Road-based Adaptation of In-Car-Infotainment Systems

Sandro Rodriguez Garzon
Daimler Center for Automotive IT Innovations
Berlin, Germany
sandro.rodriguez.garzon@dcaiti.com

Kristof Schütt
Daimler Center for Automotive IT Innovations
Berlin, Germany
kschuett@cs.tu-berlin.de

***Abstract*—This paper introduces a prototype of a highly adaptive in-car-infotainment system. The prototype processes historical user interactions in order to discover regular sequences of interaction events within similar environments. The discovered environments in form of road segments and the activities will be used to adapt the human-machine interface (HMI) in case the car approaches an environment that is likely to contain a known activity. Temporal event patterns as well as grouping criteria need to be prespecified to control the detection of characteristical activities and to define the way two road segments are declared to be similar. Furthermore, a brief technical introduction of the automotive specific process of location information preprocessing is given, which is used by the prototype to interpret its environment or to group road segments. The use cases of a button that changes its size depending on the probability of being pressed at a certain road as well as a HMI that automatically switches to a certain radio station based on historical user initiated radio station changes at similar road segments will be discussed in detail.**

Keywords-context awareness, personalization, adaptation, intelligent user interface

I. INTRODUCTION

Over the last years more and more mobile devices such as mobile phones or in-car-infotainment systems became location-aware. In particular, the high distribution of smart phones with GPS sensor spurred the use of location-based services. Several applications like searching for point-of-interests within a certain range or notifications about friends in close-by environments became popular and underlined the user demand for applications that provide location-based content.

So far, the dynamic content is generated in case the user approaches a certain environment and either prompts the system to show the content or gets notified if some predefined objects occur in the current environment. In both cases the user needs to manually define the kind of information that might be helpful in the current situation. In the former variant the user has to select a topic while in the latter variant the user has to prespecify a notion of interestingness for certain information. Both approaches use static user interests and location information for processing. But how to deal with users that have different interests depending on the location?

In order to present content based on interests that are influenced by the location the user would need to provide all individual associations between interests and locations beforehand. The expense for the user remains arguable as long as all interests are associable with user defined abstract locations like home or office. But it is getting difficult and impractical if the interests are connected to concrete locations characterized by latitude and longitude values. An interim solution would be to discover abstract locations of each user by investigating his moving behavior as showed in [1]. The abstract locations might be proposed to the user to facilitate the input of the mentioned interest and location relations.

A more pleasant approach is to discover abstract locations and the corresponding interests based on historical data of service use without the need to prompt the user to enter the interest-location relations. Such an approach would enhance several applications like restaurant search or GUI extensions like the generation of favorite button lists by considering the users interests at a certain location. An application for the restaurant search as a smart content filtering service can be realized by incorporating additional location parameter to the recommender system [2]. Complex personalization use cases like individualized favorite buttons that execute certain sequences of preferred user interactions to reduce click costs need a more sophisticated notion of interest.

This paper introduces an in-car-infotainment system prototype capable of personalizing itself based on regular user tasks at certain locations. The introduction is followed by a short survey of papers related to the fields of context-aware computing and location-based personalization. In the Section ”Examples” two personalization examples are introduced that are used throughout this paper. A detailed description of all high-level components and their interactions with each other is given in Section ”Architecture”. The Section ”Situation Recognizer” contains a step-by-step explanation of the event sequence detection and grouping process. A rather technical description of location preprocessing is given in Section ”Map Matching”. This paper ends with a summarizing conclusion and a discussion of refinements and future extensions.

II. RELATED WORK

Over the last years a lot of research was done concerning the detection of regular tasks and its visualization. In [3]

 ISBN: 978-1-61208-117-5

Shen uses graph mining algorithms to extract regular work flows of desktop use as frequent subgraphs of a graph consisting of resources and the information flows in between. In [4] Brdiczka follows a different approach by considering the time in between user interactions in addition to the type of action as described by Shen. Neither of them deals with an observation of the environment at the moment the task was executed. An interesting approach of adapting a system based on its use at certain locations is described in [5]. Coutand uses case-based reasoning to detect relations between system utilization and location to decide about future mobile phone properties in certain situations. In this case there is no need to extract complex sequences of interactions to build sophisticated task representations because a mobile phone property modification is solely related to the task of approaching a location. Thus, a task is interpretable as the action of entering a certain context.

