

User interface transfer for driver information systems: a survey and an improved approach

Fabian Hüger
Volkswagen AG
Group Research
Driver Information Systems
38436 Wolfsburg, Germany
fabian.hueger@volkswagen.de

ABSTRACT

At present, devices are establishing themselves on the mobile device market which permit personalization and expansion by means of applications. These applications are increasingly also expected in the vehicle. In contrast to mobile devices, an expandable system in the vehicle is subject to special requirements. These are discussed in this publication and existing approaches are evaluated with regard to the requirements. In this case, outplacing subsequent applications to a smartphone or web server is identified as the best solution. For user interface (UI) integration to the driver information system, final user interface descriptions are identified as a good compromise between complexity and reusability. An improved approach for UI transfer with final UI descriptions using HTML is presented. Compared to existing concepts, this achieves an improvement in the form of graphical quality, response behaviour, reusability and complexity. A prototype implementation makes it clear that a series production system can be subsequently expanded with applications that cannot be distinguished from the basic functions of the driver information system in terms of graphical quality and response behaviour.

Categories and Subject Descriptors

D.23.8 [Human machine interaction],

H.5.2 [Information interfaces and presentation]: User interfaces.

General Terms

Design, Reliability, Human Factors, Legal Aspects.

Keywords

Automotive User Interface Framework, Automotive Application Platform, Connected Car, In-Vehicle Infotainment System, User Interface Description.

1. INTRODUCTION

The sales figures for smartphones such as the Apple iPhone or devices with the Android operating system indicate that personalization and expansion ability of mobile devices can be decisive sales factors: sales figures have almost doubled from 2009 to 2010 [1]. For example, Internet services can be used

anywhere with these devices. A study has shown that the young generation in particular also expects to be able to use these services in the vehicle whilst maintaining traffic safety [2]. This includes social networks, media services and current information about points of interest (POIs) such as opening hours and prices. In addition to Internet services, there is also demand for special applications to be used in emergency vehicles, such as police cars, ambulances or also in driver training vehicles.

In most current driver information systems, it is not possible to load subsequent applications or access online services. Many requirements make it difficult to implement a system that can be expanded in this way. For example, quality regulations demand a particularly stable system. Current systems are highly complex and significantly limited with regard to resources, and must therefore be retested every time a change is made. Furthermore, the user interface must comply with particular specifications in order to guarantee a dual task capability. For example, this includes certain text and button sizes or the avoidance of visual entertainment.

Upgradeable IVI systems have been discussed since 1999 [3] but first products were announced just recently. Ford [4] and BMW [5] already have products on the market which allow use of Internet services such as Facebook, Twitter or the Pandora music service in the vehicle. Audi [6] makes it possible to search for current data such as opening hours and ratings using the Google point of interest (POI) search. Nokia is pursuing another approach with various vehicle manufacturers (OEMs): its Mirror Link (formerly known as Terminal Mode) makes it possible to display and control the screen content of smartphones in the driver information system [7].

The available approaches meet the requirements to differing extents. This publication firstly explains the requirements on an expandable driver information system, then describes and evaluates the available approaches with regard to the requirements and presents a novel approach.

2. REQUIREMENTS

Expandable driver information systems have a large number of special requirements: Avoidance of driver distraction, quality with regard to stability of the system, reusability in different systems, life cycle of applications, ease of creating services, offline availability as well as access to vehicle data.

Avoidance of driver distraction represents a primary requirement. A current study shows that driver inattention due to distraction is a causal factor in 93% of rear-end collisions investigated [8]. The

EU [9], the Alliance of Automobile Manufacturers in the USA [10] and Japan [11] define rules for user interaction with driver information systems. Driver distraction should be minimized by avoiding dynamic content and time-critical interactions as well as through the selection of appropriate contrast, text and button size. For example, video playback whilst driving is suspended even in current systems. Therefore, it is not possible to show the UI of smartphone applications in the driver information system without modification. Voice control as available for key components of the driver information system should also be possible for subsequently loaded applications in the driver information system.

For example, if the applications are integrated into the driver information system via an external interface, it must be possible to ensure that only particular applications can be presented whilst driving. Unwanted applications might be those which fail to comply with statutory guidelines regarding driver distraction, and even malicious software. The possibility for product liability actions must be avoided in the event of accidents caused by the system.

