Tumaco: Collaboration of Mobile Devices in Hypermedia-based In-Vehicle Infotainment Systems

Christian Grévisse and Steffen Rothkugel

Computer Science and Communications Research Unit
University of Luxembourg
6, rue Richard Coudenhove-Kalergi L-1359 Luxembourg
christian.grevisse.001@student.uni.lu, steffen.rothkugel@uni.lu

Abstract. In-Vehicle Infotainment systems nowadays are largely limited to one-to-one relationships with the drivers' mobile phone. In this paper, we propose Tumaco, a SOA-based IVI system, which enables the seamless integration of all devices of all passengers. Collaborative services can be executed within a cross-platform ecosystem through the use of semantic hypermedia. In addition, passengers may customize their own user experience in order to reduce mental workload. Our prototypical implementation showed the benefits of opening the user experience to all passengers on a car. Finally, we present a framework that enables developers to provide their own services within our ecosystem.

Keywords: In-Vehicle Infotainment Systems; Collaborative Computing; Customization; Hypermedia

1 Introduction

Today's In-Vehicle Infotainment (IVI) systems are often centered around a one-to-one relationship between the car's on-board unit and a single smartphone. Even upcoming industry products like Apple CarPlay or Android Auto focus on, partly cable-based, connections where the corresponding IVI UI is shown on screen, but most application logic runs on the single connected phone. There are two major issues with this approach. First, an IVI system should not just be a unidirectional graphical data stream ported from the smartphone to a car's screen [14]. Second, the mentioned one-to-one mapping does not allow a collaboration between data and services of all brought-in mobile devices of all passengers [4].

In addition, the user experience (UX) is often focused on the driver, although other passengers might also feel the need to customize their very own IVI UX. Corresponding to the philosophy of Marc Weiser, the father of Ubiquitous Computing, computing devices should evolve from being personal items to environmental ones [6]. In the IVI domain, this may only be achieved if it is possible to easily move user preferences (e.g. the design of a digital speedometer gauge)

from one car to another. This might come very handy in a car-sharing context, enabling a coherent, uniform UX. A UX known to the user avoids mental overload in opposition to sharing cognitive resources between the primary driving task and secondary tasks related to the adaptation to an unknown UX [11,3].

An IVI system communicating with only a single mobile device is probably not inline with Weiser's vision: He stated that the *real power of the concept comes not from any one of these devices – it emerges from the interaction of all of them* [20]. A heterogeneous peer-to-peer architecture introduces challenges because of the diversity of different operating systems and hardware requirements. Nevertheless, such a setup might result to be beneficial. Imagine the passengers aboard a car want to listen to some music. It would be desirable to determine, based on the music libraries of all present mobile devices, the most represented common music genre. Based on this genre, the different devices could, one at a time, be instructed to play a song and share the song's meta-information (e.g. artist, album cover, lyrics) on the infotainment screen(s). But maybe the device which is playing the current song has neither the corresponding album cover nor a data plan to download it, whereas another device might offer such a service and could help out.

In this paper, we propose Tumaco, a SOA-based IVI system with semantic hypermedia support, allowing collaboration of services among connected mobile devices. Tumaco is realised as an extension on top of existing mobile platforms, such as iOS and Android, seamlessly integrating with the genuine system. Our IVI ecosystem considers all passengers aboard a car by allowing them to customize their UX through the use of Web Components. In addition, we introduce `TumacoKit`, a framework which enables application developers to extend the Tumaco platform, without requiring architectural modifications. A prototypical implementation of Tumaco has been realised.

The remainder of this paper is organized as follows. In section 2, we summarize related work from different domains. In section 3, we present the major concepts of our approach and show their role within our introductive example. In section 4, we list some details about our prototypical implementation. We conclude and suggest some work for the future in section 5.

2 Related Work

The GENIVI Alliance has been working on an open source development platform for IVI systems based on Linux. Existing open source software components, not necessarily from the automotive domain, get adapted and integrated in order to reuse *non-differentiating* middleware features [1]. On the HMI side, the Tizen operating system has created a GENIVI-compliant IVI profile.

