing an interactive component in a way close to a runtime representation. Thus, a model-to-code transformation mapping a SBIC instance to a platform specific interactive component can be implemented with little effort.

Besides the registration information, the SBIC describes the structure, navigation flow and data flow of the user interface. Furthermore, it contains layout and design information that is weaved into the model during the generation process. Thus, the layout and design of a generated interactive component can be adapted to the layout and design of a host application.

DEVELOPMENT APPROACH

As aforementioned in the state-of-the-art analysis, the development of Service-based interactive applications for various target platforms is currently a time-consuming task involving huge manual effort. Our approach is characterized by the novel idea to only use Service annotations for modeling user interfaces. As the state-of-the-art does not offer any Service annotation model enabling the specification of application-specific aspects such as navigation or data flows, we introduced new annotation types.

To not reinvent the wheel, these types were introduced into the annotation model originating from the ServFace project [8]. Besides others, the following annotation types have been specified and included into an extension of the ServFace annotation model:

1. **Navigation Flow:** Enables the possibility to define a navigation flow between the user interfaces derived from different Service operations. With every specified navigation flow, a data flow may be associated.

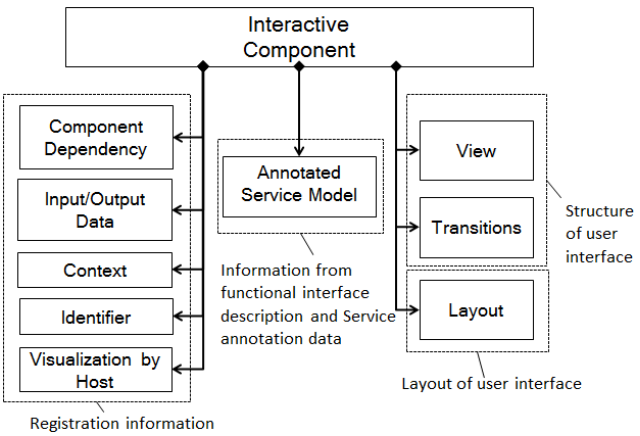

Figure 2. Structure of interactive component Meta-model

2. **Bundling:** Renders it possible to merge the user interfaces of different operations into one user interface. Furthermore, the view for input and output of one operation might be merged in one view.
3. **Provided by Host:** Marks an input parameter of a Service operation as input to the interactive component provided by the host application.
4. **Returned to Host:** Marks output parameters to be returned to the host application when the interactive component is deactivated.
5. **Component Dependency:** Defines dependencies to functionalities the host application has to provide.
6. **Initial Operations:** A subset of the operations defined in a functional interface description may be marked as initial. These operations can be accessed directly from the host application. Every initial operation is an entry point to activate the interactive component. After activating it, the user interface for the appropriate initial operation is visualized.

In order to specify these annotations, an authoring tool named Interactive Component Editor (ICE) [2] has been created. One of the views of the ICE is shown in Figure 3. After importing one or more functional interface descriptions into the tool, the tree structure of these descriptions is visualized. The nodes on the various levels (Service, operation, input/output parameters, data types) of the tree representation can be included into the modeling of the interactive component.

The screenshot in Figure 3 shows various navigation flows and data flows in a specific editor view during modeling an interactive component for a DVB-T device. The editor makes it possible to specify any of the annotations defined within the ServFace project and of the abovementioned extensions. Thus, a full featured development approach has been created enabling the model-based specification of interactive components solely with annotated functional interface descriptions.

After the modeling has been realized, the resulting annotations are serialized to a file and transferred to a remote annotation repository.

GENERATION APPROACH

The generation of interactive components is triggered on demand once an interactive component has to be embedded into a host application. In the case of the automated home scenario, the generation chain is activated as soon as a new device and thus a new annotated Web Service appears in the home infrastructure. Due to the usage of platform independent Meta-models and the platform specific parameterization of the generation approach, it can be reused for various target platforms and different host applications. The platform and host specific parameters have to be provided only once for every host application. The target specification can be reused during the generation of each interactive component that should be embedded into this host application.

The generation chain is summarized in Figure 4.

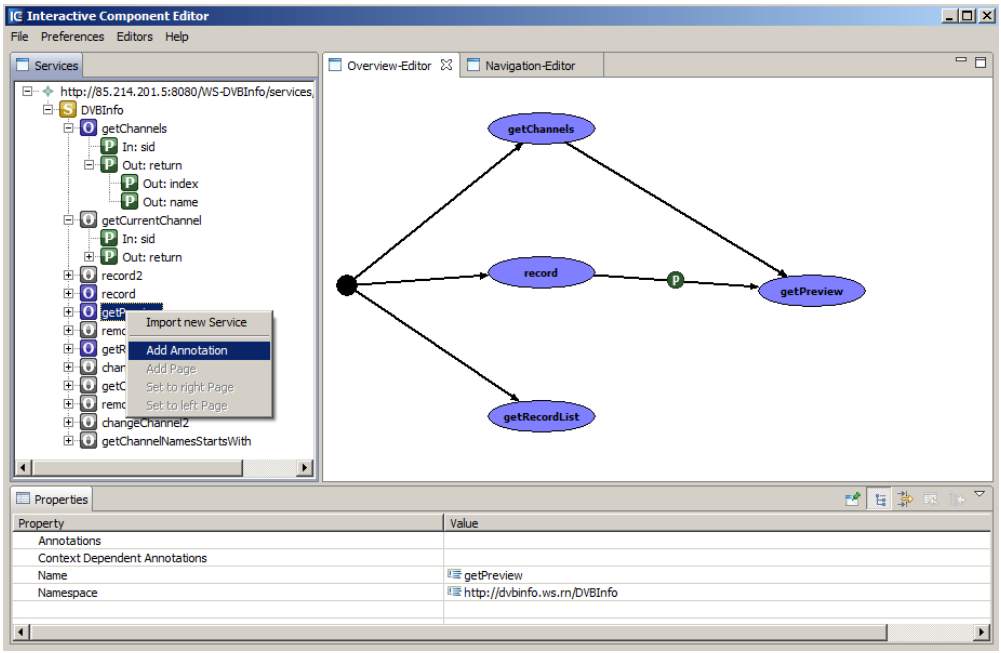

Figure 3. Screenshot of the Interactive Component Editor

In a first step, the functional interface description and the annotation file fetched from an annotation repository are parsed and merged together into a common model. For this purpose the Annotated Service Model (ASM) has been developed. It enables the representation of all ServFace annotations and the mentioned extensions. An ASM instance contains, on the one hand, the tree representation of the functional interface description with the different hierarchies (Service, operation, input/output parameters, data types) and, on the other hand, the annotations referencing a subset of the tree's nodes. This subset has been determined during the development approach applied e.g. by using the ICE.

After the ASM has been instantiated, a basic structure of the user interface with all its interactors is inferred in a first model-to-model transformation step. For representing the structure of the UI, a domain-specific model named Intermediate Service Frontend Model (ISF) adapted to the generation chain has been developed.

For instantiating the ISF, the ASM instance is traversed. By default for every Service operation one container for all input parameters and one container for all output parameters are derived and embedded into the ISF instance. This default behavior may be modified by annotations of the *Bundling* type (see previous section).

In a next step, appropriate interactors are derived from all input and output parameters passed during traversing of the ASM instance. They are embedded into the containers created in the previous step.

As the selected interactors are described within the ISF using platform specific vocabulary, this step is parameterized by a set of inference rules which determine which interactor should be selected under which condition. Besides the data types of an input or output parameter, the decision depends particularly on the set of annotations referencing this parameter.

