Evaluation of User Interface Description Languages for Model-Based User Interface Development in the German Automotive Industry

Gerrit Meixner[1], Marius Orfgen[2], and Moritz Kümmerling[2]

[1] Heilbronn University
Max-Planck-Str. 39, 74081 Heilbronn, Germany
Gerrit.Meixner@hs-heilbronn.de
[2] German Research Center for Artificial Intelligence (DFKI),
Trippstadter Str. 122, 67663 Kaiserslautern, Germany
{Marius.Orfgen,Moritz.Kuemmerling}@dfki.de

Abstract. Developing human-machine-interfaces (HMI) in the automotive industry is a time-consuming and complex task, involving different companies (car manufacturers, suppliers, translators, designers) and teams with different backgrounds. One way to improve the current problems arising from communication and documentation deficits is to formalize the specification to make it easier to read, to structure and to analyze. The project automotiveHMI aims to create a domain-specific modeling language for HMI development in the automotive industry. As part of the project, current specification processes and artifacts as well as the related roles were analyzed. During the analysis 18 criteria which should be fulfilled by a domain-specific modeling language have been identified. The criteria are used to evaluate existing modeling languages and to set objectives for the development of a new modeling language focusing the cross-company and cross-team development of model-based HMIs in the automotive industry.

Keywords: Automotive User Interface, Model-Based User Interface Development, Task Analysis, User Roles.

1 Introduction

Different studies show (e.g., [1]) that over 80% of today's innovations in the automotive industry are based on car electronics and its software. These innovations can be categorized into hidden technologies (e.g., ASP, ESP), comfort functions (e.g., navigation, communication, entertainment) or driver assistance (e.g., distance checking). Especially the last two categories have to be configurable by the driver and therefore require a certain amount of driver interaction. This results in a need for a modern and consistent human-machine-interface (HMI) which on one hand allows the configuration of these systems but which on the other hand conforms to the specialized requirements of the automotive industry. Some of these requirements are:

M. Kurosu (Ed.): Human-Computer Interaction, Part I, HCII 2013, LNCS 8004, pp. 411–420, 2013.

the interaction devices have to be integrated into a limited space; the HMI has to be intuitively usable and adaptable, since drivers generally do not get an extensive explanation and the HMI has to be very easy to use and should distract the driver as little as possible from his main task of driving.

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 HMIs. One main source of problems in the development of HMIs is the communication overhead caused by informal specifications [2]. 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 HMIs is researched in the context of model-based user interface development (MBUID) [3] and resulted in a vast number of specification languages for different aspects of HMIs in the last years. To find out which of the existing modeling languages suits the purpose of describing an HMI in the context of the automotive industry (if such a language even exists), a large-scaled analysis was performed in the German automotive industry.

The rest of the paper is structured as follows: In section 2, related work concerning description languages for HMIs are discussed. In section 3, an introduction to the project automotiveHMI will be given. Section 4 will shortly present the results of a process analysis that was undertaken in cooperation with the different project partners. Section 5 will discuss criteria for HMI modeling languages that were derived from the analysis. These criteria are then used to analyze existing modeling languages for their suitability in future HMI development processes in the automotive industry. In section 6 we will give a summary and an outlook on future work.

2 Related Work

A vast number of (XML-based) User Interface Description Languages (UIDLs) exist already in the field of MBUID. Some of the UIDLs are already standardized by e.g., OASIS, currently being standardized by e.g. W3C[1] and are subject of a continuous development process. Numerous projects and applications prove their practical suitability. Examples of a UIDL are UsiXML [4] or XIML [5].

The purpose of using a UIDL to develop a HMI is to systematize the HMI development process. UIDLs enable the developer to systematically break down a HMI 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 HMI 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 HMI. In the relevant literature several authors

[1] http://www.w3.org/2011/mbui

struggled with the challenge of a clear comparison of existing UIDLs (e.g., [6], [7]). These comparisons were mostly done on a theoretically basis by checking the language against some defined criteria. However, a comprehensive comparison focusing the state-of-the-practice (especially concerning real-world usage of model-based user interface development) in the industry with underlying real projects is yet to be drawn [8].

The aim of MBUID is to describe the properties of an HMI in sufficient detail and formality so that (semi-)automatic model transformations can be applied to generate the final HMI. The Cameleon Reference Framework (CRF) [9] identifies four different layers of a HMI in a MBUID process.

