A Pragmatic Approach to Realizing Context-Aware Personal Services

Sangkeun Lee, Dongjoo Lee, Seungseok Kang, and Sang-goo Lee
School of Computer Science and Engineering, Seoul National University, Seoul
151-742, Republic of Korea
{liza183, therocks, pyxis81, sglee}@europa.snu.ac.kr

Abstract

As network and computing technologies have improved, and the number of mobile devices has increased, realizing context-aware personal services has become one of the most important issues in pervasive computing. Semantic technology may help acquiring, organizing, and processing context information. However, most approaches thus far have adopted semantic technology with an ideal point of view, and they have failed to produce feasible performance and scalability. We propose a practical semantic technology adoption strategy for context-aware personal services with respect to conventional technologies. Based on our strategy, we present a context-aware personal service framework with acceptable performance and scalability.

1. Introduction

The advancement of network and computing technologies and the proliferation of mobile devices such as PDAs and smart phones have driven a distributed and pervasive computing environment where users are connected to the digital network anywhere and anytime. A context-aware personal service actively and autonomously adapts and provides the most appropriate service or content to users, accurately interpreting the context without much user supervision. To develop a context-aware personal service system, a number of previous approaches have adopted semantic technology for context-aware personal services. However, most of these employed a complex concept hierarchy of upper ontology, mathematical propositions, and reasoning. A strategy aimed at achieving the ideal goal of machines “understanding” user contexts as well as humans do will not produce systems with feasible performance and scalability.

Our objective was to develop a practical semantic technology adoption strategy for context-aware services and present a context-aware personal service framework based on our strategy.

The remainder of this paper is organized as follows. In section 2, we discuss related studies and projects and existing approaches. Section 3 presents our practical semantic technology adoption strategy, and section 4 describes our context-aware personal service framework. Section 5 discusses future work and summarizes our work and conclusions.

2. Related Works

Context Broker Architecture (CoBrA) [1] is an intelligent broker that supports context-aware services and enables agents to manage personal information in smart spaces using semantic web technology [2]. To support context reasoning, CoBrA provides a set of ontologies called CoBrA-Ont, an extension of Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) [3]. Context Ontology (CONON) [4], which organizes an upper ontology and some lower domain-specific ontologies, has been prototyped in Service-oriented Context-aware Middleware (SOCAM) [5]. Enhanced CoCA [6] proposes a hybrid approach that uses semantic ontology and relational schema. The system achieves better scalability via heuristic-based selective loading of relevant context data into the reasoning module.

Many previous approaches developed general context models that support flexible context representation, while others demonstrated the feasibility of application- or domain-dependent context-aware personal services. However, achieving generality and feasibility at the same time has been difficult for context-aware personal services.

3. Practical Semantic Technology Adoption for Context-Aware Personal Services

3.1 Challenges to Semantic Technology Adoption for Context-Aware Personal Services
Most existing approaches represent as much context information as possible in ontology and, although the systems are computationally intensive, most depend on ontology reasoning engines that can only manage low-level inferences and lack performance and scalability. Moreover, they tend to use complex concept hierarchies, such as upper ontologies, mathematical propositions, and reasoning, to construct context information. That is, most previous approaches have attempted to achieve the ideal goal of machines “understanding” user contexts as well as humans do, and have considered the limitations of reasoning systems as problems to be solved in the future. Hence, it has been difficult to develop a working system even with only a moderate amount of semantics.

Furthermore, existing approaches have relied on complex technologies such as description logics, artificial intelligence, and natural language processing, technologies that are very difficult to understand [14, 15], leading to unrealistic systems for end-users. Current existing ontology reasoning systems cannot process the reasoning that is required to provide complex services, because it requires processing high-level inferences involving context information, user profiles, metadata, and so on.

### 3.2 Practical Context-Aware Personal Service Implementation with Semantic Technology

A pragmatic context-aware personal service system cannot currently be built by attempting to substitute semantic technology for all previous technologies.

Therefore, our approach incorporates the advantages of semantic technology without comprising feasibility. In addition, unlike previous attempts, all of which focused on developing new ontology or inference engines, we incorporated already existing tools such as transactional data, relational database technology, and ontology reasoning engines that can process only low degrees of inference. In particular, carefully defining the roles of semantic technology and relational database technology can greatly improve the performance and scalability of context-aware services.

However, relational databases offer other advantages. For example, service providers generally have a large amount of transactional data in legacy databases, which require powerful inference engines to represent each data point in ontology. However, using relational database technology with clean data, well-defined and understood metadata, and explicit representation of relationships between entities at the schema and instance levels, simple inferences can be done via a database query language such as SQL.

Therefore, our approach focuses on data rather than inference. The system first performs a thorough analysis of information requirements against current legacy data. This forms the basis for defining usable formal vocabularies, concept hierarchies that are not too deep, and integrity constraints.

In this way, our approach provides practical application of semantic technology focused on performance and feasibility rather than mathematical completeness or decidability. Table 1 summarizes the differences between conventional approaches and our proposed approach.

<table>
<thead>
<tr>
<th>Conventional Approach</th>
<th>Practical Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus on what we don’t have upper ontology, powerful inference engines</td>
<td>Focus on what we have relational database technology, transactional data, inference engines that have limitations</td>
</tr>
<tr>
<td>Focus on Inference complex concept hierarchies FOL vs. F-Logic vs. DL.</td>
<td>Focus on data simple usable concepts low degree of inference</td>
</tr>
<tr>
<td>Satisfy scientific constraint completeness, decidability</td>
<td>Satisfy business value performance, feasibility</td>
</tr>
</tbody>
</table>

Table 1. Differences between conventional approach and practical approach

### 4. A Context-Aware Personal Service Framework

#### 4.1 Architecture

Our framework for providing useful and feasible context-aware personal services was designed based on the strategy presented in section 3. In this section, we describe the framework. To enable a flexible and easily extensible context-aware personal service, it is necessary to consider a wide range of issues. There should be an infrastructure that allows devices to communicate with each other and share information. Hardware technology related to context sensing and mobile devices should be used. Business-related issues, such as the requirements of service providers, should also be considered. Here, we present a framework focusing on software technology.