Several context-aware frameworks [6], [7] were introduced in order to extract abstract representions of the context based on the aggregation of raw sensor data. Especially, the discovery of significant locations out of multiple sequences of GPS data to predict future movements of users or to enhance mobile phones became a popular field of research [8], [1], [9]. A promising approach of Liao in [10] deals with the task of mainly relating significant locations to regular user interactions. Liao discovers significant locations and infers corresponding activities using Relational Markov Networks. The process of detecting abstract tasks like dining is supported by location dependent information like a close-by restaurant obtained from geographical databases. In contrast, our approach does not need to infer high-level abstractions of tasks because personalization relevant activities are extracted by means of prespecified temporal event patterns. Inspired by [11], we are solely interested in actions originating from the use of an in-car-infotainment system. In [11], Rogers presents the application of user-centric personalizations within the automotive environment that is based on the analysis of the user actions ”selecting preferred routes” and ”spoken feedback”. In case of our examples the user actions are ”pressing a certain button” and ”changing the radio station”.

III. Examples

In the following sections, a simple and a rather complex personalization use case within the automotive environment will be discussed in detail. The simple use case deals with a button list of the HMI that changes its visualization based on its historical use. More precisely, the size of the buttons will depend on the probability of being pressed in a certain situation. Considering a situation to be defined as the road the car is currently driving on, the buttons should dynamically change their size according to their use at a certain road. This use case is declared to be simple because the actions that should be investigated namely the button click as well as the personalization triggering situation are single points in time.

```
<route>
  <waypoint>
    <position>
      <lat>48.422017</lat>
      <lon>9.536767</lon>
    </position>
    <time>1275296924299</time>
    <direction>189.8</direction>
    <height>808.7</height>
    <speed>15.3</speed>
  </waypoint>
  ...
</route>
```

Listing 1. Section of a context log

The more complex use case deals with a HMI that analyzes user initiated radio station changes in order to propose automatic radio stations changes depending on discovered regularities. A valid regularity would be discovered if the HMI user switches to a certain radio station every time he drives through a certain village. The proposal made by the HMI should first be visualized to the HMI user and later be executed if the user agreed to the automatic execution. Therefore, the personalization is executed indirectly in contrast to the simple use case where the resizing of a button is applied immediately. The complexity of the use case is caused by the action that consists of several temporal order events, which form a significant radio station change. In this case a radio station change is declared as significant if no other radio station change followed within 5 minutes.

IV. Architecture

The in-car-infotainment prototype consists of three main components: *Context Simulator*, *HMI* and *Situation Recognition*. Additionally, context log files based on XML are used in the Context Simulator to simulate environmental aspects like the position or the direction of the car. These log files are produced by recording location relevant information either at a real world in-car-infotainment system or by a mobile phone equipped with a GPS and compass sensor. Listing 1 shows a section of an example context log file.

A rapid prototyping approach is taken to implement an exemplary human machine interface based on Action Script. Within the client-server nomenclature the HMI is called the client and the Context Simulator can be seen as the server. Such an assignment became necessary because Action Script does not allow to implement listening sockets. During initialization the HMI tries to connect to the Context Simulator and waits for incoming context data. The HMI comprises many components like a navigation system and a radio in order to provide a wide range of possibilities for personalization use cases. Figure 1 shows a screenshot of

