The Personal Adaptive In-Car HMI: Integration of External Applications for Personalized Use

Sandro Rodriguez Garzon[1] and Mark Poguntke[2]

[1] Daimler Center for Automotive Information Technology Innovations,
Berlin, Germany
sandro.rodriguez.garzon@dcaiti.com
[2] Daimler AG,
Ulm, Germany
mark.poguntke@daimler.com

Abstract. We describe an approach for integrating non-automotive applications into in-car-entertainment systems while taking account of manifold personalization capabilities within a mobile environment. Adaptive user interfaces are generated for external applications using well-known interaction and personalization concepts. The interaction concepts are defined via state-based interaction models and utilized for the integration of various applications in order to guarantee a common look and feel. Context-aware adaptations of the user interfaces are achieved by supporting the process of gathering an augmented user model with a personalization concept in form of personalization guidelines. We present and discuss an exemplary application for a personalized, safe in-car HMI that automatically adapts to the targeted design and interaction concept as well as to the personal needs of the user.

Keywords: personalization, adaptation, user modeling, user interfaces, interaction modeling, model-based development, automotive apps

1 Introduction

The American Dialect Society has chosen "App" to be the word of the year 2010 [18]. Apps are mobile applications that can be downloaded and installed on high-performance smartphones. These apps are more and more used for daily tasks and often use content from online sources. The individual extent of usage, however, strongly depends on the individual user. One user may need only one or two apps helping him getting the latest news or the wheather forecast for tomorrow. Another user may use hundreds of applications and organize important parts of their daily life around smartphone apps and web applications. What all groups of users have in common is the predictability of their behaviour in similar situations [4, 16]. They may use single functions of the same application again and again depending on their respective personal context of use.

In an in-car environment it is required to have minimized driver distraction in the human-machine interface (HMI) for infotainment applications. Furthermore,

the use of mobile devices while driving is prohibited by law in most countries. A need for seamless integration of external applications to the in-car environment arises considering the amount of possible apps a driver may want to use in their daily life.

We propose a system supporting the integration of external applications to the in-car HMI and supporting the predictability of user behaviour by adapting the HMI according to different repetitive situations. We propose to use abstract interaction models and extend these with elements for personalization. This allows to use the same abstract model as a basis for different design and interaction concepts.

2 Related Work

The above presented situation implies two aspects of user interface engineering. An approach for user interface generation is needed. In addition, the process has to support user interface personalization.

2.1 User Interface Generation

Generating user interfaces from abstract interaction representations requires an approach for abstract interaction modeling and respective transformation processes. Model-based development using a respective modeling language holds the advantages of reusability, readability and easier adaptability of user interfaces amongst others [9][7]. A common basis to start with user interface specification is a task model. Concur Task Trees (CTT) [11] provides a notation for task models that can be used to derive user interfaces in subsequent steps. The User Interface eXtensible Markup Language (UsiXML) [15] describes a modeling and transformation approach from abstract to concrete user interfaces based on the Cameleon reference framework [1]. In recent years, several approaches motivate the use of the Universal Modeling Language (UML) [10] for user interface modeling. UML has been used for software engineering processes for many years with different established modeling tools. Also, several proprietary tools for model-based development exist for user interface specification, simulation and the automated generation of code or final user interface descriptions [2] [3] [15]. However, the generation of a respective final user interface from abstract models at runtime is not covered by these approaches since they focus on design time aspects.

