A Study on Issues in Context-Aware Systems
Based on a Survey and Service Scenarios

Sangkeun Lee, Sungchan Park, and Sang-goo Lee
School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Republic of Korea
{liza183, baksalchan, sglee}@europa.snu.ac.kr

Abstract
To realize a practical context-aware system, we should be clearly aware of important issues in context-
aware systems. In this paper, we review a wide range of existing researches and the history of context-aware
systems. We derive technical issues on realizing context-aware services based on the history and explain
them. In addition, we describe seven context-aware service scenarios and discuss related issues. We believe
that the derived issues from our survey and the proposed service scenarios will be used as useful reference
for future researches.

1. Introduction
The advanced network and computing technologies and the wide spread of mobile devices such as PDAs
and smart phones make computing model more distributed and pervasive in a way that users could
connect to the network wherever and whenever. At the same time, users expect to get useful and proper
services via mobile devices based on their contexts without much concern.

A context-aware system is one that actively and autonomously adapts and provides the appropriate
service or content to users, using the advantage of contextual information without too much user
interaction. In most current systems, user should explicitly request services; however, if feasible and
practical context-aware systems are realized, providing various and new forms of services, which are more
active and autonomous, would be possible. Consequently, service providers will also get more business
opportunities.

We review existing context-aware computing researches in the past and derive related technical
issues and propose seven context-aware service scenarios. We believe that the derived issues will be
used as useful reference for future researchers working on realizing context-aware systems.

The rest of the paper is organized into following sections. In section 2, we present the history of context-
aware computing and summarize the technical issues. In section 3, we present practical context-aware service
scenarios and discuss related issues. In section 4, we conclude our work and discuss future work.

2. History of Context-aware Systems

2.1 Context Definition
It was in 1994 when the term ‘Context-aware’ was first introduced by Schilit and Theimer. They defined
context in terms of location, identities of nearby people, objects, and the changes to those objects over time[1].
Similarly, earlier researchers have tried to define context by proposing specific context examples such as
location, identity, activity, time, and so on[2][3]. In the Dey’s context definition in 1998[4], emotional status of
user is first mentioned as one of context elements. It is important that it was the first definition that considered
abstract concepts as context elements as well. In 2000, Dey and Abowd[5] proposed one of the most often
cited definitions of context: ‘any information that can be used to characterize the situation of entities that are
considered relevant to the interaction between a user and an application, including the user and the
application themselves’. We found out a tendency that the context definitions tend to get more general and
wider as time passes.

2.2 Context-Aware Systems and
Applications
There have been a wide variety of context-aware systems and applications from domain specific context-
aware systems or location-based systems to general and
extensible systems. Generally, a context-aware system is responsible for various tasks: representing context data, gathering and managing context data, service matching, and so on. Therefore, it is required to consider various aspects of context-aware systems when reviewing existing systems.

One thing we should examine is what kind of context data the system can support. In addition, we need to review what the architecture style of each system is (e.g. peer to peer or centralized server). Context information can be represented in many ways, and there are not only simple data models but also complex hierarchical or ontological data model used in more recent context-aware systems. Which context model is used in a context-aware system is one of the important aspects. So, we should review the context models used in the systems as well.

2.2.1 Location-aware Systems and Domain-specific Systems

Since early 90s, many of earlier context-aware system researchers have developed location-based service systems such as Wan[6], Sumi[7]. They share common characteristics in that they provide information according to user’s current location and they are domain-specific applications. Earlier systems use limited types of context data and are usually designed for specific purposes. Examples of application domains are various such as call forwarding [6], tour-guide[8], shopping assistant[9] and so on.

2.2.2 General Purpose Context-aware Frameworks

One of early attempts of developing a context-aware framework for general purposes is Context Toolkit proposed by Dey and Abowd in 2001. Context Toolkit provides context widgets, and they can be used as reusable software components for accessing and interpreting context data while hiding details. Therefore, Context Toolkit can support various domains of context-aware applications.

In 2002, Hofer proposed another general context-aware framework called Hydrogen[10] that can deal with various types of context information. Context Toolkit and Hydrogen both are peer to peer architecture; however, they have differences in that Context Toolkit uses a simple attribute-value tuple as context model and Hydrogen uses Object-oriented context data model.

