A Model-based Approach towards Human-Machine-Interfaces

Rainer Holve, Silke Goronzy

A Model-based Approach towards Human-Machine-Interfaces

Rainer Holve[1], Silke Goronzy[2]

1: 3SOFT GmbH, Frauenweiherstraße 14, D-91058 Erlangen, Germany
2: 3SOFT GmbH, Frauenweiherstraße 14, D-91058 Erlangen, Germany

Abstract: Specification and development of Human-Machine-Interfaces (HMI) for infotainment systems pose a difficult challenge for those automotive OEMs who claim this topic as one of their core competences. Experience shows that a major problem arises from the non optimal communication of all involved parties during the HMI development. This article describes a model-based approach towards HMI-development which combines conceptual design, specification and implementation based on a single complete HMI-model which serves as an improved means of communication between designers, ergonomists and software developers.

Keywords: Human Machine Interfaces, model based development

1. Introduction

The complexity of display-based driver information systems (DIS) in passenger cars steadily increases. While current vehicles face the driver with usually a single monitor, future systems will include fully graphics based cluster instruments, head-up displays and monitors for rear seat entertainment – all of this probably combined with speech dialogue systems. Thus, the DIS gain such a prominent role inside the vehicle that they have a considerable impact on the entire vehicle's usability and the Human Machine Interfaces (HMI) of the information systems become key factors for developing and preserving customer retention. It is no surprise that the leading automotive manufacturers (OEM) – especially those who offer cars in the premium segment – now regard HMI related issues as their core competence and no longer rely on first tier suppliers' "off the shelf" solutions.

2. HMI as OEM Core Competence

Currently, the approaches followed by the OEMs to express their HMI-related core competence can be characterized into three levels, each coming with certain tasks and challenges for both, the OEM and the first tier supplier.

2.1 Select and Modify

In this weakest form of competence, the major burden of HMI design and development remains at the first tier supplier. However, the OEM expects to be presented with various different designs from which he can choose. Also, the OEM might demand minor changes to the design such as varying the colour schemes or selecting certain fonts and bitmaps. Therefore, the OEM needs personnel with a background in design and human factors who can make qualified decisions while evaluating the design proposals of the first tier. The supplier on the other hand must be able to produce and modify several different HMI designs and present them in a form suitable for a qualified evaluation by the OEM (rapid prototyping, simulation).

2.2 Design and Specify

The currently most common approach followed by the leading OEMs focuses on a complete and detailed specification of the HMI related issues. Here, the OEM determines the look and feel for a new DIS by a user centred design process [1], i.e. a large number of different aspects need to be addressed like fonts, bitmaps, layouts, application flow, internationalization, speech dialogue etc. For this, the OEM needs to employ a group of highly skilled people who know how to use various design tools, perform usability studies and communicate their design to the first tier supplier. The supplier needs to establish a software architecture which allows to easily exchange one OEM-specific HMI with another and to incorporate the specification provided by the OEM as efficiently as possible.

2.3 Design and Develop

In this strongest form of expressing HMI competence, the OEM not only precisely specifies the HMI for his new system (see 2.2) but also performs the implementation of parts of the HMI software himself [2]. So, additionally to the detailed specification, the OEM has to deal with all responsibilities that come with SW-development including the maintenance phase which easily stretches over several years after the initial launch of the new DIS. The advantage of this approach lies clearly in a maximized reuse of the HMI software and thus drastically reduced development costs. The first tier supplier now faces the task to integrate software components developed by a new party and he has to provide a software platform that can easily be adapted to different APIs as they are required / expected by the OEM.

3. Problematic Aspects of HMI Development

Regardless of the level of competence followed by the OEM, a development project for a new DIS HMI has to deal with a number of issues which are responsible for occurring problems.

3.1 Time Pressure

Currently – and very probably in the future as well – OEMs are integrating more and more features to their automobiles. Most of these features are software driven and either provide more information to the driver or assist him/her in handling the vehicle. It is reasonable to assume that the bulk of these new functions will be integrated in or accessible through the DIS, causing an increased complexity of the HMI. While this alone is not a major problem per se, OEMs are reacting to the diversification of the automotive market by offering multiple variants of vehicles in different geographical market segments. These two aspects have to be managed within steadily shortened development cycles.

