Current Challenges in Compositing Heterogeneous User Interfaces for Automotive Purposes

Tobias Holstein[1,2], Markus Wallmyr[1], Joachim Wietzke[2] and Rikard Land[1]

[1] School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
[2] Department of Computer Science, University of Applied Sciences, Darmstadt, Germany

Abstract. Composition (i.e. merging distinct parts to form a new whole) of user interfaces from different providers or devices is popular in many areas. Current trends in the automotive area show, that there is a high interest in compositing interfaces from mobile devices into automotive user interfaces. "Apple CarPlay" and "Android Auto" are concrete examples of such compositions. However composition is addressed with challenges, especially if the parts are originally designed for different purposes.
This paper presents the problem statement of compositing heterogeneous devices. Furthermore, it presents a layer model showing architectural levels, where compositions can take place and for each of these layers challenges have been identified.

Keywords: Design; Human Factors; Ubiquitous Interoperability; Heterogeneous; Platforms; User Interface; Composition; Hypervisor; Virtualization;

1 Introduction

Every device provides a certain user interface (UI), designed for the device's purpose, i.e. the use cases, stories and specifications that developers, designers and usability experts built the interaction around when creating the device. Even when using the same type of technology in two devices, the interaction might be quite diverse, for example due to different environments or user groups targeted. Some devices integrate a multitude of different types of functionality, such as a mobile phone with camera, phone functionality as well as applications (Apps). These devices have been designed from the beginning to fulfil those purposes. However there are also cases, where devices of different purposes (heterogenous devices) are subject of a composition.

This can currently be observed in the automotive industry where car manufacturers and original equipment manufacturer (OEMs) seek alternative approaches to integrate new features. Driven, among others, to reduce development efforts and costs as well as providing new features in a short time. One approach

is the integration of mobile devices into cars [3]. The basic idea is to take advantage of already established application platforms, instead of reimplementing applications for automotive platforms, gaining effort and time.

Mobile devices are increasingly important in our everyday life as the way mobile devices are used has changed drastically over the last decades. While the first mobile phones were solely used to make phone calls, they are now used e.g. to surf the World Wide Web (WWW), to post information on social platforms and listen to music. The amount of Apps available on mobile devices increases rapidly [16, 29, 23]. This trend is supported by open platforms, fast development cycles and a large constant contributing community.

Vehicle technology has become increasingly advanced over the last decades. While the primary functionality of a vehicle is still driving from A to B, many additional features towards infotainment, entertainment, comfort systems as well as increasingly advanced driver assistance have been introduced. These features have turned car user interfaces into increasingly complex systems. Making the effort for developing, testing and maintaining automotive systems extensive, also highly due to various automotive related requirements and safety regulations. Numbers indicate that up to 40 percent of the production cost of a car are due to electronics and software [10].

The idea to integrate mobile phones and vehicles is in itself not new, however the degree of integration have increased as technology has provided new opportunities. Today, graphical user interfaces (GUIs) of mobile phones are being completely integrated into in-vehicle-infotainment systems (IVISs) [4, 18, 12]. This kind of composition of heterogeneous devices (i.e. car and phone) is though fragmentary. There are often drawbacks when the composition takes place, which offers material for an investigation into the different challenges and layers of compositions. The results may indicate whether the approach of reducing development efforts and costs can actually be successful or not.

The contribution of this paper is threefold. First, we present the problem of integrating heterogenous devices into homogenous systems with focus on automotive and mobile devices. Second, we introduce a model containing different layers, in which composition can take place and third, we show examples of challenges providing limitations and resulting constraints.

The remainder of this paper is divided into the following sections. Sect. 2 gives a short introduction to different types of UIs. It is followed by a section that goes into related work and shows the inter-disciplinarity of this research area. Sect. 4 covers the problem statement. The last sections will introduce the layer model for composition and discuss the challenges applicable with each of the layers. The paper will finish with conclusions and future work.