The topic of quality is also highly important in the vehicle. Besides high graphical quality, fast response times and system stability are essential. For that reason, each new software version of a driver information system is intensively tested prior to release. In particular, the influence of different software components on other components is tested. Model-based and therefore partly automated processes for testing driver information systems have not become established yet [12]. It cannot be imagined that an overall system test of this kind will be possible for software components that can be loaded subsequently and dynamically. To guarantee the function of the remaining system, the logic of subsequently installed applications should be stored in separate resources.

Another challenge facing expandable driver information systems is the long vehicle life cycle of eight to ten years [13]. The available services must be provided throughout the long service life of the vehicle. This is difficult to predict in particular for external web services. The "life cycle gap" must be taken into account in approaches that use integrated smartphones for the expansions: The life cycle of smartphones is only 18 months, very different from that of vehicles [14].

If subsequently loadable applications are created by the community, as is the case with current smartphones, then it must be considered that the market for driver information systems is significantly smaller and more fragmented than the smartphone market: Each year, approximately 6 million infotainment units with navigation are sold along with new cars worldwide [15]. Smartphones sales figures have achieved almost 300 million units [1]. Possibly, a software development kit (SDK) must be designed so that services can be created easily and/or existing source code extended. Due to the fragmentation of systems between different vehicles, categories and brands, it should be possible to reuse the services in different systems in order to permit more widespread use.

A caching strategy is a good idea for applications which use online content whilst driving, because data connections when on the move are normally not stable. Page-by-page navigation through web pages has been shown by experience to be inexpedient.

The App My Ride contest by Volkswagen [16] has shown that not only currently popular applications (e.g. Twitter) are required in

the vehicle, but also and in particular applications which have access to vehicle data such as speed, position or fuel consumption.

Due to the high requirements, it is clear that the approaches familiar from the consumer area can only be transferred to driver information systems to a limited extent. Existing approaches specifically for the vehicle are presented and evaluated in the following section.

3. STATE OF THE ART

Approaches which have already been published implement the required separation of applications in different ways. The separation results in the need for integration into the UI of the driver information system or UI transfer, respectively. Here too, different methods are shown and discussed.

3.1 Separation of applications

In principle, there are four variants for separating applications from the remainder of the system: outplacing applications to a separate virtual machine (VM) within the main unit of the driver information system, outplacing to a smartphone, a separate control unit (Electronic Control Unit - ECU) or a server. Figure 1 illustrates the different variants.

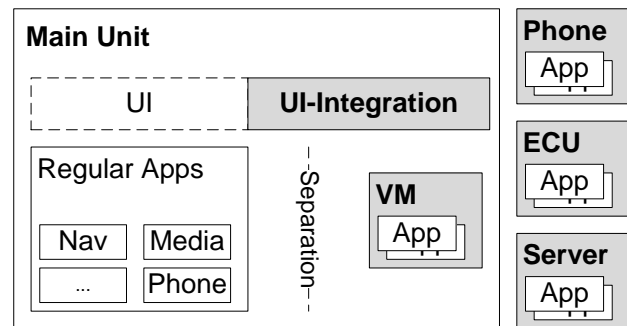


Figure 1. Different variants for separation of applications from driver information systems

Outplacing the applications within a separate VM – i.e. within a protected area – means that the remainder of the system can be left unaffected. In this case, for example, it is possible to integrate platforms from the consumer area directly [17]. A system based on virtualization is currently being developed and evaluated in the EU project entitled OVERSEE [18]. It is also possible for existing platforms such as Android to be expanded with vehicle-specific components [19]. A proof of concept [20] has indicated that simultaneous access to different partitions on external interfaces is possible in a solution of this type. Simultaneous access to graphical content and hardware availability do however still represent problems. In a virtualization solution, it may also be possible to store driver information and driver assistance functions on one device, and thereby to consolidate control units [21]. OpenSynergy [22] and Wind River [23] already have products on the market which allow virtualization. Saab has already presented a system based on Android [24].