3 The Project AutomotiveHMI

The publicly funded research project automotiveHMI (http://www.automotive-hmi.org) consists of eleven partners of the German automotive industry, two research institutes as well as one association. The industry members are car manufacturers (Original Equipment Manufacturer - OEM), suppliers and tool developers which together cover the complete development chain for automotive HMI-systems [10].

Current HMI development processes are characterized by different, inconsistent workflows and heterogeneous tool chains. The exchanged requirements are often inconsistent, redundant, incomplete and in general not machine-readable. These circumstances lead to tremendous communication efforts between OEMs and their suppliers until both get the same understanding of the new HMI-system. Mistakes and bugs are often first noticed when the supplier delivers the first software-version back to the OEM. At that late time modifications are very cost intensive and change-request negotiations become a common annoyance. A further problem is the wide range of actors from many different branches that are involved in the automotive HMI development: Computer scientists and electrical engineers work together with designers, ergonomists and psychologists in interdisciplinary teams. The HMI modeling language that is to be developed shall serve as the connective link (lingua franca) between these actors. On this account the modeling language has to be domain specific. Domain specific languages (DSL) are dedicated to a particular problem domain and their “vocabulary” is generally based upon common expressions that are typical for the domain. Thus DSLs are far more expressive in their domain than general-purpose languages would be.

The project goal of automotiveHMI is to create a modeling language that will allow the different members of the development chain to specify and exchange the requirements and artifacts for HMI development. This modeling language forms a common data interface for the different process members. It allows bridging the “digital gap” currently found in today’s HMI development due to the document-driven exchange of specification data. Building on this data interface, it is possible to exchange digital information for use in development tools without information breaks.

This results in a more efficient collaboration of development members and allows further the digital convergence of different HMI subsystems (e.g. navigation, communication, car systems). Furthermore, the modeling language provides domain-specific benefits [11], since it is tailored to the specific needs of the automotive industry: it uses terms and meanings of the domain; it allows domain experts to create and review the specification; code and domain are being brought closer together and the productivity is increased, since changes are done on the domain level, not the code level.

4 Analysis of the Development Process

As a prerequisite to the creation of the specification language, a certain selection of project and associated partners (OEMs as well as suppliers and tool developers) were interviewed first to get a comprehensive insight into the automotive HMI development processes and second to identify specific roles and requirements for the specification language. Nine partners (who are representative for the German automotive industry) were questioned in a three-step process using a questionnaire with over 160 questions as well as a day-long workshop with 2 to 10 stakeholders per partner. The overall manpower invested by all partners in the analysis phase is more than 25 person months. The results were then combined into documents that were send to the partners for review and validation. The questionnaire ranged from questions about the used software and methods, specifics about the specification and test processes and personal experience with model-based approaches. A larger part of the analysis was devoted to a free-form discussion about likes and dislikes regarding software tools, personal wishes for a future development process and currently unsolved problems in the HMI development. The reviewed results were then combined to form the abstract reference process [12] as well as the partner-independent roles and criteria.

In total we identified 7 different partner-independent roles in the development process: Interaction Designer, Screen Designers, Hardware Developer, Software Developer, Tester, Reviewer and Translator.

5 Criteria and Their Application on UIDLs

This section presents the identified and validated industrial criteria derived from the results of the analysis as well as their application on existing UIDLs.

5.1 Criteria

Table 1 describes the derived criteria as well as the roles that require them. Criteria with “General” as requiring parties are those which cannot be mapped to specific roles but were mentioned as part of overall innovations.

Table 1. The 18 criteria

C1: Versioning (Required by: Interaction Designers, Software Developers, Reviewers)
One of the main problems currently perceived is that there are different versions of the specification that are simultaneously been used. Interaction Designers refine or even change requirements that have an impact on many parts of the implementation. Screen Designers make changes to screens (affecting the implementation of that screen) or screen elements (affecting every screen that incorporates these elements). Meanwhile, Software Developers implement according to an older version and Testers derive test cases also according to the older version. Reviewers also analyze the specification and currently face the problem of either restarting the whole review activity or overlook subtle changes if a new version is released. With subcontractors of the automotive supplier added to the picture, there are multiple versions of the specification being used as a base for development activities during the project. Currently, there is no systematic way to identify the differences between these versions, which results in changes being overlooked or old problems remaining unresolved. A modeling language that allows automatic identification of these differences would reduce the inconsistency problems.