The support for a seamless integration of mobile devices in the IVI user experience recently has grown, as smart phones and tablets may actually provide services and data to improve safety and security rather than distract the driver, if they are properly integrated [2,10]. The exchange of user preferences, settings and functionalities between mobile devices and the IVI [4,3,5] as well as the use of

automotive data from the car together with the intelligence of connected mobile devices [14] can be advantageous. Rich interaction protocols may facilitate these interactions [13].

Many recent approaches are based on a service-oriented architecture. Jaguar Land Rover exchanges JSON-RPC messages between mobile devices and the IVI system to remotely regulate HVAC control [8]. Sonnenberg integrates mobile devices and the IVI system using web services [17,18,19], whose advantage is their independence of any platform or programming language. The Device Profile for Web Services (DPWS) standard allows to address services on the network [7].

An API with general media types needs additional knowledge to understand the semantics of a message [12]. However, creating a new media type for each new appliance would lead to a media type explosion [16], making it difficult to reach a consensus. There would be a lot of *superficially similar [...] but completely incompatible APIs*. JSON-LD allows self-descriptive, hypermedia-based machine-to-machine communication for truly RESTful web services by adding semantic annotations to regular JSON documents.

With Web Components, Google recently introduced reusable, encapsulated UI components into HTML5 [9]. They enable both component and data reusability, as they can be fed with JSON-LD data. Semantically enriched user interfaces, in contrast to ambiguous, locally understood UI descriptions based on pure XML or HTML, are reusable, allow cross-application interaction and ensure separation of concerns [15].

3 Tumaco

Inspired by existing SOA-based approaches, we further extended them by introducing semantic hypermedia support in order to allow a seamless integration of mobile devices in an IVI environment. Our goal is to enable collaboration while allowing *all* passengers aboard a car (including the driver) to customize their user experience based on their brought-in preferences. Linked Data is at the heart of our IVI architecture. Our approach should also allow the ecosystem to be extensible by application developers rather than by platform developers or OEMs.

An important assumption in our approach is that the car does *not* need to be directly connected to any cloud services through GSM, 3G or LTE networks. This way, network infrastructures will not need to cope with higher resource usage or overloads, as the connected mobile devices, working as relays between a car's IVI system and cloud services, can reduce the number of sent requests. Finally, the main business logic of services may stay at the side of the different connected mobile devices, which in general come with powerful hardware in comparison to embedded devices like IVI systems.

As a proof of concept, we have considered several use cases, such as showing the location of appointments saved on brought-in mobile devices on the IVI's map, configuring the layout of the car's digital speedometer with the user's preferences stored on his smartphone, playing music based on the most common

represented genre on the passengers' devices, as well as showing weather information on provided navigation points. These different use cases are representative for our three main pillars, namely collaboration of services, customization of the IVI UX and system extensibility through third-party apps.

3.1 Layered Architecture

In figure 1, we present the layered architecture of our approach. The Middleware layer, both at the IVI system and the mobile devices, includes RESTful servers and clients. The Service layer maps advertised services to their respective, platform-specific business logic. While the UI layer at the connected mobile devices is not a primary focus of our study, the UI layer of the IVI composes the user interface based on the user's preferences, the transmitted data and the available Web Components. The transmitted data (e.g. service descriptions, service requests, service input/output data) is serialized in JSON-LD, including semantic annotations to make the messages self-descriptive. This serialization binds the collaborative aspect, provided through the SOA, with the customization aspect, as Web Components may be fed with JSON-LD documents.

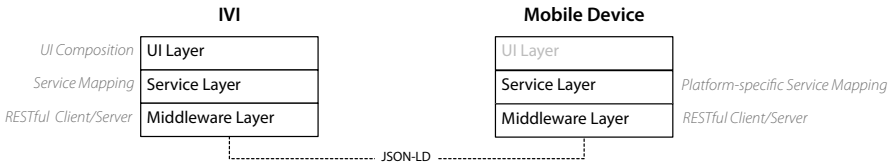

Fig. 1. Layered Architecture

3.2 SOA-based IVI Ecosystem

Both the IVI system and the connected mobile devices maintain a Service Directory within their respective Service layer. While mobile devices only hold their own service descriptions and stubs to IVI services in their Service Directory, the IVI itself knows its own services and the services from all connected mobile devices. When a mobile device registers with the IVI through the Tumaco platform app, it advertises its own service descriptions, serialized in JSON-LD, to the IVI and inquires latter's. Service descriptions include input and output types. They optionally include a UI component type, if a service performs a visual presentation. The word "type" here has to be understood as a semantic type (e.g. `Place`), not a programming language type.