Integration of mobile phones in driver information systems is nothing new. As early as 2002, it was possible to use driver information systems as a hands-free system for mobile phones by means of the hands-free Bluetooth profile. A further level of integration is achieved in systems such as the Audi Multimedia Device Interface [25], in which it is possible to access media on a smartphone connected via USB, for example. The next level is to

integrate smartphone applications, and this is discussed to an increasing extent in current research [7, 26, 27, 28, 29, 30]. Various series production systems are already implementing an integration: BMW [5], MINI [31] and Ford [4] have implemented systems in which specially developed smartphone applications can be presented and controlled in the driver information system. Toyota has announced a similar system [32]. BMW and MINI support the Apple iPhone in this case. The Ford and Toyota systems support Android and Blackberry devices in addition. Moreover, Ford offers the opportunity for voice control.

With its AutoLinQ, Continental has published an approach in which an adapted Android system running on a separate control unit (ECU) is presented in the driver information system [33].

Audi has implemented a server-based variant in the current series production system with its “Online Services” [6]. This allows access to content on the Audi site using the driver information system. In addition to the Google POI search, it is possible to use the data connection to access news, weather and travel information.

3.2 UI integration

The existing approaches differ from one another in the way that the UI of the external application is integrated into the driver information system. For a classification of the variants, the Cameleon reference framework [34] is suitable. It allows UI descriptions to be classified into different levels: The uppermost level is Tasks & Concepts (T&C) which describes the UI in the form of user objectives. The next level is the abstract UI (AUI) which describes the UI in the form of abstract interaction objects. Below that is the concrete UI (CUI) which describes the interaction modality dependent (e.g. in the form of buttons). The final UI (FUI) is the platform-dependent level and is divided into two sub-levels [35]: source code (any programming or markup language which is compiled or interpreted) and the running UI. The running UI can be described in the form of raster graphics (also: bitmaps), for example. Figure 2 illustrates the different abstraction levels.

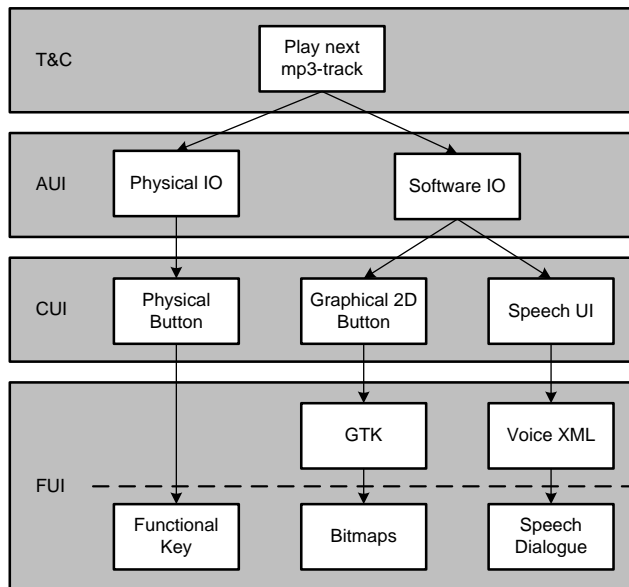


Figure 2. Abstraction levels according to the Cameleon reference framework taking an example (acc. to [36])

Existing publications implement UI integration on the AUI or FUI levels:

On the FUI level, Mirror Link uses raster graphics as the interface to the driver information system. The frame buffer of the smartphone is transferred to the driver information system via virtual network computing (VNC) [37] with the remote frame buffer protocol, where it is displayed. AutoLinQ uses video transfer from a separate control unit to the driver information system, which means raster graphics as well.¹

An example of an FUI in the form of a markup language is an approach in which hypertext markup language (HTML) is used for transferring the UI between the smartphone and driver information system [27]. In the driver information system, the HTML descriptions are interpreted according to the design and presented on specific widgets.

On the other hand, there are approaches in which the UI is generated from abstract descriptions (AUI) [36, 38, 39, 40, 41]. Generation rules make it possible to generate various FUIs from the AUI for different systems.

4. EVALUATION

We have evaluated various variants for separating the application logic and UI generation in order to develop an optimum solution.

4.1 Separation of applications

The various separation solutions are all technically feasible, but are subject to differences with regard to customer benefit, risk and economy which will be described below. Table 1 presents the differences.

