Proceedings of the Second International Conference on Automotive User Interfaces and Interactive Vehicular Applications
(AutomotiveUI 2010), November 11-12, 2010, Pittsburgh, Pennsylvania, USA

Service and User Interface Transfer from Nomadic Devices to Car Infotainment Systems

Jan Sonnenberg
Technische Universität Braunschweig
Institute for Communications Technology
sonnenberg [at] ifn.ing.tu-bs.de

ABSTRACT
Many of the emerging software applications for nomadic devices are useful in the car as well. In order to use these applications safely in the car, it is necessary to couple them with the vehicle's infotainment system and its user interface which is optimized for use by the driver. This paper describes a new approach to exchange services and user interfaces between cars and nomadic devices. Services are exchanged through dynamically generated Web Services. HTML 5 based user interface descriptions are used to access shared vehicle software interfaces as well as the remote application's logic with the help of additional mechanisms for device communication. Make and model specific design is ensured by Stylesheets that are specific for the car model. Our solution combines in-vehicle infotainment systems with external applications in a safe and secure way. Especially, there is no need for pre-defined service specific interfaces, because all interfaces are exchanged dynamically. The paper starts with a motivation, an overview of related work and outstanding challenges followed by a presentation of our approach for in-vehicle device coupling. The user interface exchange is described in detail in the third paragraph. The paper finishes with an example scenario and a conclusion.

Categories and Subject Descriptors
D.2.8 [**Human Machine Interaction**],

H.5.2 [**Information Interfaces and presentation**]: User Interfaces.

General Terms
Design, Reliability, Human Factors.

Keywords
In-car infotainment; Web Services; automotive user interfaces.

1. INTRODUCTION
Today, the integration of nomadic devices into cars is based on well defined service interfaces and corresponding user interfaces. Examples are hands-free telephony integration or audio streaming based on Bluetooth profiles. These profiles are known at the time when cars are being developed, so a tight integration into the respective hardware and software is possible. While early handheld devices relied on embedded functionality, today's more powerful Smartphones and Multimedia Players include software abstraction layers as known from stationary computers. Network hardware, graphics hardware and sensors are accessed through detailed abstraction layers. These abstraction layers allow to build intermediate software layers, like application specific protocols, rendering engines and sensor data interpretation. As a consequence, such handhelds can be adapted to the latest communication and application protocols, which is often impossible for embedded car software. While this fact illustrates the well known life cycle gap between automotive and consumer electronics, another disadvantage arises as the result of the advent of mobile software platforms that target third party developers. With the emerging growth of third party and user generated software on handheld devices such as "Apps", it becomes challenging to integrate these services in the car. Nevertheless, latest personal Smartphones host many services and personal data that are of interest in the car as well. There are e.g. location based services to find dynamic points of interest such as free parking lots or gas stations with lowest prices. Targeting mobile usage, the latest Smartphones' user interfaces own features that make them ideal for being used in the car: multi-modality, high resolution graphics, support of different screen orientations, etc.

1.1 Approaches for Remote Data and Service Integration
Data exchange between different devices is possible through storage media exchange and wired or wireless device connections. While storage media transfers only static data, device connections allow the exchange of dynamic data. Another advantage of device connections is the possibility to access remote service logic, like data decoding. Connected devices may even act as a gateway and pass data from one of their network connections to another. Wireless device connections gain increasing popularity, as they are easily or even automatically set up. The protocols used to couple connected devices vary in flexibility:

1. Vendor specific protocols are used to couple devices of the same type with each other or to couple them with vendor specific services on other devices. An example are interconnected handheld game consoles. Such protocols couple only certain services and platforms.
2. Standardized, service specific protocols are used to couple device functions of certain types. Examples are the already mentioned Bluetooth profiles for wireless

connections or USB device classes for wired connections. These protocols couple services with specified interfaces that may reside on arbitrary platforms.

3. Standardized, service unspecific protocols are used to couple self-descriptive services. Examples are Web Applications that bring their own user interfaces.

