ContextPlayer: Learning contextual music preferences for situational recommendations

Karla Okada, Börje F. Karlsson*, Laura Sardinha, Tomaz Noleto

Nokia Institute of Technology (INdT)
Manaus, AM, Brazil
{karla.gomes, laura.sardinha, tomaz.silva}@indt.org.br .

*Microsoft Research Asia
Beijing, China
borjekar@microsoft.com

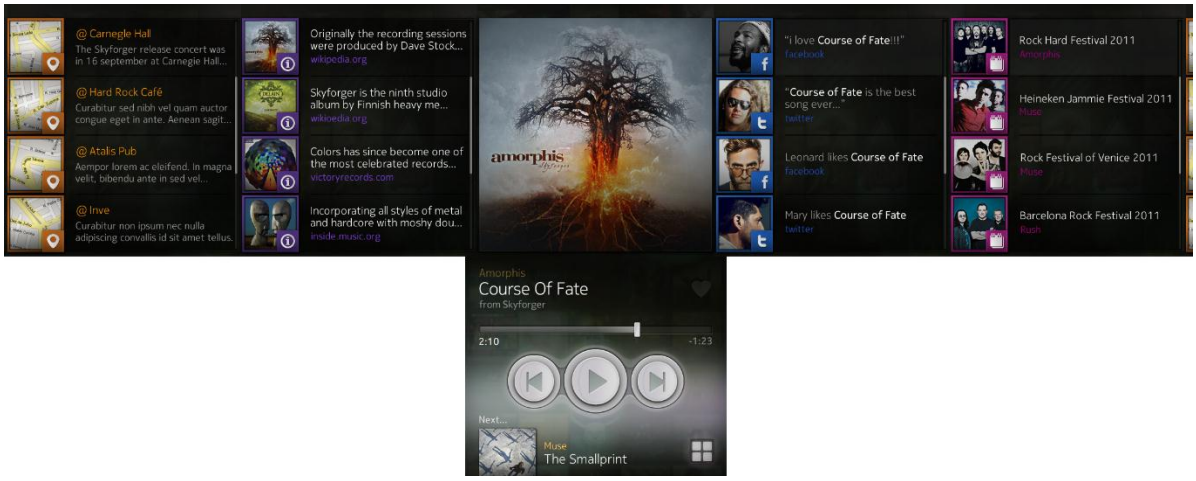

Figure 1. *Current playing song* screen and *recommendations* carrousel.

Abstract

Music listening is a very personal and situational behaviour, which suggests that contextual information could be used to greatly enhance music recommendation experience. However, making such use of mobile context, while learning user profiles, is a challenging problem. This case study presents a system for collecting context and usage data from mobile devices, but targeted at recommending music via unsupervised learning of user profiles and relevant situations. The developed data flow system supports both short enough response times and longer asynchronous reasoning on the collected data; furthermore, the mobile phone acts not only as sensor, but the mobile app is directly tied to the effectiveness of the music service user experience (UX). This work describes our system design and discusses issues related to the problem space and to usability tests on such systems, based on an international user trial.

CR Categories: • Information systems~Recommender systems • Human-centered computing~Empirical studies in ubiquitous and mobile computing

Keywords: music, context-awareness, recommender systems.

1 Introduction

Even though music listening is a highly personal and situational activity, and recommender systems for music are hardly a new idea, effectively combining contextual data and user profiles in a music recommendation service is still an open problem.

Context-aware services (CAS) are services enriched with information from their execution environment, which are able to adapt to the current context to increase their usability and effectiveness [Baldauf et al. 2007]. Context-awareness and adaptation are especially key in mobile scenarios, in which devices could sense their changing environment and act intelligently based on it.

It has been shown that, depending on the domain, at least certain contextual information can be useful for providing better recommendations [Adomavicius and Tuzhilin 2005, Do et al 2011]. Moreover, music is one such domain where quality and usefulness of recommendations can be influenced by contextual data, since people tend to listen to diverse songs at different environments or when performing specific activities [Reynolds et al. 2007].

As people carry their mobile devices everywhere and mobile content consumption is becoming more and more prevalent, using the capabilities on these devices (sensors and data access) provides a promising opportunity to improve users' music listening experience.