2.2 User Interface Personalization

The adaptation of user interfaces based on the analysis of its historical use is applied in a lot of application areas. Inspired by Mark Weiser's vision on ubiquitous computing [17], Ma [8] introduces a smart home environment that observes the resident behavior in order to automate common tasks. Using a case-based reasoning approach the smart home environment is able to adjust a TV channel

or the air conditioning depending on the residents preference within a certain context (e.g. time). Coutand[5] presents a Call Profile Service that adapts its behavior based on a detected user preference at a certain location. It observes the way the mobile phone user accepts incoming calls and adjusts the profile accordingly. A similar mobile phone application for automatic profile selection was described by Schmidt [14]. Schmidt furthermore uses the knowledge about the user's activity (walking or stationary) and the light conditions to adjust the font size within a notepad application. The difference to the former approaches lies in the fact that Schmidt's solution does not consider the user specific preference. Predefined rules are utilized to detect the current context and to adapt a device in a "one-fits-all" manner. The results of a user study presented in [4] indicate that users like to arrange the icons of their mobile phone applications by relevance and the current activity. Since relevance is a user specific preference, the prototype needs to log every rearrangement of each user and the corresponding context in order to propose user-centric and context-dependent icon arrangements.

3 Personalization of Generated User Interfaces: Use Case

In this paper we focus on integrating external applications or services into the in-car head unit while providing the possibility to personalize. We introduce the mobile application called *Supermarket Guide*. Its purpose is to provide information about nearby supermarkets and their current offers. This external application can be integrated to the in-car head unit. It may come from an external device or from an online source. The application will appear as an additional entry within the application line next to preinstalled applications like Navi and Phone. Figure 1 shows screenshots of the supermarket guide that was automatically integrated to the head unit based on an abstract model and a transformation process. The main view appears underneath the application line as illustrated in Figure 1. For each entry within the supermarket list, the user is able to view either how to approach a supermarket or a list of supermarket-specific offers.

In order to describe the personalization that happens within the HMI of the Supermarket Guide, we will introduce a sample usage situation that is illustrated in Figure 2 . Consider a user that lives in the city of Stuttgart and commutes every workday to the city of Ulm. After work, the user always drives back towards his home in Stuttgart. Sometimes - in case the user needs to buy food - the user has a stopover at one of his preferred supermarkets along the route: supermarket A in Ulm or supermarket B in Stuttgart. Immediately before approaching junction M, the user starts the Supermarket Guide and checks for interesting offers of supermarket A in order to decide whether or not to make a stopover in Ulm. If the user isn't satisfied with the offers he continues and checks the offers of supermarket B before approaching junction N. Otherwise, the user will stop at supermarket A.

