Please quote as: Hoffmann, H.; Leimeister, J. M. & Krcmar, H. (2010): Tool support for the participatory design of end user oriented applications in the automobile. In: 18th European Conference on Information Systems (ECIS), Pretoria, South Africa.

TOOL SUPPORT FOR THE PARTICIPATORY DESIGN OF END USER ORIENTED APPLICATIONS IN THE AUTOMOBILE

Hoffmann, Holger, currently working at: Kassel University, Nora-Platiel-Str. 4, 34127 Kassel, DE, holger.hoffmann@uni-kassel.de, at the time of research working at: Technische Universität München, Boltzmannstr. 3, 85748 Garching bei München, DE, holger.hoffmann@in.tum.de

Leimeister, Jan Marco, Kassel University, Chair for Information Systems, Nora-Platiel-Str. 4, 34127 Kassel, DE, leimeister@uni-kassel.de

Krcmar, Helmut, Technische Universität München, Chair for Information Systems, Boltzmannstr. 3, 85748 Garching bei München, DE, krcmar@in.tum.de

Abstract

Software is becoming more and more important to the automotive industry as a driver for innovation, thus the number of software-based systems in the automobile value creation constantly rises. Because of this trend, automakers increasingly focus on the development of end-user oriented functions. In order to be able to successfully offer such functions, the requirements of customers for those applications have to be met. The automotive sector, still oriented at developing the product "car" and organized in a strict hierarchy, lacks the means necessary to systematically elicit and analyze the customer requirements thus far. This article shows how a prototyping tool can contribute to the active integration of the customer into the development process during participatory design of applications. One central aspect of the prototyping tool presented here is that the prototypes using it can be integrated into the car and used along with applications that are already in series production. It is hence possible to conduct pilot studies of the application with representatives of its target customer group in the future environment. This enabling of customers to experience the future application allows car manufacturers to systematically elicit and analyze their users' needs. It also allows the collection and evaluation of innovative ideas by customers for new applications, as well as solutions for existing problems, for later integration into the companies' products.

Keywords: Automotive sector, participatory design, prototyping, tool, tool chain, evaluation.

1 INTRODUCTION

Software in the car has gained increasing importance as a driver for innovation and value creation in the automotive industry. As high as 90% of all innovations in and around the car are based today on innovations in electronics and software (Hardung et al. 2004, 203). Their share in the value creation reached almost 20% in 2002 and is expected to double by 2015; the software itself will then be contributing 128 bn. € to the value of a car (Hardung et al. 2004, 203, Mercer Management Consulting et al. 2004, 105-108). Novel software functions, e.g., in form of a mobile newsreader, aren't new concepts to the mobile/handheld market, but are new in the automotive industry and would allow manufacturers to fulfill their customers' rising needs for more information and comfort while driving (Tinham 2007, 40). But more importantly, they open the possibility for manufacturers to participate in the profitable business taking place after the sale – e.g., by offering or acting as a broker for value-added services (Becker 2002, 2).

In order to improve the success rate of such innovative functionality, a comprehensive knowledge of the requirements for such systems is mandatory. Along with technical and organizational requirements, which result from the automotive domain as the field of application, the future customers' requirements are especially important for these customer-oriented applications. However, the development processes in the automotive domain we encountered so far are strictly oriented at the development of a product (i.e. not software) and do not take the customer into account until very late in the process. In addition to that, the complexity of the future usage environment, the car, and usage situation, in the car and possibly while driving, further contribute to these challenges.

Our goal for this article is to present a tool that facilitates the systematic development of customer oriented software for the automobile by using prototypes as additional communication channels as proposed by Dix et al. (2004, S. 666f). In this participatory approach, we show how stakeholders from various disciplines are able to cooperate, and customers can be integrated into the requirements elicitation and development of concepts for solutions to specific problems early on. Over the past five years the design of the development process and the matching tool support have been conducted in cooperation with a major German car manufacturer. We present our findings on the integration of diverse groups of stakeholders into the creation of novel software for the automobile based on the application of this research in the field.

2 COOPERATIVELY WORKING ON THE REQUIREMENTS ELICITATION FOR CUSTOMER RELATED SOFTWARE

Having a complete and sound understanding of the users' requirements for a system is mandatory for its successful introduction to the market (Nuseibeh & Easterbrook 2000). In order to systematically build up such an understanding, it is especially important for interactive systems (in contrast to e.g., embedded software) to elicit the requirements by seizing future users' expertise in their field (Ehn 1993, Grudin 1993), ideally working directly in the future usage context (Dix et al. 2004, 458, Sharp et al. 2007, 321ff).

In the research concerning participatory design and human-computer interaction (HCI), approaches have been developed that allow the creation of solutions with multiple parties involved. Integrating the customer into the development process results in several advantages: on the one hand, the knowledge gap between user and developer (users often don't know the possibilities arising from available technologies, and developers are not completely aware of the future context in which the application should be used) is becoming narrower; on the other hand, the different perspectives of users and developers (the availability is the beginning of the applications usage for the customer, but the end of the development process) overlap further than before (see Mambrey & Pipek 1999, 2). Thus, end users and developers are able not only to elicit requirements and work on solutions together early in the

development process, but also often work through several iterations of building and rebuilding the application together (Dix et al. 2004, 466ff).