Table 1. Evaluation of variants for separating the application logic²

		VM	Pho- ne	ECU	Server
Customer value	Life cycle availability	o	-	o	o
	Available applications	-	+	-	o
	Vehicle API	o	o	o	o
	Needed interfaces	+	o	+	o
Risk	Offline availability	o	o	o	-
	External dependencies	o	-	-	o
	System stability	-	o	o	o
Cost/revenue	Certification needs	-	-	o	o
	System cost	o	o	-	+
	Initial development cost	-	o	-	o
	Cost by OEM AppStore	-	o	-	-
	App store revenues	+	o	+	+

¹ This group of variants is referred to as terminal mode below for reasons of simplicity.

² +: good, o: satisfactory, -: limited

If the customer obtains the smartphone from third parties, there could be compatibility problems due to the life cycle gap, as a result of which the customer may be limited in the choice of a smartphone. If the smartphone is sold directly by the OEM, this does allow compatibility to be guaranteed, but the problem of limited choice remains.

However, there are advantages in the selection of applications with the smartphone variant. If applicable, existing applications can be expanded and used. In the server solution, it is also possible for third parties to adapt their web services and allow them to be used in the vehicle. Given proprietary solutions within a VM or ECU, the small market share would result in developers being less interested, and the range of available applications would be smaller.

Vehicle interfaces can be used in all variants with a comparable level of complexity. In the current solution from BMW, for example, it is possible to access speed or temperature in the Twitter application.

Compared to the other variants, the smartphone variant requires users to bring their smartphone along and connect to the driver information system. The connection is wireless in current systems and can be automated using existing interfaces such as universal serial bus (USB), Bluetooth or wireless local area network (WLAN). The server variant, on the other hand, requires an online connection, which means a modem and mobile phone reception. The Internet connection can be established via a paired smartphone if necessary (this is called tethering), if permitted by the network operator.

Applications that access online data are faced with the problem of lack of network coverage. This can have a negative effect on all variants, and the server variant in particular. Suitable caching strategies and applications without page-by-page loading function could reduce these problems.

The smartphone variant is subject to risks due to the dependencies on other parties. For example, it is not possible to implement a terminal mode in standard Android devices. The security concept prohibits access to the frame buffer. Just like Apple, Google has not shown any interest so far in making modifications to permit terminal mode [42, 43]. Even if it were possible to integrate current devices, it may well be that future devices can no longer be integrated, or that updates are required at the driver information system end. There may be a similar problem with the ECU variant if a third-party operating system is installed. Possibly, operating system updates may be required for new applications, but might no longer be compatible with the hardware. The problem is known with current smartphones in which new operating systems cannot be installed on all devices.

In particular for the VM variant, there may be restrictions in terms of system stability, if there is a lack of safeguarding. There may be problems when accessing the same resources even with permanent allocations of CPU and memory capacities.

As already mentioned, it is necessary to ensure that only applications that do not distract the driver are possible in the driver information system. This can be achieved by certification, for example. However, certification may prove difficult, especially with the smartphone variant, because the smartphone is by principle an unsecure system, and communicates with the driver information system via standardised interfaces. Without functioning certification, an undesirable application could masquerade as a different one (man-in-the-middle attack).

Safeguarding would have to be ensured by crypto hardware with a protected memory area or alternative processes. Certification would also be required in a VM solution which does not separate the application areas from one another completely.

The system costs are lowest in the server variant; in particular if an existing modem and existing mobile phone contract can be used. In the ECU version, there will be costs for a complete control unit. Additional resources are required for a VM, because the resources of the driver information system are precisely tailored to the functions required. The smartphone variant will also result in additional costs for customers without smartphones.

For a separate control unit, the development costs in particular are comparatively high. The VM variant requires in addition to the VM itself an additional layer which controls resource access. Initial development and introduction of a hypervisor would also entail high development costs.

With the exception of the smartphone variant, all other variants require their own backend platform by means of which the new applications can be obtained and configured if necessary. In addition to the initial costs, a platform of this kind would entail ongoing costs above all. On the other hand, a company's own app store would deliver the greatest profit through application sales.

Overall, it is clear that there is no ideal solution amongst the different variants with regard to separation. However, it is our view that altogether the fewest disadvantages appertain to the smartphone and server variants.

4.2 UI integration

For the evaluation of the variants of UI integration (terminal mode, language, AUI) in particular the following requirements which are relevant: dual-task capability, quality, reusability and scalability, long life cycle as well as simple service development process. Table 2 summarizes the evaluation.