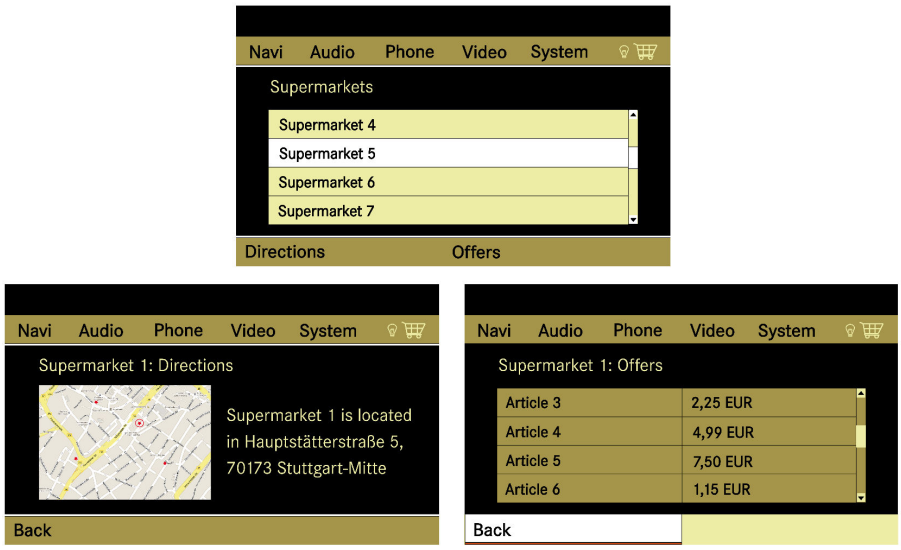

Fig. 1. Supermarket Guide

After repeating the same interactions regularly, a light bulb symbol will appear next to the icon of the supermarket application each time the car approaches the road before junction N or the road before junction M. The light bulb indicates that there is a personalized adaptation of the interaction. If the user starts the Supermarket Guide while the light bulb is visible, the supermarket application will behave differently. The application will not open the main view but a supermarket-specific offers view: either the offers of supermarket A while driving on the road next to junction M or the offers of supermarket B while driving on the road next to junction N. Generally spoken, the icon signalizes that the HMI of the Supermarket Guide has a user-specific behavior as long as the light bulb remains visible. That is, the HMI adapts its behavior according to the users needs be means of observing real world experiences.

4 Abstract Modeling for User Interface Generation

We use the roles of an application developer and an interaction designer for the abstract modeling approach. An application is developed by an application developer including a functional application interface consisting of a class diagram with attributes and operations. An interaction designer uses this interface to create an abstract interaction model using UML state charts to describe user actions and corresponding system reactions. A transformation program uses the model and generates a user interface compliant to the respective automotive HMI concept. For the transformation process rules have to be implemented mapping the abstract model elements to user interface elements for a specific concept. [12]

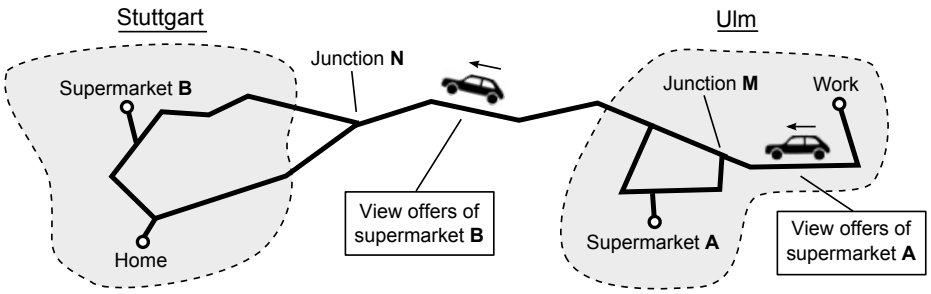

Fig. 2. Use case overview: Driving and supermarkets app

Considering the *Supermarket Guide* an abstract interaction model consists of a list representation with two selectable options for each list entry. An abstract model is illustrated in 3. The *List Overview* state contains the presentation of the supermarket entries. *Entry Detail 1* and *Entry Detail 2* present the respective information depending on the selected option, *Offers* or *Directions*, for the selected entry. A class diagram including required variables and operations as well as the interaction model using these defined elements has to be provided along with the application to be integrated which is in the given use case the *Supermarket Guide* application. Since the focus of this paper is the general approach to extend abstract interaction models with personalization adaptions we use only the illustration of a general model as in 3. Details on the modeling approach can be found in previous work [12].

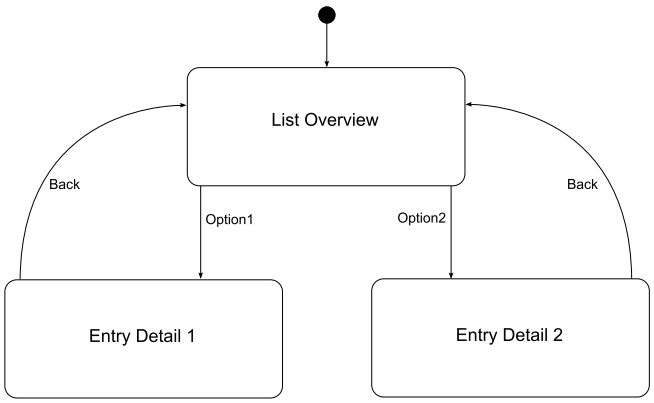

Fig. 3. General abstract interaction model for a list representation with two selectable options for each list entry

5 User Interface Personalization

In order to support user interface personalizations as described in chapter 3 we introduce a real-time component that has to deal with the process of detecting regular user behavior within similar environments. Thus, the main challenge of this so called *personalization component* lies in observing the real world user behavior and inferring the right moment to notify the HMI about an upcoming user preference. The correlation between the user preference and the real world user behavior will be stored within an augmented user model.