2 Types of User Interfaces

Different classifications of UIs types have been made [31]. These classifications also extends and evolves over time as new new technologies, inventions and compositions of different UIs appears in different types types of interaction. To

provide a foundation further chapters these subsections gives an overview of UI types discussed.

2.1 Hardware UI / Haptic UI

Hardware UIs (HUIs) are tangible, i.e. pieces of hardware that you can actually touch and feel. Standard elements found in HUIs are e.g. LEDs, flip-switches, wheels and levers.

In vehicles, buttons replaced flip switches decades ago. Advanced vehicle bus systems digitally connect everything and therefore many controls can have multiple purposes. Also combined controls like a push/turn control knob are used in modern UIs. Newer versions of this control include a touch pad at the top of the knob, which is used for gesture and script recognition [6].

In mobile devices the multi purpose use of HUIs has been there from the beginning. The number of HUIs on a mobile device has over time also been reduced, where a lot of interaction has been focused at certain places. Which brings us to display based UIs.

2.2 Display-based UI

While the display itself is an element of HUIs, because of its physical form and appearance, it is also the basis for other UI types.

Graphical UI. The first graphical user interface (GUI) has been introduced in 1981 as part of the "Xerox Star workstation" [27], which used the WIMP (windows, icons, menus, and a pointing device) interaction style. This type of UI was later adapted by Apple and other companies.

GUIs in cars are used in IVIS and also in fully digital cluster instruments, which gain in popularity. These digital solutions provide flexible options for designs, i.e. displaying different content can easily be done and by a software update additional functionality can be added without affecting the HUI.

Touch-Screen. Touch-Screens allow an interaction directly on the screen without any intermediate device (e.g. a mouse or joystick). They are commonly used in e.g. portable navigation systems, mobile phones and also in IVIS. It has became a standard method to manage the complexity and to increase usability of device interaction. However touch-screens are reluctantly used in automotive UIs, because research shows, that the distraction of the driver while using a touch-screen is seen to be too high [26]. One reason for that is the missing haptic feedback (compared to buttons/wheels). Drivers cannot feel whether a button was pressed or not and need to check the visual feedback, which causes a measurable distraction. However the distraction is lower compared to using a mobile device while driving [20].

In addition multi-touch capable touch-screens made a new interaction style possible, which is referred to as Post-WIMP. It was first introduced by Van Dam in 1997 [13]. This interaction style is used in OSs like Android and iOS.

2.3 Car UI

A car UI is a combination of multiple different UI types. A typical and well-known part is a HUI for the primary task of driving the car, including e.g. the steering-wheel, pedals, gear shift and speed gauges. If a fully digital instrument cluster is used a GUI will be part of the overall UI as well.

UIs in newer vehicles offer access to about 700 different functions (e.g. BMW Series 7 [7]). However the difficulties of handling 700 buttons, or flip switches in a dashboard, in terms of dashboard design and usage, are significant. Therefore design decisions are required to cope with the trade-off between features that are quickly accessible and features that are not [21].

Modern IVIS use a combination of touch-screens or normal screens for the GUI and a HUI for controlling it. The approaches of car manufactures differ greatly and an extreme example can be seen in Tesla's Model S, where a large touch-screen has replaced all HUIs normally presented in a center console, which is used to access all in-vehicle functions [30].

2.4 Mobile Device UI

In addition to touch-screen, GUI and buttons, mobile devices provide a wide range of sensors, which can be seen as a part of the UI. Sensors are used to enrich the user experience and to support certain use cases. A magnetometer for example can be used to change the GUI from portrait to landscape mode and vice versa. Other sensors (e.g. gyroscope, proximity sensor and accelerometer) are used for navigation, games and even fitness/health applications. A list of typical sensors as well as input/output modalities can be found in [2] and [14, p. 49].

3 Related Work

The topic of compositing, i.e. combining and/or integrating, heterogeneous devices has been covered in various research publications. Following is an overview on the current research and commercial solutions to show the different approaches in this area.

