Author(s): Goix, Laurent-Walter; Valla, Massimo; Cerami, Laura; Falcarin, Paolo.
Article title: Situation Inference for Mobile Users: a Rule Based Approach
Year of publication: 2007
Citation: Goix, L.W. et al. (2007) 'Situation Inference for Mobile Users: A Rule Based Approach,' *International Conference on Mobile Data Management*, Mannheim, Germany May 01. IEEE pp.299-303
Link to published version: http://dx.doi.org/10.1109/MDM.2007.63
DOI: 10.1109/MDM.2007.63

Situation Inference for Mobile Users: a Rule Based Approach

Laurent-Walter Goix Massimo Valla
Telecom Italia Lab, Italy
laurentwalter.goix@telecomitalia.it
massimo.valla@telecomitalia.it

Laura Cerami Paolo Falcarin
Politecnico di Torino, Italy
laura.cerami@polito.it
paolo.falcarin@polito.it

Abstract

Mobile phones are being increasingly equipped with sensors that ease retrieval of context information about a user. Context data can be aggregated with information centrally available to mobile operators and service providers, to infer higher-level information such as user "situations", easier to integrate with services. We have been conducting an internal trial monitoring the context of different users in their business life and designing rules to infer high level situations: logical location, activity and social state. In this paper we present the infrastructure and the rule-based reasoning process used for this experiment.

1. Introduction

Cellular phones are becoming users' best friend and an opportunity for mobile operators and service providers to learn more about user's context, habits and preferences on the move and to exploit this information for service adaptation. Mobile phones are increasingly being equipped with sensors that ease the gathering of such information coming from the edge to be stored, processed and reasoned in the network. Such aggregation, also including information centrally available to mobile operators and service providers, allow the inference of higher level information, such as user situations, that can more easily be linked to services for personalization or recommendation scenarios.

We have been conducting an internal trial of several users over more than a year in their business life, monitoring and gathering relevant context information as they evolve in their work ambiance, and designing rules to infer high level situations based on the raw data collected. Rule-based systems to infer information have been investigated: the novelty of our approach is to apply rule-based reasoning to the mobile environment with real users involved and to focus on three dimensions of a user's situation: logical location, activity and social state.

In the paper we introduce the trial infrastructure used to collect and process context data coming from heterogeneous and distributed sources. We then present the reasoning process and the domain rules applied to produce meaningful situations to be used for service scenarios, and how the inference process has been implemented using rule-based programming. We finally discuss related work, and present our future directions of research.

2. Reference architecture for situation reasoning

Identifying a mobile user situation requires the gathering of context information from the mobile terminal, from the mobile operator's network and from central servers that store profile data and more traditional Information Technology (IT) service related data (calendar, e-mail, etc.). These IT servers can be operated by service providers external to the operator' service layer boundary.

To derive abstract situations from such heterogeneous context sources, a context brokering architecture is used to collect, aggregate, and provide available context data. The reference architecture used for situation reasoning is illustrated in Figure 1.

In this architecture the mobile phone acts as a Context Source (CS) by sending periodically, or upon specific events, context information to a Context Broker (CB), which aggregates and stores data in a short-term cache and eventually in a long-term log memory for post-processing (Context History).

Context information is requested by the Context Broker to Context Providers (CP) that integrate external back-end systems to gather context information. Context Providers and Context Sources form the Enabler layer, which is further detailed in the following subsections.

1-4244-1241-2/07/$25.00 ©2007 IEEE

Context information is eventually requested by Context Consumers (CC): examples are context-aware applications executed on an Application Server or the same mobile terminals that request aggregated context data.

The Situation Provider (SP) is the element that implements the Situation Reasoning process, part of the Reasoning layer. The SP retrieves context data from various Context Providers (acting as Context Consumer), infers the situation of an entity, typically a user, and provides it as context information back to the platform for integration with services (acting as Context Provider). The SP, as well as its interaction with the service layer, is further described in section 4.

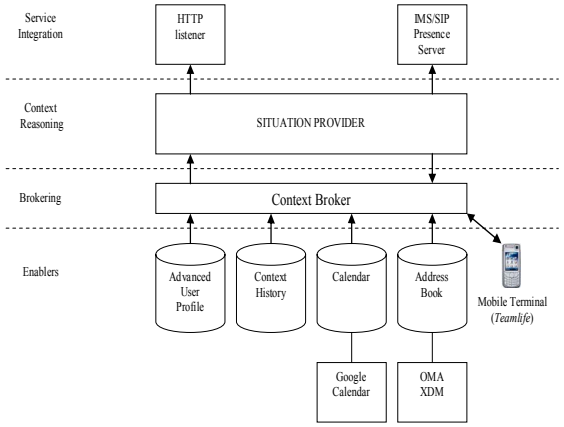

Figure 1. Global architecture of the situation reasoning process

2.1. Extracting context from the device

In our implementation the TeamLife application running on the mobile terminal is configured to silently and periodically send to the Context Broker information about: nearby Bluetooth devices, geographical position (from GPS sensor if available) and GSM/UMTS Cell-ID information. The application supports multiple user-configurable context update policies, for example to send context information only when one specific type or a number of nearby Bluetooth devices have changed.

