Metadata Aggregation for Personalized Music Playlists

A Multi-layered Architecture for an In-Car Prototype

Clemens Hahn[1,2], Stéphane Turlier[1,3], Thorsten Liebig[2], Sascha Gebhardt[1,4], and Christopher Roelle[1]

[1] BMW Group Research and Technology, Munich, Germany
[2] Ulm University, Faculty of Computer Science, Ulm, Germany
[3] Eurécom, Multimedia Communications Department, Sophia-Antipolis, France
[4] University of Munich, Department of Informatics, Munich, Germany

Abstract. The growing amount of digital music content and the increasing connectivity of vehicles raise new challenges in terms of media access for vehicle drivers. Creating easily a personalized playlist in vehicles involves a unified representation of various metadata, combined with a mobile architecture addressing media resolution and aggregation issues. This paper analyzes the technical aspects of mobile access to music metadata and its use in a personalized playlist generation scenario. A prototype illustrates this study and gives first results.

Keywords: Metadata, content aggregation, mobile architecture, playlist creation.

1 Introduction

Music is one of the most consumed media assets in vehicles. The increasing vehicle internet connectivity is bringing more multimedia content to the mobile use every day. Digital music assets are nowadays distributed on-demand by internet services for their consumption.

A typical use case in a modern vehicle is: The driver wants to quickly select specific online digital music tracks, in order to create a playlist corresponding to his tastes of the moment. Since his primary task is to drive a vehicle, this selection process has to provide first class user guidance in terms of minimal interaction, presentation and explanation. While driving, he wants to be able to influence the composition of the playlist by choosing alternative tracks or music styles, without having to reformulate the whole selection query.

The vehicle integration and adaptation of such services is raising a lot of technical challenges in terms of software architecture, network infrastructure and usability. We will address the following aspects in this paper:

G. Leitner, M. Hitz, and A. Holzinger (Eds.): USAB 2010, LNCS 6389, pp. 427–442, 2010.

- provide the user with playlist generation techniques that require few interactions but still allow granularity.
- define an efficient architecture, adapted to the mobile use and the vehicle requirements.
- make use of internet cloud metadata from external providers while containing the software complexity overhead.

We will first present the problem of playlist creation for vehicles in section 2 and describe the state of the art in section 3. After a technical discussion about the different sources of music metaknowledge according to their integrability into a mobile architecture in section 4 and we will propose a prototype of playlist generation, that takes into account architecture constraints for the aggregation of content, as well as techniques that allow the user to take advantage of it and consume easily music in a vehicle.

2 Using Metaknowledge to Create Music Playlists

From a user's point of view, the creation of a playlist is an optimization problem between the time necessary to create the playlist and the quantity of music assets which are available. Important parameters are the quantity and the quality of available information that helps the user to make his decisions. We will discuss in this section the different kinds of criteria that can be used to create a playlist and explain how a mobile device like a vehicle can access them.

2.1 Techniques for the Creation of Music Playlists

Digitalized music is a media asset that can be sorted and selected through different techniques [2].

Creating Playlists Based on Music Similarity

- **acoustic similarity:** different low-level features (MFCCs or MPEG-7) can be extracted from the audio signal and using data mining techniques to compute similarity, [10], [9].
- **expert opinion:** the music genome project [24] has identified more than 400 musical attributes that are analyzed by experts and saved in a database.
- **social information:** Social services allow users to share free-text tags, tracks, artists or genre favorites, playlists and to write comments. They often implement implicit relevance feedback mechanisms based on the monitoring of the user behavior (like the scrobbling protocol [13] from Last.FM[1]). Based on this information, it is possible to compute playlist co-occurrence [3], or to analyze common tags between playlists.

[1] `http://www.last.fm/`

Creating Playlists Based on Filtering Criteria

- **artistic performance:** The semantic description of music performances [19] can help to define playlist creation criteria: author information, performer, instrument, year of release, etc.
- **high-level acoustic features:** Based on the accoustic features mentioned previously, MPEG-7 defines high-level descriptors such as *Timbre*, *Melody* and *Tempo*. They can be considered as understandable for all users, even those with minimal music expertise.
- **genre:** Genres are defined on cultural and historical backgrounds [14]. They define commonly accepted cultural properties of music composition and performance.
- **mood:** A mood is a long lasting personal affect. The energy-stress [22], or valence-arousal [21] models have been developed by psychology research to semantically describe it. Regarding music the Moodswing [11] proposes a technique to select music according to the mood.
- **web crawled information:** The world wide web is an important source of comments on musical performances. Analyzing it allows to identify the popularity of an artist (how often the artist has been quoted in musical reviews) and his hotness (how he he has been quoted the last week).