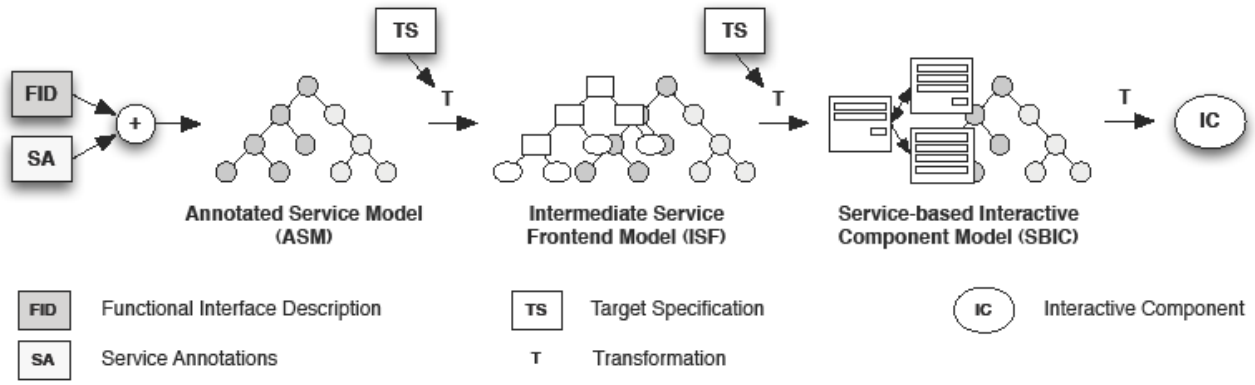

Figure 4. Generating interactive components from annotated functional interface descriptions

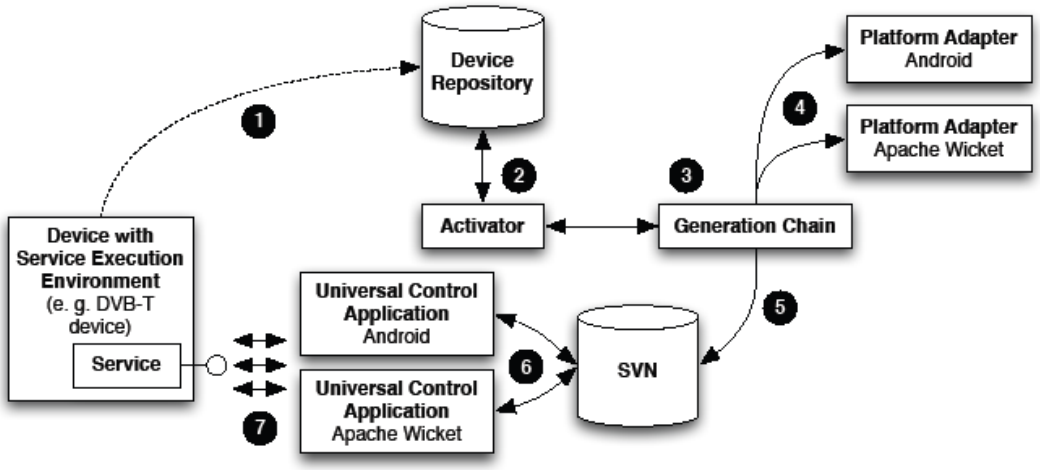

Figure 5. Usage of the generation approach within the home appliance scenario

In a second model-to-model transformation step, an instance of the SBIC Meta-model mentioned above is created from the ISF instance. Just as the transformation steps before, the implementation of this step can be reused for various target platforms due to the parameterization of the transformation. The provided parameters contain especially layout and design information for adapting the appearance of the interactive component to the appearance of a host application. This information can be made available a-priori and updated as soon as the layout and design of the host application is modified.

Finally, every SBIC instance is transformed to a platform specific interactive component that can be made available via a component repository to a host application. For every supported platform a model-to-code transformation has to be provided that takes an SBIC instance as input and returns a packaged and deployable interactive component as output.

The described generation chain builds the core of the home appliance scenario introduced above. Figure 5 shows the different steps used to make interactive components available in different universal control applications.

The approach runs through the following seven steps:

1. A device (e.g. a DVB-T device) is newly introduced into the infrastructure of Web-Service-enabled appliances. As soon as it is switched on, it is registered by transmitting the Uniform Resource Identifier (URI) of its functional interface description plus a reference to an annotation file to a device repository.
2. The device repository is monitored by a specific system component named Activator. As soon as this component detects a new device registration, it forwards the functional interface description referenced by the URI plus the annotation file to the generation chain described above.
3. The generation chain parses the functional interface description and the annotation file and transforms it first to an ISF and then to a SBIC instance.
4. Using platform-specific model-to-code transformations (named platform adapters) for all desired target platforms, the SBIC instance is transformed to platform specific interactive components.
5. These components are forwarded to a repository that is checked in fixed intervals by the used universal control applications for updates.
6. As soon as a new interactive component is available within the repository, it is downloaded to the target platform and registered in the universal control application.
7. After a successful registration of an interactive component it can be used to interact with the remote Web Service thus enabling device control.

REALIZATION AND EVALUATION

The approach has been implemented and evaluated using two heterogeneous target platforms as shown in Figure 5. As proof-of-concept for mobile devices, a universal control application and a platform adapter for the smartphone platform Android has been realized and as proof-of-concept for Web applications, the same parts of software have been implemented using the Apache Wicket framework. The repository for devices has been realized as a simple relational database. As technology for the repository for interactive components Subversion (SVN) was used. SVN has the central advantage that no additional mechanism for versioning of interactive components had to be introduced.

Figure 6 shows a screenshot of the prototypically implemented Web-based universal control application. As it is depicted in the figure, interactive components are embedded once they are registered in this application into its navigation menu shown on the left side. The initial operations (see fourth section) of each interactive component are displayed by a navigation option each. As soon as a user selects one of these options, the appropriate user interface is displayed in the center of the Web page.

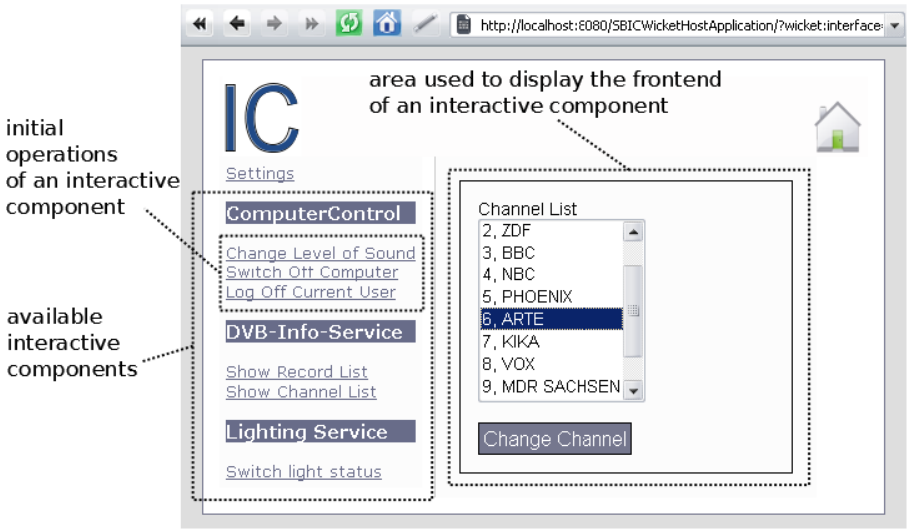

Figure 6. Prototype for the Web-based universal control application

The approach has been evaluated based on the prototypical implementation within an end user study. 25 people in the age between 19 and 31 years have used the two universal remote controllers to interact with a set of simulated home appliances. After fulfilling several minor tasks, the users participated either in a questionnaire or in an interview to get an insight into their experiences with the generated interactive components. Summarizing the results, the interactive components have been reviewed as easily and efficiently usable. All users stated that the user interface makes a consistent impression and looks like a monolithic application. No user criticized any form of inconsistency of the user interface. Thus, the component-based nature of the user interface is not recognized by end users.