Prototyping an application – meaning the partial implementation of certain aspects of the application that are usable and thus can be evaluated – represents one possible approach towards enabling participatory design (Dix et al. 2004, 241ff, Sharp et al. 2007, 530ff). For so called "high fidelity prototypes", the final system is often built up iteratively in order to reach the aspired result. Using this approach, the implementation of artifacts designed to fulfill requirements that were elicited initially or derived from general conditions are evaluated in each step of the iteration, resulting in refined and extended requirements for the next iteration. In this way, a precise picture of the requirements of the participants and their ideas for solutions to problems encountered in the process is systematically developed. The theoretical background of such cooperative work is described by Dix et al. (2004, 666f) as the Cooperative Work Framework.

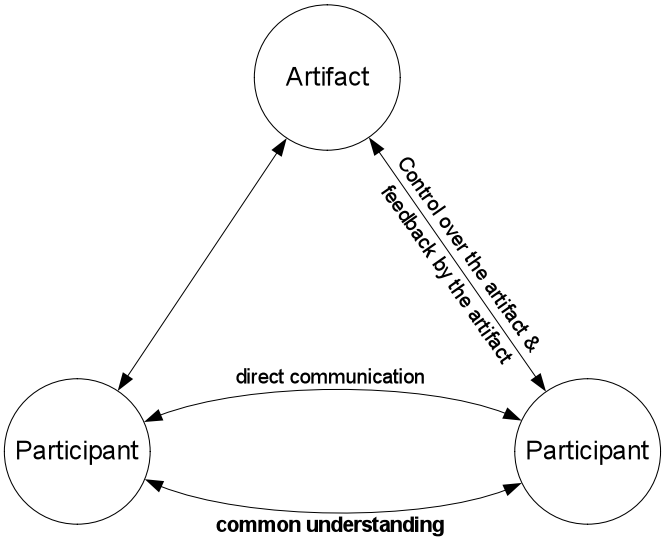

Figure 1. Cooperative Work Framework by Dix et al. (2004, S. 666f)

The essential point of this theory is that "artifacts" (manmade objects, e.g., prototypes) offer cooperating participants another way of building up a common understanding about the actual situation, in addition to the possibility of direct communication between participants on that topic. Hence, it is possible for developers of novel software based functions in the automobile to better understand the requirements of the customer as primary stakeholders by using realistic prototypes as a tool in addition to directly communicating the requirements. Additionally, observing customers while they use the prototypes in the scope of pilot projects in the future usage environment enables developers to also identify requirements that the users could not or did not express before (see Schwabe & Krcmar 2000).

3 DEVELOPMENT SUPPORT THROUGH A PROTOTYPING PLATFORM

The implementation of prototypes as artifacts to support the customer integration will result in rising expenses, as they have to be implemented from scratch. Thus, the main goal of the *Highly Integrated Modular Embedded Prototyping Platform* (HIMEPP) is to reduce the effort for developers – despite specialized (user-) interfaces found in the car. This is to be achieved by providing a standardized infrastructure and prebuilt base components, both allowing the developer to concentrate on the application logic of his system and still have a prototype ready for evaluation in the early phases of the development project. In contrast to existing (mostly iterative) rapid application development processes, the process this tool aims to support allows different stakeholders to cooperate using

specialized “workflows” for their respective domains and to iterate by re-running those in any order needed (please refer to (Hoffmann et al. 2007) for details).

3.1 Prototyping platform requirements

The goal of the design research presented in this article is to develop a prototyping tool (HIMEPP), that allows the creation and presentation of “high-fidelity” prototypes (see Sharp et al. 2007, 535) of novel software functions in the automobile. The concrete requirements from the application domain (automotive) and access to the field, in order to be able to put HIMEPP to use and evaluate it, are provided by a partner in the industry, one of the German car manufacturers. The analysis of the processes and requirements of the developers working on current innovation projects lead to the identification of different user roles for the tool, each with specific requirements. Those roles are specific to the developers in the project and are comprised to roles found in the literature about the design of interfaces supporting programmers (APIs, see (Henning 2007)) and to roles identified in the field (here: evaluation supervisors).

Platform users develop prototypes using the toolset – likely together with other parties like other developers or customers. For them the effort to realize prototypes should be as low as possible. The programming interfaces defined by HIMEPP ensure a similar usage of the infrastructure and thus a comparable programming style, disburdening collaboration in projects with multiple developers (Henning 2007). Prebuilt reusable functions help in reducing the cost of prototype development and improve reliability and quality of the prototypes created (Stritzinger 1997, 115ff, Szyperski 2002).

Platform architects can make changes to the prototyping platform and are responsible for updating and expanding the platform. To amplify the synergetic effects in the company, and improve the practicability and acceptance for the platform, HIMEPP has to be able to integrate existing tools currently in use at the company. In order to fulfill these requirements, HIMEPP has to be extendable by new elements and adaptable to new conditions. Additionally, existing parts of the platform have to be easily maintainable and reusable (Balzert 2000, 1093ff & 1102f).