However, while recently there has been much research on context-aware recommendation systems [Baltrunas et al. 2011, Woerndl 2009], most of the work available in the literature assumes (or proposes beforehand) models for context representations or for user preferences (mostly based on rating and ranking). The key difference in our approach is that it tries to use only implicit feedback and not to make any assumptions

* This work was performed while the author was R&D Lead at INdT.

about either users' preferences or what constitutes proper - a priori - feature selection.

In this paper, we i) present a system developed to explore the inclusion of contextual information into the music recommendation process, based on learning from collected usage data; and ii) describe the challenges in performing user experience (UX) evaluation of such systems.

The described system relies on user's behaviour and music metadata, i.e. besides usage data, the current system does not process low-level music features for the recommendations; song structure is not taken into account, nor is any kind of audio signal processing.

The remainder of this paper is organized as follows. The next two sections detail the initial design goals of our music recommender system and its usage of context. Section 4 describes the system architecture and the developed music discovery application (along with some reasoning on its features) with its context-matching engine. Then, in section 5, the evaluation of the system prototypes is described. Finally, the paper discusses some limitations of the system and a summary of the experience in developing and evaluating this kind of system.

2 ContextPlayer Goals and Design

The opportunity of combining mobile sensing and the personal nature of music listening activities led us to the question of how to explore ways in which data on user context can be leveraged to improve user experience in music listening. Nonetheless, as we are dealing with dynamic environments, which translate into increased demands on users' attention, such system should be "smart" and as non-intrusive as possible.

Understanding how people interact with their surroundings is key in this kind of dynamic system [Tarasewich 2003], but correctly modeling those interactions is a challenge in itself. Therefore, we have opted for a deployment-driven approach. This allowed us to gain insights on actual user needs and on how interesting different ideas would be in the real world.

Finally, a usable interface is critical to any user-facing application. Especially if one hopes to collect usage information for analysis that is somehow representative of real-world scenarios. As such, the app development followed a user-centered design process [Mayhew 1999] with the following steps: a) determine system goals; b) identify user needs; c) sketch out a high-level product design; d) prototype; and e) iterate design with evaluation results.

The design and development of ContextPlayer focused then on an early start with concept design/wireframes; not focusing on yet another music player, but on a new app that intelligently finds and suggests music content depending on the user behaviour.

Later, after an initial usability review of the design and prototype, a production-quality system was developed so that user trials could be performed and refinements made to its design.

During the first step in the process, the primary goals defined for the system were: unobtrusive, implicit feedback; good responsiveness; UI adapts to user and context; discovery of new content; and enhance music content during consumption.

To accomplish these goals the system should be responsive, both in terms of UI interaction and in getting new recommendations; and be able to properly function in both online and offline situations, while still able to make use of different data sources.

More details on the system implementation are described in section 4.

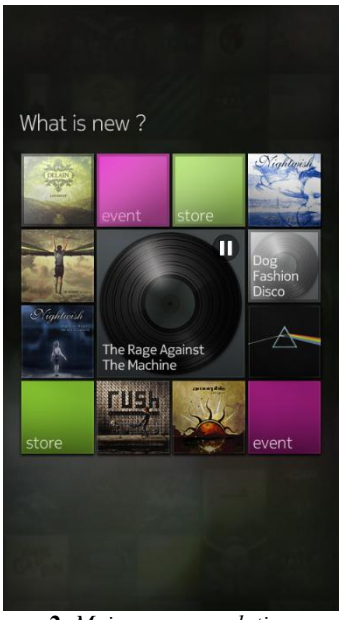

Figure 2. *Main recommendation screen.*

2.1 App Overview

The application UI allows the user to explore its local (to the phone) catalogue of songs and, from usage data, starts to suggest context-appropriate songs for playing or purchasing. Figures 1 and 2 show two of the most important screens according to this concept: a) the current playing screen (center); and b) the main recommendation screen, respectively.

The main recommendation screen (Figure 2) is the starting screen for all usage scenarios except the very first startup of the app and it is the app's main discovery hub. On this screen, the current playing song tile is surrounded by the most relevant song recommendations to play or purchase. By clicking on the currently playing tile, the user is taken to the currently playing song panel, which is surrounded by other side-scrollable panels that provide more music-related information (Figure 1).

Recommendations are derived from the user's music collection, usage of the application, and context of usage. Song discoverability on the app may also use social data, which allows it to suggest songs based on data from other people that listen to similar music or perform similar activities.

3 Context

The choice of representation framework for context has a significant impact on media applications that dynamically adapt to user needs, and, if flexible enough to address temporal evolution, can lead to powerful adaptation to user interaction [Mani and Sundaram 2007].