2.2 Adding context providers

Besides context data sent by the user device, additional context information is obtained from Context Providers registered with the Context Broker. In particular the following Context Providers have been used to perform situation reasoning.

• Advanced User Profile (AUP): The AUP is the logical repository of all profile information about users. This profile includes in particular: Identity, Registered User Devices (IMEI, BDAddress, etc temporarily associated with the user through the mobile application), SIP address, and Virtual Places, which associate a location or Bluetooth information to a name and type (according to the Location Types Registry [1]). Such locations can be generic (e.g. a specific meeting room) or meaningful for a user (e.g. his own office, own car, etc..).

• Contacts/Address Book Information: In our information model all user's contacts are centrally stored in a prototype implementation of the OMA XDM [2] (XML Document Management) enabler used as repository for address books in IMS networks. Contacts are grouped in categories to distinguish, for example, "family" members, "friends" or "colleagues". Since relationship is not standard information within XDM, we developed a specific XML extension to represent this concept within each user's address book.

• Calendar Information: To include appointments and meetings scheduled for our users, we have integrated Google Calendar as a centralized repository for event-related context information. Calendar events returned from Google Calendar's APIs [3] are converted by the provider and stored in the Context Broker cache like any other context information.

All these Context Providers return information structured using a proprietary XML-based Context Markup Language (ContextML), which provides context data as parameter/values pairs, as well as meta-information such as source, entity, scope (or context "topic"), timestamp and validity. Use of this uniform and simple ContextML language has proved to be very effective to transport, cache, aggregate and finally parse context data by Context Consumers.

3. User situation rule engineering

The reasoning process that we have implemented consists in the inference of a high-level user's situation by the means of rule-based reasoning techniques applied to user profile and context data.

Inferred information can be used to personalize existing services and to offer new, more attractive ones.

3.1 Identifying situations

We have defined "situation" the grouping of three high-level concepts, targeting very abstract and general

information that can easily be used for service personalization.

- **Logical location** (or "place type"): provides a user's position from a logical perspective, associating a meaning to the place where the user is currently located. Such location can be absolute or relative, for example when moving in a vehicle. In case of absolute position, GPS coordinates and civil location are associated to the type of place. Alternatively, BDAddress can be used as location identifier. Examples of such user location are: "office", "own_office", "meeting_room"
- **Activity**: provides information related to the user's current activity, such as "working", "formal_meeting", "late_to_meeting", "waiting_formal_meeting"
- **Social state**: tells whether the user is on its own or with other people, possibly providing their relationship. Example values are "with_colleagues", "with_friends", "with_Massimo", ecc.

In order to obtain this more abstract representation, we have defined rules which are necessary for reasoning on each concept of interest and its possible values.

In particular, we define "Context Reasoning" the process used to derive logical location and social state from raw context data, and we define "Situation Reasoning" the identification of the user's activity from context data, logical location and social state.

We have then characterized a list of target situations for a user in a normal working day: such situations refer typically to combinations of the three main concepts that can be inferred out of the identified context data.

For example, when the user has a meeting scheduled in his Calendar, the rules we have designed can automatically deduce his current activity, which in this case can be either *waiting_formal_meeting*, *late_to_meeting* or *formal_meeting*.

Considering the *waiting_formal_meeting* example, this activity will be inferred only if all the following conditions hold:

- The current user activity is working,
- The user has scheduled a "meeting" event in his/her Calendar,
- The user is currently in a "meeting room" (logical location)
- The scheduled meeting place corresponds to the current user location,
- Some participants are still missing.

Instead, if the user is not at the right place, his/her activity will be *late_to_meeting*. In this case, this activity is used as a trigger of a service that alerts the related user by SMS to remind him/her of the meeting and the expected place.

Finally, if all meeting participants are present (by analyzing nearby Bluetooth devices), the user activity will be *formal_meeting*. Such activity can easily be integrated in context-aware communication services that filter and optimize calls, for example to forward non-urgent calls to a voicemail system.

