A Model-based Approach to the Design and Rendering of Adaptive Automotive User Interfaces

Michael Feld[1], Gerrit Meixner[2], Angela Mahr[1], Anand Ramkumar Shanmugham[1]

[1]German Research Center for Artificial Intelligence,
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
[2]Heilbronn University, Faculty of Computer Science,
Max-Planck-Str. 39, 74081 Heilbronn, Germany

Abstract: Model-based design techniques have positively affected development time, cost, reliability, and effectiveness of user interfaces in various areas of modern ICT. Due to its immediate safety impact, the field of automotive UIs is particularly relevant and a primary candidate for model-based UI design. We further observe that personalized systems are gaining popularity in various mobile scenarios, such as Apple's *Siri* speech dialogue system, which exploits knowledge about the user to provide a distinct benefit. In the automotive domain though, personalization is still not commonly encountered. In this contribution, we discuss the aspects that make personalization challenging in this area, and investigate how industry-grade design principles help simplify the task. Our approach focuses on a UIML-based architecture for designing and rendering adaptive graphical dialogue systems for display in vehicles. Finally, we present an in-car parking assistance application that showcases how the framework could be employed and discuss some early numbers on its reception.

1 Introduction

User interfaces in the vehicle are in the middle of a major transition: from physical or "tangible" controls built into the consoles to visual controls rendered on the screen. In fact, as of today, new cars that do not feature any kind of pixel-based color display are becoming rare already. On the other hand, manufacturers such as Tesla with their *Model S* series now promote vehicles that replace all center console knobs with a single large touch screen display. The exclusive use of touch screens as opposed to separate controls might raise doubts as to whether it will provide the same ease of use and safety while on the road, since knobs do not require the driver to look at the control for operation after some practice. Yet, there is a strong demand from users that is driving the prevalence of screens. This is due to their flexibility with respect to different scenarios, information types, and users. The system can change the provided functions at any time if beneficial, either as a direct reaction to the user (e.g. a menu selection) or proactively. This ability in turn allows the car to satisfy one more core requirement of today's users, although it has not quite diffused into the automotive area yet. Rather, it is well known from mobile phones and home computers: the ability to personalize the Human-Machine Interface (HMI) to the user or adapt it to the current context. There are also current developments that attempt to combine the best of both worlds: to basically use a screen, but dynamically craft real tangible on-screen areas resembling buttons and other controls by employing various technologies, such as pneumatics. The actual user interfaces we see nowadays are mostly "static" though, in the sense that they generally do not take into account who is using the system, in what state this person is, or any special characteristics of the situation. This also means that they miss a chance to make the experience more comfortable, more efficient, more accessible, and also safer for the user.

There are several reasons why personalization is not as widespread in cars yet. One is missing experience with the concepts in the automotive field. The industry, being safety-focused by nature and subjected to long product development cycles, is known to be generally conservative, and waiting for some party to take the first step before the mass market follows. A thorough model-based design can help reduce these hurdles. The other major reason is that personalization is actually a complex field of research. More precisely, doing personalization properly in a way that ultimately delivers a benefit to the user, can be rather challenging [Jam03].

In this paper, we will focus on the engineering challenges that adaptive systems bring along. Functions developed for an in-car display-based HMI involve the usage of one or more graphical elements. The visual representation of these functions is carefully designed to meet the strict requirements of the industry and adhere to norms such as the Code of Practice [S$^+$06]. Therefore, it is essential to validate designs at different levels of abstraction until they are finally modeled in a graphical UI representation language that can be rendered. In the following sections, we will assemble the parts required to render adaptive UIs in the car. The first part of this paper elaborates on the aspect of model-based design the choice of UIML. Then, we deal with the question of how to acquire information about the user (generally the driver). Afterwards, we advance into the topic of adaptation and illustrate some common criteria for developing adaptation strategies for in-vehicle functions. We go on by outlining the rendering component. Finally, we show a concrete application use case in which we developed a user-adaptive parking assistance application and collected some initial user feedback on the UI displayed using our renderer prototype.

2 Related Work

As previously indicated, adaptive rendering is a rather recent application of the technology in the automotive field. However, there are related concepts published in other fields.