The first approach is rarely used for automotive device integration although some car manufacturers signed contracts with consumer device manufacturers to add coupling mechanisms for a certain device type. More often, the second approach is used which is based on established communication protocols. Modern implementations combine different connectivity features by software and add additional vendor specific online services like e.g. voice recognition and route calculation. First car manufacturers announced plans to license interfaces and development kits to third party developers in order to distribute software add-ons through their own application stores [1]. The European project AIDE developed a Bluetooth based device integration protocol with an emphasis on safe device integration [2]. Its nomadic device integration mechanism is part of a comprehensive infotainment system that prevents driver distraction by inappropriate or competing I/O events. Therefore it takes the driving situation into account and integrates handheld device communication accordingly. For example, it redirects phone calls to a voicemail message system when the driver performs a stressful driving task.

There are some interim solutions that combine the second and the third approach. An example is Universal Plug and Play (UPnP) which is based on the Simple Object Access Protocol (SOAP) and device specific control protocols [3]. In order to offer control of unknown services, devices may optionally offer Web Interfaces. Such a self-descriptive device coupling is currently only used in consumer electronic devices like e.g. network routers that offer remote configuration features.

In order to bridge the lifecycle gap between automotive and consumer electronics, a solution for the integration of current and future devices and services is the ultimate goal. While updates and extensions to the vehicle software are a solution that is currently worked on (through download and installation of verified applications into the car), this paper focuses on the third approach – self-descriptive services that bring their own user interface which is verified dynamically. A first and very simple solution for this approach is a remote frame buffer that displays the graphical user interface of the handheld in the car and provides a static mapping between vehicular and handheld controls. The benefits of this approach are limited as orientation, widget sizes, styling, and layout are not adapted to the car's user interface. More promising are Web Interfaces that are rendered through an automotive browser. While some of the above-mentioned disadvantages are solved, some still remain. Uncontrollable distraction through animations and contrast changes may appear as well as response delays that decrease driver attention. As such, an important request for the integration of self-descriptive services from foreign devices is the seamless integration into the vehicle's user interface. For this purpose, there are different proposals that base on UI model conversion [4], [5]. Some proposals even interconnect remote and local user interfaces and service logic in a smart way [6]. These approaches depend on abstract user interface descriptions that are transformed to different target platforms with the help of certain rules and patterns. While descriptions and conversions are quite complex, the resulting automatic conversion can only be as good as the considered conversion rules. On the other hand, additional work is necessary to describe the user interfaces in an abstract way.

Although there are ongoing aims for standardization of automotive protocols for device integration, the use in future mobile application software is questionable. As a matter of fact, community generated software is usually based on popular standards which is why even open automotive standards cannot guarantee that popular Smartphone applications support standardized automotive interfaces. Hence, our approach bases on existing standards and technologies that are already used for handheld application development.

1.2 Challenges and Requirements

As additional information decreases the driver's attention, several requirements need to be taken into account for device coupling and remote user interface integration. While some of them are already mentioned, there are some more [2], [6], [7]. Among others, the following requirements are especially important for device coupling and UI transfer:

- The driver should not be distracted through visual entertainment while driving. It shall be possible to switch off any dynamic information.
- Contrast, font and widget sizes shall be compliant to automotive standards.
- Information shall be presented timely and shall be prioritized where appropriate.
- Distraction by long and uninterruptible system interaction shall be avoided as well as time critical input requests. It shall be possible to resume interrupted tasks.
- The system shall respond in a clear, predictable and consistent way and report status and failure messages.

Furthermore, car manufacturers have some additional requirements concerning the layout [6]. The UI transfer shall be decoupled from HMI concepts of different models and still maintain the look and feel of the automotive infotainment system.

2. INTEGRATION OF FOREIGN APPLICATIONS INTO CARS

In order to meet the described requirements, we use some well-known technologies and extend them suitably. Firstly, we use the Devices Profile for Web Services (DPWS) for user-friendly and service independent device coupling [8]. It offers device and service discovery mechanisms as well as service independent remote method invocation based on SOAP. In addition to UPnP, it can address internet services and optionally provide secure communication. While existing coupling mechanisms use service specific communication protocols, DPWS uses IP-based communication. Thus, it is possible to describe and even generate service interfaces for remote access dynamically in software [9]. Another advantage is the use of MIME based data exchange in order to support different popular content formats and encodings. Just as UPnP, DPWS may optionally provide a Web Interface for remote control. Unfortunately, plain HTML Web Interfaces may introduce the disadvantages described earlier. A way out is the