Software, which uses virtualization mechanisms to run multiple operating systems on the same hardware platform, is often referred to as "hypervisor". A hypervisor can be classified in two different types: Type one (or native, bare metal) hypervisors run directly on the host's hardware to assign the hardware components and to manage guest operating systems. Type two (or hosted) hypervisors run as a normal application within a conventional operating-system environment. A well- known type two hypervisor software is "VMWare".

In [22] compositing of GUIs from a partitioned IVIS is shown. Partitions, i.e. multiple OSs, are running concurrently on one hypervisor. Applications running on those OSs are presented in a homogeneous GUI, which is provided by a component called "compositor". Applications have to provide a GUI that fits into the overall GUI. In this approach the physical UI has not been considered.

Instead of using the cars UI as a base for the composition the authors of [15] uses a mobile device as compositor. Therefore all data from the car is redirected to the mobile device, where it is processed and visualized. The mobile device becomes a portable IVIS.

Another approach is a link between services of a mobile device and the cars UI. In [28] services and UIs are exchanged between car and mobile device. The GUI is built dynamically by exchanging HTML5 UI descriptions. Services are connected through exchange of interface descriptions for each available service.

Bose [8] introduces a concept called "Terminal Mode" for integration of a mobile phone UI into a car UI using an extended VNC protocol [25]. Extensions include categories and authorization for applications. This approach is very similar to MirrorLink [12] which relies partly on the VNC protocol to replicate the phones display content on the remote UI.

The newest developments in the automotive industry have been introduced by Google and Apple. In order to gain access to a new market both companies try to prepare their OSs "iOS" and "Android" for the automotive consumer market. Proprietary protocols denoted as "Apple CarPlay" [4] and "Android Auto" [18] are currently developed and introduced through OEMs to the consumer market. This allows users to connect and integrate their mobile phone to the cars UI. [1] describes some limitations based on a first evaluation.

For completeness it has to be mentioned that software solutions like VNC [25], TeamViewer and Windows Remote Desktop, can be used to achieve a similar integration of GUIs for Desktop OSs (i.e. Windows, Linux, MacOS). In general these are for the WIMP interaction styles.

4 Problem Statement

Designing user interaction that integrate UIs from different devices or applications into a combined homogeneous UI is fundamentally problematic as it introduces two opposing forces. One force is to have different UIs appear in a consistent experience in the combined interaction (i.e. look-and-feel). The opposing force is to keep consistency with the fixed original UIs of the application or device. E.g. the applications from different mobile devices or operating systems appear as one UI within the vehicle, while at the same time keeping the heritage to the original user interaction of the device.

Different devices are built for different purposes and these purposes result in different design choices being made when designing the device [11]. With heterogeneous devices being integrated, the set of interfaces will vary, e.g. a mobile phone has a different set of inputs and outputs than the interface of a car. Depending how and where the integration is performed, different constraints arise for the combined user interaction, which will be discussed in the following sections.

5 Layer Model

Compositing different devices or systems can be performed at different logical layers. In this section these layer are presented. The model is inspired by the OSI (Open Systems Interconnection) model and the layers are selected around the notion of separation of concern and the separation of different functionality in hardware and software. The layers are depicted in Fig. 1, which also shows constraints and dependencies. Definitions made in lower layers may appear as constraints in higher layers and higher layers strongly depend on lower layers. Higher layers may also force requirements towards lower layers, which have to be considered in a composition.