W3C also announced the OWL (Web Ontology Language) in 2002. Basically, OWL is designed for the Semantic Web, but many context-aware systems adopted OWL as their context-model. OWL is suitable to be used as context model, because it’s flexible and expressive.

Chen’s CoBrA (Context-Broker Architecture, 2003)[11] is an agent-based context-aware system and the first context-aware system which uses OWL to model ontological context model and privacy policy language. In CoBrA system, a centralized resource-rich context broker maintains and manages the shared context data. It is CoBrA’s strong point that it allows users to define privacy policy so that it can protect privacy of users.

There are not so many existing context-aware systems which care about user’s privacy, although protecting user’s privacy is a very important issue. The Context Fabric[12] proposed by Hong is a context-aware system architecture primarily focusing on privacy protection. The system provides supports for building privacy-sensitive context-aware applications by letting users define privacy tags for each context tuple.

SOCAM[13] (Service-Oriented Context-Aware Middleware) by Gu(2004) is similar to CoBrA, because it also aims to build a middleware for context-aware services and uses OWL to model its context model; however, SOCAM adopted hierarchical approach for designing the context ontology. The ontology used in SOCAM consists of upper ontology and domain-specific ontologies.

Although OWL seemed very suitable as context model and many systems adopted it, but it brought new problems – scalability and performance of context-aware systems. Because current ontology inference engines cannot process computationally intensive high-level inferences, researchers tried to handle this problem in many ways. SOCAM reduces the scale of context knowledge and releases the burden of context data processing by separating upper ontology and domain-specific ontologies.

In 2008, Ejigu proposed Enhanced CoCA[14] (Collaborative Context-Aware Service Platform). CoCA based on HCoM(Hybrid Context Management model). To achieve better performance, CoCA uses ontology approach to manage context semantics and uses relational database approach to manage context data. By combining these two approaches and loading only relevant data according to heuristics, it achieves acceptable performance of system.

In summary, early systems use specific kinds of context such as location and are usually domain-specific application systems, but later systems are usually general purpose context-aware systems that can support various types of context. Context-aware systems tend to use simple context-data models (e.g.
attribute-value tuple) in earlier days, but later they use more complex context models (e.g. object-oriented model, graph model). After W3C announced OWL, many systems adopted ontology as their context model.

2.3 Research Issues

While surveying on context-aware systems, we find out four common research issues – Architecture Style, Performance & Scalability, Historical Context Data & User Behavior, and Privacy Protection.

There are two representative architecture styles of context-aware systems – centralized (context server) and decentralized (peer to peer). Table 1 summarizes the pros and cons of each method.

<table>
<thead>
<tr>
<th>Centralized (Context Server)</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● Easier to implement</td>
<td>● Congestion problem</td>
</tr>
<tr>
<td></td>
<td>● Simpler design</td>
<td>● Low fault-tolerance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decentralized (Peer to Peer)</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● No additional server</td>
<td>● Hard to handle dynamicity</td>
</tr>
<tr>
<td></td>
<td>● No congestion problem</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● High fault tolerance</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Comparison of Architecture Style

In a ubiquitous environment, many different devices should be able to deal with a wide range of heterogeneous data. Heterogeneity in data can be handled by standards and abstraction. Not only heterogeneity in data but also heterogeneous communication protocols and various kinds of hardware are another heterogeneity related issues.

There are many factors that could burden system performance. For example, many numbers of signals can be sent to the system at the same time; many users could access or query the same context data concurrently; the number of sensors and attached devices will increase as time goes by. A practical context-aware system should be able to handle these situations.

Historical context data and user behavior log can be used as feedback for better future services. For example, by analyzing the historical log or user behavior log, the system could automatically compose a new context-aware service or suggest a new service to service provider.

Supporting privacy protection is another issue in context-aware systems. Context-aware systems gather information from the users autonomously, so some users may feel uncomfortable that the system can use or open their information without any notice. Thus, users should be able to define their privacy policy (e.g. “my e-mail address and current location should not be accessible to other users”), and context-aware systems
should be able to protect user’s private information from the illegal accesses according to the policy.