2.2 Accessing Media Information Knowledge

Accessing the information we have defined in 2.1 is a crucial step to its aggregation and its use in playlist generation. This section describes the different types of multimedia metaknowledge sources that can be used to filter music tracks. In our context, metaknowledge means every kind of knowledge about the content that can help in selecting it among others through one or several criteria. We draw a distinction between metadata that is extracted from the content, metacontent that is delivered together with the content and metainformation that can be linked to the content.

Agent Self Extracted Metadata. A first option to access knowledge is to extract it directly from the content (see table 1). Low-level features as well as some high-level features can be accessed this way.

In spite of their advantages, we believe that the metadata extraction has currently too many shortcomings to be integrated as such in a vehicle. This is the reason why we decided not to use this technique in our prototype. However, we will consider in section 4.3 other external services that propose to deliver metacontent extracted from the content itself.

Co-delivered Metacontent. Metacontent like the performing artist, or a cover art illustration is delivered together with the audio content (see table 2). This knowledge is tightly linked to the value chain (see 4.1) of the music distribution;

Table 1. Self extracted metadata

Playlist creation	Integration in a mobile device
Advantages	
The quality of the metadata is entirely based on the quality of the extraction. The more features can be extracted, the more criteria can be used. A lot of algorithms based on similarity are available.	Extracting information directly in the client limits the need of internet connectivity.
Disadvantages	
Low-level descriptors are useful for the computation of music similarity but cannot be used as such as filtering criteria by a user with no music expertise. Pure acoustic based music selection has shown some limits and underperforms, methods based on high-level data[5].	The extraction of metadata in a multi-layered architecture does not scale to the aggregation role. Considering the chain value, the dynamic computation of such features in a client or in an aggregator does not scale to the increasing amount of available content.

from the producer to the publisher and to the online provider. It mainly consists in expert information: track name, artist name, album name, year of release. The ID3 tags were one of the first attempts to propose a standard way to deliver track, artist, album and genre information, within the MP3 mediacontainer. Afterwards then other formats have been proposed based on structured binary information of the mediacontainer, or XML formats.

Separately Delivered Metainformation. In the past years, an increasing amount of internet services aimed at federating new sources of metainformation without providing music data themselves. We can distinguish:

- textual information based on web crawling.
- music similarity: it can be based on automated music analysis (Gracenote), expert annotations (Music Genome Project), or social information (playlist-co-occurrence).
- social tagging and classification, like Last.fm or Finetune who allow their user to give free-text tags to the tracks or the artists, and compare user profiles based on the listening behavior.

Since they do not propose the content themselves (see table 3), they all need to implement some kind of identification in order to deliver the metacontent for a specific music track. This identification can be based:

- either on extracted metadata: computation of a unique fingerprint of the track [4],[1];

Table 2. Co-delivered metacontent

Playlist creation	Integration in a mobile device
Advantages	
Correctness: Co-delivered metacontent has usually a very good quality in terms of reliability.	This type of metacontent does not need any extra processing neither on client side nor on aggregation side.
Disadvantages	
Consistency: spelling differences between different providers may lead an aggregating recommender system to propose twice the same artists or oppositely to underestimate his importance. Completeness: for taxonomic values like genre or mood, the inner structure (number of genre, hierarchy, dependency between semantic concepts) can vary a lot between providers.	The whole content need to be requested even if only the metadata is necessary.

– or on the co-delivered textual metacontent: the metadata is available through a search engine using the name of the track, or the name of the artist performing it.

3 Related Work

The music information and retrieval (MIR) research has already presented recommender systems to help users to create playlists. The *Simple Playlist Generator* [15] proposes to create a playlist based on a seed song. Using the user's skipping behavior [16], it is possible to infer implicit relevance feedback and improve the playlists.

In the field of data visualization, the development of user interfaces for the display of music libraries such as in [12] or [8], has also lead indirectly to propose clients capable of creating music playlist by selecting regions on a map.

A third important aspect in the literature are the recommender systems based on user input filters. Satisfly [17] proposes to select the variance around a genre, an interpret, or an album, as well as a desired tempo or a specific time period (*i.e.* 60', 70', etc.). Musiclens [7] uses other original metadata like the intention of the music, importance of the voice or number of instruments.