SUMMARY AND OUTLOOK

This paper presented an approach to generate consistent universal controllers for Web-Service-enabled home appliances fully automatically. The approach is based on the idea to generate so-called interactive components from annotated functional interface descriptions. These components can be embedded dynamically during runtime into a host application. The applicability of the approach has been demonstrated by implementing the home appliance scenario using a Web-based universal control application and an appropriate application for a smartphone platform. The implementation has been evaluated in an end user study which confirmed the good quality and usability of the resulting user interfaces and their consistency. The control application made the impression to be monolithic though it consists of components provided for every available device.

Future work will focus in first place on improving the underlying development approach sketched in the third section. In this area it will be analyzed, how the provisioning of Service annotations may be simplified, e.g. by suggestion functionalities. Furthermore, implementing support for further target platforms is intended.

REFERENCES

1. Christensen, E., Curbera, F., Meredith and G., Weerawarana, S. Web Services Description Language (WSDL) 1.1. W3C Note. March 2011.
2. Feldmann, M., Martens, F., Berndt, G., Spillner, J., Schill, A. Rapid Developed of Service-based Interactive Applications using Service Annotations. In: Proceedings of IADIS International Conference WWW/Internet 2010.
3. Nichols, J. Automatically generating high-quality user interfaces for appliances. In CHI ’03 extended abstracts on Human factors in computing systems, pp. 624-625, ACM Press, 2003.
4. Nichols, J., Myers, B. A., Rothrock, B. UNIFORM: Automatically Generating Consistent Remote Control User Interfaces. In Proceedings of CHI'2006, pp. 611-620, Montreal, Canada , 2006.
5. Paternò, F., Santoro, C., Spano, L., D. Designing usable applications based on Web services. In: 1st Workshop on the Interplay between Usability Evaluation and Software Development, pp. 67 - 73. CEUR, 2008.
6. Paternò, F., Santoro, C., Spano, L. D., Exploiting Web service annotations in model-based user interface development. In: EICS'10 - 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 219 - 224. ACM, 2010.
7. He, J., I. Yen, T. Peng, J. Dong und F. Bastani: An Adaptive User Interface Generation Framework for Web Services. Proceedings of IEEE Congress on Services Part II, Seiten 175–182, 2008.
8. ServFace-Konsortium: *Deliverable 2.9 - Models for Service Annotations, User Interfaces, and Service-based Interactive Applications (final version).* Technical Report, SAP AG, Lyria S.A., Consiglio Nazionale Delle Ricerche, The University of Manchester, Technische Universität Dresden, 2010

A Context Taxonomy Supporting Public System Design

Romina Kühn, Christine Keller, Thomas Schlegel
Technische Universität Dresden
Nöthnitzer Str. 46, Dresden
{Romina.Kuehn, Christine.Keller, Thomas.Schlegel}@tu-dresden.de

ABSTRACT
Context awareness is the basis for a system's ability to adapt to changing conditions of its environment. This ability is especially important in the public domain where a variety of systems is used, so-called public systems. Public systems perform in public spaces and are available to all people, instead of focusing on specific user groups. They also often integrate many different devices. Thus, they need to be highly context-adaptive in many ways. However, it is very difficult to determine what context is. None of the existing definitions can serve as a guideline throughout the whole process of system development. Context relevant features need to be determined from scratch for each new system, making system design error-prone, costly and time-consuming. To support easy development of context-aware systems and applications, we propose a reusable taxonomy of context features for the public domain.

Author Keywords
Context taxonomy, Context awareness, Public system

INTRODUCTION

Ubiquitous technologies are highly applicable in spaces, where many people need to access certain services. The first vision of ubiquitous systems by Mark Weiser introduced the idea of an pervasive work environment, where many people can work together, supported by invisible and intelligent systems that surround them [23]. But the "smart office" is not the only application of ubiquitous technologies. Recent research efforts explore the usage of ubiquitous systems in the public, like hospitals, public transport systems or other public spaces [9, 5, 7]. In public spaces, many people have to access many different services, different data and use different devices to do so, like personal mobile device or public displays. Ubiquitous technologies can be used to integrate those different devices and the different services that are provided in public space. Such ubiquitous public systems have to be context-aware and to adapt to the requirements of many different kinds of users or environments.

To do so, the context of usage must be captured and then correctly classified. Based on the captured context, the system then must be able to adapt the interaction with the user. Depending on his location, a user for example needs different data and based on his abilities, he needs to interact using speech based interfaces, for example if he is blind. Based on this context classification, the system's behaviour can be modeled. In our previous work, we have developed a method to model interactive systems on the basis of the technique of Use Cases. Our method allows to model interactive components and to modify the provided interactions according to context. In this paper, we want to describe a context taxonomy that models contexts and context criteria of ubiquitous public systems. We also describe how these context criteria can be substantiated for different kinds of public systems.

Related Work

Most of the existing information systems that perfom in public systems are concerned with public transportation, often specialized for example, for the blind people [5, 2]. Another kind of public system is focusing on tourists [11, 13]. As ubiquitous technologies became popular, they were also applied in the public domain, for example integrating public displays and mobile devices or stationary information terminals [17, 21].

The idea of modeling context for ubiquitous systems is not a brand new topic [8]. Early context-aware systems are mostly location based or consider location and additionally physical conditions as a system's possible context [22, 4]. In recent years, the view on "context" has changed from a mainly physical to a broader view. Some choose to consider tasks or activities of a user as the system's context to take into account, too [18, 16]. In public systems, all of these variations of context have to be considered, but there are additional views on context that can become relevant. There is, for example, also a social context that may be important for the usage of public systems. We developed a reusable taxonomy of context criteria that are typically found in the public domain and we therefore consider essential for public systems.

This paper is organised as follows. In the next chapter, we want to present the aforementioned method for modeling interactive public systems we developed in our previous work. We will describe how this method allows to easily model such systems in a context-adaptive way. The following chapter then describes, how we modeled our context taxonomy. First, we want to describe our perception of context and the

R. Kühn, C. Keller, T. Schlegel: A Context Taxonomy Supporting Public System Design.
Proc. of 1st International Workshop on Model-based Interactive Ubiquitous System 2011,
Pisa, Italy, June 13, 2011, http://ceur-ws.org/Vol-787

terms we use to derive specific context types from relatively abstract context criteria. We then present the user-centered context taxonomy we developed. We also present exemplarily modeled Interaction-Cases that builds on our context taxonomy. We conclude the paper discussing our approach and describing work that is planned for further research efforts.

MODELING INTERACTIVE UBIQUITOUS PUBLIC SYSTEMS

In order to support the seamless integration of various devices and services in ubiquitous public systems, these systems must be properly designed and modeled. Persona and scenarios can serve as a basis to define the user's requirements and the system's behaviour [12, 1]. Based on informally described scenarios, Use Cases can be derived that describe the system's behaviour from a user's perspective. Use Cases describe the system's requirements in a more formal way.

In our previous work, we proposed the method of Interaction-Cases for modeling interactive systems [20]. Interaction-Cases can be used to describe the interaction between user and system in a semi-formal way. Types of Interaction-Cases can be predefined, they are therefore reusable. Interaction-Cases can already be defined when requirements are determined in early phases and then be substantiated up to a very specific level, that can be linked directly to Use Case diagrams and code fragments.

In order to develop context-aware ubiquitous systems, the contexts must be modeled in advance and depending on these contexts, the context-adaptive behaviour of the system needs to be modeled, too. We therefore refined our Interaction-Case method, allowing these Interaction-Cases to be context-adaptive [19]. In early design stages, an Interaction-Case can be marked as context-adaptive to a certain context. The Interaction-Case and the context definition may be very coarse-grained at first. In those early phases, the specific context features that lead to system's adaptations may not be known, but the general context criteria that influence the interaction process between system and user can already be anticipated. Therefore, it should be possible to refine the context criteria as the specification of the system proceeds.