3.2 Change Pressure

As mentioned above, an increasing number of features are to be integrated in a decreasing amount of time. Driven by the very short development cycles of e.g. the consumer electronics market, features are bound to be modified, added or removed from the current planning on a very short notice. Also, many changes become necessary when technical restraints of the target hardware and/or software platform emerge when the development is already in an advanced stadium. Lastly, it is not unheard of that certain changes are caused by "management override", simply because a certain aspect of a HMI's look & feel does not appeal to executives who review the finished system only at milestones with a relatively high grade of completion.

3.3 Communication Barrier

Through the mentioned aspects of time and change pressure, a third facet gains increasing relevance – the communication between all parties in the development process of the HMI. Currently, in most projects the communication flow is similar to the one described in Fig.1. On the OEM side, driven by the desire to describe the newly developed HMI as precisely as possible, a very heterogeneous team of graphics designers, ergonomists, speech dialogue designers, translators etc. create a set of specification documents. This set of documents (mostly simply paper based and not machine readable) is then handed over to the developer team of the first tier supplier. While this approach is – in theory – perfectly valid, real projects suffer from a couple of drawbacks.

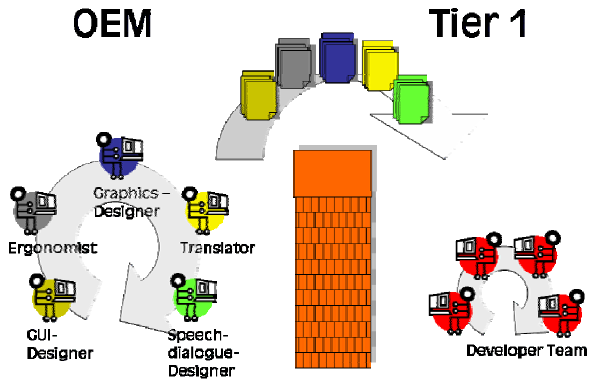

Figure 1: Classical communication flow

Firstly, the people involved all come from different areas of expertise and - necessarily - approach the project from very different viewpoints. Naturally, everyone expresses him/herself in a way that is most convenient and most suitable to the task at hand. As a result, the developers of the first tier supplier – mostly engineers – are faced with a multitude of specification documents, created with different tools, containing information in different "languages". In the end, only the final implementation of the target system will add a real semantic to the specification of the OEM since there is no way to get a complete impression of all facets covered in the specification documents.

Secondly, driven by time and change pressure, the specifications are not simply created once and then left to the OEM for implementation. Rather, the OEM's designers will expand and modify the specifications several times, even when the implementation process of the first tier is right under way. Thus, the processes on the OEM side and the tier one side are parallel and iterative. Now things can become really complicated: Is the modified specification still consistent (i.e. are all cross references between the different documents up to date)? Are all parties involved informed in time about the modifications? How do changes fit into the feature release schedule? Which sample of specifications is currently in effect as agreed upon by the OEM and the tier one?

3.4 Room for Improvement?

Given these aspects – time pressure, change pressure and communication difficulties – which when combined pose a serious threat to a successful project completion – one might ask where the room for improvement lies. Realistically, time pressure and need for changes are very much market driven and can hardly be averted from within the HMI development. Rather, we will probably have to cope with the fact that specification and realization

of HMI related issues are iterative processes and will take place in parallel.
That leaves us with the communication process used to transport ideas between the involved parties and the way they interact.

4. Model-based HMI Development

4.1 Principal idea

Instead of the "classical" and mostly unidirectional communication flow from the OEM to the tier one, based on paper bound specification documents, we propose a model-based approach (see Fig. 2).