Evaluation supervisors conduct prototype-based evaluations, in which *evaluation participants* assess novel functions and applications based on their prototypical implementation. In order to obtain the most significant results, the prototypes should be run in the car. This integration allows the supervisors to conduct the evaluations directly in the future usage environment and enables them to conduct pilot studies for novel applications directly in the field (Dumas 2003, 1099, Karat 2003).

3.2 Design principles for the prototyping platform

The requirements identified during the initial analysis correspond with the goal of software development oriented at reuse, and thus suggest the component-oriented design and implementation of HIMEPP (see Szyperski 2002). The design elements comprising a component-oriented architecture and their interrelationships are described by Gruhn & Thiel (2000), as depicted in Figure 2.

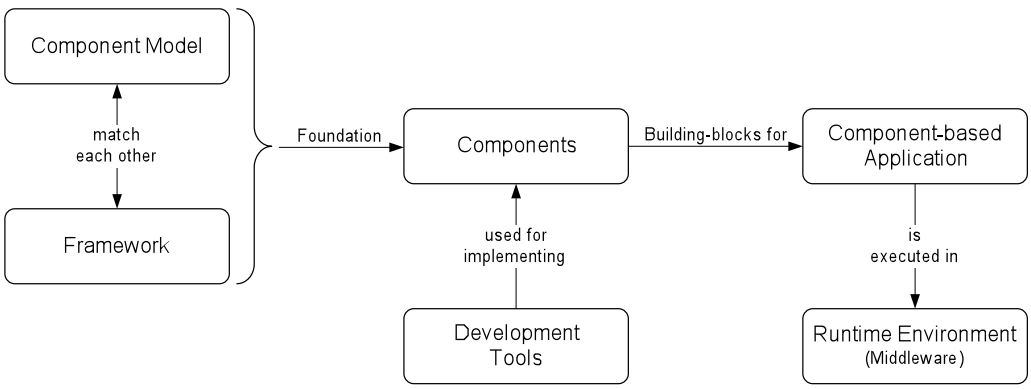

Figure 2. Elements of component-oriented architectures (Gruhn & Thiel 2000)

A *software component* is defined by Szyperski (2002) as a self contained, reusable logical unit that encapsulates certain functionalities and makes it usable through public interfaces. Components are defined for a context of application, but exist independent of concrete applications. In a *Component Model* the standard describing the implementation of component based applications is defined. This comprises all areas of the development, from the definition of interfaces to the connectivity of components and their composition into applications and their deployment (Gruhn & Thiel 2000, 21, Weinreich & Sametinger 2001, 37-45). It is the objective of the *Framework* to enforce the usage of domain-specific principles, e.g., the manner in which functions are split up into components or the uniform naming of methods, etc., by the components' developers. As a result, standardized components evolve that can be easily interconnected in the environment set by the framework, and thus can be used to compose applications. This allows, e.g., centralizing the management of information exchange by using events, and thus ruling out early on many commonly made mistakes (Stritzinger 1997, 117f, Szyperski 2002, 425f).

3.3 Implementation of the prototyping platform

Following the model by Gruhn & Thiel (2000, 20-23) as depicted in Figure 2, there are four design elements for the component oriented architecture: the *Component Model* used, a matching *Framework*, a *Runtime Environment* for the components – and for applications composed from them – and *Development Tools* used for creating components. In addition to these four elements, HIMEPP offers an additional repository of prebuilt Base Components which provide user interfaces, interfaces to the car, and other functions as a foundation for new prototypes. Figure 3 shows a concrete instance of the component-oriented architecture for HIMEPP. Wherever possible, definitions, models and tools already used within the company are used here in order to reduce the training period for using the tool, to enhance reusability of existing solutions, and hence to assist the overall acceptance of HIMEPP. This is why a solution already used in the automotive domain was sought-after for the component model; the middleware connected with that component model and also with the development tools fit for the development task. These solutions then should be adapted by designing a specialized framework and newly created base components for the purpose of prototype generation.

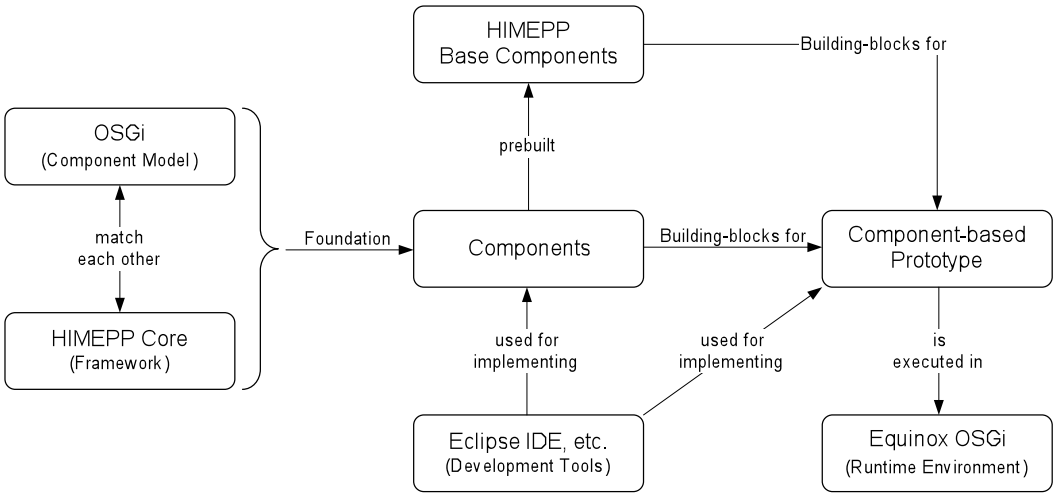

Figure 3. Design elements of the prototyping tool (based on Gruhn & Thiel 2000)