Using context-adaptive Interaction-Cases, it is possible to define the interaction process between system and user in a different way for different situations. If the system observes, for example, that the ambient noise level is very high, it can adapt its audio volume. Another example is, that if the user is blind, it is necessary to switch to audio interaction instead of visual.

The development and modeling of interactive ubiquitous public systems becomes easy and less time-consuming using Interaction-Cases. The method depends, however, on a properly modeled context hierarchy, that serves as a basis for development of context-adaptive scenarios and interactions. We therefore propose a context taxonomy for contexts in ubiquitous systems. It models contexts that can occur in the public domain and are of possible interest for public systems. The structure of the taxonomy reflects the usage of the context criterions in the iterative development of Interaction-Cases and allows step-by-step refinement of contexts from coarse-grained contexts to fine-grained context types. We will describe this structure and the context taxonomy for ubiquitous public systems in the following.

A CONTEXT TAXONOMY FOR THE PUBLIC DOMAIN

The public domain has special requirements towards information systems and a variety of contexts are possible. There are different users with a different background, different culture, knowledge etc. and a wide range of devices such as mobile devices, public displays, but also stationary information terminals. In order to capture the possible contexts that influence the interaction between a user and the ubiquitous public system, we focused on the user and the situations that can arise in ubiquitous public systems. We do not claim that our context taxonomy is complete, but it can serve as a starting point for further refinement. Which contexts are relevant and which are not depends on the system's characteristics, its structure and its purpose. The structure of our context taxonomy supports easy refinement of the contexts that are relevant for the task at hand.

Structure of the Taxonomy

We based our perception of context on the definition given by Dey and Abowd [8]:

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

For the usage of context with Interaction-Cases and for iterative refinement of relevant contexts, we describe context as different context criteria that are organized in a hierarchical taxonomy. These can be used as a first overview on possible context dimensions for public systems. We then describe different context types, that can be specified and derived from a certain context criterion [14]. Context specifications can then be substantiated from context types by allowing a system's architect to subsequently define values or value ranges for which certain context types are laid out in his system's context design. A possible structure of such a hierarchy is shown in figure 1.

- *Context* is information that characterizes situations or circumstances of an entity like a person, a place or an object [8]. The complete context a system is able to capture in a specific situation, is most likely combined of different types of context features and different values of these features. A complete system's context can, for example, be combined from a temperature of 20 degrees celcius, the availablility of visual and audio output and a certain time and location.

- *Context criteria* are different categories in which context can be defined. The context criteria are criteria that may influence a system's context and are defined on a relatively abstract level. Context criteria can be hierarchi-

cally organized. Examples of context criteria are "Climate" and "Temperature" but also "Perceptive Context" and "visual".

- A *context type* is a sub-category of a context criterion. From context criteria on an abstract level, several context types can be derived that describe features of this context criterion on a specific level. As an example, from the context criterion "visual", a system designer can define the context types called "blind" and "visually impaired".
- Context types can be specified directly by defining values or value ranges. These are called *context specifications*. A context specification for the context type "visually impaired" may be a value range capturing vision from 20% - 70% or from 71% - 99% as shown in figure 1.

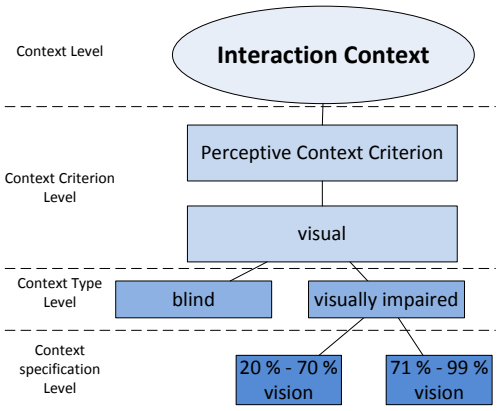

Figure 1. Example of a specific context including context criteria, types and specifications

CONTEXT IN UBIQUITOUS PUBLIC SYSTEMS

Central to the description of context in ubiquitous public systems is the user, as shown in figure 2. These systems adapt to the context they perceive in order to provide an optimized interface for the many different users that use them. We therefore started to collect the requirements of users in public systems. Based on these requirements, we differentiated several contexts that can be useful in modeling context-aware ubiquitous public systems. These context served as starting points for further refinement. In the following sections we will therefore explore these categories and the possible use in modeling interactions in ubiquitous public systems.

Interaction context

By modeling context-adaptive Interaction-Cases, it is possible to model the interaction processes a ubiquitous public system provides. Our first step is therefore to capture context criterions that directly affect the interactive process between users and systems. Ubiquitous public systems consist of different devices that provide different interaction modalities. The user may have different abilities to interact with the system, too. We modeled the different context criteria that are involved in interaction context by mapping the interaction process on part of the system and on part of the user as shown in figure 3. The system possesses input options and

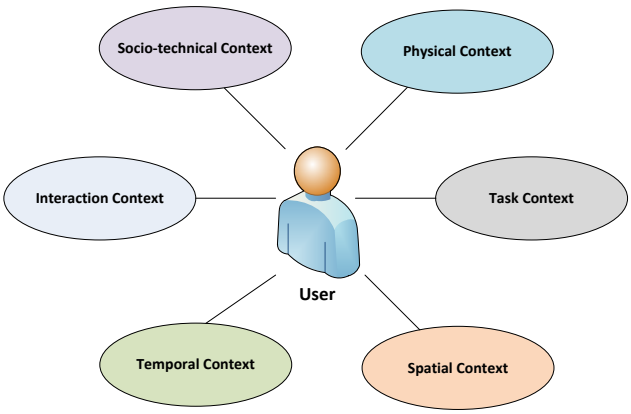

Figure 2. User-centered context

output options. In between these steps, the system processes the given input. According to this, our context criteria for the system's interaction context are the following:

- *Input*: For the context criterion of input, context types can be defined that describe the abilities of the system to get input at all. Some devices used in public systems, for example like tourist information terminals in cities, are equipped with keyboards and sometimes even a mouse-like device. Many public information systems nowadays use touch screens, sometimes in addition to keyboards [15]. From the input context criterion, it is possible to derive context types that can be used to classify the possible inputs of a system.
- *Processing*: The main task of an information system is to process data. The processing context criterion can capture the circumstances of processing in ubiquitous public systems that may influence the system's interaction towards the user. Small devices, like mobile phones, have less processing power than devices connected to a processing server, for example. The processing capabilities affect the possible interactions with the user and can therefore be modeled using the processing context criterion.
- *Output*: The output context criterion captures the abilities of the system to pass information to the user. In public systems, all kinds of public displays are known [10, 6]. Thus, most ubiquitous public systems have visual output abilities. But additional output modalities are also possible, for example speech output or haptic output interfaces.

Interaction on part of the user begins with perception. The perception abilities of the user may require the ubiquitous public system to adapt and, for example, provide different output modes. After perceiving information, a user processes the information, just like the system itself does. The user also acts in order to input information to the system or to request information from the system. We therefore captured the interaction context on the part of the user using the following context criteria:

- *Perception*: This context criterion captures how a user can perceive input. A person can perceive using his senses. Regarding the interaction with computer systems, sight, hearing and touch are the main perception channels. Context types derived from this context criterion can grasp the perceptive abilities of a user.
- *Cognitive*: The cognitive abilities of a user can be grasped using the cognitive context criterion. Children, for example, have other cognitive abilities than adults. A system can then adapt to these cognitive abilities, if they are known, and present information, for example, in simpler form.
- *Action*: The abilities of the user to act towards the system can be modeled using the action context criterion. A user can act using gestures, voice, facial expression or movement. The cognitive context criterion can be used to capture the acting abilities of a user, analogous to his perceptive abilities.