Early major work on adaptive interfaces was published by Benyon and Murray [BM88], who present the *Monitor System* – an adaptive shell which takes into account user characteristics and presents different dialogues to users. In another early work on productivity gains via an adaptive user interface, Trumbly et. al. [TAJ94] performed an elaborate study with a simulated decision-making task that adapts the computer interface based on the user expertise. They found the adaptive user interface to improve the user performance. Some early well-known adaptive system was also created by Fink et al. [FKN98] in the AVANTI project. It is one of the first attempts to make hypermedia documents adaptive, adjusting page contents and layout for e.g. elderly and disabled people. It already utilizes centralized user models stored on a BGP-MS based server. A newer and algorithmically advanced implementation is SUPPLE [GW04]. The software is able to generate a highly dynamic interface based on a functional specification (e.g. devices to be controlled), output device capabilities, and a user model. The latter is based on histories of manipulated elements (so-called user trails) and thus rather reflects a *usage* model.

There is also more recent work that deals with the runtime adaptation of user interfaces. In the INREDIS project, a solution was needed to help people with disabilities use vending machines, ATM terminals, and other devices in ubiquitous environments. Miñón et al. [MAA+11] describe an adaptive interface generator to assist with this problem. Just like our renderer, they rely on UIML to describe the user interface requirements in an abstract way. The actual adaptation process exploits the style sheet technologies XSL and CSS, while the user model stems from a (not further specified) knowledge base. A domain that shares many aspects with the automotive field in terms of the required grade of robustness are industrial and enterprise applications. Akiki et al. [ABY12] follow a model-based approach to bring adaptivity to this field, for instance to accommodate different degrees of user experience in ERP, CRM, and similar systems. They too promote UI abstraction; however, the authors choose *UsiXml* as their favored modeling language. Levia [Lei12] developed an algorithm which monitors user interaction on web pages to estimate user prioritization of widgets. It then uses this model to adjust the CSS style sheet to highlight or otherwise re-style these widgets.

Most examples mentioned adapt a graphical interface, but the concept is generally applicable to all kinds of human-machine interfaces, including speech dialogues. A personalized speech dialogue has for instance been proposed by Metze et al. [MEB+09], which non-intrusively detects the user's age, gender, and emotion and then optimizes the dialogue structure. Some of the ideas that will be introduced in the following sections are cross-modally applicable, while others focus on graphical presentation as a selected subset of adaptation effects.

3 Model-based User Interface Development

Additionally to the growing number of configurable systems in a modern car, there is a need for shorter release cycles and lower costs for the development of automotive UIs. One main source of problems in the development of UIs is the communication overhead caused by informal specifications [HG07]. This overhead is needed to reduce the ambiguity and the incorrectness of the specification document. A more formal approach to specification might reduce this overhead dramatically, leading to shorter development times, cost savings and fewer problems. (Semi-)Formal specification of UIs is researched in the context of model-based user interface development (MBUID) which could be a solution for the above problems [Mei11, HMZ11]. For a brief review of former and current MBUID approaches we refer to [MPV11].

Key aspects of MBUID are

- Abstraction and Reuse: For the description of the UI models are used which can be reused.
- Separation of concerns: Different aspects of a UI are described in different layers of a UI through different models.
- Tools: Tools are very important to support developers in creating and manipulating models and transformations.
- (Semi-)automatic transformations: Mappings between the different layers of a UI, i.e. between the different models describe their dependance.

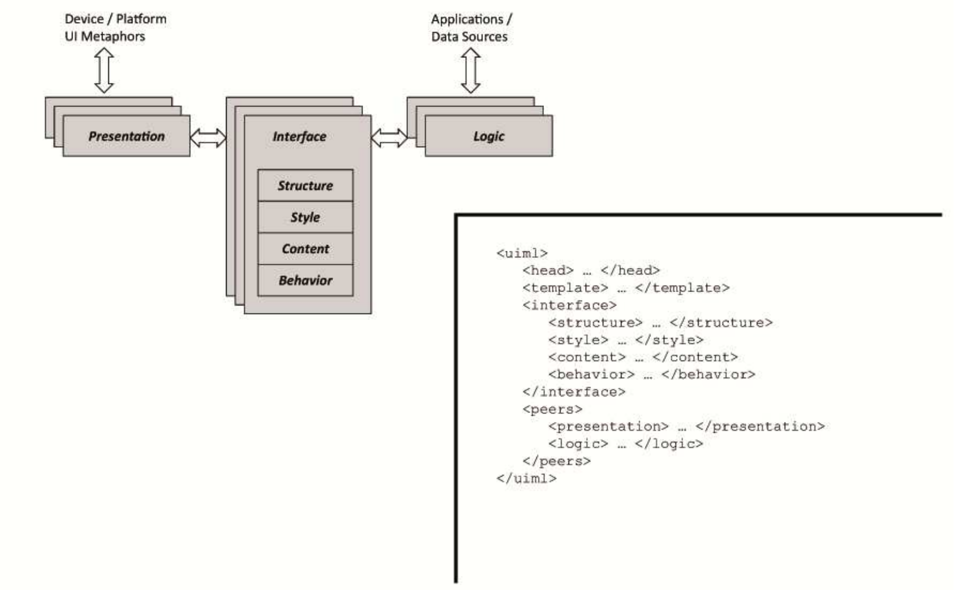
A vast number of (XML-based) User Interface Description Languages (UIDLs) exist already in the field of MBUID [SV03, GGGCVMA09]. Some of the UIDLs are already standardized by e.g., *OASIS*, currently being standardized by e.g. W3C [W3C] and/or they are subject of a continuous development process. Other approaches are e.g. based on discourse models [PFA+09].