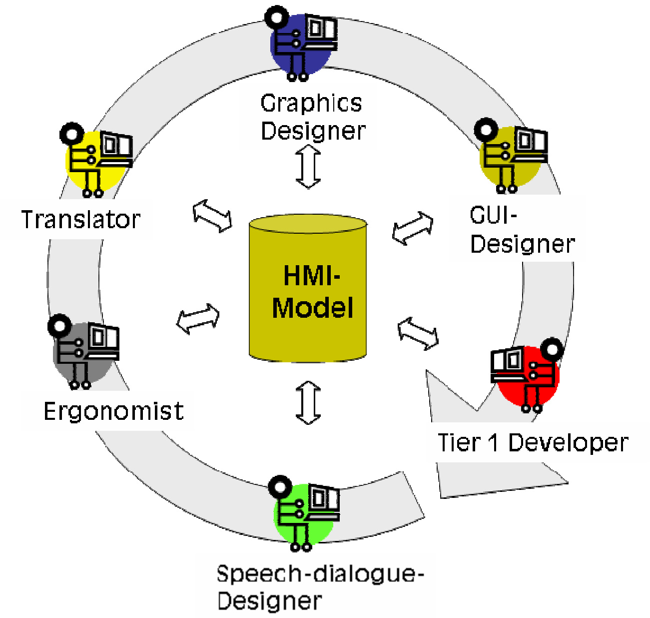

Figure 2: Model-based HMI development

The idea behind this is to establish a cooperative process around a central HMI model that contains all relevant information concerning the HMI in a uniform way.

4.2 Benefits of the model-based approach

The main goal one hopes to achieve with this approach is to keep the HMI model (i.e. the description of how the final HMI should look like) consistent at all times and to get all involved parties informed immediately when changes to the model are made. Additionally, a model that contains all relevant data in a machine readable fashion lends itself to an early simulation and rapid prototyping of the HMI. Thus, the HMI can be evaluated, "debugged" and presented for acceptance quite early in the specification process, even before the tier one starts the implementation. The tier one on the other hand can draw a massive profit from the uniform HMI model by incorporating it in his software development process by the means of automatic code generation. This can drastically decrease the development time and costs, improve the reactivity to change requests of the OEM while increasing the quality of the HMI software at the same time.

5. Requirements for a model-based approach

In order to perform properly and to deliver the mentioned advantages, there are a set of certain requirements for the HMI model, the tools used to modify it and the way to cooperate with it.

5.1 Abstraction versus detailedness

As mentioned above, the people involved in the HMI development, coming from the OEM and the supplier, form a very interdisciplinary group. Thus, the semantic features of the HMI model must be flexible enough to express the properties of the modeled HMI in different grades of abstraction according to the needs of the various involved designers/ engineers.
To give an example: Modeling, modifying, simulation and evaluation of a HMI could (and quite often actually is – at least in the early phase) quite easily be carried out by means of macromedia flash on the side of the OEM. For the tier one on the other hand such a model is hardly more useful than a paper based specification since the suitability for code generation of a flash animation is virtually nil.
The other extreme would be to have the ergonomists and dialogue designers specify the HMI in form of configuration scripts suitable for a code generator which outputs target code for the embedded hardware platform of the tier one. This is also hardly a passable way.
Thus, the means of expression provided by the HMI model have to perform a "balancing act" between the abstract view HMI designers favor and the grade of detailedness needed by the software engineers to incorporate the model into their code generation process.

5.2 Completeness

In order to not leave out certain facets of the developed HMI, the model has to cover all relevant aspects in a suitable form. To mention the most important:

<u>Graphics/Layout</u>
This concerns the arrangement of single screen menus (other common terms are view, panel or mask) as well as the look and behavior of the basic graphical elements (widgets) like e.g. buttons, sliders, dropdown-boxes or spellers. The choice of a suitable font and font sizes as well as different color-schemes for e.g. daylight and night-design are also important.

Application flow

A modern driver information system covers literally hundreds of different functions and is therefore composed of a huge number of distinct menus. These functional menus must be arranged in such a fashion that the user can easily learn and memorize how to use the system. While the application flow is mainly driven by the user's inputs via keys, rotary knobs or voice commands, internal system events must also be considered

Internationalisation

Current DIS offer the possibility to change the system's language at runtime. So, all static HMI texts have to be defined in several languages and one has to consider different character sets as well as language dependent icons and units (liter-gallon, km/h – mph, bar - psi etc.)

Variants

Although The OEM wish to establish a brand specific HMI that is as uniform as possible, it is necessary to consider variants of a HMI's look &feel. This stems from different equipments throughout an OEM's models (e.g. color/monochrom) as well as from different customer expectations in the different markets, which require different application flows.

Speech dialogue

The increasing functionality of DIS leads to an increasing utilization of speech dialogue systems (SDS) in vehicles. A complete HMI concept must therefore consider a suitable combination of graphics, haptics and speech to feature a multi-modal dialogue (see also [3]).

5.3 Ability to simulate

In order to make maximum use of a model-driven development, the HMI model must lend itself to a simulation that comes as closely as possible to reality and recognizes all aspects covered by the model. Ideally, the simulation is permanently accessible during specification without the necessity of model transformation, compilation or invocation of additional tools.