In figure 2, we show a detailed view of the IVI's Service layer. The figure shows 3 collections, which are essentially database tables, containing service descriptions, service requests and data. Two further components, the *Request*

Dispatcher and the *Data Analyser* are responsible for actions concerning the service requests among the available services.

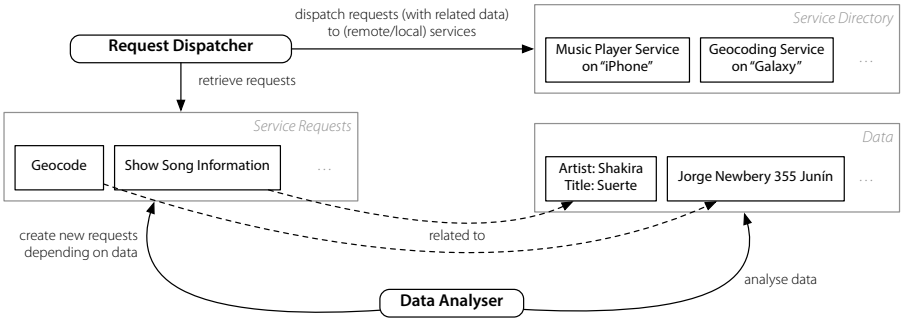

Fig. 2. IVI Service Layer

A service request may either come from a connected mobile device or from the IVI itself. The weak relation between service requests and their related data allows reusability of data items among different services. For instance, the information of the currently playing song, chosen by one connected smart device, may be reused by the IVI to ask another connected device to retrieve the corresponding album cover. Preference data, such as the background color of the speedometer, may also be pushed without a service request and directly used by the responsible Web Component. Service requests, once dispatched and the corresponding service executed, may provoke either another service request to be created (engaging a collaborative workflow) or data to be pushed.

The Data Analyser component checks, given a set of service-specific rules, whether data related to a service request is complete. For instance, the location of an appointment needs to include geocoordinates in order to be shown on the IVI's map. If the data is not complete, the Data Analyser will create a new service request to complete the data. In the previous example, this would mean that a service request for geocoding is created. Otherwise, the Request Dispatcher component executes the corresponding service request. Here, the map is reloaded to show all available appointments. If a service request maps to an IVI-own service, the IVI will execute the service's business logic. Otherwise, if the concerned service is offered by a connected mobile device, the service request is remotely dispatched through the underlying RESTful Middleware. Services, both at the IVI and on mobile devices, are defined in a descriptive, generic way, which means that the architecture or the platform app do not need to be altered in order to add new services.

Services offered by mobile devices can essentially be grouped into two categories. *Public Services* are altruistic mechanisms which reactively respond to a service request by (optionally) taking some input and returning some output. For

instance, the geocoding service may take an appointment location and return the corresponding geocoordinates. *Data Providers* proactively provide device-specific data (e.g. music library) if a matching service is offered by the IVI. In contrast to the altruistic nature of Public Services, Data Providers should be treated while respecting the user's privacy. The user should eventually be able to determine which data he really wants to share with the IVI.

