

Applying MUPE Context Producers in Developing Location and Context Aware Applications

Kimmo Koskinen¹, Kari Heikkinen¹ and Jouni Ikonen¹

¹ Lappeenranta, University of Technology, Laboratory of Communications
P.O.Box 21, FIN 53850
{kimmo.m.koskinen, kari.heikkinen, jouni.ikonen}@lut.fi
<http://www.it.lut.fi/>

Abstract. Location based services (LBS) and applications have recently emerged as a significant application area. However, location based services and application could also benefit of the dimensions of the contextual data. Multi-User Publishing Environment (MUPE) has a built-in context mediation capability that allows the application developer to concentrate on using contextual data and thus enabling rapid prototyping of location and context aware applications. In this paper MUPE context producers are applied so that the applications can exploit the different available context in a manner suitable to their logic. This paper purposes to demonstrate that context mediation is a powerful tool to speed up the prototyping process and to enable an efficient application development.

1 Introduction

Mobile devices have changed the way people use their time. These devices and the applications within them can react to various contexts with built-in sensors and contexts received through communication links. Thus, mobile devices are ideal terminals for e.g. location and context-aware services and applications due to possibilities they offer. However, applications development can be very time-consuming, as a lot of time is needed building the infrastructural support for the application and testing the application in real live test bed. This paper focuses on how the MUPE helps the developer in context mediation.

During the past few years, context and context awareness has raised a lot of research interest, especially within the Ubicomp (<http://www.ubicomp.org>) and Pervasive (<http://www.pervasive.org>) conferences. The different dimensions of the context have been quite extensively studied, e.g. in [1-3]. A good description of context computing and different layers of it is presented in [1]: context computing focuses both on identifying/collecting the context and using the collected contextual information. In [2, 3], support for context provisioning on mobile devices is discussed. It is being suggested that the mobile devices would collect the context, but that can be highly resource consuming and thus context provisioning is highly important. However, *context production relevant to* the application logic is quite often omitted in these approaches. Furthermore, most of the approaches concentrate on the semantics and ontology for

different contextual raw data. In [4], guidelines for mediation of context-aware applications are presented. These guidelines are redundant mediation technologies to support more natural and smooth interactions, facilitators to support user input and feedback, defaults contexts to minimize user mediation and ambiguity should be retained until mediation is necessary. As [4] also points out, mainly time and location are widely used on context awareness studies. Applications could benefit more, if contexts could be added or mediated to the application logic. In [5], a context toolkit for aiding the development of context-enabled application is presented. In [6], a context broker is introduced that is able to mediate between different context sources, has built-in semantics etc. Such a task, semantically bullet proof, can be very ambiguous task.

Location is possibly the most common context used. Location information is used in games [e.g. 7], GIS Applications and systems [e.g. 8], location dependant applications in sensor networks [e.g. 9], a model for ambient intelligence [e.g. 10], in guiding systems [e.g. 11]. The Pervasive conference has an annual workshop dedicated to Location and Context Awareness, e.g. in 2005 <http://loca2005.context-aware.org>.

Thus, we have concentrated on applying a solid middleware framework (MUPE [12], Multi-User Publishing Environment) to ease the development of location and context-aware applications. MUPE contains an interface for context producers that are the most important components for the purpose of this paper. MUPE context producers provide an easy way to add contexts to application logic as context producers act as mediator of contextual raw data. Any single context producer can provide context information to any MUPE application.

This paper is further structured so that Chapter II will present the characteristics of the context sources and context mediation in more detail. The approach is demonstrated with a location and context-aware application (TRIX) in Chapter III. TRIX (Tribal Exchange) is a game that uses variation of contexts in the game play. Final chapter (Chapter V) will conclude the paper.

2 Producing and Mediating Context

Applying external context in applications can be complicated. Many different information sources can produce information that could be used as contextual information both in applications and in information systems. We examine few contextual information producers that relay the information to a MUPE application. For application development, five different context producers were implemented; 3 (three) different location information producers (LIS, Ekahau and RFID-tag), weather information (environmental context) and recumbent exercise device (physiological context). These context producers fetch the information from the following information systems and information sources (see *table 1* for further details how the context information is fetched):

- LIS (Location Information System) is an information system that is a middleware platform built to collect and deliver positioning information in a wireless local area networks (WLANs).