5.4 Machine readability

Regarding the requirements mentioned as well as the described problems with the paper based specification, this demand certainly goes without saying. It is also a necessity when one thinks about how a model driven approach can be integrated in the already existing processes and tools used by the designers and developers. Generally, the format of the HMI model is arbitrary, but the general acceptance of XML and the wide availability of parsers and tools strongly recommend a XML-based storage format.

5.5 Tool based modeling process

The complexity of a HMI model that serves the desired purposes is necessarily very high. Thus, in order to work with such a model, we need a modelling tool (or a set of tools) that supports the modelling process in all stages of development and provides access to all aspects of the model's semantics. Specifically, the toolbase needs to fulfil the following requirements:

Specialized editors

The many different aspects of a complete HMI model as well as the different people who will create the model make the necessity for specialized editors apparent. The editors should provide access to the semantic model facets in a fashion that suits the professional domain of the most likely user as well as possible.

Simulation

The simulation of the HMI model should be integrated in the tool suite as seamlessly as possible, so that the user can evaluate his modifications to the model "on the fly" without the need to transform the model code or to launch external tools.

Preservation of model consistance

A HMI model of any kind that tries to fulfil the mentioned requirements will consist of literally thousands of different model elements that are highly connected and dependent on each other. Unfortunately, the more interdependencies a model has, the easier it is to mess it up with a thoughtless modification, e.g. the deletion of an element that other elements still depend on. Therefore, a tool base for HMI modelling should offer permanent consistency and syntax checks as well as the ability to gather information about the model itself, e.g. which and how many other model elements depend on a particular element.

Multi-user support

Since the whole approach described in this article circulates around a centralized HMI model which is accessed by multiple users, the support of multiple parallel accesses to the model as well as e versioning system is indispensable.

Embedding of code generators

In order to enable the tier one supplier to integrate the HMI model into his software development process, it is preferable if the modelling tool allows embedding code generators which transform certain model facets to source code for the target system. While this feature is not absolutely necessary, since the code generation can as well be based on the XML format of the HMI model as input, the direct embedding of code generators can help to speed up the development process.

Openness for extensions

This requirement stems from two sources. Firstly, the integration of a model based approach into an already existing development process calls for the ability to adapt to the existing process by integrating available or newly developed tools into the modelling tool base. Secondly, technological progress in HMI related issues will confront us with new technologies, paradigms and requirements. In order to be able to adapt the HMI modelling process to these new challenges, the ability to add new features to the tool base is essential.

6. A proposal for a HMI model

In this chapter we briefly describe the structure of a HMI model that has been applied successfully to the development of several automotive and non-automotive HMI development projects.

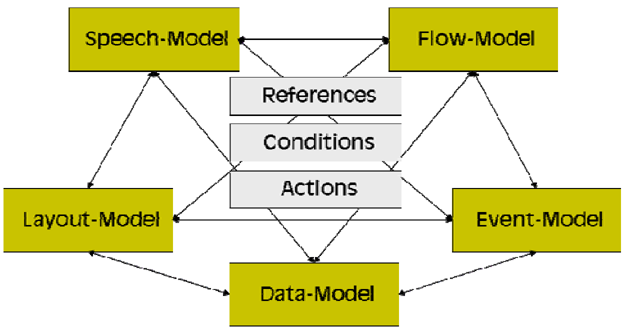

Figure 3: Structure of a HMI model

As shown in Fig. 3, the model consists of five sub-models which are interconnected by references, actions and condition. Combined, this features a HMI model that is suitable for simulation as well as for code generation for embedded target systems.
The different sub-models are depicted in the subsequent sections.

6.1 Data-Model

The requirement of an immediate “on-the-fly” simulation of the HMI model makes it necessary to sufficiently describe all data represented by the HMI during the specification.
The proposed HMI model utilizes the principle of a globally visible “data-pool” which contains all data relevant to the HMI. The other sub-models can arbitrarily refer to this data and request a notification if certain data is modified (observer pattern). Assumptions about the origin of the data are not necessary; the data-model therefore follows the motto:

„I know that the data exists
and I notice when it changes,
but I don’t care where it’s coming from.”