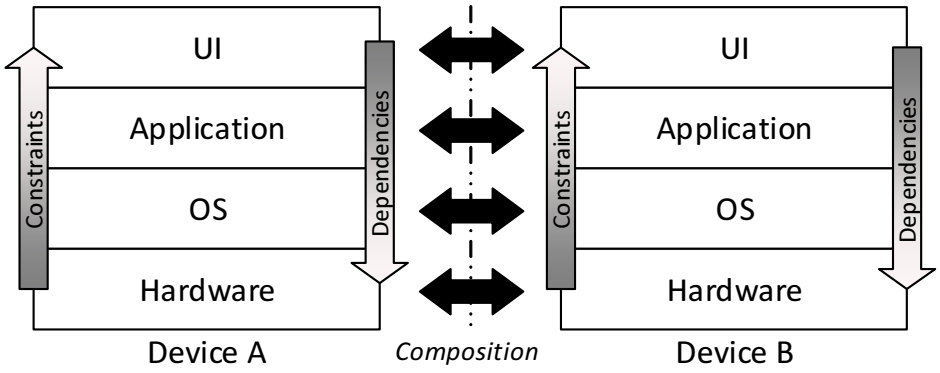

Fig. 1. Two Devices with Layers for Compositions

5.1 Hardware Layer

In the hardware layer composition is made at the physical hardware level by integrating different input and output components on mechanical or electrical level. Input components include for example buttons, knobs, microphones, joysticks and sensors. Output components include e.g. displays, speakers, vibration and LEDs.

An example of composition on this layer is using a video switch to share a display and its associated buttons between different devices. When changing channel on the display the image from the next device will be displayed and input signals will be sent to that device.

5.2 OS Layer

When a composition is performed at the OS layer, one set of hardware is used to run different OSs. This is typically done using virtualization techniques, i.e.

using a hypervisor. OS integration using a hypervisor could be done in two ways. Either by having the OSs run in parallel with no knowledge of each other, a *type one hypervisor*. Alternatively, one OS is aware of the other OSs and manages the sharing of resources, a *type two hypervisor*.

An example of composition on this layer is having an OS handling critical tasks, which has control of the display and shared hardware resources. This designated OS can then show information from other OSs on the display and send input information back to them, ensuring that critical information is always shown when necessary.

5.3 Application Layer

In the application layer different applications or services are part of the composition, running on one OS. The implementation of applications usually requires libraries or services. Those provide an application programming interface (API) for developers, giving possibilities to share information and to let different kind of services collaborate. This layer is separated from the UI layer, because different applications may provide different UIs.

An example of composition in the application layer is a vehicle navigation application where a service is integrated presenting friends locations position and messages from a communication service.

5.4 UI Layer

In the UI layer the composed experience is considered. It is at this layer where the actual input and output modalities will meet the user as one UI. When fully embracing integration at the UI layer the different parts integrated is reshaped and composed to provide a rich user experience. While on the other hand if diverse parts is put together at lower layers the composed result may be a diverse UI.

An example of integration on the UI layer are technologies such as Apple CarPlay and Android auto where an app presents an adapted UI in the dashboard with respect to the phone UI.

6 Composition Challenges

In this section we present 13 identified challenges when compositing heterogeneous devices. The following list was created based on experiences and studies, but contains only significant challenges. However, those challenges show that no architectural layer is without challenges.

6.1 Hardware Layer

Challenge 1: Compositing the different hardware interfaces of two heterogeneous devices (e.g. different CPUs or GPUs, number of buttons, or screen-sizes).

One approach to address this challenge is to create a new device that fulfils all minimum requirements of each device in a composition. For example a composition of an Android device and an IVIS, would require the new target platform to fulfil the minimum requirements [17] for the Android device as well as for the IVIS. However, targeting the minimum requirements could also mean to sacrifice features, leading to incomplete user experience.

Another approach is to create a device with the maximum available configuration of both entities. For example a composition of two devices, one device with five buttons and another device with ten buttons. The composited interface would therefore have 15 buttons. However, if an upper layer is able to handle mapping or reassigning buttons, the number of buttons could be reduced.

Challenge 2: Compositing and sharing certain hardware that have a very specific use and context, potentially resulting in misinterpretation of information or delayed reaction. An example is sharing vibration as a mean to notify the user: A mobile phone may use vibrations to inform the user about incoming phone calls or messages, whereas a vehicle could use vibrations for lane assistance alertness [9]. The user would have to determine what the source of the notification is, if the same haptic feedback was used for both scenarios. In a driving situation this could result in loss of valuable reaction time.