- Ekahau Positioning Engine is a commercial product that can provide coordinate based location information.
- RFID (Radio Frequency IDentification) is a RFID-tag that were put in several places
- FMI.FI is on-line weather information maintained by the Finnish meteorological institute that can provide environmental context such as temperature, air humidity, wind speed etc.
- Recumbent exercise bike is an exercise bike that can provide physiological context such as heart rate and cycling speed

Table 1. More detailed information about the Context Producers.

Context Source	How the context is fetched?	Communication /Interface/Protocol
LIS	Location information is based on a cell-id of an access point. The context producer (CP) is configured to follow selected hardware addresses of wireless network cards in use. The CP queries frequently the positions	Communication is done through the SOAP interface.
EPE	EPE requires radio finger printing of the used area. The context provider acts as a client for EPE. Coordinate, floor, area and probability of a user being on that location area are recorded from the location observations provided by EPE. CEP atoms are formed from the received information and updated to MUPE applications on selected intervals.	The information is fetched through Java RMI interface in two second intervals
RFID	The ID of a passive RFID tag is read by a RFID reader. After the tag is detected, designated software reads the content of a passive RFID tag and acts as a TCP server for applications. The CP gets the information about the detection and changes the relevant CEP atom.	The reader had a serial interface which is used to inform that a tag is detected. TCP is used as a protocol to carry the context information.
FMI.FI	The weather context provider fetches information (once in hour) from the web pages and parses selected weather parameters from the page. The parsed results are sent to the MUPE application in a CEP atom format.	HTTP is used to obtain the information from the web pages.
Recumbent	A software component was implemented for reading exercise data form the bike and serving data over a TCP connection. A protocol for transferring exercise data over a TCP connection was also designed. This protocol enabled a client to request data from the recumbent. The context producer gets the exercise data via TCP and converts it to a CEP atom format.	This information is read via a serial interface in a format which is defined by the manufacturer. The exercise data is transferred via TCP to the context producer.

Figure 1 show how contextual information is processed for a MUPE application. The context information is fetched from various sources by the context producer components. They relay the source specific context data to the context manager middleware

component. This component stores the context data and maps the context source *object id* to a application specific *object id*. The most important functionalities are conditionally (if-then-else) structured XML scripts that can be used to make Java method calls in the MUPE server. The MUPE server is contained in the MUPE World Manager component. This component contains the Context manager object. This object contains methods for processing the received contextual information. Context information is given in the parameters of the method of the object.

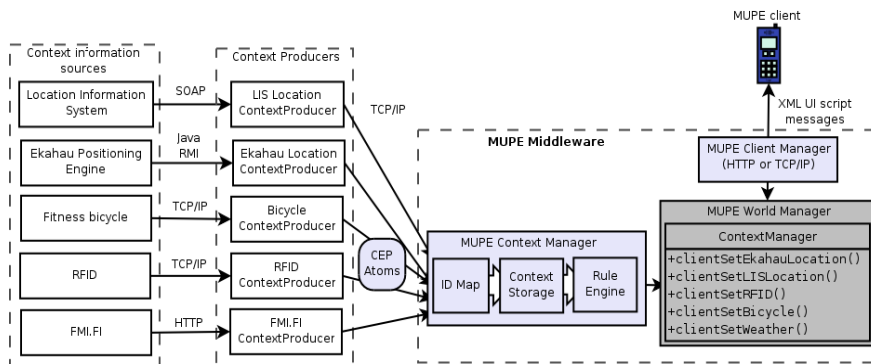


Fig. 1. Context Originating systems and MUPE context processing.

Let us examine one context information source (LIS) and the context processing in more detail; *Figure 2* presents a message scheme how location information is transferred from LIS to a MUPE application by using LIS context producer and MUPE context manager. *First*, a request (see *Listing 1*) by LIS Location Context Producer to the LIS system is made (the scheme also shows used parameters both for request and answer messages). *Secondly*, LIS answers to the request. See *Table 2* for examples of implemented SOAP request methods and triggers. Other implemented SOAP requests were *GetLocationInfo* (about a specific cell id), *GetLocationList* (lists all cell ids). Other implemented SOAP triggers were *DeleteTrigger* (deletes the trigger) and *Fire-Trigger* (follows the user/device within LIS).