Once defined, all elements of the data-model are available to all other sub-models and a notification mechanism informs about changes to the data-model anytime. A variant mechanism within the proposed HMI model allows for holding certain subsets of the data-model in arbitrarily many variants which enables the modeler to handle aspects like skinning (daylight, nighttime design) and internationalization (different text translations and language dependent layout parameters).

6.2 Event-model

The event-model describes all events, which affect the specified HMI (e.g. user input via keyboard or speech, external events like an incoming phone call or system events like error messages or warnings). As the content of the data-model, elements of the event-model are globally visible and can be referenced, fired ore processed by other sub-models.

6.3 Layout-model

The layout-model contains the description of all views (also called masks, panels, screens, contexts) within the HMI-model. Views are constructed from widgets (graphical base elements with a predefined behaviour like buttons, sliders, checkboxes etc.). The widgets are linked to the data-model to provide them with the necessary data as well as to the event-model to trigger actions after occurrence of certain events, e.g. user input.

6.4 Flow-model

The application-flow-model combines all defined views to a complete menu structure. The model uses UML-like statecharts which are illustrated with small depictions of the referenced views (see Fig. 4). The transitions between the certain states are labeled with events from the event-model and can also be extended with conditions to model even complex application flows.

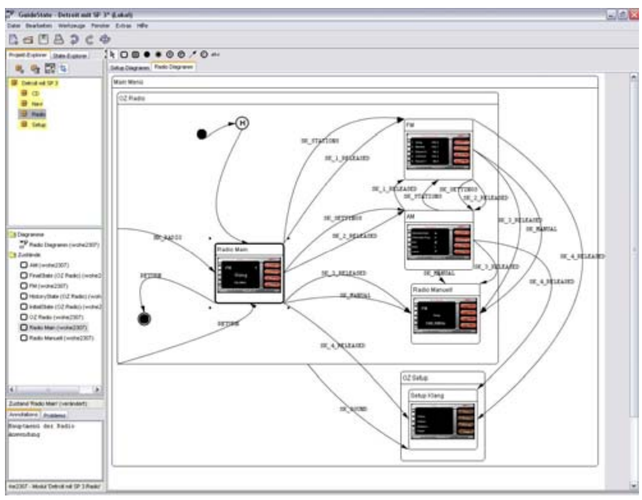

Figure 4: Modeling the application flow

6.5 Speech-model

The speech-model allows to describe truly multi-modal HMIs in combination with the graphics/haptics HMI by adding vocabularies, grammars and text-to-speech prompts to the states of the flow-model and to define speech triggered events in the event-model. A detailed description of the speech-model and its integration in the overall HMI model is given in [3].

7. Model-driven HMI development process

The described HMI model has already proven suitable in various development projects for automotive as well as for non-automotive HMIs. The underlying development process, however, is very closely coupled. This means, that all involved parties are required to share a central model. While this process is quite mature, effective and very efficient, there is still room for improvement. Depending on the level of trust, the geographical proximity or the underlying business model between OEM and tier one, it might be desirable to feature a more loosely coupled development process (see Fig. 5). In such a process, the OEM periodically provides an increasingly complete simulation model to the tier one who extends the model to get an implementation model suitable for code generation. However, since the two parties no longer share a single HMI model and work quite asynchronously we now face the need for a merge process that integrates changes in the OEM's simulation model into the (older) implementation model of the tier one. The implementation of such a model-merge is quite a challenge and the efforts undertaken by the authors are not yet completed.