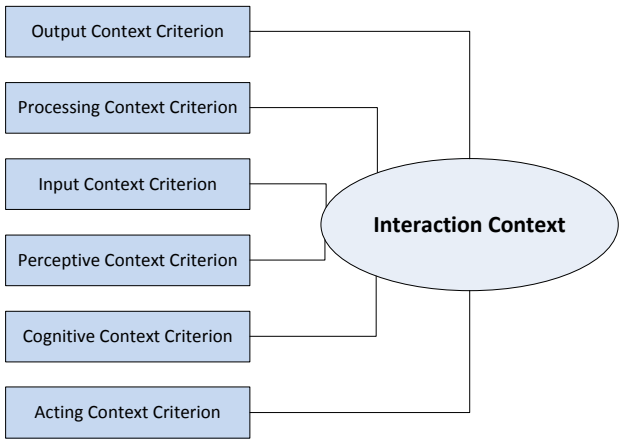

Figure 3. Interaction context

The different sides of interaction context are comparable. From a certain point of view, the input context criterion and the acting context criterion capture the same type of context, for example, speech input. The same is true comparing output and perception context, capturing, for example, visual input. We distinguished a system's interaction context from the user's interaction context. Using two different "sides" of interaction context means, that it is possible to perceive that the user is blind, which is a context type deriving from the perception context criterion. At the same time, it is possible that the system is only able to give visual output (output context criterion). This situation can only be observed using a perceptive context that is distinguished from an output context.

Socio-technical context

Another interesting aspect of context is the socio-technical context in figure 4. We identified four socio-technical context criteria [19] which we divided, depending on their focus, in user and system modelling context criteria. The following social-technical context criteria are user-centered.

- *Sociological context criterion*: With the sociological context criterion we describe the rules which people in public systems are following. These rules allow us to model possible scenarios for different sociological contexts and so affect the usage of ubiquitous computing in public systems. For example, it is a common rule not to disturb other people in surroundings like churches with mobile phones or other devices or to request people's personal data where others can see it.
- *Organizational context criterion*: The organizational context criterion describes a third party like organizations which are somehow involved in public systems. This context criterion can model the different conditions and possibilities of, for example, public transport organizations, supplier or other organizations that are associated with public systems.

Besides the user-centered context criteria there are two system's socio-technical context criteria which are described as follows:

- *Operational context criterion*: In public systems there is a multitude of processes, procedures and activities which are not directly visible to a user. These operations can be summarized in the operational context criterion. For example, activities or procedures like to operate the turnout in a control center can affect this criterion.
- *Technical context criterion*: Another system centered criterion is the technical context criterion. It includes all technical abilities of a system, for example, the ability to show real-time data or just data which can not be updated automatically.

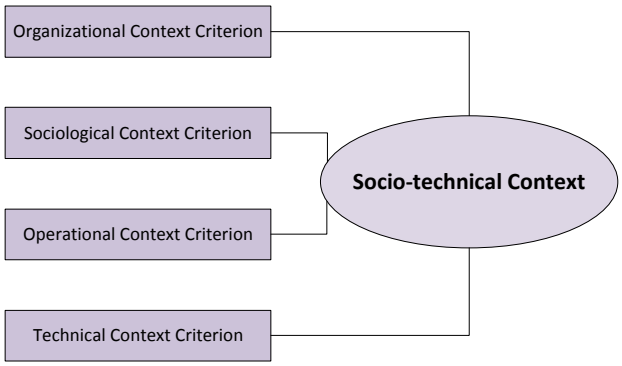

Figure 4. Socio-technical context

Further contexts

Beside the contexts we described above, there are some further contexts that affect the usage of ubiquitous systems in the public domain. We briefly characterize these in the following paragraphs.

Physical context
Physical context of ubiquitous public systems captures, for example, temperature, humidity, ambient noise level or brightness. The context criterion "ambient noise level" can be relevant, for example, for adapting the output volume of speech output as the ambient noise level raises.

Task context
There are several approaches to capture task or activity based context [18]. The task the user wants to complete, does influence the interactions he pursues. We therefore plan to capture different task-based contexts for public systems in our future work.

Spatial context
The spatial, or location-based, context is described in other projects and papers e.g. by Bauer et al. or Bellavista et al.[3, 4]. Spatial context can, for example, capture the location of a user but also the user's movements, which means whether the users walks or stands.

Temporal context
As already described in our previous work, the temporal context contains absolute and relative time [19]. Time aspects can, for example, affect the presentation of data both on a mobile device and public displays. Further contexts related to time are conceivable. We will explore these aspects and their possible use in ubiquitous public systems in future work.

USAGE OF CONTEXT FOR MODELING INTERACTION

In this section, we want to give a short example of the usage of context-adaptive Interaction-Cases. Given the context hierarchy in figure 1, the input of data in a public system can be modeled in different ways. Our example of a public system is a public transport system. In such a setting, people want to retrieve information on timetables of buses or trains. We therefore modeled the Use Case `retrieveTimetable-Information`, as displayed in figure 5. In order to request information on a timetable, the user needs to specify the location and time of departure. The Use Case thus contains an Interaction-Case `enter departure information`.

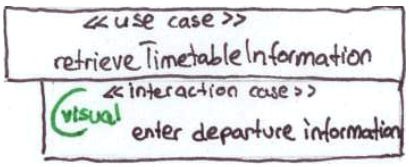

Figure 5. Use Case `retrieveTimetableInformation` and associated first Interaction-Case

The system we modeled as an example should adapt to the perceptive abilities of its users. The Interaction-Case `enter departure information` is therefore modeled context-sensitive. It can be adapted regarding the context criterion "visual". We modeled two Interaction-Cases that implement the given Interaction-Case for two different context criterions. The first is the "normal" Interaction-Case that uses keyboard input to acquire the departure information. It is shown in figure 6 on the right.

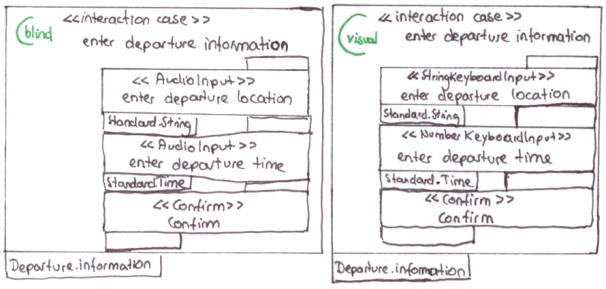

Figure 6. General Interaction-Case and context-adaptive derivation

However, if the user of the system is blind, he is not able to use a normal keyboard to provide the departure information. In this case, the system should switch to audio interaction. The Interaction-Case in figure 6 is therefore modeled for the context-type "blind" and the system adapts the input modalitiy to audio. Using this modeling technique and our context taxonomy for public systems, it becomes possible to model context-adaptive interactive ubiquitous public systems easily, already beginning in early design phases using pen and paper.

CONCLUSION AND FUTURE WORK

In this paper we presented a structure for a context taxonomy that supports modeling and development of context-aware ubiquitous public systems. Using a context taxonomy for the public domain as a basis and implementing the method of modeling Interaction-Cases, it becomes possible to define interactions between system and user in an iterative way. The structure of context we proposed also supports the step-by-step refinement not only of Interaction-Cases, but also of the involved contexts, as shown exemplary in the previous section. We also presented a context taxonomy for the public domain that can be used as a starting point to model contexts for ubiquitous public systems.

Our goal is to enlarge the taxonomy we described above and to refine the context criteria in the future. It is, for example, possible, to refine the input context criterion within the interaction context with respect to the data type that can be entered via the different input channels. Some input channels may, for example, only be relevant or active for input of special data types. We also want to explore the possibilities of deriving context-adapted Interaction-Cases automatically using rule-based substitution of certain Interaction-Steps. We hope to further improve the modeling method and the underlying context taxonomy and therefore to improve and facilitate the development of ubiquitous public systems.