3. Practical Context-aware Scenarios

In this section, we propose seven context-aware service scenarios. The scenarios are very important for developing a context-aware system, because they can give researchers insight and detailed hints for developing their system; however, according to our survey, most scenarios presented in the previous papers are too simple (e.g. set phone to vibrate in meeting[14]), or they assume too ideal situation. (e.g. an agent knows everything about every users[11][15]) Therefore, we composed seven realistic service scenarios of a context-aware system concerning diverse aspects of context-aware computing and summarize the issues related to each scenario. We believe that our scenarios will be useful for future researches.

3. 1 Iced Coffee Preparation

On a very hot and humid day, Matt is returning to his home by a bus. When 5 minutes left to arrival, his PDA rings and displays the following message: “It is very hot day. Is it okay to prepare a cup of iced coffee at your home? Yes/No” He clicks “Yes” and the coffee pot in Matt’s room starts to make iced coffee. When Matt arrives at his place, U-home terminal attached on wall notifies that the iced coffee is ready.

This scenario shows a functional collaboration between multiple devices. Each device in the scenario serves different role for one service. This scenario also shows that a context-aware system should consider device substitution for services (e.g. If A carries cellular phone instead of a PDA, participating devices could be changed for the same service.)

3. 2 Umbrella Suggestion

One day in the morning, when Matt is about to leave his place, u-home management terminal sounds alarm and displays a message: “According to Forecast.com, there is a 70 percent possibility that it will rain today, the system recommends you prepare your umbrella.” After seeing this message, Matt prepares an umbrella and leave.

This scenario gives an example of a context-aware service triggers a service based on external information. To realize this type of service, the system should be able to utilize information from various external sources such as web sites, web services, etc.

3. 3 Recommendation based on User’s Schedule

Jack made an appointment to have beers with his friends at a bar and added this schedule on his calendar application on his PDA. On the day of the appointment, when Jack is about to drive his car to his work, the system installed in his car PC sounds alarm and displays the message: “According to your schedule, you’re supposed to drink alcohol tonight, the system recommends you not drive, this is the information for buses and trains.....” Jack remembers the appointment and decides to take subway train instead of driving to his work.

Events added to calendar by a user can be used as very useful context data in that they are explicitly provided by the user, and the scenario above shows how a context-aware system can use them. A calendar event consists of date, time, description of the event, and so on. However, it is not easy for a context-aware system to utilize the schedule information of users in that they do not have semantic structures. Providing predefined semantic tags is one candidate solution. If users can easily add semantic tags to the schedule events, the events can be used as useful context information.

3. 4 At a Bus Stop

[Case#1] Jack has just arrived at a bus stop. At the moment, Jack’s cellular phone rings alarm sound with displaying the following message: “Waiting time info: circulation shuttle bus arrives in 10 min. commuter bus arrives in 5 min.”

[Case#2] Jack gets curious how much time he would wait at the bus stop on his way to a bus stop. So, he searches the information on context-aware service application installed on his PDA. When he clicks ‘bus waiting time info’, PDA displays the waiting time information of the nearest bus stop.

Some useful context information such as location of bus keeps changing rapidly, thus context-aware systems should be able to catch up the rapidity of changing information. Case#1 and Case#2 show the different levels of autonomy of similar services. The information is delivered fully autonomously in Case#1, while the same information is delivered in non autonomous way in Case#2. Context-aware systems should support both.

3. 5 Automatic Environment Setup for Better Sleep
<table>
<thead>
<tr>
<th>Context-aware Systems</th>
<th>Derived Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Architecture Style</td>
</tr>
<tr>
<td>System</td>
<td>Performance &amp; Scalability</td>
</tr>
<tr>
<td>Context Models</td>
<td>Utilizing Historical Context Data</td>
</tr>
<tr>
<td>Context Definition</td>
<td>Privacy Protection</td>
</tr>
<tr>
<td>Fundamentals</td>
<td>Device Collaboration</td>
</tr>
<tr>
<td></td>
<td>Device Substitution</td>
</tr>
<tr>
<td></td>
<td>External Context Source</td>
</tr>
<tr>
<td></td>
<td>Utilizing Schedule Data</td>
</tr>
<tr>
<td></td>
<td>Real-time Context Data</td>
</tr>
<tr>
<td></td>
<td>Level of Autonomy</td>
</tr>
<tr>
<td></td>
<td>User feedback</td>
</tr>
<tr>
<td></td>
<td>Ad-hoc Network</td>
</tr>
<tr>
<td></td>
<td>Using Legacy Data as Context</td>
</tr>
</tbody>
</table>