As such, a lot of effort has been put into creating flexible and abstract models of context. However, there is a tradeoff between abstraction and context-sensitivity [Lieberman and Selker 2000]. To avoid pitfalls in the complexity of the context model, we pragmatically define context for the purposes of the application described in this case study as "a finite set of sensed conditions collected from a mobile device that could affect a given user's music-listening behaviour".

Situations outside this scope are not considered and changes in user preferences through time are tracked and represented in a user model, not in the context itself.

Mani and Sundaram [2007] also point out that context attributes can only be decided on a per application basis and that context is related to knowledge and cannot be understood independent from it. These findings are in line with our initial intuition during brainstorming and helped inform our representation of contextual information.

Starting from simple information that has proved useful in context-aware ubiquitous computing [Dey 2001], ContextPlayer uses activity, environment, location, and time as key attributes.

Activity information is derived from readings from the device accelerometer, and its output classifications include: idle, walking, bicycling, running, etc. The environment attribute is the results of inferring the type of physical space where the user is at a given moment by getting audio input samples and comparing them to a trained database of classified samples. Possible outputs include: meeting, office, bus, among others. The location information consists of geographical coordinates (latitude, longitude), that are clustered into representative regions. Lastly, time is discretized into five time windows within a day and into a bucket for each of the seven days in a week.

These context attributes can be seen, according to the semantic context interpretation and abstraction layers presented in [Bettini et al. 2010], as belonging to the High-Level Context layer, where the lower level sensory information is semantically interpreted.

Besides the sensorial information, as context is related to knowledge, we also represent some domain knowledge as part of the event/context. For this, music metadata is attached to the context and extra events are generated for addition or removal of music files from the user device. Data collected from the web is also used to enhance the available music-related information.

This representation of context allows the system to detect contexts such as 'Every Wednesday in the morning the user goes jogging and listens to rock songs' or 'The user purchases songs while on the bus during early morning hours on weekdays'.

4 System Implementation

To realize the goals described in Section 2, the system combines device-side and server-side components. The basic dataflow starts with the mobile device acting as sensor, collecting context and usage information. The data is transferred to a backend platform (that also handles music-specific metadata and has access to external data sources) for processing and generation of recommendations. The resulting data structures are then returned to the device to be used by the music discovery app.

Even though most of the data heavy lifting is performed on the system backend, the mobile client needs to not only detect the current context, but to have enough data and flexibility to recommend songs by itself (having some level of autonomy for cases when the backend is either not available or did not yet provide new data). As this case study focuses on the mobile app, the contextual backend platform (CBP) is omitted. More details can be found elsewhere [Karlsson et al. 2012].

The device-side components of the system can be described as two separate processes (Figure 3): a Data Provider; and the Music Discovery Application itself.

4.1 Data Provider

The main responsibility of the Data Provider is to collect three sets of data for processing: i) music metadata, from each track on the user's music library on the device; ii) music listening habits of the user (which song was played, when, and for how long); and contextual data (i.e. data from sensors available on the device), that can later be reasoned over.

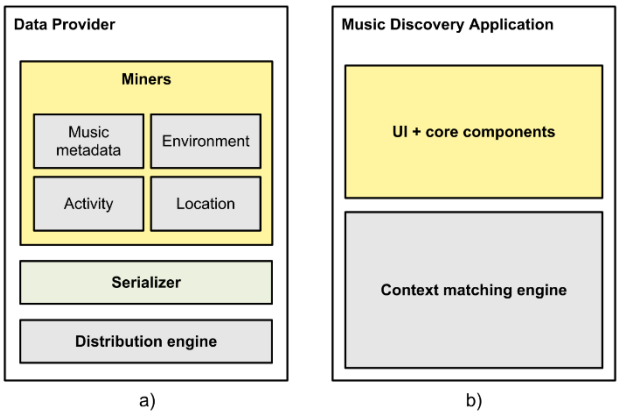

Figure 3. *Device-side modules.*

A set of specialized miners is responsible for gathering data from a specific context attribute sub-module (activity, environment, and location). A final music metadata miner observes both the user's local music library (keeping the system aware of eventual addition or removal of songs) and the device's underlying media framework (for music playing events to notify the system).