C2: Management of Variants (Required by: Interaction Designers, Screen Designers)
In the early phases of specification but also every time major changes are required, there might be different design alternatives being discussed. The discussions result in one of the variants being picked and the others being discarded. These decisions are not documented systematically and therefore are not traceable. If a certain design decision is questioned later in the development process, it is difficult to argue why this was done. Also later high-level changes (e.g. to remove or alter features) might make those design alternatives interesting again. It should therefore be supported to annotate parts of the specification as variants that were discarded or to document groups of design alternatives together with the decision systematically.

C3: Definition of State Machines (Required by: Interaction Designers)
The dynamic properties of an HMI are typically modeled as a state machine. Each state relates to a screen of the final HMI and is annotated with information about interaction elements or information shown to the driver. Each transition stands for a change from one screen to another either as the result of driver interaction (e.g., pressing a button or making a selection) or a system event (e.g., a popup indicating a warning). The state machine contains information that is both easy to formalize and can be used to automatically derive data for other roles (e.g., testing). Currently, state machines are specified using presentation or spreadsheet programs, resulting in specifications that cannot be automatically checked or transformed.

C4: Support for Abstract Screens (Required by: Interaction Designers, Screen Designers)
An important part of the communication between Interaction and Screen Designers is the definition of abstract screens. These screens describe the available interaction

Table 1. (*continued*)

elements (e.g. selecting a radio station or informing the driver about traffic) without restricting the Screen Designer to a specific modality or layout. These abstract screens are currently specified in documents, limiting the readability and therefore the usefulness for automatic transformations. Also, finding the required abstract elements for a screen in a specification document is more difficult compared to a possible plug-in for the Screen Designer's graphics editing program.

C5: Support for Concrete Screens (Required by: Screen Designers, Software Developers)
Screen Designers provide examples for layout and design as part of the specification. Currently, the different formats (standard graphics formats or sound formats) are difficult to integrate in the word-processor-written specification, resulting in consistency problems since these examples are stored externally. Automatic tracking of the consistency of the examples with the rest of the specification could solve these problems.

C6: Annotations (Required by: Interaction Designers, Software Developers, Reviewers)
The main goal of the modeling language is to reduce media breaks, therefore keeping a consistent body of information related to HMI specification and development. Informal notes and annotations on different parts of the specification allow designers and reviewers keep reminders, questions and to-dos in place with the related information. It can also be used to extend the informal parts of the specification on demand to make up for formal constructs that the language currently lacks.

C7: Definition of Informal Requirements (Required by: Interaction Designers, Screen Designers)
An ideal specification would be completely formal, with no need for interpretation, no inconsistencies and no duplicate information. In a real development process, this level of formality is neither feasible nor desired. Sometimes it is easier to add informal text to a transition or screen than to specify it exactly instead of conveying the intent. It is therefore important to be able to annotate formal information like state machines or requirements with pictures, drawings or informal text. This high-level information also helps the creators of the specification in the creation process, since it allows to mark parts of the specification for later revisiting or to document intermediary results.

C8: Metadata for Requirements (Required by: Interaction Designers, Software Developers)
Currently, specification documents are very large and therefore difficult to handle. This includes checking for consistency, checking for changes between versions and finding of relevant parts for a certain activity. To improve the first and last point, metadata for requirements could help to sort and filter requirements based on properties such as author, change date, related parts of the HMI or type of requirement. Metadata could also be used to keep outdated requirements or design alternatives in the specification for traceability, decision management and documentation reasons.

Table 1. (*continued*)

C9: Traceability for Requirements (Required by: Interaction Designers, Software Developers, Reviewers)
Since the specification is subject to frequent changes of varying severity, quickly identifying the affected parts of the specification when changing or removing a requirement helps in ensuring consistency and making sure that no outdated information remains in the specification. A way to allow this is to include traceability information in the specification. Artifacts of the development process are related to one another (e.g. if a requirement is refined, links to its refinements are created), allowing automatic retrieval of all affected parts.

C10: Referencing External Data (Required by: Interaction Designers, Software Developers)
The specification of an HMI consists of several types of information, including examples, like animations, prototypes and mockups. Since the examples are too informal to be integrated into the modeling language, they have to be referenced unambiguously. Generally, there are different types of information in the development of an HMI which allow different levels of abstraction or formalization. It is useful for automatic transformation to formalize those where it makes sense. Less formal information should generally be allowed to be stored in an external location and be referenced where appropriate.