The purpose of using a UIDL to develop a user interface (UI) is to systematize the UI development process [Pue97]. UIDLs enable the developer to systematically break down a UI into different abstraction layers and to model these layers. Thus it is for example possible to describe the behavior, the structure and the layout of a UI independently of each other. Existing UIDLs differ in terms of the supported platforms and modalities as well as in the amount of predefined interaction objects that are available to describe the elements of the UI.

The aim of MBUID is to describe the properties of a UI in sufficient detail and formality so that (semi-)automatic model transformations can be applied to generate the final UI. The *Cameleon Reference Framework* (CRF) [CCT+03] identifies four different layers of a UI in a MBUID process:

- Task and Domain: Describes the tasks that the user wants to perform as well as domain concepts.
- Abstract User Interface: A description of the UI which describes the dynamic behavior and the interaction objects of the UI in a modality-independent matter.
- Concrete User Interface: A modality-dependent but (programming) language-independent description of a UI that allows specifying the visual or auditive properties (depending on the used modalities) of the UI.
- Final User Interface: Expresses the UI in terms of implementation-dependent source code. A FUI can be represented in any UI programming language (e.g., Java Swing toolkit) or mark-up language (e.g., HTML). A FUI can then be interpreted or compiled.

Between these levels there are different relationships e.g., reification covers the inference process from high-level abstract descriptions to run-time code.



```
<uiml>
   <head> … </head>
   <template> … </template>
   <interface>
      <structure> … </structure>
      <style> … </style>
      <content> … </content>
      <behavior> … </behavior>
   </interface>
   <peers>
      <presentation> … </presentation>
      <logic> … </logic>
   </peers>
</uiml>
```

Figure 1: The UIML Meta-Interface Model.

The CRF recommends a four-step reification process: a Concepts-and-Tasks Model is reified into an Abstract UI which in turn leads to a Concrete UI. A Concrete UI is then turned into a Final UI, typically by means of code generation techniques; Abstraction is an operation intended to map a UI representation from one non-initial level of abstraction to a higher level of abstraction. In the context of reverse engineering, it is the opposite of reification; Translation is an operation that transforms a description intended for a particular target or context of use into a description at the same abstraction level but aimed at a different target or context of use. Furthermore, the CRF provides a unified understanding of context-sensitive UIs rather than a prescription of various ways or methods for tackling different steps of development.

4 The User Interface Markup Language

The User Interface Markup Language (UIML) is a meta-language that permits a declarative description of user interface in a device independent manner. The intended principle of the UIML is to give a vendor-neutral, standardized representation of any user interface which can be mapped to the high level development languages. The current version is UIML 4.0 for which standard definitions are available [HSL$^+$09]. Since May 2009, UIML 4.0 is a standard of the Organization for the Advancement of Structured Information Standards (OASIS).

For the work described in this paper we have chosen UIML as a standardized UIDL to be suitable for our project. UIML is an efficient way to portray the structure, style, content and behavior of a user interface using a high-level markup language. UIML gives a device-independent technique to illustrate a UI and it is analogous to XML and it provides a way to create our own data format.

It does not hold any widget set specific information and contains only generic elements. Finally, the abstract ideas can be mapped to their detailed matching element depending upon the system requirement. The interface section of a UIML document consists of five components (see Figure 1): *structure*, *style*, *layout*, *content*, and *behavior*:

- *Structure* describes a hierarchy of interaction objects and their relation-ships.
- *Style* specifies the properties of the components, e.g., text, color, or size.
- *Layout* defines how the components are arranged relative to each other (spatial constraints).
- *Content* separates the content of the interface from the other parts and is referenced in the components' properties.
- *Behavior* describes, for example, interaction rules or actions to be triggered under various circumstances (specifies the dialog model).