The music playing data is represented in the form of scrobbling events, annotated with the contextual information at the time of the event. A serializer layer is responsible for translating the mined data into intermediary objects, assembling the scrobbling events structure, and serializing the events.

The distribution engine handles the transmission of the data streams to the CBP.

4.2 Output from the CBP

Based on the data received from the distribution engine, augmented by external data sources - e.g. a music catalogue from an international e-commerce music store, an artist influence graph, historic song purchase data, etc. - the CBP generates a user music profile (UMP) and runs a series of recommenders. The output of these recommenders - playlist seeds, songs to buy, relevant music events - is then sent back to the device in an intermediary format to be used after context matching.

4.3 Music Discovery App

As the app behaviour depends on the user context, its most important module is the Context Matching Engine (CME). The CME uses data from the miners and the UMP to decide what the most relevant recommendations are.

4.3.1 Context Matching Engine

As the CBP receives scrobbling data, it gets more details about the user music preferences to eventually update the UMP and append better contextual recommendations. Whenever the application detects new UMP updates, it fetches this data and caches it locally to use it as input for feeding context-specific structures. Relevant contexts in the UMP are represented by arrays of context attributes.

In order to decide when and what songs to play, the CME takes three steps: i) query context from the latest sensor readings (miner cache); ii) search for the best possible match considering the relevant contexts in the UMP and the seeds associated to it; and iii) assembly a playlist containing songs based on these. If eventually the user context changes, the engine decides whether the current playlist is no longer suitable for the user and repopulates it with the new context's songs by following again the aforementioned steps.

At some point, it may happen that a playlist runs out before the active context changes. In this case, the user would probably not be satisfied if the player abruptly stops playback whilst there could be many other possibly enjoyable songs in the device. There are also the scenarios where the engine does not find an exact context match, i.e., none of the UMP entries matches the current context's attributes. For such cases, the CME resorts to a cascade of fallbacks, as shown in Figure 4. Such fallbacks are also useful in the cold start and offline scenarios, as described below. Context match and fallbacks are seamlessly integrated.

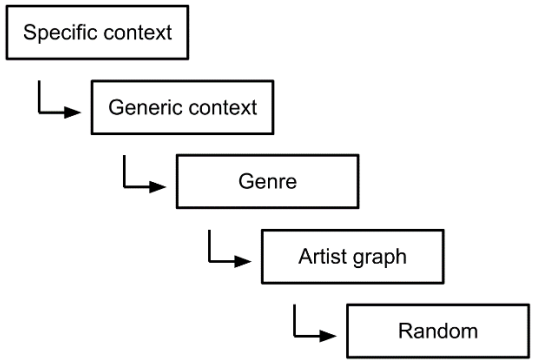

Figure 4. *Cascade of playlist fallbacks.*

4.3.2 Application UI

All app usage scenarios are designed around the concept of having the app *just work* and the system recommend songs related to the current user context, without demanding much effort from the user. When launching the app for the first time, the user is prompted to loads his/her collection of music files.

As these are recognized by the app, the user sees a new screen (the Music sea, shown in Figure 5) that represents his/hers music collection. Once the first of them is played, the main recommendation panel (MRP) starts to be populated with different kinds of recommendations – songs for purchase, songs for the current context, relevant nearby music events, etc; each type of recommendation having a different tile color.

As the user interacts with the applications, the user profile is refined, and finer-grained contexts can be identified. Thus, the relevance of the recommendations will improve dynamically. However, this process can take time, depending on how much the user interacts with the app or if connectivity is available.

4.3.2.1 Cold Start and Offline Scenarios

When running the client application for the very first time, nothing is yet known about the user's music preferences. Hence, some mechanism must exist to suggest songs in the MRP. If connectivity is available, the device will show some tiles recommending top selling songs in that area (based on the user's current location, or on a global ranking).

Independently of connectivity, the system will try to execute the context matching process and, if there is no match, it will go through the fallback steps shown in Figure 4. As the user still does not have a UMP at this point, there will be no relevant context or favorite music genres. Nonetheless, there needs to be a way of establishing some sort of song relevance metric among the available songs, giving the user a feeling he/she is not just listening to a meaningless song sequence. In such scenario, the system will start two processes: i) building a histogram based on the songs currently loaded on the device; and ii) loading an offline-created "artist graph" (deployed with the app).