The selection of a suitable *Component Model* is based on its prior usage in the domain and the possibility to integrate existing solutions developed at the automotive manufacturer. The OSGi Platform, which specifies a lightweight dynamic service model for Java, meets both requirements and has thus been chosen as the underlying *Component Model as well as Middleware* (runtime environment for components). The OSGi technology is not only very well suited for component

oriented development of dynamic applications, it is increasingly often used in the automotive domain, e.g., at Audi, BMW, Daimler, Ford, Siemens VDO (now Continental) and Volvo (OSGi Alliance 2007a, Saad 2003). The Eclipse IDE is used as a *Development Tool*, since it is based on OSGi itself and uses the same Middleware later used for running the prototypes. This enables platform users as well as platform architects to quickly test new code directly on the development machine; inside the IDE, the time-consuming copying to the target machine with its runtime environment and testing/debugging there is not necessary. Useful functions that are not offered by the Eclipse IDE directly, e.g., for the deployment of applications to the target machine in the car, are supported by additional tools.

Special requirements for the prototyping tool derived from the application domain are satisfied by the specifically tailored *Framework*, called HIMEPP Core, and the prebuilt *Base Components*. HIMEPP Core provides the architectural framing applied to all components, so that all components complying with it can be combined. This architectural framing is based on the design guidelines for frameworks and their interfaces (see Cwalina & Abrams 2007, 29, Henning 2007). Some of the requirements imposed by the design guidelines are met by using components already defined by the OSGi Platform (see OSGi Alliance 2007b):

- OSGi Declarative Services are used to describe the offered functionality as well as dependencies between components. This makes it possible to dynamically start and stop individual components as needed and thus use the limited resources on the target machine optimally.
- The OSGi Event Admin manages the synchronous or asynchronous communication between components. This enables the developers to create strictly modular applications, where each of the used components can easily be updated or exchanged.
- The OSGi Log Service helps debugging applications during system development – one of the challenges of component oriented programming – but it can also be used to track users' activities during the prototype evaluation.

HIMEPP Core offers a central interface for all HIMEPP compatible components to use in order to standardize the usage of those predefined services, resulting in the OSGi components to be used in a similar manner (Henning 2007, Williams 2001). The HIMEPP framework also simplifies the correct usage of the different functions and ensures their consistent usage across different HIMEPP projects (Henning 2007). In addition to this standardized interface towards the OSGi platform, HIMEPP Core also offers functions helpful for prototyping which are not at all or insufficiently defined and implemented by the OSGi platform.

As the framework element of the architecture, HIMEPP Core also defines design principles for components, i.e., states how components are to be designed and supports the developers in building them. This ensures that components designed for the use in this framework can be combined into applications, but it does not directly help building applications. This assistance in building applications is the benefit of the HIMEPP Base Component repository. This repository contains prebuilt components that provide HIMEPP developers with frequently used functions or interfaces (to the user or the car) for inclusion into their own development projects. As one option, Boone (1999, 199f) proposes to *analyze the daily operations* to indentify such essential components that are based on systems or applications currently available. By using this identification of required basic functions in the prototyping tool the base components can be derived, classified and grouped. In addition, the functions needed to satisfy initial requirements (e.g., the inclusion of existing solutions of the company with the new tool) extend this set of components further. Overall, six groups of functions are identified in this way, each containing two to four base components.

Figure 4 shows which applications (rows) use which functional building blocks (columns), as well as the source of the data used by the application. The applications are divided into those representing the current software version in cars available today (top block) and those applications coming from current research projects (bottom block). The functional building blocks are divided into interfaces for

system output (left block) or system input (middle block), and interfaces for data processed or displayed by the application (right block).