While the interface section describes the general appearance of the user interface, the peers section of a UIML document describes the concrete instantiation of the user interface by providing a mapping onto a platform-specific language (i.e., interface toolkits or bindings to the application logic):

- *Logic* specifies how the interfaces are bound to the application logic.
- *Presentation* describes the mapping to a concrete interface toolkit, such as *Java Swing*.

Furthermore, a UIML document includes an optional `head`-element for providing meta-information and a concept that allows the reuse of predefined components. These so-called "templates" are predefined interaction objects, which can be easily instantiated with concrete parameters derived from the application data model.

UIML with a generic graphical vocabulary (like e.g. as demonstrated in [APQAS02]) lies in the layer of the Concrete User Interface of the CRF. UIML rendering is the process of converting UIML into a standard UI that can be presented to the user through a system with which the user can interact.

Rendering can be done in two major ways: 1) Compiling UIML into a higher level programming language which creates the display and interaction of the UI described in UIML. 2) Interpreting UIML using a program through a function call from a higher level language that displays the UI and allows interaction.

When interpreting, both the UIML document and the renderer are needed to create and run the UI. The benefit of interpreting is that in the UI can be changed at run-time to adapt the UI e.g. towards a changing user model. Therefore we have chosen the interpretation method in this work to render the UIML document.

5 Acquiring User Models

In order to build user-adaptive applications, one needs to have knowledge about the user - collectively termed a user model or user profile. In the most general case, such a model contains anything associated with that person that may aid the application in making decisions with respect to what to present and how to present it. Some common examples are identifying, demographical, and physiological information about the user (e.g. name, age, height, address, etc.). Advanced properties might describe preferences, skills, limitations, and needs (e.g. favorite music, driver's license, swimming skills, shortsightedness, etc.). Knowing that a person has bad sight could for example justify the selection of a larger default font size. The more powerful and complex user models have emerged from artificial intelligence, most prominently from dialog systems [WK89] and are nowadays used in a plethora of other fields.

One particular question of a renderer for the vehicular domain is how to acquire the user model. In theory, there are several options, but not all are equally suitable. The most obvious would be an explicit input given by the user, either on her own initiative (e.g. user changes the display color theme) or triggered by the system (e.g. system asks for the user's name). A different category of acquisition methods relies on implicit and often non-interactive input, which can be intrusive (e.g. bio-sensors measuring the skin conductance indicating stress) or non-intrusive (e.g. camera monitoring facial expression and pupil size). Finally, the system can also work on existing data sources in order to link (e.g. from an external user profile server), mine (e.g. extract from e-mail messages), or infer (i.e. combine existing facts) new knowledge.

The storing and managing of user models is an entirely different task. Due to the increasing number of in-vehicle systems that depend on and share complex knowledge, we favor the use of a central knowledge store that is based on an ontology. An Automotive Ontology that is tailored to the specific characteristics of the domain (such as a fast-changing spatial context) has been proposed in [FM11]. This ontology is also backing the demonstration application we introduce in the forthcoming sections. The model has been extended to cover the specialized case created for the user study.

6 Planning Adaptation Strategies

A user model and its maintenance are two prerequisites for personalization. The more obvious question from an application point of view is which aspects of the application should be personalized and how the system should map from knowledge in the user model to adaptation effects. A simple solution would have the user specify the desired effects directly (e.g. through configuration), in which case we speak of an adaptable rather than an adaptive application. The latter could potentially provide a better usability, since users do not have to configure the system manually, something that is often difficult particularly for those users that would be in need of it (e.g. elderly persons, users with disabilities; see [FKN98]). This is the core case we are focusing on.

Planning for an adaptive system can be decisively more rewarding, but requires a careful planning and involves certain challenges, also known as usability challenges [Jam03]: predictability (the user can predict system behavior to a certain degree), comprehensibility (the user can understand the system behavior), controllability (the user can prevent adaptation if not desired), obtrusiveness (adaptation should occur without adding effort to the user), breadth of experience (adaptation should not restrict the user experience), and privacy. A system employs adaptation strategies to specify how it should adapt its presentation (form) or behavior (function) if the current user/context model satisfies some condition. In the automotive domain, the following user and context model properties have been pointed out to be particularly suitable for adaptation:

- Static (or infrequently-changing) user characteristics: age, gender, vision, hearing, physical abilities, (driving) experience...
- Dynamic user state: workload, fatigue, emotion, schedule...
- Driving situation: traffic density, urban vs. rural, byway vs. motorway...
- Environmental situation: weather, vision, road conditions, road signs...
- Communication situation: dialogue system context, passenger dialogue, phone calls...