C11: Open, Standardized and Free Modeling Language (Required by: General)
There have been efforts in the past (e.g., [2], [14] [15] [16]) to define modeling languages tailored for the automotive industry. Most of these failed because they had no support from tool vendors. An important requirement for a language that can be used in industrial development is that the language is open, standardized and free. Standardization allows different companies to create tools or plug-ins to other tools that allow interoperability. Since adopting a standard requires changes, the transition can be made easier when the language carries no license costs. Openness guarantees that the language can be extended to contain additional information that might become useful in the future.

C12: Simple Integration with Tools (Required by: Interaction Designers, Screen Designers, Software Developers, Reviewers, Translators)
Currently, exchange of information between roles is achieved through text, images or example programs. Even if specialized tools that support modeling exist, the models have to be exported as text or pictures, hindering automatic import at the receiver's side. A modeling language that can hold all the information of the HMI design process needs to be easily integratable with current generic and specialized tools to be useful.

C13: Support for Iterative Approach (Required by: Interaction Designers, Hardware Developers, Software Developers)
Even without drastic high-level changes like the addition or removal of features, HMI development is iterative in nature. A modeling language for HMI specification must therefore support different versions as well as an easy way to compare or merge these versions.

Table 1. (*continued*)

C14: Support for Prototyping (Required by: Interaction Designers)
Currently, when specifying the dynamic behavior of a user interface, interaction developers have no way to actually experience what they specified. They have to wait until the implementation is done to evaluate the usability and to suggest improvements, which results in higher costs, since it occurs late in the development process.

C15: Support for Different Modalities (Required by: Interaction Designers, Hardware Developers, Software Developers)
Current HMIs already support different input and output modalities. While visual output is the main output modality, warning signals, tactile feedback and speech output is also used in current automobiles. Speech input is a newer feature that complements buttons and touchscreens as a further way to interact with the HMI. Since the future is likely to improve these modalities or even add further ones, a modeling language should support specifications for different modalities.

C16: Flexibility of Processes (Required by: General)
Industry processes are subject to continuous improvement. Also different vendors or suppliers use different development processes. A modeling language should therefore be flexible enough to be used with any process. It is important that the language puts no constraints on the development process. It would otherwise hinder innovations which would result in poor acceptance.

C17: Sufficient Abstraction of the Modeling Language (Required by: Hardware Developers, Software Developers, Testers)
A current problem of HMI languages is their concrete applicability. They are either very specialized (e.g. to support specific development processes or to be used as an exchange format for developer tools) or they are too generic (e.g. to describe abstract user interfaces (AUI). For a language to be usable in an industrial development process, it has to describe dynamic and static properties of a user interface on different abstraction levels without imposing unnecessary constraints on the development process.

C18: Referencing of Information (Required by: Interaction Designers)
Redundant information in a specification decreases maintainability and increases the chance for contradictions. Information should therefore not be repeated, but instead referenced. For example, instead of defining the position of a widget that appears on a number of screens, the position could be put into a template that serves as the base of these screens.

5.2 Application of the Criteria and Evaluating Existing Approaches

The 18 criteria were applied to a number of current available UIDLs, including those with a focus on the automotive industry as well as domain-independent ones.

Four UIDLs developed by the automotive industry (Infotainment Markup Language (IML) [15], OEM XML [14], AbstractHMI [16] and ICUC XML Daimler

[2]) and seven general UIDLs (e.g., Dialog and Interface Specification Language (DISL) [13] and USer Interface eXtensible Markup Language (UsiXML) [4]) were analyzed. The modeling languages were analyzed using the 18 criteria. A summary of the results are shown in Fig. 1. Generally, OEM-XML met the most criteria (9 of 18), followed by ICUC and UsiXML. However, this means that the most suitable language only met half of the criteria required by the industry.

None of the languages supported C1, C7, C8 or C9. While the automotive languages matched more criteria than the general ones, they still did not provide enough features to be used in all stages of the actual development. It was therefore concluded that a new language is required by the industry that matches all 18 requirements.