 ISBN: 978-1-61208-117-5

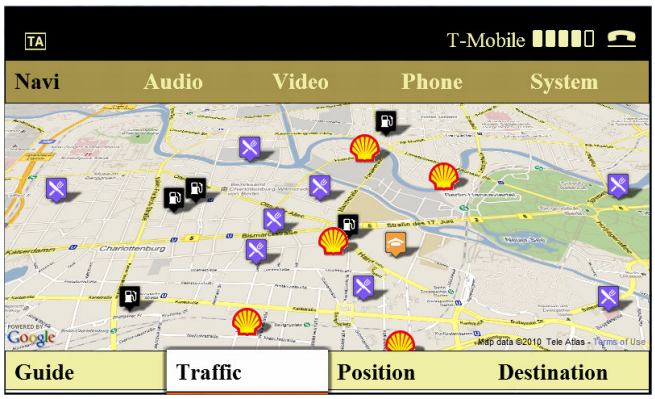

Figure 1. Screenshot of the HMI

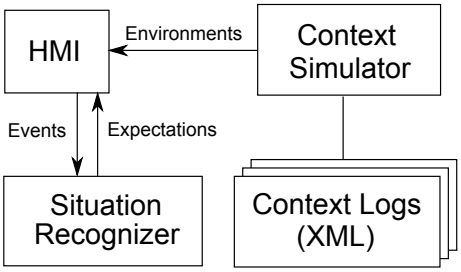

Figure 2. Overview architecture

the HMI in the navigation system modus.

In order to personalize the HMI based on its historical use we need to log all user interactions and the corresponding context. This is realized by sending all interaction data and indirect user interactions like location change to a separate component called Situation Recognizer. Since the Situation Recognizer does not differentiate between a direct or indirect interaction both events are summarized under the term *interaction event*. As stated above, the HMI acts as a client and connects to the Situation Recognizer at startup time. An overview about the whole architecture is given in Figure 2.

Each interaction event comprises of a header and a body section. The header section contains event attributes that are common to all events regardless of their type while the body section contains type dependent attributes. Listing 2 shows an example event that needs to be fired by the HMI to signalize a radio station change.

The `nr` attribute contains a successive number within a certain session while the `level` attribute denotes the abstraction layer of an event. So far, we identified 3 levels: *Raw events*, *logical events* and *instance events*. Raw events can represent a touch or a press of a button of the central command unit(CCU) while logical events can signalize a recognized gesture or a click (e.g. constructed of the raw events touch down and touch up). The raw events will be thrown globally at any time, whereas the instance events will only be thrown if a unique instance of an HMI element is affected. An example can be a click onto a certain button of the global application line(e.g. Navi). In this case the corresponding HMI element id is stored in the event attribute `sourceId`.

The body contains event attributes that belong to a certain type of instance event. Consider an interaction event that originates from a certain button of an application line signalizing a button click. In this case the button label might be included in the body to permit the Situation Recognizer to differentiate between clicks of different buttons that are part of the same button list.

The Situation Recognizer processes all interaction events and notifies the HMI about approaching a situation that will likely be the environment for a known interaction event sequence. In other words, the Situation Recognizer discovers regular interaction sequences and informs the HMI about future interaction event values and its environments. Therefore, the notification must contain all relevant situation information as well as future values of certain environmental aspects and its probabilities. These information will be used by the HMI to display a human-readable form of a discovered frequent interaction sequence as well as to execute personalizations. The interpretation of the notification message comprising use case dependent data is supported by a description of the message format specified beforehand.

Considering the complex example introduced in Section III the notification must contain the situation information comprising of the road name and the future radio station. Listing 3 shows the corresponding notification. The message will be interpreted by the HMI in the following way: The current road is called ”Main Street” and a user initiated switch to the radio station with name ”KissFM” is likely to appear now. The notion of the word ”likely” is discussed in Section ”Situation Recognizer”. The notification of the simple use case must contain the label of the button and the corresponding probability of execution.

V. Situation Recognizer

As stated above, the Situation Recognizer deals with the task of processing user interactions to discover frequent interaction sequences. These frequent user interaction sequences will be used to inform the HMI about the future steps of the user. To analyze sequences of events and its environments it is necessary to split the search process into 3 successive steps: *Action Discovery*, *Action Grouping* and *Situation Discovery*. Thereby, each step is supported by a use case description to keep control of the whole process by e.g., reducing the search space or by defining a fading function for an irrelevance factor for each user action. The use case descriptions must be defined by an expert together with the HMI developer.