Finally, we also wanted to open the ecosystem in a way that third-party apps may offer their services. This is achieved through Dynamic Services. While the platform app already offers a set of Public Services (e.g. geocoding) and Data Providers (e.g. music library), other apps may offer their service to the platform app and thus indirectly to the IVI (e.g. retrieve the weather of a location, provide navigation points). Both Public Services and Data Providers can be offered by other apps. At the IVI side, this separation is not visible, as the dispatching of any remote service is abstracted from the service execution. In order to offer services, an app needs to register them at the TUMACO platform app. This way, the platform app may advertise these dynamic services to the IVI system, which may later use them the same way as those directly offered by the platform app. Incoming service requests for dynamic services are relayed to the offering app. The inter-app communication process is platform-specific, but it always requires a way to address a precise service within a given app and optionally hand over some input data. The generic sequence of exchanged messages is shown in figure 3. We implemented `TumacoKit`, a Cocoa Touch framework for iOS which follows the described mechanism. An app needs to expose a URL scheme and identifier to be addressable from within the TUMACO platform app and handle incoming URL queries, possibly containing data in the form of URL parameters. An Android-specific implementation could realise the same mechanism by exposing Intent Filter Actions, data being exchanged and encapsulated within Intent Extras.

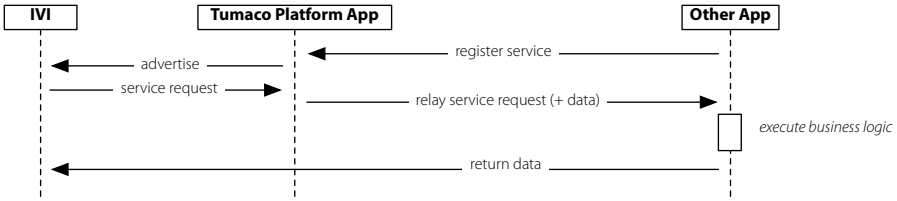

Fig. 3. `TumacoKit`: Service Integration

3.3 Custom User Experience & Web Components

To allow a consistent user experience for all passengers aboard a car, including the driver, the TUMACO platform app offers the user the possibility at the

connection to choose a screen. For the moment, we consider 5 screens: instrument cluster, central console, copilot, rear left, rear right. The screen - mobile device relationship is a one-to-one mapping, as a person may only sit at one seat at a time. The central console is a screen users may not check into, as it regroups all information sent from all connected mobile devices to give a global overview of the provided data to the driver (e.g. relevant location data). Due to privacy reasons, the other 4 screens show only the specific data transmitted from the corresponding mobile device. Customization can be achieved, as the user may design the layout of his specific screen by choosing the Web Components he wants to see, in the order and size he wants them to be put and even with the design options he wants (e.g. speedometer background color). For the driver, these customization aspects are especially important, as they may reduce confusion when using an unknown car with his own design preferences stored on his mobile phone. As Web Components are custom elements that can be imported into regular HTML5 code, they can be reused within the different screens, assigned with screen-specific data.

3.4 Use Case: Consensus on Music in Carpooling Scenario

In figure 4, we show how our introductive example is realised within Tumaco. After exchanging service capabilities, mobile devices detect that their Music Library Service (a Data Provider) matches the semantic types of the IVI's Music Collection Service and thus send their music library. The IVI detects the most represented common music genre and instructs a device, here the Android phone, to play a corresponding song. Latter now sees that the IVI has a service to display song information and sends the song's meta-information. Now, the IVI's Data Analyser component sees that this service also wants the album cover, which has not been transmitted by the Android device. Gracefully, the iPhone device provides a corresponding service and is instructed to download and send back the corresponding album cover. Now, the IVI has all necessary data and displays them in a Music Player Web Component at the UI layer. This example shows a complex workflow, where one service execution may generate a new service request. With Tumaco, it is thus easily possible to realise a high level service, distributed among different iOS and Android devices, without the need to change these mobile platforms. In addition, as the involved services are declared in a descriptive way and are semantically annotated, the RESTful Middleware layer and the above lying Service layer can reason and take decisions without human intervention.

4 Implementation

We have built a prototypical implementation of the Tumaco ecosystem, including platform apps for both iOS and Android. The IVI is represented through a Django web server running on a Raspberry Pi (Model B). We have tested the implementation on an iPhone 5, running iOS 8.4 and on a Samsung Galaxy S