ACKNOWLEDGEMENTS

Part of this work has been executed under the project IP-KOM-ÖV funded by the German Federal Ministry of Economics and Technology (BMWi) under the grant number 19P10003O.

REFERENCES

1. Aoyama, M. Persona-scenario-goal methodology for user-centered requirements engineering. *Requirements Engineering, IEEE International Conference on 0* (2007), 185–194.

2. Banaître, M., Couderc, P., Pauty, J., and Becus, M. Ubibus: Ubiquitous computing to help blind people in public transport. In *Mobile Human-Computer Interaction - MobileHCI 2004*, S. Brewster and M. Dunlop, Eds., vol. 3160 of *Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, 2004, 535–537.

3. Bauer, M., Becker, C., and Rothermel, K. Location models from the perspective of context-aware applications and mobile ad hoc networks. *Personal and Ubiquitous Computing 6* (2002), 322–328. 10.1007/s007790200036.

4. Bellavista, P., Corradi, A., Montanari, R., and Stefanelli, C. A mobile computing middleware for location- and context-aware internet data services.

5. Bertolotto, M., O'Hare, G. M. P., Strahan, R., Brophy, A., Martin, A. N., and McLoughlin, E. Bus catcher: a context sensitive prototype system for public transportation users. In *WISE Workshops*, B. Huang, T. W. Ling, M. K. Mohania, W. K. Ng, J.-R. Wen, and S. K. Gupta, Eds., IEEE Computer Society (2002), 64–72.

6. Brignull, H., and Rogers, Y. Enticing people to interact with large public displays in public spaces. In *In Proceedings of the IFIP International Conference on Human-Computer Interaction (INTERACT 2003)* (2003).

7. Cheverst, K., Davies, N., Mitchell, K., and Friday, A. Experiences of developing and deploying a context-aware tourist guide: the guide project. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, ACM (New York, NY, USA, 2000), 20–31.

8. Dey, A. K., and Abowd, G. D. Towards a better understanding of context and context-awareness. In *Computer Human Intraction 2000 Workshop on the What, Who, Where, When, Why and How of Context-Awareness* (2000).

9. Favela, J., Rodríguez, M., Preciado, A., and González, V. M. Integrating context-aware public displays into a mobile hospital information system. *IEEE Transactions on Information Technology in Biomedicine 8*, 3 (September 2004), 279–286.

10. Greenberg, S., Boyle, M., and Laberge, J. Pdas and shared public displays: Making personal information public, and public information personal. *Personal and Ubiquitous Computing 3* (1999), 54–64.

11. Hristova, N. Ad-me: A contextsensitive advertising system. In *University of Maynooth* (2001), 10–12.

12. Kazman, R., Abowd, G., Bass, L., and Clements, P. Scenario-based analysis of software architecture. *IEEE Software 13*, 6 (1996), 47–55.

13. Klante, P., Krösche, J., and Boll, S. Accessights - a multimodal location-aware mobile tourist information system. In *Computers Helping People with Special Needs*, K. Miesenberger, J. Klaus, W. Zagler, and D. Burger, Eds., vol. 3118 of *Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, 2004, 627–627.

14. Korpipää, P., and Mäntyärvi, J. An ontology for mobile device sensor-based context awareness. In *Modeling and Using Context*, P. Blackburn, C. Ghidini, R. Turner, and F. Giunchiglia, Eds., vol. 2680 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003, 451–458.

15. Maguire, M. A review of user-interface design guidelines for public information kiosk systems. *International journal of human-computer studies 50*, 3 (1999), 263–286.

16. Ni, H., Zhou, X., Zhang, D., and Heng, N. Context-dependent task computing in pervasive environment. In *Ubiquitous Computing Systems*, H. Youn, M. Kim, and H. Morikawa, Eds., vol. 4239 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, 119–128.

17. Peterson, M. P. Pervasive and ubiquitous public map displays. In *ICA UPIMap2004* (Tokyo, 2004).

18. Prekop, P., and Burnett, M. Activities, context and ubiquitous computing. *Computer Communications 26*, 11 (2003), 1168–1176.

19. Schlegel, T., and Keller, C. Model-based ubiquitous interaction concepts and contexts in public systems (2011).

20. Schlegel, T., and Raschke, M. Interaction-cases: Model-based description of complex interactions in use cases. In *Proceedings of IADIS International Conference Interfaces and Human Computer Interaction* (Freiburg, Germany, 2010).

21. Storz, O., Friday, A., Davies, N., Finney, J., Sas, C., and Sheridan, J. Public ubiquitous computing systems: Lessons from the e-campus display deployments. *IEEE Pervasive Computing 5*, 3 (2006), 40–47.

22. Want, R., Hopper, A., Falcao, V., and Gibbons, J. The active badge location system. *ACM Transactions on Information Systems 10*, 1 (1992), 91–102.

23. Weiser, M. The computer for the 21st century. *Human-computer interaction: toward the year 2000* (1995), 933–940.

Navigating the Personal Information Sphere

Simon Thiel
Fraunhofer IAO
Nobelstraße 12
70569 Stuttgart, Germany

Andreas Schuller
Fraunhofer IAO
Nobelstraße 12
70569 Stuttgart, Germany

Fabian Hermann
Fraunhofer IAO
Nobelstraße 12
70569 Stuttgart, Germany

ABSTRACT
A major trend in information society is integration of personal information from different sources. Digital data about persons, their behavior, their content and their social structure is merged into the personal information sphere; a multi-dimensional space containing information related to a person. In the research project di.me[1], funded by the EC, a userware is developed to support the user managing it's personal sphere on multiple platforms.

Main requirement for the userware is to help the user keeping an overview on his personal data, while giving a powerful tool for changing all kind of aspects, like e.g. changing access rights, merging information from different sources and structure it according to his mental model. For this an easy to understand, but rich visualization of information and relations is required. A user-interface concept describes, how a user can navigate through his information sphere and which artifacts support managing it.

This paper describes the user-interface concept within the di.me userware, giving special focus on navigation and visualization of the personal information sphere.

Keywords
Personal Information Sphere, Context-Awareness, Intelligent User Interfaces, Augmented Identity

INTRODUCTION

Social networks play an increasing role in the online community. To stay connected with colleagues, friends and family (multiple) accounts on various social network platforms are quite common. Keeping track on the various accounts, updates and changes, however becomes more and more difficult. One goal of di.me is to integrate several of these platforms into one personal information sphere that is controlled by the user. Assembling the profile information with files and information stored locally on the desktop helps to compile a rich semantic model of the user's personal information sphere. An easy to use, multi platform user interface enables the user to manage his personal sphere and to keep control on the information he shares.

[1] di.me = "digital.me: Userware for the Intelligent, Intuitive, and Trust-Enhancing Management of the User s Personal Information Sphere in Digital and Social Environments"

Approach

General approach of di.me project describes the development of a userware consisting of a private service and several clients for accessing it. The private service can be installed on a private server or hosted at a third-party (e.g. cloud-) provider. The user can access the private service via the di.me client running on his desktop computer or by using a mobile application.

Personal Information Sphere

Core concept in di.me is the Personal Information Sphere (PS). The metaphor of a sphere (Figure 1) containing references to all information related to a person has been established in previous projects of Fraunhofer IAO [Schu09]. Information in the PS is structured along a semantic model containing meta-information about each information artifact. The meta-information covers classical elements (e.g. known from file systems) like access rights, information about the owner, date-of-creation and date-of-last-edit. But also further history information about changes of ownership is assigned to the artifact. In the semantic model the concepts (e.g. information artifacts) are interlinked by relations. These give indication about instance, composition, aggregation, or general association relationship between two concepts. For management of the personal information sphere, it's important to categorize and cluster the user's information artifacts. Humans require structure to control and oversee large amounts of information artifacts. This structure can be predefined following some reasonable default categorization or can be defined by the user following his mental model. In the best case a general default structure can be expanded and adapted by the user. This higher level of abstraction the user reaches by categorization enables him to control larger amounts of artifacts, keeping track of granted permissions and navigating through his data.