In case of the Supermarket Guide, the personalization component observes the way the user interacts with the application-specific HMI at different locations. If the personalization component detects a user-specific regularity within a certain region it will notify the HMI about the user preference every time the user enters the region. This information can be processed by the interface in order to provide personalized functionality.

5.1 Personalization Component: In- and Output

Considering the personalization component of the Supermarket Guide as a black box, it is necessary to notify the black box whenever the user opens the supermarket offers view. This information gives the black box a first glance onto the amount of times the user likes to view the supermarket offers. Enriching the notification with the supermarket-specific name allows the black box to detect which supermarket-specific offers are favored by the user. Additionally, the user prefers to query different supermarket-specific offers at different locations. Therefore, the black box also needs to get notified about the environment in which the user interaction is happening. Hence, different digital traces need to be observed by the black box in order to detect and output regular environment-specific user preferences.

The content of the different digital traces can be distinguished by its origin. The event of "opening a supermarket offers view" as well as the supermarket-specific name are declared to be application specific knowledge. In contrast, the environment-specific information - in our use case: the road - is declared to be a system-specific knowledge. This kind of knowledge is provided by the system independently of the type of applications that are preinstalled on the head unit.

All events originating from human-computer interactions are called *interaction events* regardless of its origin. The black box itself does not differentiate between events having different origins. Thus, the task of detecting regular situation-dependent user behavior is based solely on the processing of a single stream of interaction events. In other words, the personalization component provides two interfaces: An input interface for interaction events and an output interface for notifications about the occurrence of a situation that will likely contain a regular user interaction.

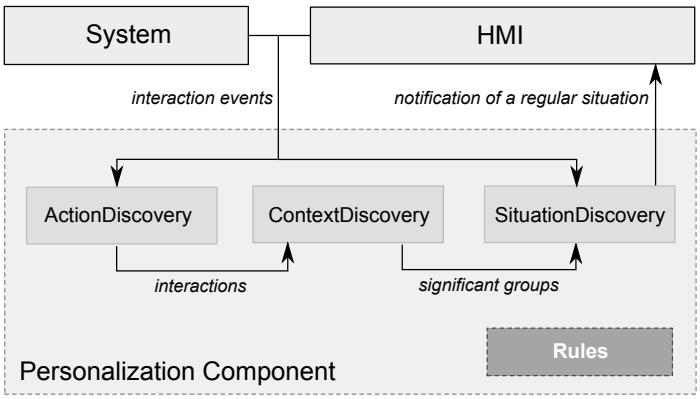

Fig. 4. Overview: Personalization

5.2 Personalization Component: Internals

Since every application has its own user interface with different personalization scenarios it is necessary to let an expert define the way the personalization component should feed its augmented user model. The expert, known as the *personalization designer*, configures the personalization component by means of personalization guidelines in form of a scenario specification file . Therein, each configuration step is closely coupled with a certain subprocess of the personalization component. In the following, every subprocess will be introduced briefly with its required configuration steps.

Action Discovery Firstly, each incoming interaction event will be passed to the *action discovery* subprocess. The purpose of this process is to detect sequences of interaction events that are supposed to be relevant concerning the specific scenario. The concrete occurrence of such a sequence is called *action*. Considering the Supermarket Guide, an action contains the event that is generated by the HMI in case the user enters the supermarket offers view. But such an action is only a valid detected action if the user remains at least 10 seconds at the supermarket offers view. This additional condition restricts the action discovery process to detect actions that are originally intended by the user and not executed by mistake. These conditions are defined by the personalization designer by means of a *temporal event pattern* that describes the relevant action in a general way. The result of the subprocess is an action consisting of a sequence of one or more concrete interaction events.