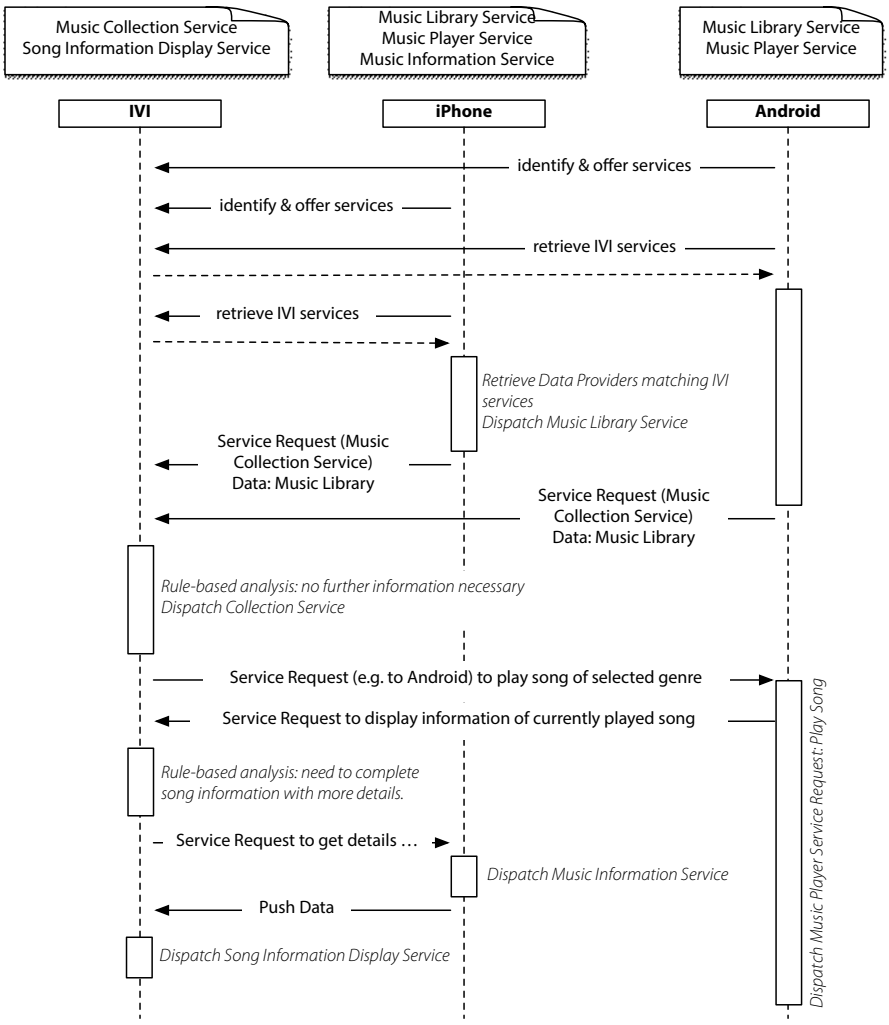

Fig. 4. Complete example: Consensus on Music

III, running Android 4.4. The implementation of the platform apps has shown the differences, advantages and disadvantages of building a common ecosystem using the iOS SDK in Swift and the Android SDK in Java. In particular, we use a WANET between the Raspberry Pi and the iPhone, and an infrastructure WLAN between the Raspberry Pi and the Galaxy S III, as Android does not support ad-hoc networks by default. The IVI's UI layer is built using HTML5, JavaScript and custom Web Components (e.g. navigation map, speedometer gauge, music player). For the moment, there only exists an iOS implementation of `TumacoKit`. Future work will include an Android implementation based on Intent actions and Extras equivalent to URL queries used in iOS.

5 Conclusion & Future Work

We proposed a SOA-based IVI system which seamlessly integrates mobile devices through semantic hypermedia support. Tumaco extends existing mobile platforms without requiring any changes to the genuine distribution. Services may engage in workflows and are both platform-independent and self-descriptive through semantics. `TumacoKit` allows mobile application developers to extend the ecosystem with their own services, without requiring architectural modifications. All passengers aboard may customize their user experience through Web Components, fed with semantically enriched data. We provided a full example of a complex workflow, which has been implemented in our prototype.