Figure 2: Issues of Context-Aware Systems

When Jin goes on her bed, the system sets the most suitable temperature for sleeping considering various aspects and plays a calm and quiet music. The system keeps logging her sleeping status. When Jin gets up in the morning and checks her cell phone, she sees the message: “Did you sleep well? Yes / No” She could not sleep well, because it was too hot for her, so she selects “No.” The system asks several questions to her and uses the answers as feedbacks for later services.

The scenario shows an example of using feedback. Processing user feedback is important in maintain and enhancing services. The feedback is more important in context-aware services. Context-aware services interact with the daily lives of the users with some degree of autonomy. If there is no proper way to reflect user feedback, the users may be afflicted with useless, sometimes harmful services all around their lives without any means of rejection. The user feedback can be given both implicit and explicit ways. The system should be able to process these various kinds of feedback.

### 3.6 At a Conference Room

Jin is supposed to present her research paper at a conference room. Jin goes into the conference room and starts up the presentation application. At the moment, the brightness of room is being adjusted by the system. Listeners can see the presentation slide that is being presented by Jin, through their PDA, laptop, or other portable devices.

This scenario tells us an example of cooperation among devices nearby. In the scenario, devices only use information which can be achieved from the devices around them. If a system supports devices to make an ad-hoc network that is constructed without a centralized server, then there is no reason to burden the burden of central context server for realizing services like the scenario #6. However, it is still a problem that it is difficult to handle the dynamic of ad-hoc network.

### 3.7 Personalized Department Office

Chan has forgotten applying for “graduation exam”. If he does not apply today, his graduation can be delayed. When Chan comes to lab at 10:00AM and logs in his desktop computer, the monitor displays the following message: “Notification from Computer Science Department: You should apply “graduation exam” by tomorrow. Please visit department office.”

This scenario shows an example of using legacy system’s database as context-data. This service exploits not only the physical sensed information that “Chan is sitting in front of his desktop PC”, but also the virtual information which derived from legacy database. Using both physical context data and virtual context data can produce more various useful services.

### 3.8 Other Issues Derived from Our Scenarios

Terminal selection is one other important issue. Information should be delivered to the most accessible device to user’s context. If the user sits at the desk, the desktop computer is most accessible. If the user carries his PDA, the PDA is most relevant terminal. This functionality is required in various kinds of context-aware services.
Identifying the entities in university campus is also an important issue. Many entities, like shuttle bus, cafeteria, and students and so on, appear in above scenarios. The infrastructure should maintain the model which identifies these entities. It requires some hardware infrastructures like RFID tags or well distributed sensors. Software infrastructures like well defined context model are also required.

The infrastructure should support the development of context-aware services. Context-aware service may require collaboration of multiple devices, or act with more high level of autonomy. Timely delivery can be very important in some services. Identifying the pattern of event is important. The infrastructure for context-aware services should support the development of services considering all these characteristics of context-aware services.

4. Conclusions

In this paper, we illustrated the history of context-aware systems based on our survey and derived technical issues. Additionally, we propose seven elaborately made context-aware service scenarios and discuss related issues. Figure 2. summarizes issues presented in this paper.

Especially, we conclude that protecting privacy and security, achieving performance and scalability, utilizing historical context-data, are at the early stages. Moreover, we need to consider collaboration of multiple devices, utilization of context information from external sources, and exploitation of user feedback.

In conclusion, we believe that this paper can help the future context-aware researchers by giving the bird’s eye view of important issues in the research domain.

5. References


6 Acknowledgements

This research was supported by a grant (07High Tech A01) from Hightech Urban Development Program funded by Ministry of Land, Transportation and Maritime Affairs of Korean government.

This research was supported by the Ministry of Knowledge Economy, Korea, under the Information Technology Research Center support program supervised by the Institute of Information Technology Advancement. (grant number IITA-2008-C1090-0801-003)