Context Discovery Each detected action will be received by the *context discovery* subprocess. This subprocess groups actions that happened within similar environments. Considering the Supermarket Guide, two actions are grouped in

case the car was driving on the same road at the moment the action was detected. A group is declared as *significant* if it contains at least a minimum number of actions. Only significant groups are contributing to the augmented user model as valid regularities. This becomes necessary in order to ignore user interactions that are executed rarely. The subprocess is guided by the personalization designer through the specification of the environmental factors which are relevant to the scenario.

Situation Discovery The last step - *situation discovery* - deals with processing each significant group in order to detect similar situations. Considering the Supermarket Guide, the user expects the HMI to personalize itself in case he opened the offers view of a specific supermarket several times at the same location. Since the input stream of interaction events also contains location events it is straightforward to consider the stream of interaction events in conjunction with the regularity found in the previous subprocess. The personalization designer decides by means of a temporal event pattern how both information streams are joined to detect the desired situation. For the Supermarket Guide, the interweavement of both streams is fairly simple: Look for location events with a road similar to the road of a significant group. If a situation is found the personalization component will notify the HMI. The reader is referred to [13] for a more detailed example.

5.3 Extension of the Interaction Model

The HMI in turn is in charge of providing the application-specific interaction events and to execute a personalization in case it is notified about a similar situation. Since the HMI is generated based on an interaction model, it is necessary to extend the interaction model in order to deal with the personalization task.

Considering the HMI of the Supermarket Guide, two interaction events needs to be generated: "Opening a supermarket offers view" and "Leaving a supermarket offers view". Since we have already specified a state for the supermarket-specific offers view it is straightforward to define entry and exit actions to fire the required events. We assume that all states implicitly fire entry and exit events. Thus, it is not necessary to explicitly define these kind of actions.

As mentioned in Section 3, the personalization of the Supermarket Guide will be announced by the light bulb symbol appearing in the application line next to the shopping cart icon. This personalization icon is an internal feature of the system and is reused for every application that offers personalization capabilities. In order to jump directly into the supermarket offers view at the start of the application, it is necessary to add a conditional transition immediately after the entry point. Figure 5 illustrates the modified interaction model with the extension colored in blue. It forces the HMI to check whether the personalization component announced a personalization. If there is no announcement the HMI will show the main view. Otherwise, the HMI will show the supermarket offers view of the announced supermarket. A second conditional transition is added

in order for the interaction model to be reusable for other applications with different personalizations.