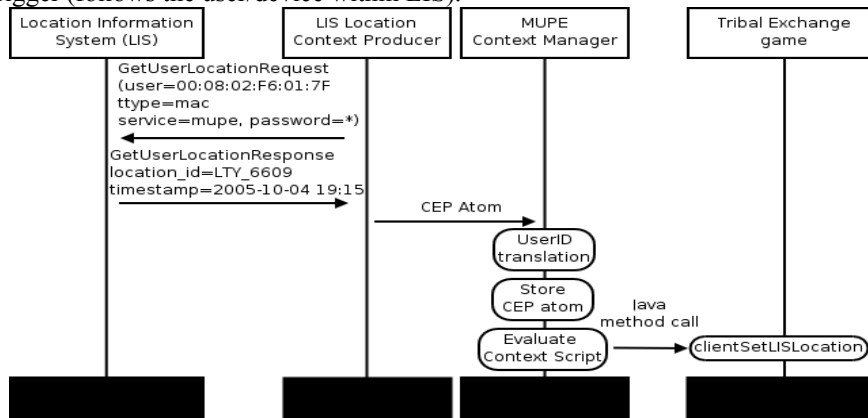


Fig. 2. Context Mediation sample from the Location Information System to the Application.

Table 2. Two Implemented SOAP methods and triggers.

Method	Description	Parameters
GetUserLocation	A SOAP request that obtains the last known location of a given MAC address.	<i>target</i> : target of a request <i>caller</i> : caller of a request <i>service</i> : the service that requests <i>password</i> : service password <i>ttype</i> : type of a target (e.g. mac-addr) returnvalue : <i>GetUserLocationResponse</i> , that contains <i>cell-id</i> and the <i>timestamp</i>
SetTrigger	A SOAP trigger that assigns the given trigger to a specific user.	<i>target</i> : the user to be followed <i>service</i> : the service that requests <i>password</i> : service password <i>uri</i> : namespace for the trigger <i>proxy</i> : URL address of the method returnvalue : trigger <i>ID-number</i>

Listing 1 shows (below) an example of data that is received from the LIS and further converted into XML for CEP-protocol. A CEP-message informs that user defined with hardware address 00:08:02:F6:01:7F is located in a cell defined as LTY_6609. In addition the message includes the timestamps of the observation and the description of the cell.

```
<atom name='LISLocation'
  timestamp='2005-9-5 1:23:26,00 +1'
  source='http://gamesrv.wlpr.net:1234'
  userId='00:08:02:F6:01:7F'>
<string name='timestamp'>2005-10-04 19:15:35... </string>
<string name='location'>LTY_6609 </string>
<string name='description'>...building, 6th fl. </string>
</atom>
```

Listing 1. Cell information in CEP-XML format.

Thirdly, after getting the cell-id on a given time the context producers transforms the information into a CEP atom form and gives it to a MUPE context manager. The context manager can carry out actions based on the received information. Any application can use the same logic and so each application does not need a component of its own. Inside the MUPE context manager there is three phases; first the CEP atom object is converted into an ID identifiable by an application (see *listing 2*).

```
<addUserMap>
  <userMap>
    <contextProducerName>Wlan-positioningsystem
    </contextProducerName>
    <contextProducerId>00:08:02:F6:01:7F
    </ contextProducerId>
    <serverId>Brian
    </serverId>
  </userMap>
</addUserMap>
```

Listing 2. Conversion of a CEP atom.