Each of these conditions may require that the system adjusts its features. The exact effect depends on the type of function being adjusted, yet several important categories can be identified:

- Adapting dialogue strategies: skipping or adding prompts, adding confirmations, offering help...
- Adapting presentation mode: input interaction technique, widgets, output modality...
- Adapting presentation style: text size, font, colors, layout, sound volume, speaking rate of speech output...
- Re-scheduling: lead time or duration of warnings, information priority...
- Adapting the granularity: providing information in a more or less detailed way
- Filtering the contents: removing items from a list
- Changing the contents: providing different information, e.g. route selection

```
<Rules>
  <Strategy name="MakeVisibleStrategy">
    <Activation UserModelProperty="ageClass" value="senior">
      <ApplyTo class="button">
        <Property visible="true" />
      </ApplyTo>
      <ApplyTo id="image1,image2">
        <Property visible="true" />
      </ApplyTo>
    </Activation>
  </Strategy>
</Rules>
```

Figure 2: Example adaptation rule file. Here, a single strategy aimed at elderly users (by matching the user property *ageClass* to value *senior*) applies two effects when activated: making all hidden buttons visible as well as showing two specific images (*image1* and *image2*).

7 Renderer Architecture

Combining the previously discussed aspects into a single architecture leads us to the renderer design proposed in [FMM$^+$13]. Its overall function is to read and interpret a standards-compliant UIML input document, adapt it using a knowledge base and adaptation strategies, and finally display it in a Flash host, which might be running the in-car HMI. The rendering pipeline consists of a three-layer process: There are two layers dealing with adaptation: The *UI Adaptation Layer* adapts the style, while the *Data Adaptation Layer* adapts the content. The former has access to adaptation strategies specified in a rule file providing the mapping between user model properties, UI elements, and the adaptation effects by indicating the properties depending on which the adaptation effect has to be applied (see Figure 2 for an example). Last, there is one layer handling the actual rendering. For a more detailed description of the renderer architecture, we refer to the aforementioned source.

8 Evaluation Using an Automotive Application

The first in-car function implemented in our lab to use the user-adaptive Flash renderer was a personalized parking assistant. The purpose of this IVIS (in-vehicle information system) is to help the user quickly find a suitable parking space, for instance when she is driving in an area she is not familiar with, or when she would like to find out where there are still places left. The function was created with personalization in mind: For users with specific needs, it should automatically optimize its presentation to maximize usage efficiency. The application was intended for demonstration and evaluation purposes. In order to simplify the presentation scenario, it uses only graphical output and is limited to touch-based interaction.

The parking assistant consists of three separate “screens”: a search screen, a parking information screen, and a payment screen. The search screen (Figure 3, top) can be invoked

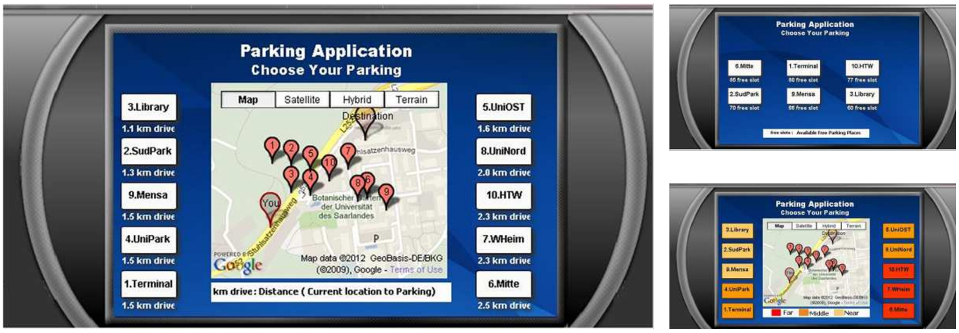


Figure 3: The parking assistant's *parking search* screen. *Left:* non-adapted version; *Right Top:* adapted for increased workload; *Right Bottom:* adapted for decreased sight.

by the user during normal navigation to request information about nearby parking spaces. It is also possible for the system to launch it pro-actively when it determines the possible need for a parking (e.g. when getting close to the destination). The parking information screen shows details about a selected parking space and provides further commands, such as access to payment options. The payment screen allows the user to pay her ticket from inside the vehicle for metered spaces. The screens are linked by command buttons.