	C1: Versioning	C2: Management of variants	C3: Definition of state machines	C4: Support for abstract screens	C5: Support for concrete screens	C6: Annotations	C7: Definition of informal requirements	C8: Metadata for requirements	C9: Traceability for requirements	C10: Referencing external data	C11: Open, standardized and free modeling language	C12: Simple integration with tools	C13: Support for iterative approach	C14: Support for prototyping	C15: Support for differetnt modalities	C16: Flexibility of processes	C17: Sufficient abstraction of the modeling language	C18: Referencing of information	**Sum**
IML					X					X			X		X	X		X	6
OEM-XML		X	X		X					X			X	X	X	X		X	9
ICUC			X		X					X			X	X		X		X	7
AbstractHMI			X		X					X			X			X		X	6
Teresa-XML				X								X							2
UIML					X					X	X				X	X		X	6
useGUI					X						X		X	X		X		X	6
DISL			X	X															2
UsiXML				X	X					X	X		X		X	X			7
XIML				X	X	X				X						X		X	6
SEESCOA XML				X						X						X			3
Sum	0	1	4	5	8	1	0	0	0	8	3	1	6	3	4	9	0	7	

Fig. 1. Results of the criteria application

6 Summary and Outlook

This paper presented (parts of) a large-scale analysis performed on automotive HMI development processes in the context of the project automotiveHMI. The analysis led to the identification of representative roles and requirements (criteria) which are relevant for the definition of a HMI modeling language in the automotive industry. After presenting the criteria, it was discussed how today's domain-specific as well as general-purpose UIDLs do not match these criteria, which led to the need for a new, domain-specific language that could become an industry standard for the (German)

automotive industry. The project automotiveHMI is currently working on the realization of this domain-specific language.

Acknowledgements. The research described in this paper was conducted within the project automotiveHMI. automotiveHMI is funded by the German Federal Ministry of Economics and Technology under grant number 01MS11007.

References

1. Danneberg, J., Burgard, J.: A comprehensive study on innovation in the automotive industry, http://www.oliverwyman.com/pdf_files/CarInnovation2015_engl.pdf
2. Huebner, M., Gruell, I.: ICUC-XML Format. Format Specification Revision 14. Elektrobit (2007)
3. Hussmann, H., Meixner, G., Zuehlke, D. (eds.): Model-Driven Development of Advanced User Interfaces. SCI, vol. 340. Springer, Heidelberg (2011)
4. Vanderdonckt, J., Limbourg, Q., Michotte, B.: USIXML: A User Interface Description Language for Specifying Multimodal User Interfaces. In: Proc. of the W3C Workshop on Multimodal Interaction, Sophia Antipolis, France (2004)
5. Puerta, A., Eisenstein, J.: XIML: A Universal Language for User Interfaces. RedWhale Software, http://www.ximl.org/pages/docs.asp
6. Souchon, N., Vanderdonckt, J.: A Review of XML-Compliant User Interface Description Languages. In: Jorge, J.A., Jardim Nunes, N., Falcão e Cunha, J. (eds.) DSV-IS 2003. LNCS, vol. 2844, pp. 377–391. Springer, Heidelberg (2003)
7. Guerrero García, J., González Calleros, J., Vanderdonckt, J.: A Theoretical Survey of User Interface Description Languages: Preliminary Results. In: Proc. of Joint 4th Latin American Conference on Human-Computer Interaction, Los Alamitos, USA (2009)
8. Meixner, G., Paternó, F., Vanderdonckt, J.: Past, Present, and Future of Model-Based User Interface Development. i-com 10(3), 2–11 (2011)
9. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computer 15(3), 289–308 (2003)
10. Kuemmerling, M., Meixner, G.: Model-Based User Interface Development in the Automotive Industry. In: Proc. of the 3rd International Workshop on Multimodal Interfaces for Automotive Applications, Palo Alto, USA, pp. 41–44 (2010)
11. Ghosh, D.: DSL for the Uninitiated. Communications of the ACM 54(7), 44–50 (2011)
12. Hess, S., Gross, A., Maier, A., Orfgen, M., Meixner, G.: Standardizing Model-Based IVI Development in the German Automotive Industry. In: Proc. of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, Portsmouth, USA, pp. 59–66 (2012)
13. Bleul, S., Schaefer, R., Mueller, W.: Multimodal Dialog Description for Mobile Devices. In: Proc. of the Workshop on XML-based User Interface Description Languages, Gallipoli, Italy (2004)
14. Brunhorn, J.: XML-Sprache zur Beschreibung von HMIs für Infotainmentsysteme und Kombiinstrumente. Language Specification 1.0. OEM Arbeitskreis HMI Methodik (2007)
15. Jud, A.: Präzise Syntaxdefinition einer Modellierungstechnik für Infotainment-Systeme. Master-Thesis, Technische Universität Berlin (2007)
16. Reich, B.: Abstrakte Beschreibung automobiler HMI-Systeme und deren Erweiterung für neue Dienste. Master-Thesis, Universität Ulm (2008)