As mentioned earlier, both the ontology and relational database approaches have strengths and weaknesses for modeling data [16]. Thus, we separated our framework into two layers, Service Request and Service Processing, and assigned the roles of semantic technology and relational database technology for each layer. By combining the advantages of both approaches, the framework guarantees performance, scalability, and flexibility. In addition, it provides an environment that can use legacy databases as content sources. Figure 1 presents the framework architecture.
In the framework, **Context Manager** controls the abstraction levels of context information based on predefined Context Rules. Such as when sensors sense and collect low-level context information (e.g., GPS and temperature readings), Context Manager aggregates this information and transforms it into high-level context information such as street name, city name, and weather.

**Service Trigger Manager** monitors high-level context information transformed by the Context Manager. A Service Rule specifies what kind of service should be triggered in what context. If Service Trigger Manager identifies a current context match as a predefined Service Rule, then it triggers a service request to proper service provider. **Service Repository** is where a context-aware personal service definition is stored. A service definition includes the information of the service provider and the required context data to be sent to the service provider. A service request message includes the destination service provider and the context data that are needed to process the service.

If a service provider gets a service request message, **Request Processor** performs actual service processes using the input context data from the Service Trigger Manager. It chooses an effective algorithm to provide the appropriate content or service. Request Processor sends the processing results to the **Interface Manager**, which manages all of the interfaces between various sensors and devices, and decides the most appropriate device and display format for the service.

**Log Manager** manages all the logs of changes in context data, context abstraction, service triggering, and so on. Log data can be used in many ways. For instance, Context Manager can track the older context information and rollback the context abstraction results. **Log Mining Manager** mines the log data to get useful information/insight for supporting better context-aware personal services.

As explained above, we separated the framework into Service Request and Service Processing layers. Service Request Layer focuses on context information.

To capture a context and trigger an appropriate service request, the context-aware personal service system should be able to understand the semantics of context. The system controls the abstraction levels of much low-level information and transforms it into manageable numbers of high-level context information. Also, we designed the simple models of context data and rule representation that are expressively powerful enough to be used in real applications. Then, reasoning engines perform inferences with a reasonable amount of context information, thereby increasing performance.

**Service Processing Layer** manages a number of large-sized content databases such as music and movie databases. We assume that the semantics of the content data are relatively less important than the context information and thus use relational database technology. However, simple inferences can still be processed using this method by maintaining well-defined and clean data.

### 4.2 Implementation and Use Case

We implemented the prototype of our framework, named **CAP** in Java using J2SE v1.5. **CAP** uses mysql Ver 14.12 Distrib 5.0.51a for storing context and rule data, and uses XML-RPC protocol for communication. It uses simple representation of rule and context data in XML format. We also implemented a simple reasoning engine and adopted it in our framework. To test our prototype, we’ve constructed several demo context-aware applications. Because hardware devices are not currently ready, we implemented virtual devices that emulate hardware devices such as sensors, PDA, Car PC, and so on, and used them in our demo applications. Below is an example of our implemented demo scenarios.

**Use Case: Gas Station Recommendation Service**

A main actor, Matt, is driving his car to the airport. The system in the car performs the following steps. As soon as the fuel-alert indicator lights, the system matches the current context to predefined **Gas Station Recommendation Service**. Then it triggers **Gas Station Recommendation Service**, and sends the context information, including location, route, time, and user identification, to the proper service provider. After the service provider receives all of the information necessary for processing **Gas Station Recommendation Service**, it picks a ranking algorithm for the service, and finds the best 10 (this number can change) gas stations, using the user profile data that they have and Matt’s context that they received from CAP. Then the service provider sends the list of gas stations to the system, which displays the list on the interface in...
Matt’s car. Matt decides to go to the second gas station on the list, and changes his route.

Although we constructed a demo application that uses large size of Gas Station Data and various context data such as user’s location, we found out that CAP was able to process the requests with acceptable scalability. This is because separating the processing many gas station data from the processing context data supports better performance. Another reason is that we used relatively simple context-data and user rule model, and we didn’t allow complex reasoning in our system. However, with using these simple models, we were able to construct many useful demo applications such as we presented above. In addition, although we presented only one implementation scenario, the framework can also provide various types of services such as context-aware automatic device configuration services or context-aware seamless device switching services.

5. Conclusions and Future Work

We developed a novel approach for realizing a practical and useful context-aware personalized service, and presented a strategy with respect to conventional technologies, such as relational databases. We emphasized well-defined and clean data, rather than inference itself and mathematical constraints.

Our framework is domain independent, easily extensible, and could be used in various types of context-aware personal services, based on our semantic technology adoption strategy. Finally, we illustrated how CAP was used to support the implementation of a demo context-aware personal service with acceptable performance and scalability.

In the future, we plan to add the privacy and security feature to our framework, and research rule mining for more intelligent services. We are currently working on improving the prototype of the framework presented herein, and comparing its performance and scalability to other approaches. We also have a plan to test and evaluate our system using actual hardware devices.

6. References


7. Acknowledgments

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)