This artist graph represents a set of artists and their influences. The graph was manually created by music specialists and includes 300 popular artists. The histogram of local songs is used as a proxy for the user music preferences and the graph provides a measure of local similarity among songs while the UMP is created. Together they are used to suggest songs that might be of interest to the user. If, by chance, the user local music library does not include any song by an artist on the graph, the system resorts to playing random songs. From the moment a song is selected to play and scrobbling data is generated, the UMP starts to be formed.

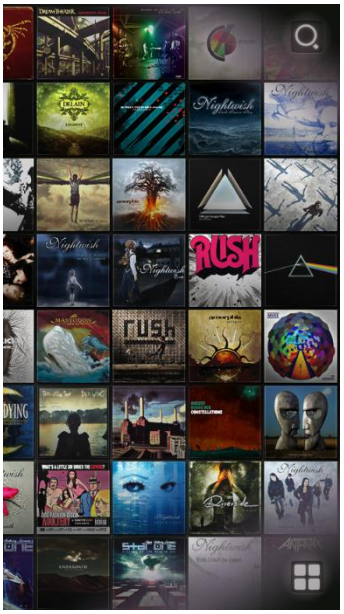

Figure 5. *Music sea.*

4.3.2.2 Purchase, Events, Social, and More

As previously mentioned, the goals of the system include the discovery of new content and augmentation of music consumption experience through the addition of related information and media. This is realized in part in the MRP for recommendation types. But, more completely, by the panel carrousel that is side-scrolable from the current playing song screen (Figure 1). Panels exist for all the different recommendation types; nearby events; songs other people listen in similar situations; video clips, news, and Wikipedia information on the current playing song or on related artists; and the history of user-listened songs.

5 Experiments and UX evaluation

Evaluation of the quality of a recommender system can be seen from three different points of view: functional testing, quality of recommender algorithm output, and usability evaluation.

Functional tests were already part of the development process. As high precision and recall do not always mean higher user satisfaction, there is growing consensus that recommender systems should focus less on the offline evaluation of algorithms, and focus more on user-centric approaches. Aspects such as the presentation and interaction have a significant impact on the user experience [Konstan and Riedl 2012], but

many other factors affect such analysis [Knijnenburg et al. 2012].

To make matters yet more complicated, the developed system presented in this case study also uses unsupervised learning, thus individual user experience also varies with usage of the system. As such, we have opted to focus on a qualitative evaluation based on UX trials. Trying to more easily single out specific points in the application usage, some features initially developed were disabled to reduce the feature space to be analyzed.

After design reviews, a two-step trial was executed to evaluate both the concept and the actual contextual recommendations. In both trials there were no restrictions concerning age or education level when recruiting users. The only desired attribute was for them to be active music listeners.

At first, a small test was performed with 10 users - 5 male, 5 female, from age 18 to 32 - focusing on the comprehension of the recommender concept. For this trial a set of initial tasks had to be completed by the participants; which later were interviewed to gather their preferences, needs and opinions on possible improvements. This first experiment generated 3,704 scrobbling events and allowed us to track how users' initial to final perceptions improved.

Some key points from the first trial were:

- All users seemed very interested in the recommendation of events;
- Six users were initially confused by the lack of direct control over which songs were played. But, towards the end of the experiment, most users were satisfied with the system recommendations (7 out of 10);
- Four of them mentioned that the recommendations exceeded their expectations. Three users declared to be always curious to see what the system would suggest;
- While nine of the participants were not used to buy songs online, all of them considered the experience of discovering a new song interesting and attractive and commented that they felt encouraged, by the recommendations, to buy new songs.

Based on this feedback, refinements were made to the UI and some minor changes were applied to the system data flow. A more extensive trial was initiated with 60 users in four different countries (Brazil, Finland, UK, and the USA), in which 59 users generated additional 22,467 scrobbling events.

Some of the most challenging problems facing mobile interfaces include the constantly changing context of usage and the limited user attention given to the device and application [Heo et al. 2009]. Given that, the intent of this larger trial was to have users trying the complete system (including improvements related to feedback from the first trial) without specific tasks to be performed. Vouchers valued at GBP 80.00 were given to users for song purchases inside the app, as an incentive to explore and try out the system.

Evaluating a contextual recommender system is a non-trivial task as many factors are at play at once, and most of the UX models for recommenders do not include context properly [Knijnenburg et al. 2012]. We thus follow Kuniavsky's [2003] definition of good UX for a system which states that, although it varies from person to person and task to task, it is possible to have a good approximation by making the system "functional, efficient, and desirable to its intend audience".