This converted information is stored and a script engine component is informed about an arrival of a new context. This script engine component contains the if-then-else conditional functionalities. These conditions can reference the stored information inside the MUPE context manager. If these conditions are met, the actions can be carried out in the MUPE application (server). The MUPE middleware platform contains tools for making such scripts. *Listing 3* shows an example script of a method call named `clientSetLocation`.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<script id='lis' <-- namespace addresses (see list 4)
<if> <atomChanged>
  <atomRef userId='' name='LISlocation'> </atomRef>
  </atomChanged>
<actions> <mupeCall>
  <![CDATA[2::clientSetLocation ${userId}
  {LISlocation::location}]]>
  </mupeCall>
</actions>
</if>
</script>
```

Listing 3. Example of a triggered script.

The context sources were not optimal in many manners:

- Weather information can be updated more frequently than an hour to the MUPE server, but this is redundant as our context source (FMI.FI) updates the information only in hourly base. We could have different sensors of our own to provide similar and more frequent data.
- For production system, it would definitely be more efficient to make the recumbent reader programs to output CEP atoms directly. The server part for recumbent and context producer is not directly embedded to our system currently. In the next stage context producer should do this job directly. However, a separate context producer can handle a bigger load than the reader program (which could reside on a less powerful computer). Similar approach should be taken with RFID-information also.
- The Mobile Phone can also act as a reader by using NFC (Near Field Communication) shells inserted into a mobile phone. However, for the moment there is not open source software available for this purpose.

2.1 MUPE Analysis

This short subchapter purposes to compare the MUPE approach in both addressing context information (e.g. location) and technical details. This analysis is partially based on Korpipaa Ph.D. thesis (Blackboard-based software framework and tool for mobile device context awareness). Korpipaa introduces the definitions for context and context awareness (mostly Dey's definitions) and furthermore, offers critique related to the context awareness research. Korpipaa lists three major areas:

- Determination of an appropriate set of canonical contextual states may be difficult or impossible. In MUPE use of context is application oriented, and thus this dynamic aspect of context can be overlooked as the handling of the collected context does not vary in the application logic unless the developer has designed it so.
- Determination of what information is necessary to infer a contextual state may be difficult. In MUPE this aspect is the same. However, yet again, it is up to developer to decide the level and the use of collected context. In the beginning, MUPE was built for mobile games, and thus the approximation of real context is enough. However, we can argue that MUPE is not an optimal solution for ‘real’ applications.
- Determination of an appropriate action from a given action may be difficult. In MUPE this is the Java action call. Dedication to Java may be in some cases a clear bonus, but for some application areas a disadvantage.

Korpipaa also compares different context frameworks (e.g. Dey’s context toolkit). Context toolkit consists of components that provide for applications the functionalities for handling the context. Korpipaa also lists a number of client-to-server models, blackboard-based context architecture models and models that are somewhat related to the context management. MUPE is listed in the last category. Comparing MUPE to other alternatives some aspects can be pointed out:

- MUPE has a networked blackboard-based context engine with some similar features to Korpipaa solution (e.g. support for context requests)
- MUPE context engine is designed differently, as the contexts come from outbound entities and thus MUPE does not need context to operate. However, the applications can directly benefit from various contexts
- The response times in MUPE are lower as the applications are in the server side (not in the client). However, the protocol support and security issues in MUPE need to much better, e.g. due to privacy reasons
- Context handling is different, as MUPE handles context in compound structures. However, sensors could send a lot of information and is not feasible to do all processing (and consume the battery) in the mobile device .
- MUPE does not directly support any databases. LIS system in this paper contains databases, but MUPE does not directly access those databases.
- MUPE does not address context recognition, customization of contextual information and context relevant application control

3 How Applications Benefit of Context Mediation

Applications may have very different type of use for context data. Still there are common operations that the context aware applications have to handle. When a new context data arrives from the context source, actions need to be done based on the data. To demonstrate the possibilities of a context use, a mobile game named Tribal Exchange (TRIX) was implemented. TRIX is a game, which combines virtual world and physical game environment together. Game world is constructed from a map, where