Most of this research effort has focused on finding innovative ways to use metaknowledge for the creation of playlists. However, they gave rather little focus to the analysis of the quality of the metaknowledge for an online scenario

Table 3. Separately delivered metainformation

Playlist creation	Integration in a mobile device
Advantages	
Those data are independent from the content provider, so they are not influenced by commercial orientations and it allows addressing simultaneously different content providers. Some kind of information like popularity, can only be gathered through transverse crawling methods (web information, radio charts, etc.), that content providers do not provide.	Most of those services can be abstracted and aggregated in a multilayered architecture.
Disadvantages	
Correctness: folksonomy and other user generated content need often extraprocessing of normalization [18]. The lack of consistency in the codelivered metacontent may cause problem to retrieve linked metainformation.	The architecture of such system is less efficient than former systems, since the query of new metadata requires bidirectional exchange of information.

where the content changes every day, and the way to integrate it efficiently in a mobile infrastructure. Moreover, most of them did not address the topic of vehicle clients which have limited user interaction possibilities.

4 Prototype

Before giving the details of our prototype we think it is important to analyze what are the different sources of content and metacontent that we need in order to perform playlist generation in a vehicle using daily changing on-demand media.

4.1 Role Definition

The music industry is a complex ecosystem that has been dramatically changed by the digitizing of music assets [20]. The generation of playlists and more generally the selection and display of media content can be schematically positioned at the end the value chain, after the production and publication of content (see figure 1).

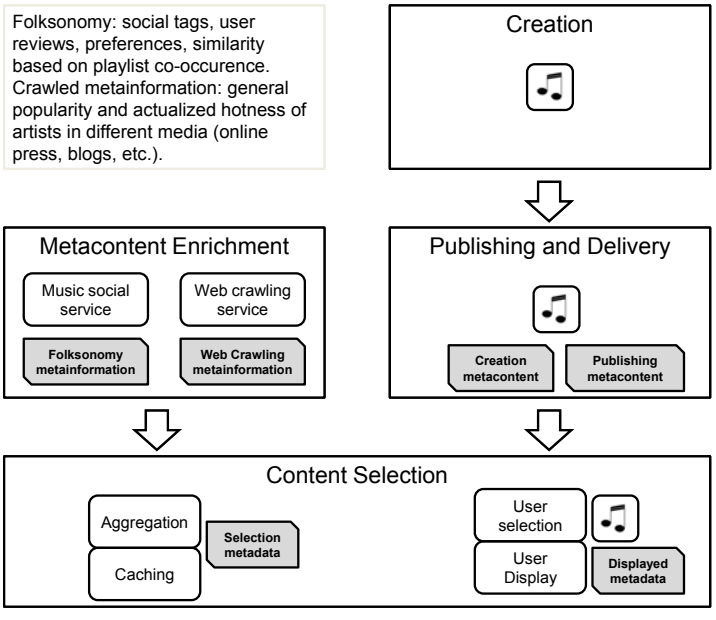

Fig. 1. Extension of the value chain, with internet services

With the development of internet as a media distribution channel, new services have emerged which do not propose content but rather metadata. We identify mainly two categories of them.

- Web crawlers and cloud services: They provide metadata based on information gathered from the web.
- Social services: They provide information based on social services.

On the one hand, the increasing amount of information available through this services simplifies some processing tasks for the clients as we announced in 2.1 but on the other hand they involve a multiplication of interfaces which would lead to a serious overhead in the software complexity of vehicle clients, and increase the latency needed to access those services. New functional roles need to be developed in order to achieve:

- The aggregation of multiple providers, like music providers and metacontent providers while maintaining a low software complexity.
- The caching of metacontent to interconnect structured information and provide reactive user interfaces.

4.2 Functional Architecture

Our prototype focuses on the implementation of a multilayered architecture.