3.2 Selecting the rule language

In order to share the information provided by several Providers as facts within the reasoning module we have decided to translate ContextML data in RuleML [4]. Rules designed to infer situations have also been represented in RuleML. This choice has enabled us to express facts and rules in a standard and non-ambiguous machine-readable formalism, providing knowledge base independence from the underlying rule engine. We could hence change rule engine without rewriting the knowledge base, by using translators. However, this approach had some limitations as RuleML does not have all the logical language features, which are already present in "native" rule languages like JESS [5].

3.3 Asserting facts from context data

As we needed to apply rules on data coming from different providers, we had to translate these data in facts, written in RuleML.

Simple user's data like attribute-value couples provided by the ContextML language can be easily translated in simple facts composed by a sequence of strings; for example the IMEI code of user's phone can be written as an ordered fact, like this:

```
(assert (IMEI user 123453357776666))
```

Ordered facts are simply Jess lists, where the first field (the *head* of the list) acts as a sort of category for the fact.

In order to represent complex and structured data as facts, we have used a technique known as "reification": complex data is represented with a set of *n* simple facts sharing the same symbol, which acts as a sort of category identifier for this set.

An example of structured fact could be: "For the user 3357776666 the connection with the cell 222-123-456-789 represents a location type 'office'". This information is represented by the three following facts:

```
(assert (user location 3357776666))
(assert (user location 222-123-456-789))
(assert (type location office))
```

Every time reification became necessary, we have also added a specific fact "identity" which ties the category identifier (in the example, *location*) with the symbol that was referred unambiguously to the user (*3357776666*).

```
(assert (identity 3357776666 location))
```

3.4 Designing rules

All situation rules have been designed using the following general template:

```
(defrule  [context pre-conditions] =>
  (assert  ([new situation]))
  (retract ([no longer valid situation]))
```

having less aggregated context information on the left-hand-side and more aggregated and high level context on the right-hand-side. Rules can then be composed in an incremental way having as pre-conditions a Logical Location (*meeting_room*) and a Social State (*with_colleagues*) from the application of previous rules, and inferring from them an Activity (*informal_meeting*), typically retracting the former situation contemporaneously.

4. Inferring situation

4.1 The situation reasoning process

The situation reasoning module we have developed performs six sequential tasks described in Figure 2.

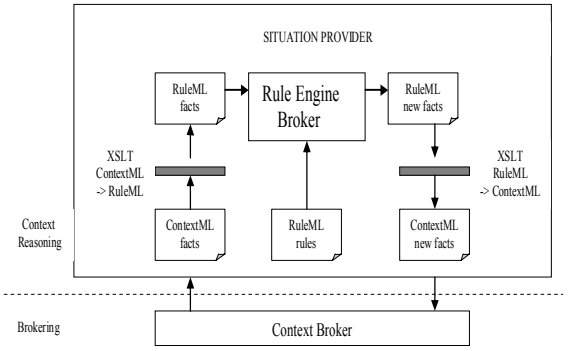

Figure 2. Situation reasoning process

The situation reasoning process is defined as follows:

1. Specification of RuleML rules for situation inference (only performed once);

(For each inference cycle)

2. Facts extraction from the Context Broker (and retraction of old facts);
3. Knowledge Base population through the automatic translation of facts from ContextML to RuleML through an XSL [6] stylesheet;
4. Execution of rules on the knowledge base in the Rule Engine Broker that decouples from the actual underlying rule engine;
5. Possible assertion of new facts in RuleML, depending on executed rules;
6. Internal notification of the newly deduced facts to the Situation Provider.

4.2 The situation provider

The Situation Provider is a Context Provider developed as Stateless EJB and run on a JBoss Application Server.

Situation information is computed both by interacting remotely with the Context Broker to retrieve context information from external Context Providers, and by interacting locally with the Rule Engine Broker for situation inference. The Rule Engine Broker is a software module that integrates multiple rule engines and provides a single abstract interface based on RuleML. Our implementation relied on JESS 7.0 as underlying rule engine.

4.2.1. Retrieving context information. The SP interacts with the Context Broker to retrieve context information required for situation inference. In our implementation the following providers are involved: CH (Context History) for location-based information (GSM cell, GPS coordinates & civil location) as well as nearby Bluetooth devices identifiers provided by the device; AUP (Advanced User Profile) in particular for the user Virtual Places and the registered user devices' information (BDAddress); Calendar Provider, for each meeting's name, location and participants' email addresses; Address Book, for information related to full name, SIP URL and relationships (for example friends or colleagues).