Table 2. Evaluation of the UI integration variants³

	Final UI - Terminal mode	Final UI - Language	Abstract UI
Guarantee of conformity with driver distraction guidelines	-	o	o
Quality	-	o	o
Reusability & scalability	-	o	+
Life cycle	-	o	o
Service creation	o	o	-

The requirement for dual-task capability and compliance with driver distraction guidelines can be solved in principle using all three variants. A dual-task-capable UI can be created both in terminal mode and in interfaces based on a markup language or abstract descriptions. In comparison, however, it is more difficult with terminal mode to ensure that only compliant UIs are displayed because, in principle, raster graphics can contain anything. In all other approaches, suitable rules in the generator or

³ +: good, o: satisfactory, -: limited

interpreter can be used for forcing a display in accordance with the guidelines right on the interface level.

In terms of quality, there may be differences in the graphic display quality and responsiveness. In order to avoid blurring due to scaling in terminal mode, it is necessary to ensure that the frame buffer provides raster graphics in the appropriate resolution for the driver information system. Considering the variety of smartphones and tablets, this is difficult to ensure. In addition, there can be problems with terminal mode with regard to responsiveness to dynamic interactions such as scrolling through a list. Moreover, problems can occur in terminal mode if an application is expecting touchscreen inputs but the driver information system only permits voice inputs or physical inputs (e.g. using a rotary push button).

Reusability of applications in new systems or scalability to different product classes is provided in particular in the markup language and AUI variants. For example, if a list containing ten entries is to be displayed, the interpreter in the driver information system can use a specific layout to display six entries with adequate size on large screens, but only four entries on smaller screens. This behaviour would have to be implemented externally in terminal mode, but this is not part of the specification at present [44]. A UI description also has the advantage that new rules will enable new layouts to be considered in future.

Furthermore, there may be difficulties in terminal mode due to external dependencies. An external interface for exchanging UI descriptions (e.g. HTTP) is more likely to remain available in new devices than access to the frame buffer.

Compared to the other approaches, the service development process is relatively complex in the solutions based on abstract descriptions and model-based UI generation. Besides model compliant UI descriptions, generation rules need to be defined when specific design specifications are required.

In summary, it is clear that the variants which use UI descriptions offer advantages compared to terminal mode. An interface in the form of a language-based final UI represents a good compromise between complexity and reusability.

5. WEB SERVICES FOR UI TRANSFER

As in [27], we use dynamically generated web services for UI transfer. The applications are running on a separate application platform which can be any of the discussed variants. In our approach, the model of the application (that is logic and data) is accompanied by a view component which generates final UI descriptions in the form of HTML and publishes it using an HTTP server. These UI descriptions contain multiple pages with hyperlinks where each page or view has one of several predefined view types and appropriate view elements. View types can be lists, input lists, speller screen displays or wizards with defined layouts, for example. View elements can be list items or buttons, for example. The browser in the driver information system maps the views to system specific layouts. The layouts assigned to the view types can differ in this case between different systems (e.g. systems with different display sizes), with the effect that the same application can be displayed differently. The UI extension provides a set of specified view types, configures them according to the specifications of the browser and passes on user inputs to the browser. For multiple applications multiple browser instances are possible. The browser also implements functional interfaces

for reading or modifying other services such as navigation or vehicle data. Figure 3 illustrates the architecture of this approach.

Instead of writing new applications, it is also possible to expand existing smartphone applications to include a vehicle UI.