In an attempt to represent these effects we map these three key areas into four related components (perceived quality, appeal, system-related experience, and outcome-related experience) on a recent framework for the user-centric evaluation of recommender systems [Knijnenburg et al. 2012] and use these as guiding categories in a questionnaire to apply to the trial users.

Some interesting findings were:

- The majority of users enjoyed the application UI, describing it as "cool", "wow-factor", etc.
- Pieces of the UI were not automatically clear. Many users had difficulty understanding the situations concept and how it affected the actual recommendations;
- 21 users would prefer to have regular music player features available to choose specific songs to play;
- Almost all users mentioned that the music sea concept was interesting, but that having a search function would be extremely interesting;
- Due to the limited amount of content related to events, 30% of the users complained about it degrading the app experience. Even if other recommendations were fine.
- 63% of the tested users preferred the system to have faster recommendations than a system which requires user input to reveal their preferences;
- For some users, the population of the UMP took a few days to start improving their recommendations, which caused a bad impression on the system experience.

Overall, the user feedback was supportive of the app idea for recommending songs. Nonetheless, almost 50% of the users had suggestions to improve the notifications of context changes or the reason for receiving specific recommendations on the UI, which suggest that even if contextual recommendations are useful, users are interested in an explanation for them. Following the experience on the first trial, most users also highlighted the discovery and purchase of new songs as very positive.

On the negative side, the exploration of the user library of songs and the lack of ability to play specific songs were mentioned by all users as a major problem. How to integrate both regular player features and recommender features (or isolate the usability analysis of either) remains an open issue.

Fifteen of the users also did not use any of their bonus for purchasing songs. Unfortunately, no concrete reason can be given for this behaviour as the questionnaire did not cover it.

Interestingly - perhaps due to differences in noise level between situations Brazil and the ones used to train the classifier - the environment classifier would misclassify the context environment for Brazilian users and this attribute ended up not being relevant for their contexts.

6 Problems and Limitations

While the user-centered design process applied to the described case study helped us to iterate on the system design and explore the problem space, developing, and evaluating contextual recommender systems is still a challenge endeavor. Both from the conceptual and practical points of view.

The distinction between perception and evaluation is subtle but important; the former denotes whether certain system aspects register with the user, while the later denotes whether the perceived aspect has any relevance to the user [Knijnenburg et al. 2012]. Both aspects should be tracked separately, but due to the trials' design, it was not possible to collect data at this level.

The trials suggest that the simple context matching utilized in this case study provided adequate representation for relevant contexts. However, we identified extensions points that could be explored - without the model facing over-generalization

problems - to improve its efficacy. Context attributes such as user mood or the current weather/season can affect music-listening behaviour and proxies to track them could be implemented.

The domain knowledge used along with the context attributes could also benefit from some form of tagged data (such as song mood, for example). Tags could be crowdsourced through the app itself or another music data source might be tapped for it.

As users need to continuous interact with users for the model to learn their preferences, the bootstrapping could be improved by smartly exploring recommendations. One possible way for this would be to use genre information and relationships. This seems worth of investigation as many users mentioned genre-based scenarios during the app trials.

Lastly, as multiple context attributes come from classifiers attaching some semantics to lower-level sensor readings, it might be necessary to use finer-grained training sets. The 'environment' case from the second trial illustrates this point.

On the practical side, multiple issues ended up affecting the system evaluation. These issues can be categorized as: i) expectation mismatch; ii) infra-structure / environmental; and iii) unanticipated usage-related problems.

One example of expectation mismatch was the fact that many users kept requesting player features when the purpose of the app was not that. A strategy to make the goal of the system clear (or to cover the most impactful missing scenarios) is key.

Many of the Brazilian users faced connectivity issues (constant network switching from 3G to 2G), which caused problems in the timely population of the UMP, reception of new recommendations and on the purchase flow of new songs. Even though we planned for the offline usage scenario, these issues were more prevalent than expected. Any assumptions on environmental conditions or infra-structure availability need to be documented and tracked to avoid compromising the trials.

Moreover, when evaluating the app in a larger scale, even though it was functionally correct, unexpected issues still affected the experience of many users. One example of this turned out to be the reliance on album covers as a major item on the visual-heavy app UI. While for most songs, the album cover data could be read from their files' metadata, this became an issue in two cases.