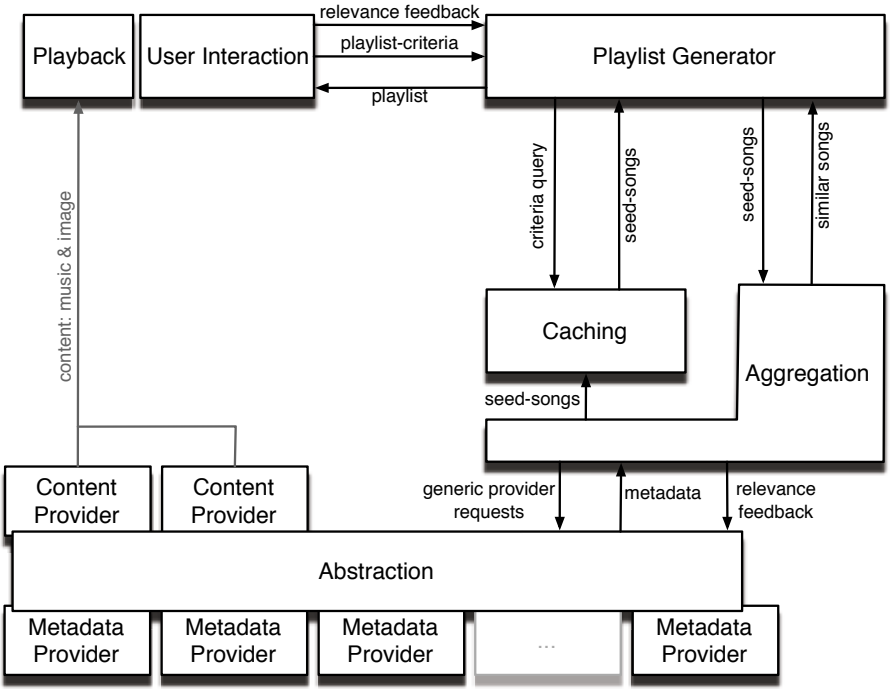

Fig. 2. The functional components of the architecture for generating playlists in a mobile environment

Abstraction. Since the prototype uses different kinds of metadata- and content-provider to have access to a comprehensive set of knowledge, it needs to access them all in a common way. This component abstracts their functionalities by providing common functions like *search for track* or *similar tracks for* a specific song to other software components. Thereby the precise implementation of the several providers is hidden to the frontend and the playlist generator.

Aggregation. The data is collected and merged from different providers through the abstraction component. This component builds up information entities which hold all required metadata. The set of required metadata is defined by the caching component.

During the aggregation of data different metadata sources have to be assembled. Our prototype uses metadata provided by experts as well as metadata provided by the so called folksonomies. To assemble them the system has to consider factors like spelling differences in track or artist names and the various kinds of identifiers.

Caching. This component caches a representative sample of the digital available music of our content providers. In order to best support the user in his playlist

generation process, the system must react to the user interactions very fast. Short reaction times reduce the risk of distraction from the primary task of the vehicle-driver. Thereby, it reduces the cognitive disorder.

These cached information permits a multi-criteria playlist generation. The cached tracks have to support the following attributes:

- name of track, artist and album;
- year of release;
- genre and mood information;
- ratings of the popularity;
- user specific ratings;
- album cover;
- download URL of the audio data.

The Aggregation component provides this component with metadata, collected from different providers. This data collections and caching process is scheduled as a background process, independently from the playlist generation process.

Playlist Generator. This component receives from the vehicle, resp. from the driver, a set of criteria which are the constraints for the desired playlist. By matching these criteria with cached songs it receives some seed-songs. These seed-songs provide the starting-point of the generated playlist. Combined with similar tracks to these seed-songs (over the Aggregation and Abstraction component), a playlist is automatically built up (for details see section 4.2 - the interlacing strategy).

User Interaction & Playback. By interacting with this component the user can control the whole system. As described he can select different criteria for his playlist. He can combine the following criteria in order to tell the system his current music tastes:

- **Genre:** A hierarchicaly structured tree with 8 top level genres likes *Rock*, *Pop*, *Jazz*, *Classical* and so on;
- **Mood:** A set of 25 moods, ordered in a valence-arousal grid;
- **Popularity:** A three stepped scale from *underground* over *mainstream* to *hot*;
- **Year:** A period of time or the exact year of publication;
- **Origin:** A personalized option like music from the own repository or loved songs.

For each selected criteria a preview of the playlist is presented. The songs of the preview derive from the caching component. Therefore they can be displayed very fast.

After the user has selected all his playlist constraints, the Playlist Generator builds up a list of tracks which fit these criteria. Thereupon the Generator returns