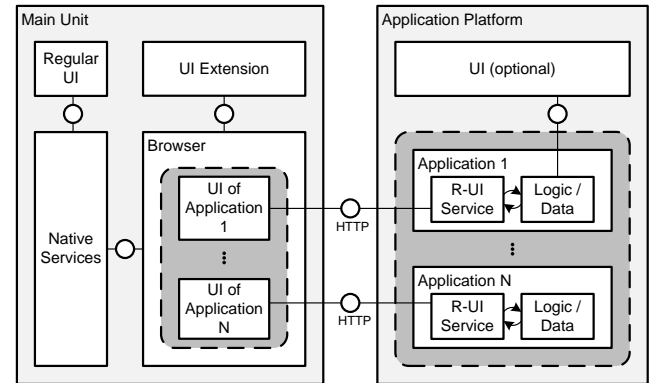


Figure 3. Web services for UI transfer

6. IMPLEMENTATION & RESULTS

The architecture described has been implemented as a prototype in a series production driver information system. As an application platform a smartphone was utilized. The flexibility and utility of Java meant that Android was selected as the smartphone platform. Implementation on other platforms is also possible due to the standardized interface (HTML via HTTP). For application development, a library serves as an additional abstraction layer. It generates HTML from source code and passes on events to the logic. View updates are possible via listener or own worker threads, and are integrated into the HTML descriptions by the library. By using the library, implementation is possible fully independently and without knowledge of the HTML.

On the application side, the interface is implemented by a framework service on which the applications register centrally by means of an API. The service provides the HTML UI descriptions on an HTTP server, and informs the driver information system about the applications via UDP broadcasts. The driver information system discovers the broadcasts, builds a list of the applications and integrates them in an application list within the regular UI of the driver information system. To consider the limited resources of the driver information system, the browser is a thin HTML parser specifically implemented for its purpose.

In addition, an app store is implemented which can be accessed using the driver information system, smartphone or a browser. In the course of this, it proved to be sensible to link the applications to an account. In this way, after the applications have been purchased in a browser, for example, they could be installed automatically on the smartphone.

Implementation showed that smartphone applications can be integrated into a current series production system. HTML was revealed to be a suitable language in order to integrate applications into the UI of the driver information system in a high quality manner. The graphic quality and responsiveness cannot be distinguished from the basic functions of the driver information system.

Implementation of several applications, including applications for Twitter, Google Mail and Last.fm, made it clear that the selected

view types are sufficient for a large proportion of current applications.

Taking the example of the Last.fm Android application which has its source code available online [45], it was possible to demonstrate that existing applications can be expanded with a UI for driver information system in a straightforward manner. Figure 4 shows the UI of Last.fm on the smartphone and a possible UI in a driver information system.

In addition, a codecamp during which students spent a week developing smartphone applications for a driver information system showed that applications could be created in a convenient manner using the abstraction by means of library.



Figure 4. Last.fm on a smartphone and in a driver information system

7. CONCLUSION & OUTLOOK

Discussion of the requirements for an expandable infotainment and evaluation of existing approaches have made it possible to establish that outplacing subsequent applications on a smartphone or web server currently represents the best solution. A language-based interface with final UI descriptions was identified as a good compromise between complexity and reusability for UI integration. The concept presented for UI transfer using web services and HTML for final UI descriptions represents an improvement on existing concepts in terms of graphic quality, responsiveness and reusability (compared to terminal mode) and complexity (compared to abstract descriptions). A proof of concept has demonstrated the feasibility and quality of the approach.

In our approach, common UI descriptions allow transfer of a UI to different driver information systems. Unfortunately, the level of abstraction goes against the grain not only of reusability but also of design freedom. This is especially important when a UI should be very similar to a specific operational concept. Further work will investigate how more degrees of freedom in design are possible while maintaining reusability.

Also, further work will investigate if the presented approach is suitable for applications which run on a web server accessed via mobile radio, or if further caching strategies are necessary for the case that connectivity is slow.

In addition, a suitable method must be developed for safeguarding the interface, in particular for smartphone applications. Terminal mode can only guarantee a secure interface if certificates can be stored in a secure area of the smartphone[44], which can only be achieved on a hardware basis. It can be assumed that using the UI description in our approach will make improved methods possible.

8. ACKNOWLEDGMENTS

Thanks to all project partners within Volkswagen AG and within the research project “Connected Cars in a Connected World” (C3World).