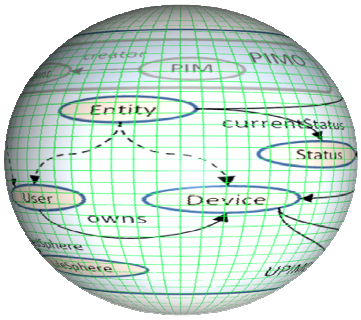

Figure 1: di.me Personal Information Sphere

A straight forward representation of this categorization can be implemented by establishing concepts for each category (or tag) aggregating the referring information artifacts or sub categories. Details about the implementation of the di.me semantic model are beyond the scope of this paper and will be discussed in separate publications. For describing the user interface, two lines of categorization are of particular interest: Categorization of persons or contacts into "Groups" and categorization of content and profile information objects as "Information Categories". (Figure 2)

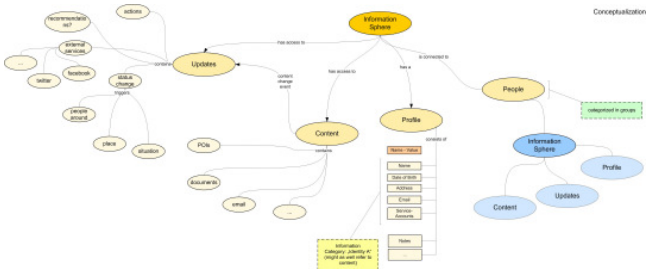

Figure 2: Conceptualization of UI-concepts

Sharing the Personal Information Sphere

di.me supports mechanisms for sharing information with persons known to the user. To give the user maximum flexibility in defining, who should be able to access a specific resource on the one hand, but on the other hand provide functionality for rich structuring and reach a high usability, the sharing mechanism allows for sharing information objects to single persons, but also supports sharing whole categories to groups (see Figure 3). However, it's clear that assigning single pieces of information to single persons will lead to a strong fragmentation of the access model and there is a high risk for the user to loose track of the permissions granted. Therefore one challenge for the UI-concept is to encourage the user using groups and categories for setting access rights. Also the system should come up with suggestions about rearranging existing groups, creating new groups or to aggregate a set of information objects in a new Information Category.

Recommendations from di.me

Helping the user to structure his personal information sphere is one application of recommendations coming from the system. Central concept of di.me userware is to establish a powerful tool, enforcing the user to manage his personal data and avoiding to restrict the user's actions. So the system will not change (the structure) of the PS proactively, but give recommendation that the user is free to accept, adapt or deny. Within his daily schedule a user will typically have some time-slots, when he for example enjoys browsing his social networks, sorting persons into groups and photos into categories. So, recommendations don't force themselves to the user, but are supporting him when there is time for it.

di.me gives recommendations regarding to aspects as follows:

- Organizing persons in groups: adding persons to groups, splitting groups, merging groups, etc.
- Organizing information objects in Information Categories, e.g. via tagging: adding information objects to categories, merging categories, splitting categories, structure categories, etc.
- Detection of not yet specified situations
- Sharing information objects/categories with peers/ groups
- Disclose or hide status updates

To produce these kinds of recommendation, the semantic model in di.me is required to provide semantic for all information objects, categories, groups and situations. In a continuous process the recommendation engine is reasoning on the various aspects of the personal information sphere and produces recommendations accordingly.[2]

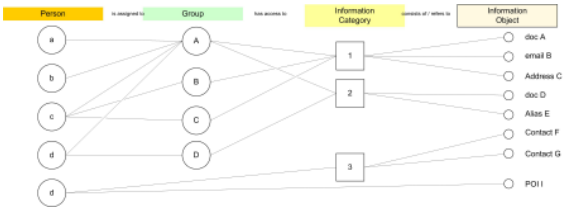

Figure 3: Categories and Access Rules

Integration of Services

Internet communities are broadly using social networks with many platforms [Bloch10], according to [Meek10] the usage of social networking exceeds the usage of email since 2009. A main target for di.me userware is to provide a platform for integration of existing user accounts for many social network platforms. For this, external services can be accessed via di.me. Personal information provided on a social network platform is presented as part of the personal information sphere in di.me. To avoid redundancy and provide convenient synchronization, artifacts like name, address, etc., relevant on many platforms are automatically merged and can be managed in a single place. It's clear that not all external platforms support APIs for editing data, but wherever it is available, automatic updates of personal information should be provided.

Another aspect of social third-party services is that they provide a rich variety of channels for communication. A concept of di.me is to support communication with a person by use of the known channels shared with this person. An example: As a result of merging his Skype contacts with di.me, Max knows that Peter is reachable via Skype, however he also knows Peter's email address. So, when

[2] Details about this recommendation engine will also be subject of further publications.

sending a message to Peter, Max get's the choice: whether he wants to send it via Skype or email it.

Personal History

For all external services di.me is able to access, it will also capture status information and messages or updates coming in. The presentation of these updates is the central entry point for the user. These updates (form the system or from other users) are shown on the home view of di.me. An implementation of such a view has to provide rich filtering options to give the user control on which information to be displayed. Filtering is not only to be done manually, but generally set according to the situation the system recognizes. (Recognition of situations will be explained in the following paragraph.) Depending on the filter setting some of the updates won't be displayed when incoming. However, all incoming updates are stored in the personal history. The history is a timeline that refers to updates and events collected by di.me. So, di.me provides a view to browse and search the personal history. Again, rich filtering and sorting options allow following different aspects of personal history. Such aspects can be persons, communication channels or abstract life spheres. Filtering can also take place according to situations.

Context Sensitive UI & Situations

For further control of the client behavior in di.me, particular in the mobile context, the client is able to perceive environmental and activity information, which is assembled in the user's personal context. Following rules set up by default or specified by the user, situations are derived from life content information. These situations and particularly the change of situations are a source for triggering multiple reactions in di.me. The change of situation may cause the system to:

- Change the configuration of updates shown to the user, and how they will be alerted
- Update specified contacts about this situation change.
- Update external services about change of situation. (E.g. a service tracking persons and showing people around.)
- Provide access to information to specified persons or groups.
- Show recommendations about potential actions.

Following the specified rules, the user interface adapts according to changes of the situation recognized. This mainly concerns the set of data accessible to the user. The recommendation feature allows for interacting with the user. The simple principle of: suggestion, accept, adapt and deny, lays a basis for improving the recommendation mechanism by reinforcement learning technologies.

Implementation

The di.me project started in November 2010. Therefore development is still in a conceptual phase. However, UI prototypes showing approaches for navigation and visualization have been developed and will be introduced in the following section. The di.me approach covers clients for desktop and mobile devices. Within this paper, navigation aspects will be shown for a mobile scenario, while an example for visualization of the personal history is given for a desktop context.

Navigation

For mobile application development of di.me client is subject technical restrictions: The client has to comply with limited processing power and a rather small display. Thus the UI is required to restrict each view to a minimum number of details shown at a time. Nevertheless, the challenge is to give the user as much (useful) information as possible. So, the mobile scenario is most interesting from a UI-design perspective.

For the main navigation of the di.me client six items and correlating top-level views have been specified: (Figure 4)

- Home
- MySphere
- People
- Timeline
- Situations
- Settings

Since, on a smart phone, the display size is quite restricted, "Situations" and "Settings" are reachable via a "more" menu item.