The personalization aspects of the application revolve around the following user (and in one case context) properties: *Regular visitor*, *increased workload / stress*, *decreased eyesight*, *restricted movement*, and *young children as passengers*. Each of the three application screens can change its appearance based on the value of one or more of these properties, which can take the Boolean (for simplicity) values *false* or *true* in this application. The adaptation strategies were created following an initial design review in order to improve usability in the respective cases. For example, when the user is under stress (possibly caused by time pressure or dense traffic), the number of options on the search screen is reduced, following the assumption that in such a situation, a careful comparison of many options would be more difficult (Figure 3, middle). If, on the other hand, the user has decreased eyesight, possibly caused by age, the application color-codes the distance to the parking into the button rather than having a separate text label, and uses the gained space to increase the font size (Figure 3, bottom). These personalization features were implemented using adaptation rule files as described earlier provided to the renderer. Table 1 summarizes all adaptation scenarios implemented in this application for the *parking search* screen. Similar adjustments were made to the other two screens. It has to be noted however that in a real-world application, strategy design should be based on a more elaborate justification than a single design review iteration.

The fully functional prototype of this application was used to obtain some initial figures on the reception of the personalized application by test users. The goal of this brief study was to learn about the possible positive or negative effects of the adaptation, as well as about the capability of the renderer architecture to deliver it.

Model Property	Adaptation Effect(s)
Regular visitor	• Parkings sorted by previous preference • Graphical map hidden
Increased workload	• Reduced number of options • Graphical map hidden, less visual clutter • Added information on free lots
Decreased sight	• Increased button and font size • Distance color-coded into button (plus key)
Movement disabilities	• Icon added for accessible parkings • Shows information on number of available parking lots for disabled instead of distance
With young children	• Shows information on distance from parking to destination instead of distance to parking • Parkings sorted by distance to destination

Table 1: Table of adaptation strategies implemented in the parking assistant's *parking search* screen. The strategies essentially map user model properties to renderer effects.

8.1 Method

The design of this user study was planned to encourage test subjects to use the application by giving them short tasks in the application context, e.g. selecting a parking or paying the ticket. Subjects were put into different situations resembling adaptation scenarios by orally introducing them to it ("Imagine you were…") and then asking them to act and later rate accordingly. While this is clearly not an ideal way of projecting characteristics such as bad sight on persons, the small dimension of the study prevented the recruitment of diverse groups of subjects that were naturally in the required state. This limitation has to be taken into account when interpreting the study results.

The study consists of 1 baseline condition and 5 adaptation conditions. In each condition, participants had to complete a set of simple tasks (locating a parking, obtaining parking information, paying a ticket) using the parking assistant application running on a desktop computer. The baseline task did not contain an adaptation effect and showed a "neutral" default user interface. The other conditions each had one of the user model properties set, e.g. *regular visitor*, which caused the corresponding adaptation effects to be applied by the renderer. Subjects that had difficulties performing the tasks were allowed to ask questions. After each condition, the participants had to fill out a questionnaire relating to the performed task. The study was conducted using a standard desktop computer.

8.2 Metric

In the questionnaire, the user was asked for the following feedback:

1. List of changes identified in the user interface compared to the (non-adapted) base-

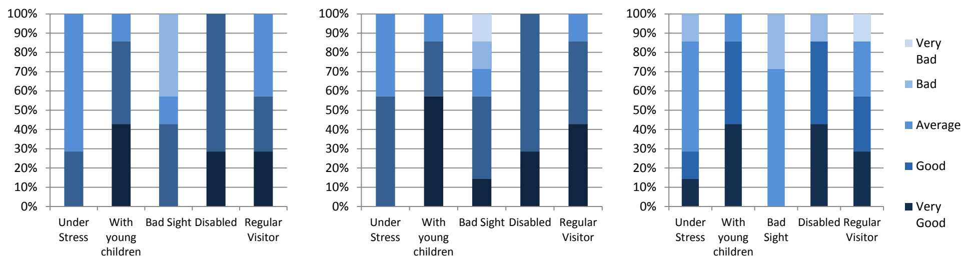


Figure 4: Subjective ratings of three aspects of adaptation for the individual conditions. *Left:* general idea; *Center:* chosen strategy; *Right:* implementation usability.

line

2. Rating of the idea of performing adaptation in the given situation
3. Rating of the usefulness of the applied adaptation strategy in general, regardless of implementation details
4. Rating of the usability of the adapted version as implemented

All ratings were made on a five-point scale. Apart from the questionnaire, while the user was performing the task, the experimenter also made note of the task success and the number of comprehension questions asked by the subject for each task.