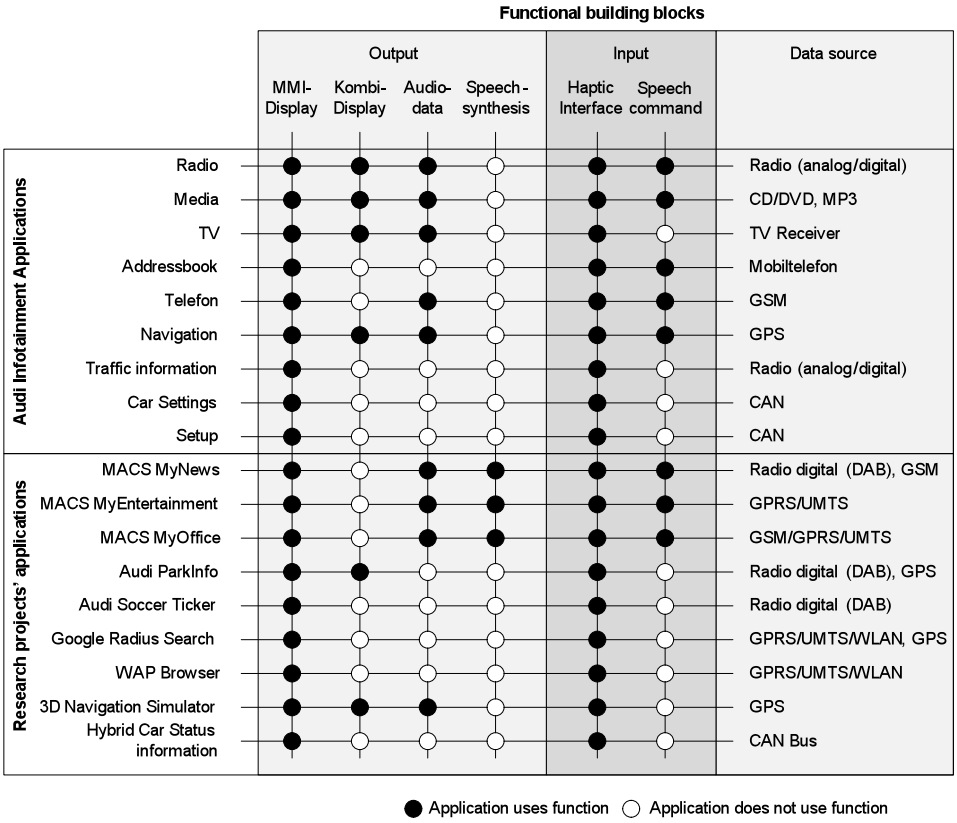

Figure 4. Identification of base components based on current applications (own figure)

- Components for audio- and speech interfaces provide user interfaces for prototypes. The group contains a component to play audio files, one that can read texts using speech synthesis software and one component recognizing commands based on a recognition grammar. These components enable drivers to use the applications via voice; they neither have to take their eyes off the road to pick up information, nor do they have to take their hands off the steering wheel in order to operate the system.
- Components for the haptic interface also provide a user interface. The functionality is split up into two distinct base components: the first component realizes the communication with the controller device integrated in the car; the second component interprets the communication semantically. This semantic component enables developers to evaluate different controller concepts without the need to chance the application logic of the prototype.
- The group of components for bus communication in the car is comprised of the components needed for accessing the car's CAN bus system: one component to send and receive CAN signals in the platform, and a pair of components for selecting relevant CAN signals and interpreting them semantically. This enables developers to incorporate the car's state into their application and also alter the state from the application.
- Interfaces for mobile data transfer or detection of the local context (e.g., position) are summarized as components for mobile services. Currently, a component for using a 3G GSM modem, enabling the downloading of Internet content and a component for receiving GPS signals allowing the localization of the car, are readily available for HIMEPP users.
- Components that allow the integration of external components (e.g., existing software in the company) are grouped as external component interfaces. HIMEPP offers an interface to the

graphical user interface prototyping environment in use by several departments to simulate screens that are compliant to the corporate standards.

- The final group consists of service and support components that ease the development of a prototype. They are comprised of components for use in an application (e.g., loading system properties, displaying a splash screen), as well as components used during development and testing (e.g., simulators for the haptic interface or simulation of GPS routes).

The base components described above represent a construction kit containing the most important features for the development of a prototype for the car. Developers can thus focus on developing their application logic, and thus the redundant implementation of interfaces, e.g., towards the user, can be omitted. The respective user interface semantics components additionally provide the usability concept compliant to the corporate standard. However, adhering to the principle of the separation of concerns while breaking up the functions into components results in a set of modular components that can easily be exchanged, e.g., for evaluating a different usability concept, without the need of changing the application logic of the prototype itself.

4 EXAMPLE: DEVELOPMENT OF AN INTERACTIVE APPLICATIONS FOR MULTIPLE TARGET GROUPS

The example presented here is demonstrating the development of a prototype for a complex interactive application. For the application we chose a "brain training" game, that should be usable by adults as well as children in kindergarten for two reasons. First this allowed us to test the ability of the tool to cope with uncertainty due to changing requirements coming from two very different user groups. Second, games are highly interactive, allowing us to put most of the prebuilt base components to use. As this is one of the first attempts to use the tool no real end user involvement took place in the development process itself. However, possible characteristics of the application were discussed in a small heterogeneous team using the tool and early prototypes. Additionally representatives of different target groups (regular drivers, children in kindergarten age) had a chance of seeing the final prototype and giving feedback on properties like usefulness and ease of use.

4.1 Scenario: "HighIQ" as an entertainment application

"HighIQ" is an application designed for use in traffic jams or, in a different stage of expansion, for rear seat entertainment. The main goal of the application is the playful training of cognitive abilities as a diverting activity in the car. In order to achieve this, three "mini games" which are tailored to the target audience, are used to train a certain aspect at a time. The HIMEPP platform comes to use early to let the users decide which mini games they would like to see, based on mock-ups of the actual games. Some of these ideas were first implemented rudimentarily to test their fitness for use in the car (with a small screen and mainly voice based interaction). Based on these tests, different combinations of mini games were then combined to applications; the combination receiving the best results in user evaluation then was defined as the usage scenario and completely implemented. Hence, HIMEPP was used to build multiple prototypes that could be evaluated early in the development process, limiting the more complex and costly development of more sophisticated prototypes to the most promising ideas.