 ISBN: 978-1-61208-117-5

```
<event nr="2156" level="I" type="StationChanged" sourceId="Radio" timestamp="1277112493971">
  <StationChanged>
    <frequency>http://www.kissfm.de</frequency>
    <stationName>KissFM</stationName>
  </StationChanged>
</event>
```

Listing 2. Example of an interaction event

```
<proposition uc="RadioChange">
  <action>
    <parameter>
      <stationFrequency>
        http://www.kissfm.de
      </stationFrequency>
      <stationName>
        KissFM
      </stationName>
    </paramenter>
    <environment>
        <street>Main Street</street>
    </environement>
  </action>
</proposition>
```

Listing 3. Example of a situation notification

A. Action Discovery

The first step namely Action Discovery uses a temporal event pattern of the use case description to extract a certain event sequence out of the stream of interaction events. For this purpose the complex event processing engine Esper [12] is selected, which analyzes event streams based on pattern descriptions written in Esper's event processing language(EPL). The resulting concrete sequence of events is called *action*.

In case of our complex use case, the temporal event pattern to extract significant radio station changes looks as follows:

$$StationChange \rightarrow Timer(5min, not(StationChange))$$

The temporal event pattern of the simple use case is omitted because it comprises only of the button click event. The resulting actions contain either a concrete radio station changed event or a concrete button click together with the corresponding context in form of a road segment id.

B. Action Grouping

The Action Grouping subprocess collects all actions and groups them by prespecified criteria. Each group should contain event sequences that occurred within similar environments. The notion of similarity is defined within the use case description. Therefore, each use case may contain different criteria to compare event sequences. A group is called significant if the amount of actions exceeds a preconfigured limit. Only significant groups are considered in the next step.

In the complex example use case, the actions should be grouped by similar radio stations and nearby road segments. Thus, each resulting group is characterizable by a single radio station and a set of road segment ids. The resulting group properties of the simple use case contain a single button label and a single road segment id because a button click is only related to a single road segment and not to a subgraph of the whole road network.

C. Situation Discovery

Finally, the Situation Recognizer uses the group properties to parametrize a prespecified temporal event pattern in order to extract similar situations. Thereby, the group properties are interpreted as a description of the environment the actions occurred in. If a similar situation is found the HMI will be informed about the group properties containing the environmental aspects as well as future values of use case relevant attributes.

The temporal event pattern that triggers a notification of the complex example use case may look like this:

$$StationChange\ and$$

$$StationChange.station\ != \ Group.station$$

$$\rightarrow$$

$$Context.location == Group.location$$

In words, if the last radio station change switched to a radio station that is different to a radio station property value of a discovered group and if the location of the current context is similar to the location of the same group then notify the HMI about the detected situation. The temporal event pattern of the simple use case contains only the location comparison. An overview about all steps discussed in this section is given in Figure 3.

So far, a situation notification is only triggered in case a significant group was found. This implies the instantiation and parametrization of a temporal event pattern for the detection of the corresponding situation in case a certain amount of similar actions occurred within the same environment. Considering the radio example such a notion of the statement "likely to happen" could be adequate but insufficient for the dynamic buttons example. For the execution of resize commands it is necessary to know the probability of a button click within a certain environment cause the different

 ISBN: 978-1-61208-117-5

buttons are influencing one another concerning their sizes. To calculate the probability of an occurrence it is necessary to relate the amount of occurrences to the amount of non-occurrences. But how does the prototype discover non-occurrences? It is impractical or almost impossible to detect and record all non-occurrences of actions that are unknown so far. Therefore, the prototype does only observe a non-occurrence of an action if the action happened at least once. Additionally, the detection of non-occurrences will only be constrained to an environment defined by the temporal event pattern of the corresponding situation. Thus, the detection of non-occurrences starts with the detection of a situation and ends if the environment changes concerning the use case relevant context attributes.