9. REFERENCES

- [1] Gartner, “Gartner says worldwide mobile phone sales grew 35 percent in third quarter 2010; smartphone sales increased 96 percent,” November 2010. [Online]. Available: <http://www.gartner.com/it/page.jsp?id=1466313>
- [2] S. Bratzel, L. Lehmann, and R. Tellermann, “i-Car: The young generation and the connected car.” [Online]. Available: www.auto-institut.de
- [3] R. Lind, R. Schumacher, R. Reger, R. Olney, H. Yen, M. Laur, and R. Freeman, “The network vehicle - a glimpse into the future of mobile multi-media,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 14, pp. 27–32, 1999.
- [4] Ford Motor Company, “SYNC AppLink: The free software program from Ford that gives SYNC users voice control of smartphone apps,” December 2010. [Online]. Available: http://media.ford.com/images/10031/-SYNC_AppLink_1210_HR.pdf
- [5] BMW Group, “BMW offers new interface for extended iPhone connectivity. The special option Apps.” February 2011. [Online]. Available: <https://www.press.bmwgroup.com/pressclub/p/pcgl/-pressDetail.html?id=T0097916EN>
- [6] Audi, “Audi Online Services.” [Online]. Available: http://www.audi.de/de/brand/de/neuwagen/kommunikation/-audi_online_dienste/navigation.html
- [7] R. Bose, J. Brakensiek, and K.-Y. Park, “Terminal mode – transforming mobile devices into automotive application platforms,” in *Proceedings of the Second International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2010), November 11-12, 2010, Pittsburgh, Pennsylvania, USA, 2010*.
- [8] V. L. Neale, T. A. Dingus, S. G. Klauer, J. Sudweeks, and M. Goodman, “An overview of the 100-car naturalistic study findings,” in *Proceedings of the 19th International Technical Conference on the Enhanced Safety of Vehicles, Washington D.C., Jun. 6–9, 2005*.
- [9] European Union, “Recommendations on safe and efficient in-vehicle information and communication systems: update of the European Statement of Principles on human machine interface,” December 2006.
- [10] Alliance of Automobil Manufacturers (AAM) Driver Focus - Telematics Working Group, “Statement of Principles, Criteria and Verification Procedures on Driver Interactions with Advanced In-Vehicle Information and Communication Systems,” June 2006.
- [11] Japan Automobil Manufacturers Association (JAMA), “Guideline for In-vehicle Display Systems - Version 3.0,” August 2004.
- [12] L. Duan, A. Höfer, and H. Hussmann, “Model-based testing of infotainment systems on the basis of a graphical human-machine interface,” in *Second International Conference on Advances in System Testing and Validation Lifecycle*, 2010.