6.2 OS Layer

Challenge 3: Compositing common hardware resources in virtualized OS environments. While one OS itself provides a level of abstraction from the hardware, the hypervisor will face challenges with singleton hardware resources when integrating two or more OSs. An example is how to decide which input signals should be routed to which of the OSs. A keyboard device might be assigned to one OS, therefore it cannot be used within another OS. The determination of which input belongs to which OS might also depend on higher layers, for example the current state of an application.

Challenge 4: Compositing different OS behaviours. When multiple OSs share the same screen, but aren't aware of it, challenges occur when it cannot be made sure that certain outputs are received by the user. This can result in some functionality not working as expected. An example from mobile systems is the notification system. In a single OS a message shown on top of all other applications, where the user has to take an action to proceed, will be visible as expected. One approach in a shared environment is to define a set of rules, e.g. priorities or other rule sets to make sure that OSs can force important messages to be shown on the screen. However defining these rules is yet another challenge and the way these notifications are presented is part of the UI layer.

Challenge 5: Compositing OSs without the full set of hardware. An OS might require a certain hardware component in order to provide full functionality, which may not be available in the given hardware layer. An approach is to provide virtual hardware via the hypervisor, which on one side offers the expected interface to the OS and on the other side works with the actual given hardware layer. For example buttons of a mobile device, which are not available in a center

console of a car, can be replaced by virtual hardware and triggered through software. The challenge increases when interfaces are more complex, such as e.g. the routing of voice commands.

Challenge 6: Compositing different devices behaviour experience. As an extension of *challenge 5* the hypervisor may potentially provide a virtual device working as an adapter between the embedded platform and the OS, however the user experience might be significantly affected due to the challenges supporting all potential use cases. Such example is trying to virtualize the input of a touch screen when only buttons are available. Certain operations will be difficult to transform (e.g multi touch operations). A more hypothetical example is the virtualization of a mobile device magnetometer. The usage of such sensors is fundamentally different in a car than a mobile device. A car might use it for crash detection and the mobile device to detect if the device is flipped into different directions. Using the sensor of the car would not cover all use cases of the mobile device, as the car cannot be flipped over. One approach could then be to use a button, which if pressed releases a signal in order to trigger a rotation by 90 degrees. The mapping of incompatible interfaces can however lead to limitations, e.g. allowing only rotations by 90 degrees, which will be a constraint to upper layers.

6.3 Application Layer

When compositing multiple services or APIs into a new application, the application will obviously depend on the services and APIs used. These dependencies are likewise *challenges*, and the following examples will elaborate on that.

Challenge 7: Composition with dependencies on the APIs provided by applications and services. If properly used the application will work as long as the APIs deliver the requested data and accept the given input. However, if an API is not used according to its specification or the expected use cases, the application might work in this version, but may break when the next API update occurs.

Challenge 8: Composition using external services causes a strong dependency. Integration into social- or information providing services can be attractive, but if a service is changed or stops to exist the application will fail. An example is using web services providing GUIs via HTML5. Even though the web service delivers HTML5 according to standard, it might contain HTML5 elements which are not supported by the renderer application in the car. This challenge and *challenge 7* become significant worse when development cycles and update rates of diverse systems is taken into consideration.

Challenge 9: Compositing two devices at application layer using remote connectivity protocols, e.g. VNC or MirrorLink. The OSs of each device run independently, but provide their services and GUIs via protocol to remote applications. The remote application must then handle the challenge within interaction with the OS, i.e. sending the equivalent of the Mouse, Keyboard or touch command needed, while the same application receives an update of the GUI.