The adult version of "HighIQ" described above consists of three parts. Part one tests the Stroop–Effekt (1935), by writing the name of a color using a different color for the text and asking the user to tell which color the text is written in. The complexity lies in the fact that differing color names and text colors result in two interfering stimuli. In the second part, the users have to solve easy mathematical equations as quickly as possible, thus using a brain activity higher than trying to solve hard questions slowly (Kawashima 2005). The user's task in the third part comprises selecting one of four objects in a list based on a picture of the shadows cast by the object in question, thus training the visual cortex of

the brain (see Velez et al. 2005). After completing the three parts, the results achieved (i.e., the percentage of correct answers) by the user are presented in a spider web diagram.

Since this application is too complex for children in kindergarten age, and thus would not be fun for them to play, some adaption is necessary. While sticking to the concept of multiple mini games, the second part (mathematics) is discarded and the third part (3D shadows) is exchanged for a different game. Instead of the shadows cast by an object, a comic book character is presented, and the children have to answer which of the four other characters presented originates from the same series. This does not train the visual cortex, but it helps children to learn the allocation of numbers as the answer is given as digits from 1-4. The representation of the score is also adapted; instead of a complex diagram, the system will say (depending on the number of correct answers), "You did a (very) good job!"

4.2 Core design elements and implementation of the application

The core design elements of the application are deducted from the application logic as it is described in the scenario above and from the user interfaces needed. Overall, eight different elements are needed:

- *Welcoming screen* for the user (corporate standard compliant)
- *Main menu* to select game or training (corporate standard compliant)
- *Settings menu* to tailor the application (corporate standard compliant)
- *Display for the mini games* (close to the corporate standard)
- Integration of the *haptic interface* to navigate the menus
- *Speech synthesis* to greet the user, explain the games, ask questions and present the results
- *Speech recognition* to answer questions or navigate menus
- Connection to the *CAN bus* to access the car's infrastructure (e.g., testing speed)

The first three design elements have already been covered by the application used at the company to prototype user interfaces. The implementation of the application logic for the HighIQ application(s) should be close to the same corporate standards, but as this kind of application does not yet exist, there are no concrete standards to comply to. The integration of the haptic interface is carried out using the two base components, thus separating the hardware communication from its semantic interpretation. Texts are synthesized as speech by the corresponding base component. Speech recognition is also added by the respective base component, allowing the recognition of commands in the form of recognition grammars for complex interactions. The current implementation of both speech interfaces allows the use of English or German. A "push-to-talk" (PTT) base component registers for the messages sent by the PTT button at the steering wheel to the car's bus and makes them available in HIMEPP. A matching PTT semantic interprets those signals and activates the speech recognition engine when the PTT button was pressed.

The screenshots of the finished prototype in Figure 5 are examples for the corporate standards compliant menus of the application (top left), the adapted display for the mini games using the math game as an example (top right), as well as the different instances of the visual mini game for adults and for children (bottom left and right). These screenshots are identical to what the user sees in the display integrated into the automobile's dashboard. Note the different menu names for the English (left) and German (right) version of the prototype, which can be switched using the settings menu.

Using HIMEPP as a prototyping platform allowed the evaluation of different visualization options for the application at development time. This is true for determining the menu structure that most customers found most usable; the main advantage was finding the best assessed technically feasible combination of mini games – i.e., the application logic with the highest acceptance – at the very beginning of the project. HIMEPP also supported the development in later stages by allowing customers to evaluate whether or not the current implementations of the application logic met the requirements initially elicited, i.e., whether or not the developers correctly understood and satisfied the requirements.

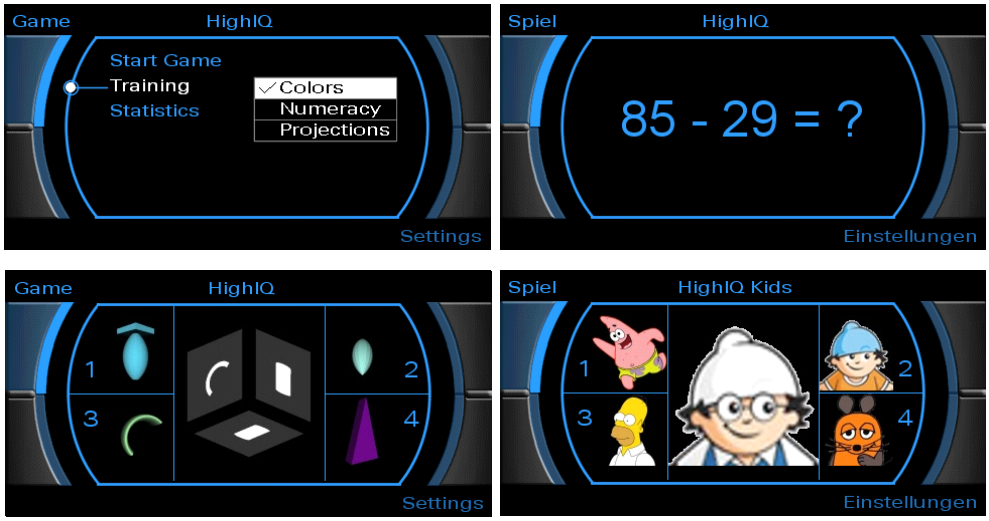

Figure 5. Prototype screenshots (own figure)