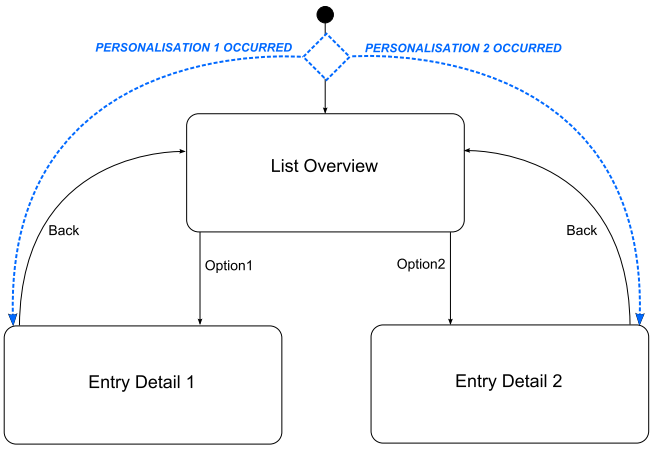

Fig. 5. Adapted State Diagram for Personalization

6 Discussion

Up to this point, the interaction model of the Supermarket Guide was used to generate the application-specific HMI. Hence, the external application gets a well-known HMI concept rather than providing its own. The benefit of having an abstract interaction model becomes obvious in case a second application needs to be integrated. Consider the integration of the social networking platform facebook [3] [6] using the same general interaction model. Figure 6 illustrates a proposed HMI concept for an in-car facebook application. Since the main structure of the HMI with its main view and two detail views matches with the HMI of the Supermarket Guide, it makes sense to reuse the interaction model with a different parametrization. Substituting the data sources and modifying the specification for the personalization component are the only modifications that are necessary. The latter modification arises from the fact that the facebook application will be personalized in a slightly different manner. The personalization decision to enable direct jumps into one of the detail views does not depend upon a location information provided by the car system but from the abstract location that is assumed to be provided by the facebook service. Out of the perspective of the personalization component it does not matter if the interaction events are generated by a system component or by an external source. Even multiple

[3] Facebook is a trademark or registered trademark of Facebook, Inc.

sources like location and the kind of persons within the car can be used to decide whether to show the wall or info views.

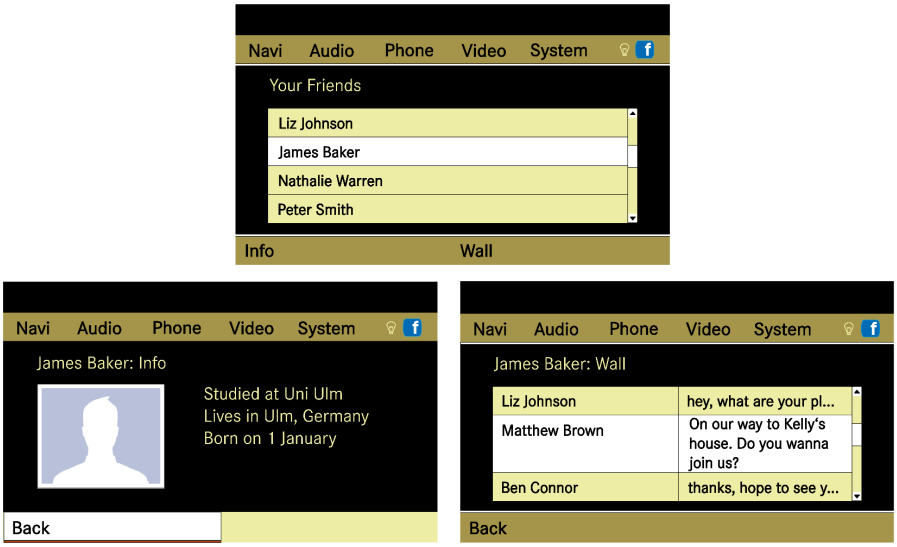

Fig. 6. Simple Facebook Application based on the general abstract interaction model

Thus, the major benefit of the presented approach lies in the unobtrusive integration of external and adaptive applications by using well-known interaction concepts. New adaptive applications can be developed faster because the automotive-specific HMI concepts are known in advance. The developer isn't in charge of dealing with the special issues of user interface development within the automotive environment like distraction reduction. Additionally, all potential adaptations of the HMI are also known in advance since they are an integral part of the interaction model. On the one hand, the user has a clear understanding about the adaptation that might happen. On the other hand, the application developer in conjunction with the personalization designer is still able to to specify the generic conditions describing the moment an adaptation might occur.

It remains to be seen whether it is practicable for a personalization designer to specify all the personalization conditions in a generic form using a temporal event pattern as described in Subsection 5.2. In our approach, the interdependence between different digital traces must be known a priori to properly configure the subprocess of the personalization component. Sometimes it is not predictable which environmental factors are influencing a user's intention of changing for example the radio station.

Another question arises concerning the way personalization is detected. In some cases it might be reasonable to integrate the personalization detection

within the application itself. For example, the automatic reordering of application-specific list entries depending on the user interest is an application-specific personalization. It does not make sense to share these kind of detected regularities with other applications and thus do not require a system-wide augmented user model. But, as stated above, one of the main benefits of user interface generation based on abstract models is the reuse of interaction models for different applications. Thus, extending the interaction models in order to handle personalizations might affect multiple applications. A system-wide personalization component would be able to detect a regularity in an application A and could provide this information to an application B. The user interface of application B would then adapt itself since it is originally based on the same interaction model. Even newly installed applications would instantly profit from regularities that were detected so far.