Table 4. Similar tracks for two seed song, collected from Last.fm's web service

seed song 1: Mando Diao - Gloria			seed song 2: Kasabian - Fast Fuse		
T1,1	Mando Diao	High Heels	T1,2	Kasabian	Take Aim
T2,1	The Libertines	Can't Stand Me Now	T2,2	Arctic Monkeys	Fire and the Thud
T3,1	Johnossi	Man Must Dance	C4	**Editors**	**Munich**
C1	**The Kooks**	**Do You Wanna**	T3,2	White Lies	Death
T4,1	Sugarplum Fairy	She	T4,2	Franz Ferdinand	Turn It On
T5,1	The Hives	Walk Idiot Walk	T5,2	Arctic Monkeys	Potion Approaching
C2	**The Hives**	**Tick Tick Boom**	C1	**The Kooks**	**Do You Wanna**
T6,1	Johnossi	18 Karat Gold	T6,2	The Libertines	Can't Stand Me Now
T7,1	Razorlight	Wire To Wire	C3	**Razorlight**	**America**
C3	**Razorlight**	**America**	C2	**The Hives**	**Tick Tick Boom**
C4	**Editors**	**Munich**	T7,2	Kaiser Chiefs	The Angry Mob

the playlist to the user. Each track of the list has a download URL attribute. So the vehicle can request the songs directly from the content providers and playback them.

Query Adaption via Preference Relaxation. In the case that a filter combination will not produce any results due to conflicting criteria the search query is gradually relaxed along the path of the category taxonomy. In particular, the most specific categories are replaced with their respective more general super categories until there are matching results which can be combined to a playlist of reasonable size. This approach is easily extensible to more sophisticated relaxation mechanism which incorporates additional domain knowledge into account such as those described in [23] for instance.

Fast Filter Criteria Preview through Pre-Cached Content. In order to support the driver in his playlist generation process a preview of the desired playlist will be offered. Each time the filter criteria changes a new preview demonstrates its influence on the final result.

The user can add as many filters as he wants, while filters from different categories are linked by a logical `AND` and filters from the same category are linked by a logical `OR`. We expect the user to naturally assume that kind of linkage.

Track Interlacing Strategy and Parallel Metadata Aggregation. Once a preview has been requested, the backend starts automatically a playlist generation process in background and saves the results. If the user is pleased with the preview, the playlist is transferred from the backend to the frontend, otherwise, it is deleted.

Since we are using different metadata providers to generate the playlist we had to develop an algorithm to combine the results. The playlist generation

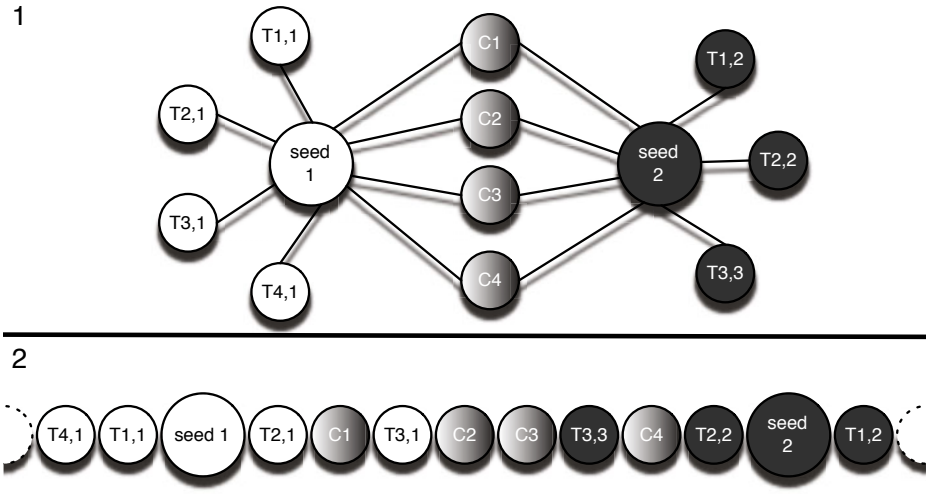

Fig. 3. The interlacing method to generate playlists with smooth track-to-track intersections. At the top two seed-songs with similar tracks (T) and common similar tracks (C). At the bottom the generated playlist with interlaced tracks for two seed-songs.