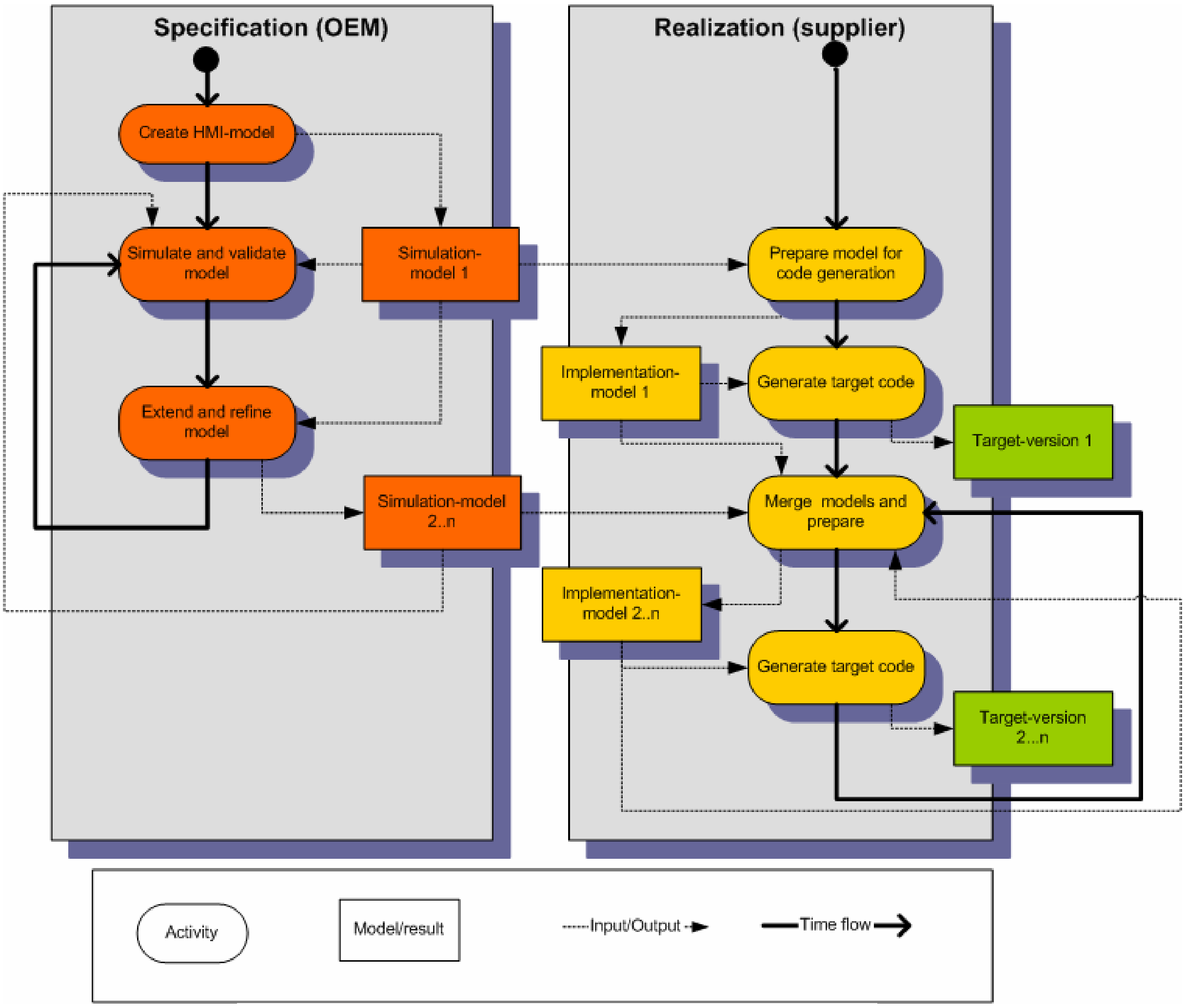

Figure 5: Loosely coupled development process

8. Conclusion

This paper has motivated the need for a model driven approach towards HMI development to overcome communication difficulties between the involved parties. The requirements for such a HMI model as well as for the underlying tool base were outlined. The described model driven approach developed by the authors has proven successful in several embedded HMI development projects. While the closely coupled model drive development process based on this approach is quite mature, further work is necessary to allow for a loosely coupled process which requires the ability to merge two HMI-models.

9. References

[1] W. Hamberger, St. Christ, T. Bouaziz: "*Toolsuite for multi-modal operating concepts: Usability-engineering from concept to code*" (in german), pp. 881 - 890, VDI-Berichte 1907, 2005.

[2] R. Bruckmeier, D. Rossberg, M. Wagner, A. Huber: "*Human-Machine-Interface: Software becomes OEM core business*" (in german), pp. 891 - 893, VDI-Berichte 1907, 2005.

[3] S. Goronzy, R.Kompe, R. Holve: "*Fully exploiting the potential of speech dialog in automotive applications*", Proceedings of 3rd European Congress ERTS – EMBEDDED REALTIME SOFTWARE, Toulouse, France, 2006.

10. Glossary

API Application Programming Interface

DIS Driver Information System

GUI Graphical User Interface

HMI Human Machine Interface

OEM Original Equipment Manufacturer

SDS Speech Dialogue System