4.2.2 Providing situation information. External interfaces are provided as REST-like interfaces through HTTP GET requests by a servlet front-end that returns ContextML content. Upon remote HTTP invocation, the servlet retrieves an instance of a Situation bean implementing the situation reasoning process and asks him for inferring the entity's situation at runtime. The newly produced facts (if any) are then converted back into ContextML format from RuleML using a second XSL stylesheet to output the inferred situation.

We have also designed the SP to monitor a group of users (by inferring user's situation at each change of their raw context) and send a remote notification whenever a user's situation changes. Notifications are provided as external HTTP requests (for example to activate a service) or through a SIP PUBLISH towards a predefined SIP Presence Server acting as "Presence Network Agent" for integration into IMS-based networks.

5. Related work

Nowadays in the area of service personalization, some service providers are coping with context data acquisition, interpretation, and aggregation, but there are different opinions on what should be considered as "context". In order to set a boundary between what is context and what is not, we follow Schilit et al. [7] view of context as the user's location, the social situation and the nearby resources.

Pappas et al. [8] have introduced the concepts of primitive context, active primitive context and current context .to model the "discrete adaptation capability of a system" and its evolution over time. While they use a predetermined formalization based on three levels, our implementation relies on a flexible hierarchy of generic context levels. Furthermore some conflicts could occur in their model when more primitive contexts become active. Our paradigm relies on a knowledge base, which is always consistent: as in a rule engine the rules execution order is not deterministic, we have defined an execution priority among rules that could conflict (i.e. fire simultaneously). We have thus managed the contemporary use of new facts assertion and retraction of previously asserted facts that are no longer valid.

Kolari et al. [9] have implemented a context-aware service platform that allows designing and managing context-aware applications, which has been validated by several users. While they have provided an ad-hoc Java implementation, we have used a rule-based approach to derive higher-level information from raw context data in a more flexible way: in our case rules, depending on their effectiveness, can be easily inserted in (or removed from) the knowledge base without changing the core logic of the application, for example to provide further situations as needed.

Yau et al. [10] have provided a similar hierarchical Situation modelling and reasoning, describing the knowledge base with OWL-DL [11] and transforming it in first-order logic predicates to perform situation reasoning. While they cope with knowledge representations and transformations they do not consider reasoning on situations with a limited temporal validity; in our approach, whenever a situation changes or is no more valid it is removed from the knowledge base, thus we rely on rule engine features to keep contexts and derived situations synchronized with dynamically changing raw context data coming from context providers.

6. Conclusions and future work

This paper introduced a rule-based approach for inferring situation of mobile business users out of raw context data collected both on their mobile devices and within the network. We are currently working at the optimization of the rule-based reasoning process as well as the design of rules for other types of situations, namely *travel* and *home* ambiances. Regarding context and situation information, we are investigating the gathering of further information related to the user behaviour, like the applications they use, and are thinking about adding further dimensions to the *situation* information in the future. Finally an architecture to distribute the reasoning process between terminals and server components is being considered.

7. References

[1] Schulzrinne H., Tschofenig H., "Location Types Registry", IETF RFC 4589, July 2006. http://www.ietf.org/rfc/rfc4589.txt

[2] XML Document Management (XDM) Specification 1.0, Open Mobile Alliance (OMA http://www.openmobilealliance.org/release_program/xdm_v1_0.html

[3] Google Calendar Data APIs and GData, http://code.google.com/apis/calendar/overview.html

[4] RuleML: The Rule Markup Initiative, http://www.ruleml.org/

[5] JESS: the Rule Engine for the Java Platform, http://herzberg.ca.sandia.gov/jess/

[6] World Wide Web Consortium, XSL Transformation (XSLT) 1.0, W3C Recommendation, http://www.w3.org/TR/xslt

[7] Schilit, B.N., Adams, N., and Want, R. "Context-Aware Computing Applications". In *Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications*, p. 85-90, 1994.

[8] Pappas A., Giaffreda R., Hailes S., "A Design Model for Context-Aware Services Based on Primitive Contexts", in *Workshop on Advanced Context Modeling, Reasoning and Management (UBICOMP 2004)*, Sept 2004

[9] Kolari, J., Toivonen, S., "Kontii - Context-aware Mobile Portal", *ERCIM News n°54*, 2003

[10] Stephen S. Yau, Junwei Liu, "Hierarchical Situation Modeling and Reasoning for Pervasive Computing", in *Proc.ceedings of 3rd Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS 2006)*.

[11] World Wide Web Consortium, OWL Web Ontology Language. On-line at http://www.w3.org/TR/owl-ref/