process is based on the preview tracks which consist in a list of n seed songs; $\mathcal{S}[0] = \{s_1, s_2, ..., s_n\}$. For every s_k there is set of m_k recommendations $\mathcal{R}_{s_k} = \{t_{1,k}, t_{2,k}, ..., t_{m_k,k}\}$ which is retrieved from metadata providers. Our algorithm process incrementally and takes the first seed song s_1 and searches in $s_2, ..., s_n$ a song s_k such that the cardinal of $\mathcal{C}_{s_1,s_k} = \mathcal{R}_{s_1} \cap \mathcal{R}_{s_k}$ is maximum, that is to say that the recommendations of s_1 and s_k have the maximum of tracks in common (see figure 3-1). The algorithm then carries on with the set $\mathcal{S}[1] = \{s_k, s_2, ..., s_{k-1}, s_{k+1}, ..., s_n\}$ where s_1 has been removed, until $\mathcal{S}[n-1]$ when the set of seed songs is exhausted. This way, we create an ordered listed chain of seed-songs $\mathcal{S}' = \{s_1, s'_2 ..., s'_n\}$ where s'_k are a permutation of s_k and sets of common songs which can have different cardinality $\mathcal{C}_{s_i,s_j}$. A playlist can be created by placing the common songs between the seed songs as following: s'_i, $\mathcal{C}_{s'_{i+1},s'_i} \setminus \mathcal{C}_{s'_i,s'_{i-1}}$, s'_{i+1}, $\mathcal{C}_{s'_{i+2},s'_{i+1}} \setminus \mathcal{C}_{s'_{i+1},s'_i}$, where $\mathcal{C}_x \setminus \mathcal{C}_y$ is the difference between the sets C_x and C_y.

The figure 3-2 and the table 4 illustrate how the remaining tracks that are not common to the seed songs (*i.e.* they belong to the complementary of $\mathcal{C}_{s'_i,s'_{i+1}}$ in $\mathcal{R}_{s'_i} \cup \mathcal{R}_{s'_{i+1}}$) are interlaced between the common tracks completing the result, in order to create a playlist smoothly going from a seed song to another.

In order to deliver a playlist to the user as fast as possible, the collections of similar songs are (a) parallel retrieved and (b) the playlist generation is split in multiple parts. The parallel request for similar tracks accelerates the generation

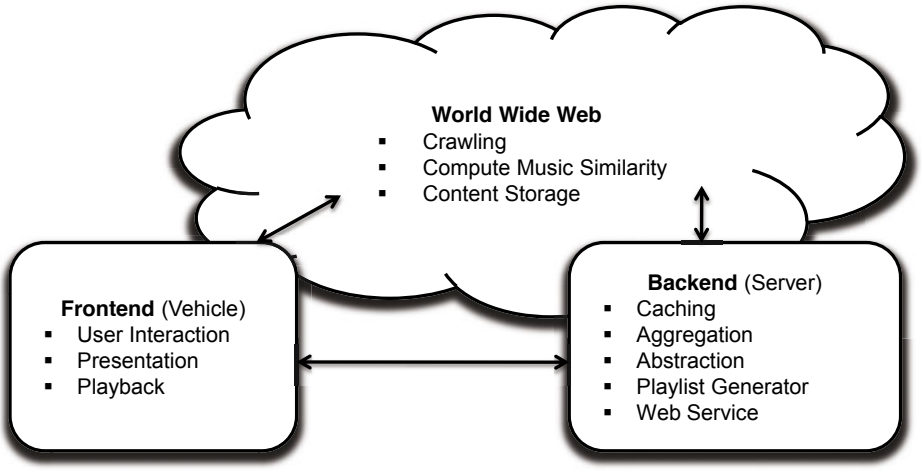

Fig. 4. The hardware architecture of the prototype

process. Depending on the latency of the service providers, waiting for the response takes a significant amount of time in generating the playlist. By splitting up the playlist a first part of it can be delivered in an acceptable delay to the user. Thereupon, while listening to the first tracks of the playlist, the other parts can be built up in background.

4.3 Deployment Architecture for a Mobile Use

The presented functional components have to be deployed on an adequate hardware architecture. The prototype is split into two main applications: firstly the graphical user interface (the so called frontend), with whom the user can interact in the car and secondly a server (backend). A conceptual model of this architecture is presented in figure 1. The discussed functional components are mapped to the corresponding hardware components.

Frontend. The UI-Prototype we implemented for the frontend is written in Flash. The input/output-devices used in the prototype are the controller knob (a push-shift-rotate controller) for input and the central information display (CID) for visual output. The music playback performs over the car audio system. The flash application runs in a Web-Browser based on Webkit and specifically developed to read commands from the controller. Commands are forwarded to the flash run-time using JavaScript. The browser is capable of handling gzip compression over HTTP, which helps reducing the latency of exchanging requests.

We use a 7 Series BMW for our experimental vehicle. It is equipped with an UMTS router and the described visual and haptic interface. A picture of the user interface in the vehicle is presented in figure 5.

Backend. We installed on a server a servlet container (Glassfish) and a relational database (MySQL). The functional component *Caching* is mapped to the relational database. The components *Abstraction*, *Aggregation* and *Playlist Generator* are deployed in the servlet container . The *Abstraction* handles the access to the several providers by implementing the web service API over WSDL or REST. The *Aggregation* and *Playlist Generator* implement the business logic.