6.4 UI Layer

Challenge 10: Compositing different graphical design languages to form a homogeneous experience. The challenge is, that each type of device or service has different visualization and branding guidelines [5, 19, 24], which are more or less mandatory to follow when designing for that type of device. For example the look-and-feel of a car UI will differ from the look-and-feel of a mobile device UI.

Challenge 11: Compositing different sets of usage scenarios from the original user interfaces. A mobile device and its applications are designed for use on other distances from the user than a vehicle display. An example is screen utilization. With the mobile display the user can vary the distance depending on eyesight and content displayed. Whereas on the vehicle display text size etc. must be at a size that support users with varying eyesight.

6.5 General Challenges

Challenge 12: Compositing with dependencies and constraints of other layers. As already indicated by earlier presented challenges, a decision at one layer may lead to constraints and dependencies in other layers, i.e. new challenges. An example for this is the screen resolution in the hardware layer, affecting the design of the visual appearance of a GUI.

Challenge 13: Compositing devices with different life cycles causing diversity and constraints over *time*. This is a type of challenge that exists in a general level, applicable across all layers. For example a car may be bought every 10 years, while a mobile device may be replaced every two years. 10 years ago smart-phones were fundamentally different than today and predicting what will come in the future is merely making a guess.

7 Conclusion

In this paper we have studied the composition of heterogeneous user interfaces. Furthermore we defined a model where composition approaches can be divided into different architectural layers. Last, but not least we discussed the challenges, limitations and constraints that relate to these layers. While the list of challenges is not complete for each layer, it shows that no architectural layer is without challenges. Based on identified approaches and challenges it seems that no approach can fully composite two heterogeneous interfaces without substantial adaptations at one or both sides.

It is possible to convert basic inputs from one element to another and it is also possible to pass basic outputs from one element to another. However, this is only a basic level of composition. In order to really composite a homogeneous experience the whole device composition must be considered, from the physical attributes to the combined UI, considering both output and input methods.

8 Future Work

There are several areas in composition of heterogeneous interfaces where we would like to extend our work. One area is to perform further evaluations on each layer to further investigate the constraints and limitations.

Another intent is to explore deeper into the usability, usefulness and user experience on the UI layer, in order to provide better user interaction of composited devices.

References

[1] Airbiquity: Apple carplay: Ready for connected car prime time? online (2014), http://www.airbiquity.com/news/blog/airbiquity-white-paper-apple-carplay-ready-connected-car-prime-time/ (Accessed 2014-09-19)

[2] Allan, A.: Basic Sensors in iOS: Programming the Accelerometer, Gyroscope, and More. " O'Reilly Media, Inc." (2011)

[3] Alliance, O.A.: Open automotive alliance - a global alliance of technology and auto industry leaders. online (2014), http://www.openautoalliance.net/ (Accessed: 2014-11-01)

[4] Apple: Apple carplay - the best iphone experience on four wheels. online (2014), \url{https://www.apple.com/ios/carplay/}, https://www.apple.com/ios/carplay/ (Accessed: 2014-10-16)

[5] Apple: Designing for ios. online (2014), https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/ (Accessed: 2014-10-20)

[6] Audi: Dialoge - das audi-technologiemagazin. online (January 2014), https://www.audi-mediaservices.com/publish/ms/content/de/public/broschueren/2014/01/14/Dialoge_-_Das_Technologiemagazin_01-14.standard.gid-oeffentlichkeit.html

[7] BMW: Bmw technology guide: idrive. online (2014), http://www.bmw.com/com/en/insights/technology/technology_guide/articles/idrive.html (Accessed: 2014-10-20)

[8] Bose, R., Brakensiek, J., Park, K.Y.: Terminal mode: transforming mobile devices into automotive application platforms. In: Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications. pp. 148–155. ACM (2010)

[9] Brandt, T., Sattel, T., Böhm, M.: Combining haptic human-machine interaction with predictive path planning for lane-keeping and collision avoidance systems. Proceedings of 2007 IEEE Intelligent Vehicle Symposium (2007)