7 Conclusions and Future Work

We presented an approach to consider user-centric adaptation mechanisms within the definition of state-based interaction models. It was demonstrated that a properly specified interaction model together with a clearly defined personalization is sufficient to integrate non-automotive applications into the automotive environment. Particularly, the special requirements for the automotive environment enforced us to develop a guided personalization process in order to guarantee a predictable and comprehensible HMI behavior. Considering our approach, an application developer is required to select an adequate interaction model which is provided for example by a car manufacturer. The personalization remains still flexible and can be configured by the application developer together with the personalization designer via the configuration of the personalization component.

A working prototype for the HMI generation process [12] as well as a prototype for the personalization concept [13] were demonstrated previously. In future work, these implementations have to combined to further evaluate the presented approach.

References

1. Cameleon project homepage. Accessed on 31 January 2011.
2. Eb guide: Integrated tool chain for developing multi-modal hmis. Accessed on 17 April 2011.
3. Multimodal teresa - tool for design and development of multi-platform applications. Accessed on 31 January 2011.
4. M. Böhmer and G. Bauer. Exploiting the icon arrangement on mobile devices as information source for context-awareness. In *Proceedings of the 12th Int. Conf. on human computer interaction with mobile devices and services*, pages 195–198, 2010.
5. O. Coutand, S. Haseloff, S. L. Lau, and K. David. A Case-based Reasoning Approach for Personalizing Location-aware Services. In *Workshop on Case-based Reasoning and Context Awareness*, 2006.

6. Facebook Inc. `http://www.facebook.com`, Accessed on 15 April 2011.
7. K. Luyten. *Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development.* PhD thesis, Transnationale Universiteit Limburg, 2004.
8. T. Ma, Y.-D. Kim, Q. Ma, M. Tang, and W. Zhou. Context-aware implementation based on cbr for smart home. In *IEEE Int. Conf. on Wireless and Mobile Computing*, pages 112–115, 2005.
9. G. Meixner. *Entwicklung einer modellbasierten Architektur fr multimodale Benutzungsschnittstellen.* PhD thesis, TU Kaiserslautern, 2010.
10. OMG. Uml 2.2 superstructure specification, 2009.
11. F. Paternò, C. Mancini, and S. Meniconi. Concurtasktrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Int. Conf. on HCI*, 1997.
12. M. Poguntke and A. Berton. One application, one user interface model, many cars: Abstract interaction modeling in the automotive domain. In *Proceedings of the 3rd Workshop on Multimodal Interfaces for Automotive Applications*, 2011.
13. S. Rodriguez Garzon and K. Schütt. Discover significant situation for user interface adaptations. In *Proceedings of the 3rd Workshop on Multimodal Interfaces for Automotive Applications*, 2011.
14. A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. d. Velde. Advanced interaction in context. In *Proceedings of the 1st Int. Symposium on Handheld and Ubiquitous Computing*, pages 89–101, 1999.
15. J. Vanderdonckt, Q. Limbourg, B. Michotte, L. Bouillon, D. Trevisan, and M. Florins. Usixml: a user interface description language for specifying multimodal user interfaces. In *Proceedings of W3C Workshop on Multimodal Interaction WMI*, pages 1–7, 2004.
16. A. Vetek, J. Flanagan, A. Colley, and T. Kernen. Smartactions: Context-aware mobile phone shortcuts. In *Proceedings of Int. Conf. on Human-Computer Interaction*, volume 5726 of *Lecture Notes in Computer Science*, pages 796–799, 2009.
17. M. Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, pages 75–84, 1993.
18. B. Zimmer, G. Barrett, and A. Metcalf. *"App" 2010 Word of the Year, as voted by American Dialect Society.* American Dialect Society, 2011.