World Wide Web. The several providers for metadata and audio/image-content reside in the world wide web. These providers aggregate dynamically metadata by crawling the web for music related content. They also support services to identify tracks and deliver similar music to given seed-songs. We use the music-catalog from Rhapsody[2] with over 9 Mio. songs as content-provider as well as metadata-provider. Additionally Gracenote[3] supplies the prototype with mood-information. The social network Last.fm and the web crawler The Echo Nest complete the list of the metadata providers that we have used.

The backend offers a web service that can be accessed by the frontend. This service is designed in a RESTful style [6]. The data is transferred between front- and backend in XML format over HTTP.

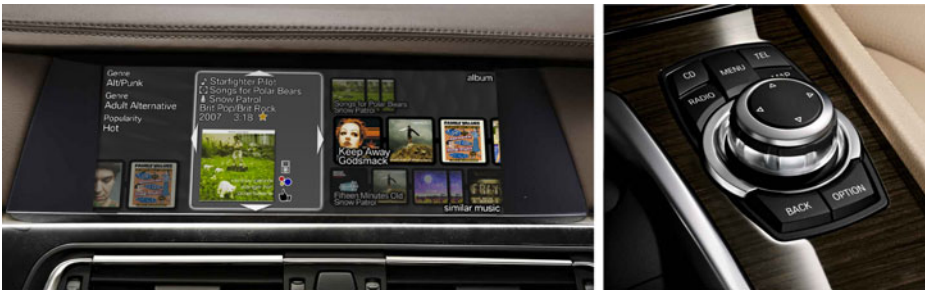

Fig. 5. The experimental vehicle. left: graphical user interface, right: controller knob.

First Results. We monitored the response time of the server to the client based on the simulation of 1000 user preview queries that generated around 1600 queries from the frontend to the backend, with a number of playlist criteria from 1 to 6. As depicted in figure 6, the response time does not depend of the number of criteria (still it is only database selects on the different rows of the track table) and remains reasonable from user experience since the user does not have to wait longer than 3 seconds. The variability in the results can be explained by the jitter of mobile communications, the very changing latency of internet web services. In order to permit an almost immediate start of the final playlist, its first three tracks are always composed of cached tracks, while

[2] `http://www.rhapsody.com`
[3] `http://www.gracenote.com`

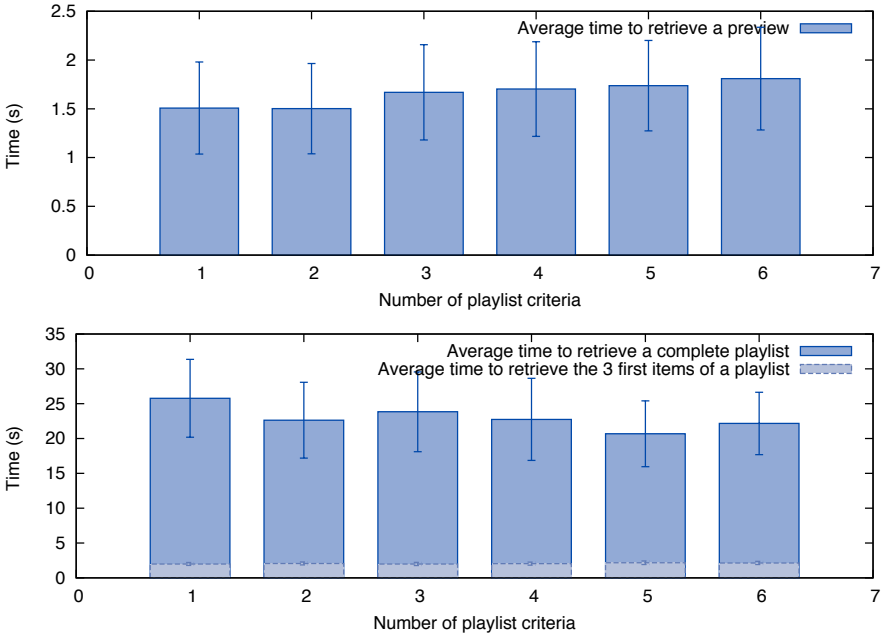

Fig. 6. Response time for the preview and the playlist over 1000 user queries

our system retrieves additional metadata in the background with the algorithm described formerly. As a result, the rendering of the playlist can start right after the preview.