Our hypothesis for this study was that there would generally be a positive tendency for the rating of the adaptive versions. 7 users (4 male, 3 female) with ages ranging from 22–27 years participated in the study. The users had driving experience of around 4 years and moderate user experience with navigation systems.

8.3 Results

Figure 4 (left) shows the subjects' ratings of their attitude towards adaptation in the given scenario (question 2). We can see that the idea of adaptation for the *Disabled*, *With Young Children* and *Regular Visitors* are considered more appropriate by the users than adaptation in case of *Bad Sight* and *Stress*.

A similar result was recorded for the users' opinion on the adaptation effect suggested by the parking assistant application in the particular situations (question 3). Figure 4 (center) shows that the adaptation effect for *Disabled* was considered most useful, followed by the condition *With Young Children* and then *Regular visitors*.

The usability rating (question 4) incorporates details of the adaptation effect as well as technical aspects of the renderer. Figure 4 (right) indicates a similar picture: Subjects were rather positive about the *With Young Children* adaptation, but more reluctant about the *Bad Sight* implementation.

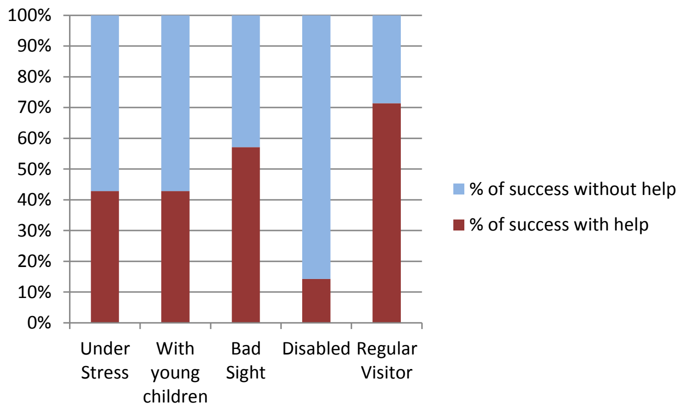


Figure 5: Percentage of successfully completed tasks with and without support by the experimenter.

The chart in Figure 5 shows the percentage of users who asked for help solving the task and the percentage of users who completed the task without any help. This can be seen as another (more objective) indication of how usable the adapted interface was. According to these numbers, the most intuitive interface was the *Disabled* adaptation.

8.4 Discussion

The overall results reveal that people were generally positive about the personalized versions of the parking assistance application. They also show that the rating depends significantly on the individual condition and implementation, which comes as no surprise since it is well known that adaptation strategy design is a critical factor. We can also learn from the numbers that the effects were sufficiently transparent so that more than 80% of participants identified at least some of them. Keeping in mind that people were not truly in the state that should trigger the adaptation during real usage, these numbers are merely an indication. People were able to solve the given tasks (some with help), though the ratings for the implementation were on average slightly worse than the other condition-specific ratings. This may indicate that the renderer in its current state is working overall well, but usability could possibly still be optimized. The fact that it was not possible with this study to clearly separate the technical from condition-specific design aspects has to be taken into account as a limitation.

9 Conclusion and Future Work

We have shown how existing standards can be leveraged to build a rendering architecture that combines two aspects: abstraction from a final user interface, so it can be integrated into an MBUID process that is part of modern automotive HMI design tool chains, and user-adaptive behavior. We have further presented a personalized parking assistant application, which was designed completely in UIML and was made user-adaptive through declarative adaptation rules.

Going beyond the work in this paper, we have identified adaptation strategies as one area that will need further investigation. Our current solution provides a basic mapping, but a standard should be the long-term goal. Also, conflict resolution is not yet supported. Finally, UIML as a basis for specifying automotive UIs has yet to be evaluated in a more complex scenario, such as on-board navigation systems.

References

[ABY12] P. Akiki, A. Bandara, and Y. Yu. Using interpreted runtime models for devising adaptive user interfaces of enterprise applications. In *14th International Conference on Enterprise Information Systems (ICEIS 2012)*, Wroclaw, Poland, 28 June – 1 July 2012. SciTePress.

[APQAS02] M.F. Ali, M.A. Perez-Quinones, M. Abrams, and E. Shell. Building multi-platform user interfaces with UIML. In *Proc. of the 4th International Conference on Computer-Aided Design of User Interfaces*, pages 255–266, 2002.

[BM88] D. Benyon and D. Murray. Experience with adaptive interfaces. *The Computer Journal*, 31(5):465–473, 1988.