[10] Broy, M.: Challenges in automotive software engineering. In: Proceedings of the 28th international conference on Software engineering. pp. 33–42. ACM (2006)

[11] Buxton, B.: Sketching User Experiences: Getting the Design Right and the Right Design. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007)

[12] Consortium, C.C.: Mirrorlink. online (2014), http://www.mirrorlink.com/ (Accessed: 2014-10-22)

[13] van Dam, A.: Post-wimp user interfaces. Commun. ACM 40(2), 63–67 (Feb 1997)

[14] Dorau, R.: Emotionales Interaktionsdesign, vol. 1. Springer (2011)

[15] Geier, M., Becker, M., Yunge, D., Dietrich, B., Schneider, R., Goswami, D., Chakraborty, S.: Let's put the car in your phone! In: Proceedings of the 50th Annual Design Automation Conference. p. 143. ACM (2013)
[16] GmbH, A.: Number of android applications. online (2014), http://www.appbrain.com/stats/number-of-android-apps (Accessed: 2014-11-01)
[17] Google: Android 4.4 - compatibility definition. online (2013), http://source.android.com/compatibility/android-cdd.pdf
[18] Google: Android auto. online (2014), \url{http://www.android.com/auto/}, http://www.android.com/auto/ (Accessed: 2014-10-16)
[19] Google: Android design. online (2014), https://developer.android.com/design/index.html (Accessed: 2014-10-20)
[20] Heikkinen, J., Mäkinen, E., Lylykangas, J., Pakkanen, T., Väänänen-Vainio-Mattila, K., Raisamo, R.: Mobile devices as infotainment user interfaces in the car: Contextual study and design implications. In: Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services. pp. 137–146. MobileHCI '13, ACM, New York, NY, USA (2013)
[21] Kern, D., Schmidt, A.: Design space for driver-based automotive user interfaces. In: Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications. pp. 3–10. ACM (2009)
[22] Knirsch, A., Theis, A., Wietzke, J., Moore, R.: Compositing user interfaces in partitioned in-vehicle infotainment. In: Boll, S., Maa, S., Malaka, R. (eds.) Mensch Computer 2013 - Workshopband. pp. 63–70. Oldenbourg Verlag, Mnchen (2013)
[23] Microsoft: Microsoft by the numbers - the windows phone store features more than 300,000 apps and games. online (2014), http://news.microsoft.com/bythenumbers/index.html (Accessed: 2014-11-01)
[24] Microsoft: Windows design guidelines. online (2014), https://dev.windows.com/en-us/design (Accessed: 2014-10-20)
[25] Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual network computing. IEEE Internet Computing 2(1), 33–38 (Jan 1998)
[26] Rümelin, S., Butz, A.: How to make large touch screens usable while driving. In: Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications. pp. 48–55. ACM (2013)
[27] Smith, D.C., Irby, C., Kimball, R., Verplank, B., Harslem, E.: Designing the star user interface: The star user interface adheres rigorously to a small set of principles designed to make the system seem friendly by simplifying the human-machine interface. Byte. April pp. 242–282 (1982)
[28] Sonnenberg, J.: Service and user interface transfer from nomadic devices to car infotainment systems. In: Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications. pp. 162–165. ACM (2010)
[29] (TechCrunch), S.P.: itunes app store now has 1.2 million apps, has seen 75 billion downloads to date. online (2014), http://techcrunch.com/2014/06/02/itunes-app-store-now-has-1-2-million-apps-has-seen-75-billion-downloads-to-date/ (Accessed: 2014-11-01)
[30] Tesla: Example ui. online (2014), http://www.teslamotors.com/sites/default/files/blog_images/model-s-photo-gallery-14.jpg (Accessed: 2014-10-22)
[31] Wikipedia: User interface — wikipedia, the free encyclopedia. online (2014), http://en.wikipedia.org/w/index.php?title=User_interface&oldid=630749818 (Accessed: 2014-11-03)