D. Implementation

The Situation Recognizer component is implemented using Java with Esper as its event processing unit for java applications. At startup time the use case descriptions are parsed and all action discovering patterns are generated as EPL statements. Additionally, the global context that gets appended to each incoming event gets initialized. This guaranties that all use case relevant environmental aspects are related to single events. The global context definition is part of each use case description. A preprocessing step for the context in form of a road detection algorithm is outsourced into a separate library described in the next Section.

VI. Map Matching

In order to group actions by their occurrence on certain roads it is necessary to first define the type of information we like to group. In mobile applications the location information often comprises of a latitude and a longitude value. Applying well-known density-based clustering algorithms like DBSCAN [13] or MRStream [14] would result in arbitrary clusters like the sum of all locations or groups of rectangular regions. In case of an automotive environment the clusters do not consist of arbitrary two dimensional regions because the car movements are always constrained by roads. Therefore, a cluster in the sense of a car is either a road segment or subgraph of the whole road network. However, GPS sensors provide latitude and longitude values that need to be transformed into a unique road id identifying a certain road segment. With the help of the road segment id it would be possible to define a notion of similarity between road segments by considering their road network distance.

The transformation of the latitude and longitude value into a position that lies on the road network is called map matching. The presented prototype includes an implementation of a map matching algorithm that is used to preprocess the location data before starting the action grouping process. It is essentially the first part of an algorithm proposed by Hummel [15]. For our purposes this rather simple implementation is sufficient.

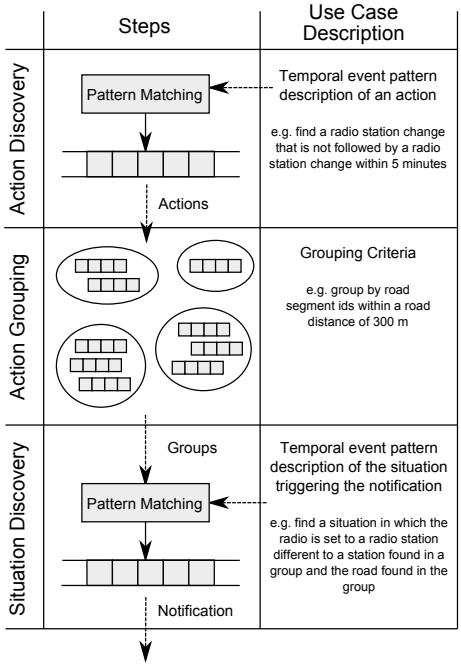

Figure 3. Overview Situation Recognizer

The topological data, needed to perform map matching, is based on the open source data provided by the OpenStreetMap project [16]. Before this data can be used, it has to be converted and inserted into a PostGIS database. The conversion was done with the free version tool called OSM2PO [17].

As mentioned above, the similarity between road segments needs to be calculated to decide if two road segments belong to the same group or not. Two road segments are declared to be similar if the road distance in between does not exceed a certain amount. The calculation based on the topological data is realized by pgRouting presented in [18].

It is also necessary to present the results in a human-readable way to be comprehensible for the HMI user. Therefore, a the street name needs to be extracted from the resulting road segment id. This service is also provided by the Map Matching library.

VII. Conclusion

We have presented a highly personalizable in-car-infotainment prototype that is capable to adapt itself based on the detection of regularities of its use in recurring environments. Architectural aspects as well as implementation details were discussed in order to describe primarily the technical problems.

The examples throughout this paper were mainly focused on use cases that try to adapt the HMI based on regular user

 ISBN: 978-1-61208-117-5

interactions at certain roads. However, it is up to the expert to decide, which dependencies should be used for a use case. For this purpose, we introduced the use case descriptions that are specified by an expert and used by the Situation Recognizer to control the whole process. While specifying the use cases the expert may decide to group button clicks only by time intervals and radio changes by time intervals and road segments.