Some users had many old music files; from before placing the album cover in the file metadata was a common practice. This led to the system being unable to properly show huge amounts of songs, effectively making it impossible to use. Some other users had too many songs from compilation albums (songs by various artist under one collection), resulting in app screens filled by the same album cover. As they actually represented different artists, the UI became very confusing.

These issues influenced the study negatively and mitigation plans should exist for any future experiments. More mini trials might also have shed light on the two mentioned UI issues, before the larger scale trial took place.

7 Concluding Remarks

This work presents a case study on a context-aware system for situational music recommendation whose goals were to explore the problem space and allow a good degree of quality to the user experience. The system employs a number of techniques to deal with the necessary data flow and to generate quality recommendations. The implementation of ContextPlayer provided many insights into the challenges of such systems.

Results suggest that the system can properly identify and recommend songs per context. Even though our experience raised many interesting issues on how to appropriately design and evaluate such systems, context-specific music recommenders are complex and much remains to be explored in this area, especially when involving social data.

We have also identified the need to associate songs semantically; hence, we plan to focus more effort on building a knowledge model to be used for contextual recommendation in the music domain. Additionally, we intend to extend the CME to perform in-device analytics for responsiveness and power optimizations.

References

ADOMAVICIUS, G., & TUZHILIN, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. In *IEEE Transactions on Knowledge and Data Engineering*, *17, 6*, 734-749.

BALDAUF, M., DUSTDAR, S., ROSENBERG, F. 2007. A survey on context-aware systems. In *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, 4, 263-277.

BALTRUNAS, L., KAMINSKAS, M., LUDWIG, B., MOLING, O., RICCI, F., AYDIN, A., LUKE, K.H., SCHWAIGER, R. 2011. InCarMusic: Context-Aware Music Recommendations in a Car. In *12th International Conference on Electronic Commerce and Web Technologies*.

BETTINI, C., BRDICZKA, O., HENRICKSEN, K., INDULSKA, J., NICKLAS, D., RANGANATHAN, A., RIBONI, D. 2010. In *A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing* 6, 2, 161-180.

DEY, A. K. 2001. Understanding and using context. In *Personal and Ubiquitous Computing Journal*, 5(1), 4-7.

DO, T. M. T., BLOM, J., GATICA-PEREZ, D. 2011. Smartphone usage in the wild: a large-scale analysis of applications and context. In Proceedings of the *13th International Conference on Multimodal Interfaces.*

HEO, J., HAM, D. H., PARK, S., SONG, C., YOON, W. C. 2009. A framework for evaluating the usability of mobile phones based on multi-level, hierarchical model of usability factors. In *Interacting with Computers*, 21, 4, 263–275.

KARLSSON, B. F., OKADA, K., NOLETO, T. 2012. A Mobile-Based System for Context-Aware Music Recommendations. In Artificial *Intelligence Applications and Innovations – Adv. in Information and Communication Technology*, 382, 520-529.

KNIJNENBURG, B.P., WILLEMSEN, M.C., GANTNER, Z., SONCU, H., NEWELL, C. 2012. Explaining the user experience of recommender systems. In *User Modeling and User-Adapted Interaction 22*, 4, 441-504.

KONSTAN, J.A., RIEDL, J. 2012. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, *22*(1-2), 101-123.

KUNIAVSKY, M. 2003. Observing the user experience. San Francisco: Morgan Kaufmann Publishers.

LIEBERMAN, H., SELKER, T. 2000. Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal*. 39, 3-4, 617-631.

MANI, A., SUNDARAM, H. 2007. Modeling user context with applications to media retrieval. In *Multimedia Systems*, 12, 339-353.

MAYHEW, D. J. 1999. The usability engineering lifecycle. In *CHI'99 Extended Abstracts on Human Factors in Computing Systems*. 147-148. ACM.

REYNOLDS, G., BARRY, D., COYLE, E. 2007. Towards a Personal Automatic Music Playlist Generation Algorithm: The Need for Contextual Information. In *Proceedings of the 2nd Audio Mostly Conference: Interaction with Sound*, 84–89.

TARASEWICH, P. 2003. Designing mobile commerce applications. *Communications of the ACM - Mobile computing opportunities and challenges*, vol. 46, 12

WOERNDL, W., BROCCO, M., EIGNER, R. 2009. Context-Aware Recommender Systems in Mobile Scenarios. *Intl. Journal of Information Technology and Web Engineering*, *4, 1*, 67-85.