each tribe (player) has multiple houses. One of the houses acts as headquarters (village) and the other houses are housing quarters. Idea of the game is to protect houses from different nature catastrophes. Houses are protected by building walls around the houses. The game is divided into phases. If a tribe has successfully protected all their buildings in the end of the phase, they will get another building. Otherwise unprotected buildings are removed from the map. A wall can be built only on a flat ground. Rocks, forest and water prevents building.

Building of a wall requires tribes to collect codes from the game environment. Codes are free form text strings. Codes map to the utilities in the virtual world. These utilities can be pieces of wall or tools to remove obstacles like rocks. Wall pieces are owned by tribes and exchanging building blocks with other tribes plays an important part in the game. These game pieces can be searched by using location information. Code pieces have a location, which is available from the location context produced from LIS. The players define their device address and after that they can request codes in their location. A player can use only one code at a time, even if the current location would contain more codes. Another context used in TRIX is weather-related contextual data. These are temperature, humidity and wind speed. New weather input changes durability of different wall pieces. Example of a weather effect is that walls built of ice start melting if temperature rises above zero. When the durability points of a wall piece are used up, the wall breaks and must be replaced with a new wall piece. The durability changes are defined with context manager scripts. The rules (e.g. `<![CDATA[2::clientDoDamage {${userId}} {128} {-2}]]>`) have different if conditions; in the sample above, if the value of humidity is between 80 and 100, the clientDoDamage function triggers an event in the game that decreases the properties of an wall piece (in this case, a straw, identified by the number 128) by two (which in a game logic decreases the durability of a straw by two). Other weather rules were temperature, air pressure, wind speed, wind direction and cloudiness. Furthermore, events (in the area) such as lightning strike(s), forest fire, earthquakes, flood, tsunami, meteorite rain etc. have their own rules. Mostly, the effects were scaled down, because some events (e.g. earthquakes in Finland need only 2.0 Richters) do not take place frequently.

4 Conclusions

This paper presented how MUPE application platform can be applied in developing location and context-aware applications. As the context production resides on the server side, the mobile terminal resources are not exhaustively consumed and leaving the scarce resources for the mobile device. The paper presented several context producer implementations that are an integral part of the application logic. The context production is implemented by creating special context producer components, which provide a uniform access to context data. This set-up allows the developer to concentrate on the application and content design. The applications can benefit from the context mediation because the trigger scripts can be reusable and can be part of any application logic.

References

1. A.Ferscha, C.Holzmann and S.Oppl, Context Awareness for Group Interaction Support, Proceedings of MobiWac 2004 Conference.
2. J.I.Hong and J.A.Landay, An Infrastructure Approach to Context-Aware Computing, Human Computing Interaction (HCI) Journal 16, 2-4 (2001), pp. 287-303.
3. R. Mayrhofer, An Architecture for Context Prediction, Advances in Pervasive Computing, DC in Pervasive 2004 Conference, vol. 176, April 2004, pp. 65-72.
4. A.K.Dey and J.Mankoff, Designing Mediation for Context-Aware Applications, ACM Transactions on Computer-Human Interaction, Vol. 12, No.1, March 2005, pp. 53-80.
5. D.Salber, A.K.Dey and D.Abowd, The Context Toolkit: Aiding the Development of Context-Enabled Applications, Proceedings of CHI'99.
6. H.Chen, T.Finin and A.Joshi, An Intelligent Broker for Context-Aware System, Proceedings of Ubicomp 2003 conference.
7. O.Rashid, I. Mullins, P.Coulton and R.Edwards, Games: Extending cyberspace: location based games using cellular phones, Computers in Entertainment (CIE), Volume 4 Issue 1, January 2006.
8. C.Wang, X.Xie, L.Wang and W-Y.Ma , Components of GIR: Detecting geographic locations from web resources, Proceedings of the 2005 workshop on Geographic information retrieval GIR '05, November 2005.
9. T.He, C.Huang, B.M.Blum, J.A.Stankovic and T.F.Abdelzaher, Range-free localization and its impact on large scale sensor networks, ACM Transactions on Embedded Computing Systems (TECS), Volume 4 Issue 4, November 2005.
10. I.Saitoh, Reconfigurability and location-based services: A location model for ambient intelligence, Proceedings of the Smart objects and ambient intelligence: innovative context-aware services (sOc-EUSAI '05), October 2005.
11. W.Yue, S.Mu, H.Wang and G.Wang, Guiding and navigating: TGH: a case study of designing natural interaction for mobile guide systems, Proceedings of the 7th international conference on Human computer interaction with mobile devices & services MobileHCI '05, September 2005.
12. MUPE portal, <http://www.mupe.net>.
13. P.Korpiää, Blackboard-based software framework and tool for mobile device context awareness, Ph. D. Dissertation, University of Oulu, Finland.