mapping of HTML elements to basic automotive widgets and a coupling of these widgets to the remote service in a way that is controllable by the vehicular software. This allows the car infotainment system to control the user interface layout as well as remote service interaction and the display of results. Therefore, no styling information is transferred. The styling information is shipped with the car and adapted dynamically to meet car UI design directives. The car infotainment system suppresses animations and renders fonts and widgets in an appropriate size. Depending on the light conditions, different day and night designs are used. As latest Smartphones support landscape and portrait orientation, many services are designed to support both orientations. Thus, a basic HTML structure optimized for different orientations can be used to render a service on a handheld device and in the car whatever orientation is used. Nevertheless, there might be complex widgets, that cannot be used in the car directly. As all widgets are described by hierarchical widget classes inside the HTML based user interface description, it is up to the car to use an alternative widget that fits to the needs or to omit the element. If a widget is known to be handheld specific, it is also possible to provide an alternative in the user interface description. It might even be better to change the modality, e.g. to change from text to speech for pop-ups.

In order to support multiple handheld devices and access their hardware capabilities, an adapted web based application framework is used as additional middleware layer. While classic HTML based applications cannot access device features, such a framework provides access to certain platform APIs via JavaScript. So it is possible for HTML based applications to access geo location interfaces, address books, media archives, etc. [10] In the following, these HTML applications that access device APIs through a middleware layer are called Web Applications. The term 'Web Service', on the other hand, is used for server applications with direct device API access.

3. USER INTERFACE EXCHANGE

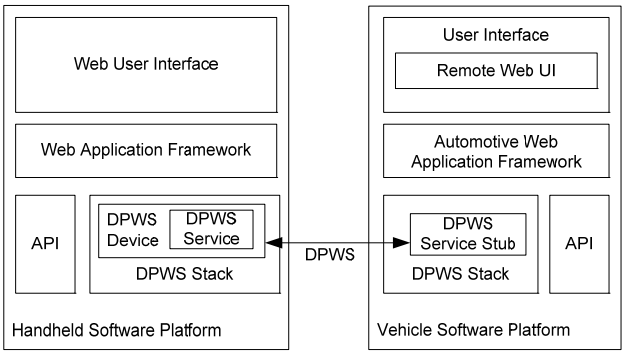

Figure 1: Service and user interface exchange

As depicted in figure 1, our approach for user interface transfer is based on the following two characteristics:

1. A device dependent Web Application Framework is used to offer access to a set of generic device APIs. Additionally, device dependent DPWS Web Services are used to offer these generic device APIs to remote devices without providing a user interface. Both the Web Application Framework and the DPWS Web Services need to be implemented for each supported platform.
2. Device independent Web Applications implement applications that access device APIs through the Web Application Framework. These Web Applications are rendered with device specific stylings and can be used on multiple platforms.

To share such a Web Application with other devices, an additional Web Service is generated which links to the Web Application and thereby provides a user interface. This Web Application is downloaded to the remote device and rendered according to its local styling. As this downloaded Web Application is known to be a remote service, it behaves differently in comparison to local Web Applications: Special indicators within the HTML describe whether remote APIs shall be used or local APIs shall be preferred, if available. Depending on these indicators, the Web Application Framework decides whether to use the generic remote APIs coupled through Web Services or the local device APIs.

4. EXAMPLE SCENARIO

Consider an Economic Driving Application that uses the hardware accelerometer of a Smartphone to collect information on the intensity of the acceleration and braking while driving[1]. In this paper we present a similar example application that has two views, the first view is showing a colored smiley indicating the current driving behavior, the second one is showing a table listing records of the driving behavior over time. Such an application can run by its own on a Smartphone, all necessary sensor equipment being present. As this application is driving related, it is also interesting to integrate it into the car's infotainment system.

Figure 2: User interface of the economic driving application on the handheld and on the car infotainment

Our application consists of an HTML user interface description and a JavaScript routine that retrieves the accelerometer data from the handheld hardware through the web application framework. Based on this data, it calculates the driving behavior and visualizes it with a smiley. In order to use this Web Application in the car, the DPWS stack creates a new Web Service proxy and advertises it to other devices active in the local network. Besides a general description of the application, this DPWS service offers access to the application data and to the required interfaces of the

[1] Several comparable applications of this type exist:
Toyota Sweden AB,"A glass of water", www.aglassofwater.org
Fiat Group Automobiles, "eco:Drive", www.fiat.com/ecodrive
DriveGain Ltd., "DriveGain", www.drivegain.com
(all September 2010)