[CCT$^+$03] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.

[FKN98] J. Fink, A. Kobsa, and A. Nill. Adaptable and adaptive information provision for all users, including disabled and elderly people. *New review of Hypermedia and Multimedia*, 4(1):163–188, 1998.

[FM11] Michael Feld and Christian Müller. The Automotive Ontology: Managing Knowledge Inside the Vehicle and Sharing it Between Cars. In *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2011)*, pages 1–8, Salzburg, Austria, November 2011. ACM.

[FMM$^+$13] Michael Feld, Gerrit Meixner, Angela Mahr, Marc Seissler, and Balaji Kalyanasundaram. Generating a Personalized UI for the Car: A User-Adaptive Rendering Architecture. In *UMAP*, pages 344–346, 2013.

[GGGCVMA09] J. Guerrero-Garcia, J.M. Gonzalez-Calleros, J. Vanderdonckt, and J. Muoz-Arteaga. A theoretical survey of user interface description languages: Preliminary results. In *Proc. of Joint 4th Latin American Conference on Human-Computer Interaction*, pages 36–43, Los Alamitos, United States, 2009. IEEE.

[GW04] K. Gajos and D.S. Weld. SUPPLE: automatically generating user interfaces. In *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 93–100. ACM, 2004.

[HG07] M. Hübner and I. Grüll. ICUC-XML Format. Format Specification Revision 14. Elektrobit, 2007.

[HMZ11] H. Hussmann, G. Meixner, and D. Zühlke. Model-Driven Development of Advanced User Interfaces. *Studies in Computational Intelligence*, 340, 2011.

[HSL+09] James Helms, Robbie Schaefer, Kris Luyten, Jean Vanderdonckt, Jo Vermeulen, and Marc Abrams. User interface markup language (UIML) specification version 4.0. Technical report, OASIS Open, Inc., 2009. Draft.

[Jam03] A. Jameson. Systems that adapt to their users: An integrative Overview. In *Tutorial presented at 9th International Conference on User Modelling*, 2003.

[Lei12] L. Leiva. Interaction-based user interface redesign. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 311–312. ACM, 2012.

[MAA+11] R. Miñón, J. Abascal, A. Aizpurua, I. Cearreta, B. Gamecho, and N. Garay. Model-based accessible user interface generation in ubiquitous environments. In *Human-Computer Interaction – INTERACT 2011*, pages 572–575. Springer, 2011.

[MEB+09] F. Metze, R. Englert, U. Bub, F. Burkhardt, and J. Stegmann. Getting closer: tailored human–computer speech dialog. *Universal Access in the Information Society*, 8(2):97–108, 2009.

[Mei11] G. Meixner. Modellbasierte Entwicklung von Benutzungsschnittstellen. *Informatik Spektrum*, 34(4):400–404, 2011.

[MPV11] G. Meixner, F. Paternò, and J. Vanderdonckt. Past, Present, and Future of Model-Based User Interface Development. *i-com*, 10(3):2–11, 2011.

[PFA+09] Roman Popp, Jürgen Falb, Edin Arnautovic, Hermann Kaindl, Sevan Kavaldjian, Dominik Ertl, Helmut Horacek, and Cristian Bogdan. Automatic Generation of the Behavior of a User Interface from a High-Level Discourse Model. In *HICSS*, pages 1–10, 2009.

[Pue97] A.R. Puerta. A model-based interface development environment. *Software, IEEE*, 14(4):40–47, 1997.

[S+06] Jürgen Schwarz et al. Code of Practice for the Design and Evaluation of ADAS. Project deliverable in the Preventive and Active Safety Applications Integrated Project (PReVENT), October 2006.

[SV03] N. Souchon and J. Vanderdonckt. A review of XML-compliant user interface description languages. In *Proc. of the 10th International Workshop on Interactive Systems: Design, Specification and Verification of Interactive Systems*, pages 377–391. Springer, 2003.

[TAJ94] J.E. Trumbly, K.P. Arnett, and P.C. Johnson. Productivity gains via an adaptive user interface: An empirical analysis. *International Journal of Human-Computer Studies*, 1994.

[W3C] W3C Working Group Model-Based User Interfaces. http://www.w3.org/2011/mbui/. Retrieved October 05, 2012.

[WK89] Wolfgang Wahlster and Alfred Kobsa. User Models in Dialog Systems. In Wolfgang Wahlster and Alfred Kobsa, editors, *User Models in Dialog Systems*, pages 4–34. Springer Berlin, 1989.