# Generating a Personalized UI for the Car: A User-Adaptive Rendering Architecture

Michael Feld[1], Gerrit Meixner[2], Angela Mahr[1], Marc Seissler[3], and Balaji Kalyanasundaram[1]

[1] German Research Center for Artificial Intelligence, Saarbrücken, Germany
[2] Heilbronn University, Faculty of Computer Science, Heilbronn, Germany
[3] German Research Center for Artificial Intelligence, Kaiserslautern, Germany

**Abstract.** Personalized systems are gaining popularity in various mobile scenarios. In this work, we take on the challenges associated with the automotive domain and present a user-adaptive graphical renderer. By supporting a strictly model-based development processes, we meet the rigid requirements of the industry. The proposed architecture is based on the UIML standard and a novel rule-based adaptation framework.

**Keywords:** Automotive, Personalization, User adaptation, UIML.

## 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. Being more flexible, the system can adjust the provided functions (e.g. navigation, other assistance, infotainment) at any time, even pro-actively. They can take into account who is using the system, in what state this person is, or any special characteristics of the situation. This ability that results in a more personalized experience is well-known from mobile phones and home computers [2], but it has not been adopted by the car industry yet. Among the reasons are the absence of a rendering architecture that complies with the special requirements imposed by the industry on HMI development processes.

In addition 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 is the communication overhead caused by informal specifications [5]. This overhead is needed to reduce the ambiguity and the incorrectness of the specification document. A more formal approach can 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) [8,6]. For a brief review of former and current MBUID approaches see [7].

A vast number of (XML-based) User Interface Description Languages (UIDLs) exist already in the field of MBUID [3]. Some of the UIDLs are already standardized by e.g., *OASIS*, currently being standardized by e.g. W3C [1] and/or they

S. Carberry et al. (Eds.): UMAP 2013, LNCS 7899, pp. 344–346, 2013.

are subject of a continuous development process. The purpose of using a UIDL to develop a user interface (UI) is to systematize the UI development process [8]. UIDLs enable the developer to systematically break down a UI into different abstraction layers and to model these layers. Thus it is possible e.g. to describe the behavior, the structure and the layout of a UI independently of each other. Existing UIDLs differ in terms of supported platforms and modalities as well as in the amount of predefined interaction objects for describing UI elements.

## 2 Renderer Architecture

In our project we developed a design that combines the aspects of user modeling, rule-based adaptation, and rendering in a single architecture as outlined in Figure 1. The rendering pipeline consists of a three-layer process: There are two layers dealing with adaptation (layers 1 and 2) and one layer handling the actual rendering. These two aspects are generally independent, although they have been optimized for this scenario. The input to the renderer is a standards-compliant User Interface Markup Language (UIML) [4] document. The output is the rendering, which is directly displayed in a browser or stand-alone Flash application when the appropriate host Flash document is loaded. The coordination of the process is performed by the Adaptation Assistant component. The *UI Adaptation Layer* adapts the user interface elements of the original input UIML based on a user model and set of rules. The user model is obtained from a knowledge base component called *KAPcom* (Knowledge Management, Adaptation and Personalization Component). The adaptation rules are specified as an XML-based rule file. This file can contain multiple sets of tags to represent the rules. It provides a 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. Examples of effects that can be performed are: adding/removing images, reordering UI items, changing fonts, changing the layout etc. For instance, there could be a rule that increases the margin around

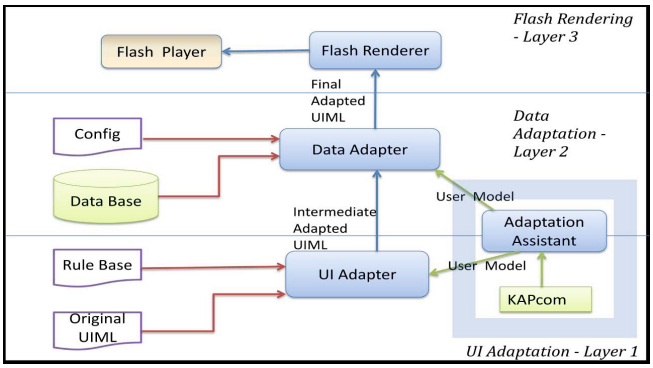


**Fig. 1.** Architecture of the proposed rendering component

**Fig. 2.** In-car HMI

buttons for elderly users or at high speed. The output of this layer is an adapted UIML file that may have some of the layout elements changed. This document is then processed by the *Data Adaptation Layer*. It performs adaptation based on the same user model, but it changes the content instead of the layout. This might affect for instance text labels. It is connected to a separate data store representing the current application data model (e.g. route information). This results in a further adapted version of the UIML code. The *Rendering Layer* finally takes the fully adapted UIML file and produces the Flash output. It consists of parser, layouting, and object generation modules that are implemented completely in *ActionScript*. The major work of this component consists of mapping the UIML elements to Flash widgets, thereby re-creating the original layout on the fly.

## 3 Conclusion and Outlook

We have leveraged existing standards to design 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. As part of our research, we have further created a personalized parking assistant application that utilizes the renderer, and which was designed completely in UIML (Figure 2). A small user study conducted with this application has confirmed that people were indeed seeing a benefit in the adaptive version, but also that the appropriateness depends largely on the use case. In the near future, we expect numerous personalized applications starting to appear in the vehicular context.

## References

1. W3C Working Group Model-based User Interfaces, `http://www.w3.org/2011/mbui/` (retrieved October 05, 2012)
2. Gajos, K., Weld, D.: SUPPLE: automatically generating user interfaces. In: Proc. of the 9th International Conference on Intelligent User Interfaces. ACM (2004)
3. Guerrero-Garcia, J., Gonzalez-Calleros, J., Vanderdonckt, J., Muoz-Arteaga, J.: A theoretical survey of user interface description languages: Preliminary results. In: Proc. of Joint 4th Latin American Conference on Human-Computer Interaction, pp. 36–43. IEEE, Los Alamitos (2009)
4. Helms, J., Schaefer, R., Luyten, K., Vanderdonckt, J., Vermeulen, J., Abrams, M.: User interface markup language (UIML) specification version 4.0. Tech. rep., OASIS Open, Inc. (2009) (draft)
5. Hübner, M., Grüll, I.: ICUC-XML format specification revision 14. Elektrobit (2007)
6. Hussmann, H., Meixner, G., Zuehlke, D. (eds.): Model-Driven Development of Advanced User Interfaces. Studies in Computational Intelligence, vol. 340. Springer, Heidelberg (2011)
7. Meixner, G., Paternò, F., Vanderdonckt, J.: Past, present, and future of model-based user interface development. i-Com 10(3), 2–11 (2011)
8. Puerta, A.: A model-based interface development environment. IEEE Software 14(4), 40–47 (1997)