5 DISCUSSION AND FUTURE RESEARCH PERSPECTIVES

The integration of customers into the design of novel software functions, which meet their needs concerning information and comfort while driving, is currently one of the major challenges in the automotive industry. Using the tool described in this article, it will be possible in the future to successfully do so by jointly developing prototypes representing customers' requirements, and evaluating solutions found together early in the development process. This enhances the common understanding between different stakeholders and also allows customers to bring in their own ideas. From our initial case studies (6, overall), we have learned so far, that the HIMEPP platform supports the prototyping efforts and piloting the applications in four aspects:

- HIMEPP Core is a development framework improving the initial learning curve for new developers and lowering the hurdles in inter-developer cooperation.
- HIMEPP Base Components provide reusable essential interfaces, especially towards the user and the car, so that developers can focus on implementing their application's logic.
- HIMEPP's modular architecture allows updating parts of the platform and adding new components easily. This is true even for interfaces not compliant with the car manufacturer's current standards.
- HIMEPP's tight integration into the vehicle allows conducting an evaluation by observing the customer interacting with the system in the target environment (Williams & Helbig 2007). This helps understand how people would use the system on a daily basis.

The HIMEPP platform is currently in use in projects of our industry partner, at the U.S. laboratories of its corporation and at two universities. In the scope of these projects, it was shown that interaction concepts new to the industry (e.g., the Nintendo Wii's Nunchuk, featuring a three-axis accelerometer for motion sensing and an analog joystick with two buttons) can be integrated into the HIMEPP platform for evaluation, and that mobile applications can be systematically developed and described in preparation for series development. So far, prototypes were developed in early stages of the projects and then used as a basis for discussion concerning the course of action to be taken in the projects. Using the final prototypes integrated into the car in pilot studies, it was possible to ask lead users for an evaluation of the application and simultaneously draw conclusions concerning usage safety from the driving behavior while using the system (Williams & Helbig 2007). Conducting the numerous projects using the HIMEPP platform as a tool suggests that the kind of design of mobile applications is feasible, which is to be proven empirically. There are, however, several implications and as yet

unanswered questions for further research with regard to technological, methodical and theoretical aspects.

From a *technological* perspective, several opportunities arise for research concerning participatory design of applications in the vehicle. The ongoing widening of the systems' borders in the car opens it increasingly for the integration of external mobile devices, and the availability of online connectivity allows the integration of web-based applications. In order to learn about the most relevant trends for users, it should be possible for them to generate their own prototypes. Hence, HIMEPP will be expanded by another abstraction layer, allowing the visual design of application by "dragging and dropping" prebuilt functional blocks, thus enabling even laymen to create their own applications.

Concerning the *methodical* point of view, the participatory design has to be described more systematically and formally in order to be usable as a solid instance of design oriented science. Only in doing so does it becomes a reliable and validated tool for research projects. In the scope of this article, existing approaches in participatory design, prototyping and pilot studies have been combined and employed for designing a tool, easing further participatory design projects. In addition to the formal description of the process, the interface of the pre-development processes towards the serial production have to be identified, as they are a necessity for systematically deducting and describing the implicit requirements represented in the prototypes built. Achieving this would allow the use of the implicit information gathered in a cooperative way in the fixed processes of product development in the industry.

The *theoretical* contribution of participatory design lies in showing how future customers can be integrated into the development process. In order to develop a holistic understanding for such a development approach, understanding the different factors influencing it is essential. A first step in that direction would be the analysis of the theoretical backgrounds of research areas focusing on the interaction of groups of people in general, and computer mediated interaction in detail. Possible examples for this are knowledge and innovation communities, or the field of open innovation.

Participatory design of applications can be an essential tool for manufacturers in the automotive domain in the future to be able to integrate their customers into the development process. Customers can then not only become the source for explicit and implicit requirements, they can also point out possible solutions for problems arising in the process. This makes participatory design an essential factor for ensuring consumer satisfaction and helps eliminate costly misguided development early in the process. IS research can then contribute to this development by taking the role of a link between the different scientific disciplines and mediating between them when further extending participatory approaches into the industry.

References

Balzert, H. (2000). Lehrbuch der Software-Technik: Software-Entwicklung. Spektrum Akademischer Verlag, Heidelberg u.a.

Becker, W. (2002). Network of Automotive Excellence als Lösungsansatz für den Wandel in der Entwicklung. 2nd European Engineering User Conference, Brüssel.

Boone, J. (1999). Harvesting Design. In Building Application Frameworks: Object-Oriented Foundations of Framework Design(Eds, Fayad, M. E., Schmidt, D. C. and Johnson, R. E.) John Wiley & Sons, New York, p. 199-210.