Web Application Framework. Knowing the name and origin of the application, the car infotainment system can download the remote application and display a button to start it. During the DPWS export process, the JavaScript routine is modified. The method to access the accelerometer is indicated as a remote function. Once the application is executed as a remote web application in the car, this indication allows the Automotive Web Application Framework to forward accelerometer data requests to the handheld through DPWS communication. However, if a local accelerometer is available, the Automotive Web Application Framework may also use that. Additionally, the Economic Driving Application has an optional entry for the average fuel consumption per time period. While this data is not available on the handheld and is omitted there, the application can access this data when it is running in the car and it can show this optional entry. The different representations are depicted in figure 2 which shows the table view on the Smartphone and in the car infotainment. Although both representations are based on the same logic, resources and HTML UI structure, model specific Stylesheets and widgets are used for rendering the application in the car.

5. RESULT AND CONCLUSION

The presented coupling and user interface transfer mechanisms allow seamless mapping of different input controls and widgets to multi-platform applications. Applications may access local data and logic as well as remote data and logic. Beyond that, applications may access specific automotive APIs that may be standardized in future. Fallbacks are included for the case that certain hardware features and corresponding service APIs are not available. The requirements described in section 1.2 are met:

- Driver distraction through visual entertainment is suppressed because widgets are rendered in an automotive styling without animations.
- Contrast, font and widget sizes are adapted to the car by special Stylesheets combined with dynamic styling.
- Although the Automotive Web Application Framework cannot increase service response times, it can prioritize information and offer feedback like a please-wait-message.
- The automotive Application Framework may interrupt interaction at any time and may resume it accordingly.
- A consistent feedback output is achieved and additional status and failure messages may be given.

Web Applications however cannot be as tailored to the automotive platform as services developed by the car manufacturer, but it is still better to access handheld services through the car's HMI than using services directly on the handheld while driving. In the end, the user decides if he likes to use certain services. Nevertheless, the car manufacturer keeps the influence on control and service verification.

Future work will focus on the transcoding of platform specific user interface descriptions that are used on modern Smartphone platforms into HTML in order to couple arbitrary applications with the car. Another outstanding task is the evaluation of the driver distraction.

6. ACKNOWLEDGMENTS

Thanks to all my colleagues at the Institute for Communications Technology and to all my project partners within the research project "Connected Cars in a Connected World" (C3World).

7. REFERENCES

[1] Ford Motor Company, "Ford SYNC (R)", http://www.fordvehicles.com/technology/sync/, July 2010.

[2] Callum, S et. al.; Integration of Nomadic Devices: The AIDE Use Case, AIDE Deliverable D3.4.3, Information Society Technologies (IST) Programme, July 2008

[3] UPnP Device Architecture 1.0, v.1.0.1, UPnP Forum, 2006.

[4] Breiner, K.; Maschino, O.; Görlich, D.; Meixner, G., Towards automatically interfacing application services integrated into an automated, model-based user interface generation process, 4th International Workshop on Model Driven Development of Advanced User Interfaces. International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI-2009), February 8, Sanibel Island, Florida, United States

[5] de Melo, G.; Honold, F. Weber, M.; Poguntke, M.; Berton, A.; Towards a Flexible UI Model for Automotive Human-Machine Interaction, Proceedings of the First International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2009), p. 47-50, September 21-22 2009, Essen, Germany. DOI= http://doi.acm.org/10.1145/1620509.1620518

[6] Stolle, R.; Saad, A.; Weyl, D.; Wagner, M., Integrating CE-based Applications into the Automotive HMI, SAE World Congress 2007, April 16-19, 2007, Detroit, Michigan.

[7] Commission Recommendation of 22 December 2006 on safe and efficient in-vehicle information and communications systems: update of the European Statement of Principles on human machine interface, Official Journal of the European Union, February 2007, Brussels, Belgium

[8] Devices Profile for Web Services (DPWS), OASIS Standard, July 2009.

[9] Sonnenberg, J.; A distributed in-vehicle service architecture using dynamically created Web Services, 14th IEEE International Symposium on Consumer Electronics, June 07-10, 2010, Braunschweig, Germany.

[10] Stark, J., Building iPhone Apps with HTML, CSS, and JavaScript, O'Reilly Media, Sebastopol, California, February 2010