- [13] K. Chatterjee and W. Kumaresh, "A road map for telematics," *The McKinsey Quarterly*, www.informinc.com, 2002 Number 2, pp. 1.
- [14] Inform Inc., "Cell phones: A poster child for extended producer responsibility," www.informinc.com, January 2004, pp. 1.
- [15] iSuppli, "Worldwide relative yearly unit sales for autos and major auto electronics categories." [Online]. Available: <http://evertiq.com/news/14632>
- [16] Volkswagen, "App my Ride - Volkswagen App Contest," August 2010. [Online]. Available: <http://www.app-my-ride.com/>
- [17] A. Schiefer, V. Gruhn, and R. Hrushchack, "Vesba - a middleware oriented architecture for virtualized embedded systems," in *CARS 2010, April 27, Valencia, Spain*, 2010.
- [18] A. Groll, J. Holle, C. Ruland, M. Wolf, T. Wollinger, and F. Zweers, "Oversee - a secure and open communication and runtime platform for innovative automotive applications," 2010. [Online]. Available: https://www.oversee-project.com/fileadmin/oversee/scientific_publications/-OVERSEE_Paper_Escar_Final.pdf
- [19] M. V. M. Macario, G. Torchiano, "An in-vehicle infotainment software architecture based on google android," in *IEEE International Symposium on Industrial Embedded Systems, 2009. SIES '09.*, 2009, pp. 257–260.
- [20] H.-U. Michel and D. Kaule, "Modern architecture - virtualisation techniques in the environment of automobile electronics development," *Elektronik*, vol. 7, pp. 27–30, 2010.
- [21] G. Smethurst, "Re-defining embedded solutions," in *Intel ERIC*, 2010. [Online]. Available: http://download.intel.com/corporate/education/emea/event/irc/files/presentations/deu/-T2_GrahamSmethurst.pdf
- [22] OpenSynergy GmbH, "Coqos product information." [Online]. Available: <http://www.opensynergy.com/en-/Products/COQOS>
- [23] F. Walkembach, "Wind river automotive software solutions," in *Intel ERIC*, 2010. [Online]. Available: http://download.intel.com/corporate/education/emea/event/irc/-files/presentations/deu/T2_FranzWalkembach.pdf
- [24] Saab, "World First from Saab: Saab IQon - Open Innovation in Car Infotainment," March 2011. [Online]. Available: <http://media.saab.com/en-us/press-releases/2011-03-01/-world-first-saab-saab-iqon-open-innovation-car-infotainment>
- [25] P. Kohlschmidt, D. Niederkorn, and B. Fargel, "Integration of consumer devices in vehicle systems," *VDI Berichte*, vol. 2000, pp. 573–582, 2007.
- [26] R. Stolle and D. Weyl, "Integrated HMI for mobile consumer devices," in *Euroforum Mobile Infotainmentsysteme im Automobil*, 2006.
- [27] J. Sonnenberg, "Service and user interface transfer from nomadic devices to car infotainment systems," in *Proceedings of the Second International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2010), November 11-12, 2010, Pittsburgh, Pennsylvania, USA*, 2010, pp. 162–165.
- [28] —, "A distributed in-vehicle service architecture using dynamically created web services," *Consumer Electronics (ISCE), 2010 IEEE 14th International Symposium on*, vol. 14, pp. 1–5, 2010.
- [29] M. Eichhorn, M. Pfannenstern, D. Muhra, and E. Steinbach, "A soa-based middleware concept for in-vehicle service discovery and device integration," *Intelligent Vehicles Symposium (IV), 2010 IEEE*, vol. IV, pp. 663 – 669, 2010.
- [30] F. Szczublewski, L. Jalics, and M. Krage, "Nomadic device connectivity using the ami-c hmi architecture," in *SAE Congress*, 2009.
- [31] BMW Group, "Driving fun, the enjoyment of music and social networks: MINI Connected with new functions." April 2011. [Online]. Available: <https://-www.press.bmwgroup.com/pressclub/p/pcgl/-pressDetail.html?d=T0105773EN>
- [32] Toyota, "Entune." [Online]. Available: <http://-www.toyota.com/entune/>
- [33] Continental, "AutoLinQ," 2010. [Online]. Available: <http://-www.autolinq.de/>
- [34] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, Laurent, and J. Vanderdonck, "A unifying reference framework for multi-target user interfaces," *Interacting with Computers*, vol. 15, pp. 289–308, 2003.
- [35] W3C, "Model-Based UI XG Final Report," May 2010. [Online]. Available: www.w3c.org/2005/incubator/model-based-ui/XGR-mbui/
- [36] G. de Melo, F. Honold, M. Weber, M. Poguntke, and A. Berton, "Towards a flexible ui model for automotive human-machine interaction," in *International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2009.
- [37] T. Richardson, Q. Stafford-Fraser, K. Wood, and A. Hopper, "Virtual network computing," *Internet Computing, IEEE*, vol. 2, pp. 33–38, 1998.
- [38] A. Hildisch, J. Steurer, and R. Stolle, "Hmi generation for plug-in services from semantic descriptions," in *International Conference on Software Engineering, Workshop on Software Engineering for Automotive Systems*, 2007.
- [39] K. Breiner, O. Maschino, D. Görlich, and G. Meixner, "Towards automatically interfacing application services integrated in a automated model based user interface generation process," in *4th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI-2009), February 8, Sanibel Island, Florida, United States*, 2009.
- [40] R. Stolle, A. Saad, D. Weyl, and M. Wagner, "Integrating ce-based applications into the automotive hmi," in *SAE World Congress, Detroit, Michigan*, 2007.
- [41] M. Weber, A. Berton, and B. Reich, "Abstract description of automobile HMI systems," *i-com*, vol. 8, pp. 15–18, 2009.
- [42] D. Lavrinc, "Nokia wants to control your car," March 2011. [Online]. Available: <http://www.autoblog.com/2011/03/24/-nokia-wants-to-control-your-car/>
- [43] "Terminal mode - member companies." [Online]. Available: <http://terminalmode.org/en/for-members/>

- [44] J. Brakensiek, "Terminal mode technical architecture - release version 1.0," October 2010. [Online]. Available: <http://www.terminalmode.org>
- [45] "The official last.fm android application suite." [Online]. Available: <https://github.com/c99koder/lastfm-android>