The Home view serves as navigation cockpit for the di.me client application. It consists of two views "Updates" and "Tips", where by default the updates view is shown when the application gets activated. Depending on the (filter) settings and the situation currently detected, the scope of updates to be shown is determined. Each of the updates shown in the list can be selected to see more details of the update, containing the full text, the sender as well as some timestamp. This detail view also serves as starting point to jump to the senders profile or to give direct reply on the update. In the "Tips" view, recommendations coming from the di.me system get displayed. Selecting a tip again opens a detailed view on the recommendation. Here the user can chose to follow (accept) the recommendation, adapt or ignore it.

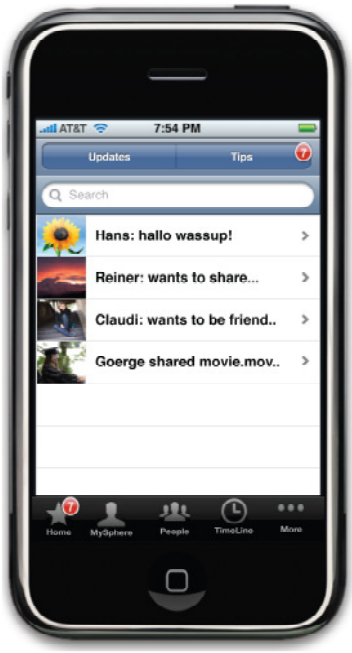

Figure 4: Home-Screen (Design Prototype)

As discussed in the approach, recommendations can contain a suggestion to restructure the groups of a personal information sphere. Managing groups on a small display can easily become a confusing task. When showing the people view, the user can switch into management mode by turning the phone by 90 degrees. (Figure 5) In this view, the landscape format is split into two columns showing persons and groups at the same time. The user can now perform drag and drop operations to add persons to groups or remove them accordingly. Filter mechanisms for both columns can be set separately, reducing number of persons and groups respectively shown in the lists. This management view is provided also for structuring information categories for the view MySphere.

Figure 5: Organizing Groups (Design Prototype)

MySphere allows the user to browse and search the personal information sphere. Again, rich filtering mechanisms allow for reducing the amount of information shown in this view. On the top of the view a search field allows for searching arbitrary information objects or categories.

In Situations the user is able to control and define the situations recognized and to specify the client's behavior accordingly.

Timeline shows a simple visualization of the personal history quite similar to a calendar application.

Visualization

The UI design for the desktop client of di.me adopts the look and feel that was introduced with the mobile platform. Also the main menu is structured following the six main menu items. Since the design of the navigation is not very much restricted, on the desktop challenges for visualization are of particular interest. As an example this paragraph introduces the visualization of the personal history (Timeline) on the desktop UI.

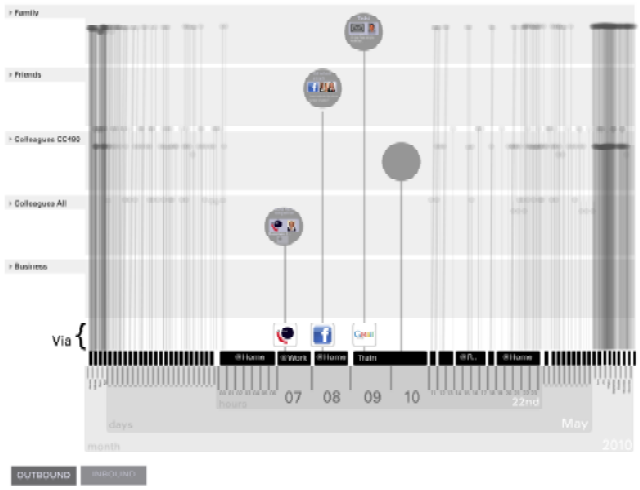

Figure 6: Visualization of History (Design Prototype)

As described previously, di.me userware is collecting incoming updates and upcoming events in the personal history. Updates are coming from various sources and over time a vast amount of items get collected. Providing an easy to use, intuitive UI for browsing such a history is one main challenge for visualization in di.me.

Figure 6 shows the current design prototype for visualization of the personal history. The timescale is shown on the bottom of the page, zoomed in the middle to the highest granularity selected. When approaching the borders of the view, the timescale is zoomed out, showing a whole year as a horizon. Swim lanes with horizontal orientation describe groups that have been senders or receivers of communication acts (updates received/sent). Bubbles situated within these swim lanes indicate acts of communication sharing a close semantic relationship. The user can browse through the history by scrolling right or left, zoom in and out at the current position, select a topic to

follow it through his history or jump into a bubble to read the details of each update.

Related Work

Related work for di.me can be found in many different aspects. Social network mash-up tools are a fast evolving field with some prominent examples. Diaspora [Diaspora] providing the possibility of hosting personal data on a private server shares some general ideas with di.me. Other approaches integrate social networks into the web browser. Examples are [Flock] and [Rockmelt]. Personal information manager cover another aspect of di.me, a technology well established, however typically restricted to email, chat and calendar functions. The di.me functionality of mining a personal sphere, in terms of a semantic desktop search, is based on related work resulting from the NEPOMUK project [Groz07].

Concerning the UI-aspects of di.me a survey to the general topic of augmented identity can be found in [Herm09]. Related work concerning visualization of social networks was done by [Fre00]. Visualization of timelines and personal history has been discussed by [Plai98] with focus on medical logs. Fundamental considerations about scaling and zooming within a timeline can be found in [Bade2004].

Summary

Main goal of di.me userware is to establish a tool for integration of personal information from local sources as well as from social network platforms into the personal information sphere. For this information about persons, their behavior, their content and their social structure is merged into a semantic model, where it can be browsed, searched and managed. The di.me userware client supports the user, when accessing his personal information sphere, giving him a powerful tool to control his personal data. For further support the di.me system gives recommendations about restructuring groups and categories and sharing information with interested contacts. To give orientation to the user, an easy to understand, but rich visualization of information and relations is described. A user-interface concept illustrates, how a user can navigate through his information sphere and which artifacts support managing it.

This paper described the basic concepts of dime and gave insight into the navigation of the di.me user-interface prototype for the mobile platform. Visualization aspects have been discussed on the example of visualizing the personal history on the desktop platform.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union Seventh Framework Program (FP7/2007- 2013) under grant agreement n° 257787

REFERENCES

[Fre00] Visualizing social networks; Freeman, L.C.; Journal of social structure, 2000

[Bloc10] E. Bloch, The 2010 Social Networking Map, http://www.flowtown.com/blog/the-2010-social-networking-map

[Diaspora] Diaspora project ; http://blog.joindiaspora.com/what-is-diaspora.html

[Rockmelt] Rockmelt, social media web browser; http://www.rockmelt.com/

[Meek10] Internet Trends - Presentation from CM Summit; M. Meeker, Scott Devitt, Liang Wu; Morgan Stanley, 07.06. 2010

[Herm09] Challenges for User Centered Smart Environments; Fabian Hermann, Roland Blach, Doris Janssen, Thorsten Klein, Andreas Schuller, Dieter Spath; HCII 09, 2009

[Schu09] Mobile Soziale Netzwerke : Interaction Patterns zur Fusion realer und digitaler Welten; Schuller, Andreas; Kniewel, Romy et. al.; Berichtband des siebten Workshops des German Chapters der Usability Professionals Association e.V., Stuttgart 2009, S. 184-188.

[Plai98] An information architecture to support the visualization of personal histories; C. Playsant, B. Schneiderman, R. Mushlin; Information Processing & Management, Elsevier 1998, p. 581- 597

[Bade2004] Connecting time-oriented data and information to a coherent interactive visualization; R. Bade, S. Schlechtweg, S. and Miksch; Proceedings of the SIGCHI conference on Human factors in computing systems, 2004, p. 105-112

[Groz07] The nepomuk project-on the way to the social semantic desktop; Groza, T., Handschuh, S. et al.; Proceedings of I-Semantics, 2007, p 201-211