In the future, we plan to integrate available car data, either via the ubiquitously represented OBD-II interface or upcoming Web APIs accessing the CAN bus [14], into the Tumaco ecosystem. Privacy is of utmost importance, therefore the user needs to be given further control over which data gets shared with the IVI and on which screens it shall be accessible. New forms of displays such as HUDs on the windshield, replacing classical dashboards, may bring new opportunities for UI customization, to enable the driver to fully control his augmented reality user experience. Finally, services or even games over different cars could enable a new kind of social interaction in traffic jams.

References

1. G. Alliance. BMW Case Study. `http://www.genivi.org/sites/default/files/BMW_Case_Study_Download_040914.pdf`.
2. S. C. Boll, A. L. Kun, P. Fröhlich, and J. Foley. Automotive User Interface Research Moves into Fast Lane. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 2525–2528, New York, NY, USA, 2013. ACM.
3. S. Diewald, T. Leinmüller, B. Atanassow, L.-P. Breyer, and M. Kranz. Mobile Device Integration and Interaction with V2X Communication. In *19th World Congress on Intelligent Transport Systems (ITS)*, 2012.
4. S. Diewald, A. Möller, L. Roalter, and M. Kranz. Mobile Device Integration and Interaction in the Automotive Domain. In *AutoNUI: Automotive Natural User Interfaces Workshop at the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2011)*, 2011.

5. S. Diewald, A. Möller, L. Roalter, and M. Kranz. DriveAssist – A V2X-Based Driver Assistance System for Android. In *Automotive HMI Workshop at Mensch und Computer 2012*, 2012.
6. M. Ebling and M. Baker. Pervasive Tabs, Pads, and Boards: Are We There Yet? *Pervasive Computing, IEEE*, 11(1):42–51, January 2012.
7. M. Eichhorn, M. Pfannenstein, D. Muhra, and E. Steinbach. A SOA-based middleware concept for in-vehicle service discovery and device integration. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 663–669, June 2010.
8. M. Feuer. Remote Vehicle Interaction - Architecture High Level Description. `https://download.automotivelinux.org/POC/PoC_Spec/15-456-POC-Tizen3-RVI-HLD_Draft4_clean.pdf`, 2014.
9. E. Gasperowicz. Creating semantic sites with Web Components and JSON-LD. `http://updates.html5rocks.com/2015/03/creating-semantic-sites-with-web-components-and-jsonld`, 2015.
10. F. Hüger. User Interface Transfer for Driver Information Systems: A Survey and an Improved Approach. In *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '11, pages 113–120, New York, NY, USA, 2011. ACM.
11. C. Kennington, S. Kousidis, T. Baumann, H. Buschmeier, S. Kopp, and D. Schlangen. Better Driving and Recall When In-car Information Presentation Uses Situationally-Aware Incremental Speech Output Generation. In *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '14, pages 7:1–7:7, New York, NY, USA, 2014. ACM.
12. M. Lanthaler and C. Gütl. On Using JSON-LD to Create Evolvable RESTful Services. In *Proceedings of the Third International Workshop on RESTful Design*, WS-REST '12, pages 25–32, New York, NY, USA, 2012. ACM.
13. S. Murphy, A. Nafaa, and J. Serafinski. Advanced service delivery to the Connected Car. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, pages 147–153, Oct 2013.
14. S. Murphy, P. Szczechowiak, J. Serafinski, and T. Zernicki. Car-mobile integration for advanced telematics services delivery. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, pages 71–76, Jan 2014.
15. H. Paulheim and F. Probst. A Formal Ontology on User Interfaces Yet Another User Interface Description Language. In *2nd Workshop on Semantic Models for Adaptive Interactive Systems,(SEMAIS)*, 2011.
16. L. Richardson, M. Amundsen, and S. Ruby. *RESTful Web APIs*. O'Reilly Media, 2013.
17. J. Sonnenberg. A distributed in-vehicle service architecture using dynamically created web Services. In *Consumer Electronics (ISCE), 2010 IEEE 14th International Symposium on*, pages 1–5, June 2010.
18. J. Sonnenberg. 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*, AutomotiveUI '10, pages 162–165, New York, NY, USA, 2010. ACM.
19. J. Sonnenberg. Connecting in-vehicle entertainment with CE devices via Multi-Platform Web Applications. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 515–516, Jan 2011.
20. M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):66–75, 1991.