Cwalina, K. and B. Abrams (2007). Richtlinien für das Framework-Design. Addison-Wesley, München.

Dix, A., J. Finlay, G. Abowd and R. Beale (2004). Human-Computer Interaction. Pearson Prentice Hall, Harlow.

Dumas, J. S. (2003). User-Based Evaluations. In The Human-Computer Interaction Handbook(Eds, Jacko, J. A. and Sears, A.) Lawrence Erlbaum Associates, Mahwah, p. 1093-1117.

Ehn, P. (1993). Scandinavian Design: On Participation and Skill. In Participatory design: principles and practices (Eds, Schuler, D. and Namioka, A.) Lawrence Erlbaum Ass., Hillsdale, p. 41-77.

Grudin, J. (1993). Obstacles to Participatory Design in Large Product Development Organizations. In Participatory design: principles and practices (Eds, Schuler, D. and Namioka, A.) Lawrence Erlbaum Ass., Hillsdale, p. 99-119.

Gruhn, V. and A. Thiel (2000). Komponentenmodelle. Addison-Wesley, München.

Hardung, B., T. Kölzow and A. Krüger (2004). Reuse of Software in Distributed Embedded Automotive Systems. International Conference on Embedded Software (Ed, Buttazzo, G.) ACM Press, Pisa, p. 203-210.

Henning, M. (2007). API Design Matters. ACM Queue, 5 (4), 24-36.

Hoffmann, H., J. M. Leimeister and H. Krcmar (2007). A Framework for Developing Personalizeable Mobile Services in Automobiles. Fifteenth European Conference on Information Systems (Eds, Österle, H., Schelp, J. and Winter, R.) University of St. Gallen, p. 938-949, St. Gallen.

Karat, J. (2003). Beyond Task Completion: Evaluation of Affective Components of Use. In The Human-Computer Interaction Handbook (Eds, Jacko, J. A. and Sears, A.) Lawrence Erlbaum Associates, Mahwah, p. 1152-1164.

Kawashima, R. (2005). Train Your Brain: 60 Days to a Better Brain. Kumon Publishing North America, Teaneck.

Mambrey, P. and V. Pipek (1999). Enhancing Participatory Design by Multiple Communication Channels. In Human-Computer Interaction: Communication, Cooperation, and Application Design, Vol. 2 (Eds, Bullinger, H.-J. and Ziegler, J.) Lawrence Earlbaum Ass., London, p. 387-391.

Mercer Management Consulting, Fraunhofer IPA and Fraunhofer IML (Eds.) (2004). Future Automotive Industry Structure (FAST) 2015: die neue Arbeitsteilung in der Automobilindustrie. Verband der Automobilindustrie, Frankfurt am Main.

Nuseibeh, B. A. and S. M. Easterbrook (2000). Requirements Engineering: A Roadmap. 22nd International Conference on Software Engineering, ICSE'00 (Ed, Finkelstein, A. C. W.) IEEE Computer Society Press,

OSGi Alliance (2007a). Automotive Electronics Market. Vol. 2007.

OSGi Alliance (2007b). OSGi Service Platform Release 4 - Service Compendium Version 4.1. OSGi Alliance, San Ramon.

Saad, A. (2003). Prototyping bei der BMW Car IT GmbH. JavaSpektrum, (2), 49-53.

Schwabe, G. and H. Krcmar (2000). Piloting a Socio-technical Innovation. 8th European Conference on Information Systems ECIS 2000, Vol. 1 (Eds, Hansen, H. R., Bichler, M. and Mahrer, H.) p. 132-139, Vienna.

Sharp, H., Y. Rogers and J. Preece (2007). Interaction Design. Wiley, Chichester.

Stritzinger, A. (1997). Komponentenbasierte Softwareentwicklung: Konzepte und Techniken für das Programmieren und Modellieren in Smalltalk. Addison-Wesley-Longman, Bonn.

Stroop, J. R. (1935). Studies of interference in serial verbal reactions. Journal of Experimental Psychology, (18), 643-662.

Szyperski, C. (2002). Component Software - Beyond Object-Oriented Programming. ACM Press, New York.

Tinham, B. (2007). Mobile IT: for real. Manufacturing Computer Solutions, 13 (2), 40-41.

Velez, M., D. Silver and M. Tremaine (2005). Understanding Visualization through Spatial Ability Differences. In Proceedings of the 16th IEEE Visualization 2005(Eds, Chen, B. and Gaither, K.) IEEE Computer Society Press, Minneapolis, p. 511-518.

Weinreich, R. and J. Sametinger (2001). Component Models and Component Services: Concepts and Principles. In Component-Based Software Engineering - Putting the Pieces Together(Eds, Heineman, G. T. and Councill, W. T.) Pearson Education, Upper Saddle River, p. 33-48.

Williams, J. D. (2001). Component Framework Worth the Struggle. Application Development Trends, Vol. 2008 (Eds, Varhol, P. and Mackie, K.).

Williams, M. and R. Helbig (2007). Probandenstudie - Evaluierung des Dienstes MACS MyNews. In Mobile Dienste im Auto der Zukunft (Eds, Reichwald, R., Krcmar, H. and Reindl, S.) EUL Verlag, Lohmar, p. 205-272.