5 Conclusion and Future Works

We have analyzed the issues encountered when tackling the topic of personalized playlist creation in a vehicle. Our vehicle scenario involves usability aspects like selecting different metadata filters to create a playlist, overcoming the latency of some internet services and proposing the user alternative choices to modify the final result.

We have presented a prototype to illustrate how the main functional components, user display, caching, aggregation and abstraction can be deployed in a mobile architecture. We have noticed that even if this deployment gives satisfying results it could be improved to provide a more reactive interface for the preview of user queries.

We believe that new web technologies that will be implemented in mobile devices like HTML 5 browsers or Adobe Air, allow the development of efficient caching methods on the client side. Combined with a synchronization mechanism with our backend, a future version of our client will be able to give immediate previews of user queries avoiding the latency of backend requests.

References

1. Allamanche, E.: Content-based identification of audio material using mpeg-7 low level description. In: ISMIR (2001)
2. Berenzweig, A., Logan, B., Ellis, D.P.W., Whitman, B.P.W.: A large-scale evaluation of acoustic and subjective music-similarity measures. Computer Music Journal 28(2), 63–76 (2004)
3. Bernhardsson, E.: Implementing a Scalable Music Recommender System. Master's thesis (2009)
4. Cano, P., Batlle, E., Kalker, T., Haitsma, J.: A review of algorithms for audio fingerprinting. VLSI Signal Processing 41(3), 271–284 (2005)
5. Celma, O.: Music Recommendation and Discovery in the Long Tail. Ph.D. thesis, Universitat Pompeu Fabra, Barcelona, Spain (2008), `http://mtg.upf.edu/~ocelma/PhD/doc/ocelma-thesis.pdf`
6. Fielding, R.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, Citeseer (2000)
7. Finetunes: musiclens - in tune with you (2010), `http://finetunes.musiclens.de/` (accessed June 9, 2010)
8. van Gulik, R., Vignoli, F.: Visual playlist generation on the artist map. In: Proceedings of the International Conference on Music Information Retrieval ISMIR, Citeseer (2005)
9. Haitsma, J., Kalker, T.: A highly robust audio fingerprinting system. In: ISMIR (2002)
10. Kastner, T., Allamanche, E., Herre, J., Hellmuth, O., Cremer, M., Grossmann, H.: Mpeg-7 scalable robust audio fingerprinting (May 2002)
11. Kim, Y., Schmidt, E., Emelle, L.: Moodswings: A collaborative game for music mood label collection. In: Proc. Intl. Symp. Music Information Retrieval (2008)
12. Lillie, A.S.: MusicBox: Navigating the space of your music. Master's thesis, School of Architecture and Planning, Massachusetts Institute of Technology (September 2008)
13. Audioscrobbler Ltd. Audioscrobbler. The social music technology playground (2010), `http://www.audioscrobbler.net/` (accessed June 9, 2010)
14. Pachet, F., Cazaly, D.: A Taxonomy of Musical Genres. In: Proceedings of the 1st Conference of Content-Based Multimedia Information Access (RIAO), Paris, France (April 2000)
15. Pampalk, E., Gasser, M.: An implementation of a simple playlist generator based on audio similarity measures and user feedback (2006)
16. Pampalk, E., Pohle, T., Widmer, G.: Dynamic playlist generation based on skipping behavior. In: Proc. of Int. Symposium on Music Information Retrieval (2005)
17. Pauws, S., van de Wijdeven, S.: User evaluation of a new interactive playlist generation concept. In: Proc. Sixth International Conference on Music Information Retrieval (ISMIR 2005), Citeseer, vol. 11, p. 15 (2005)
18. Peters, I., Weller, K.: Tag gardening for folksonomy enrichment and maintenance. Webology 5(3) (2008)
19. Raimond, Y.: A Distributed Music Information System. Ph.D. thesis, Queen Mary, University of London (November 2008)
20. Rayport, J., Sviokla, J.: Exploiting the Virtual Value Chain. The McKinsey Quarterly (1), 21–22 (1996)

21. Russell, J.: A circumplex model of affect. Journal of personality and social psychology 39(6), 1161–1178 (1980)
22. Thayer, R.: The biopsychology of mood and arousal. Oxford University Press, USA (1989)
23. Wagner, M., Liebig, T., Noppens, O., Balzer, S., Kellerer, W.: Towards Semantic-based Service Discovery on Tiny Mobile Devices
24. Westergren, T.: The music genome project (2010), `http://www.pandora.com/mgp.shtml` (accessed June 9, 2010)