VIII. Future Work

In the near future we plan to implement a feedback mechanism of the HMI in order to skip notifications that are not processed by the HMI. At the current stage, the Situation Recognizer works independently of any real execution of a personalization use case at the HMI. Therefore, it might happen that certain HMI personalization proposals were deactivated by the HMI user without notifying the Situation Recognizer. In this case, it is necessary to inform the grouping process to delete the corresponding groups or/and to prevent the grouping process to rediscover the same groups.

The approach of specifying and configuring personalization use cases in a generic way without the need to reprogram the Situation Recognizer offers a wide range of possibilities for rapid prototyping of personalizable applications. We plan to implement and test several new HMI adaptations that are based on the analysis of sequences of user interactions containing different road segments as well as time intervals. Especially, the prediction of the destination based on regular environments would enhance the usability of future navigation systems.

References

[1] D. Ashbrook and T. Starner, "Learning Significant Locations and Predicting User Movement with GPS," in *Int. Symposium on Wearable Computers*, 2002, pp. 101–108.

[2] G. Adomavicius and A. Tuzhilin, "Context-Aware Recommender Systems," in *2nd Workshop on Context-Aware Recommender Systems*, 2010.

[3] J. Shen, E. Fitzhenry, and T. G. Dietterich, "Discovering frequent work procedures from resource connections," in *Proceedings of the 13th Int. Conf. on Intelligent user interfaces*, 2009, pp. 277–285.

[4] O. Brdiczka, N. M. Su, and J. B. Begole, "Temporal Task Footprinting: Identifying Routing Tasks by Their Temporal Patterns," in *14th Int. Conf. on Intelligent User Interfaces*, 2010, pp. 281–284.

[5] O. Coutand, S. Haseloff, S. L. Lau, and K. David, "A Case-based Reasoning Approach for Personalizing Location-aware Services," in *1st Workshop on Case-based Reasoning and Context Awareness*, 2006.

[6] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. Journal of Ad Hoc and Ubiquitous Comput.*, vol. 2, no. 4, pp. 263–277, 2007.

[7] A. K. Dey and G. D. Abowd, "The context toolkit: Aiding the development of context-aware applications," in *Proc. of the Conf. on Human Factors in Computing Systems*, 1999, pp. 434–441.

[8] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello, "Extracting Places from Traces of Locations," in *Proc. of the 2nd ACM Int. Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, 2004.

[9] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen, "Discovering personally meaningful places: An interactive clustering approach," *ACM Trans. Inf. Syst.*, vol. 25, p. 12, 2007.

[10] L. Liao, D. Fox, and H. Kautz, "Location-Based Activity Recognition using Relational Markov Networks," in *Proc. of the Int. Joint Conf. on Artificial Intelligence*, 2005.

[11] S. Rogers, C. nicolas Fiechter, and C. Thompson, "Adaptive User Interfaces for Automotive Environments," in *Procs. IEEE Intelligent Vehicles Symposium 2000*, 2000, pp. 662–667.

[12] EsperTech Inc., "Complex Event Processing," http://esper.codehaus.org/, Last access: 2010-12-20.

[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of 2nd Int. Conf. on Knowledge Discovery in Databases and Data Mining*. AAAI Press, 1996, pp. 226–231.

[14] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, "Density-based clustering of data streams at multiple resolutions," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 3, pp. 1–28, 2009.

[15] B. Hummel, "Map Matching for Vehicle Guidance(draft)," 2006, http://www.mrt.uni-karlsruhe.de/z/publ/download/hummel2006b.pdf, Last access: 2010-12-20.

[16] OSM Project, "Open Street Map," http://www.openstreetmap.org/, Last access: 2010-12-20.

[17] C. Moeller, "OSM2PO," http://www.osm2po.de, Last access: 2010-12-20.

[18] pgRouting Project, "pgRouting," http://www.pgrouting.org/, Last access: 2010-12-20.

 ISBN: 978-1-61208-117-5