Survey and Trend Analysis of Context-Aware Systems

Sangkeun Lee*, Juno Chang**, and Sang-goo Lee*

*School of Computer Science and Engineering, Seoul National University,
Seoul 151-742, Republic of Korea
E-mail: liza183@europa.snu.ac.kr

** Div. of Digital Media, Sangmyung University
Seoul 110-743, Republic of Korea

Abstract

It has been over a decade since the concept of context and context-awareness first introduced in computer science research domain. It seems obvious that the context-awareness is acknowledged as a very important step in ubiquitous computing, and many researchers are working on this topic for realizing it. In this paper, we present a survey of a number of existing context-aware systems. As a result of the survey, we outline a general process in context-aware systems and explain design considerations of context-awawre systems. In addition, we summarize characteristics of representative existing context-aware systems in history and present a trend analysis of the systems. In conclusion, we suggest future work for better context-aware systems.

Key Words: context-awareness, ubiquitous computing, context-aware middleware, service, framework, application

1. Introduction

The advanced network and the wide spread of mobile devices such as PDAs and smart phones make computing paradigm more distributed and pervasive. In this environment, users expect to get useful services and information via their mobile devices; however, because the number of services and the amount of information has been also rapidly increasing, it is difficult and time-consuming for users to request proper services or information by themselves.

A context-aware system is one that actively and autonomously adapts its operation and provides the most appropriate functionalities to users, using the advantage of people's contextual information [1]. Realizing context-aware systems is considered as a solution for the problems presented above [2]. If we had an ideal context-aware system, we would not have to worry about requesting appropriate services or information, because the system would provide what we require using our context without our requesting. Although there have been many researches on context-awareness since early 90s, context-aware systems are

not broadly available [3]. Thus, in this paper, we surveyed a number of existing researches on context-awareness especially focusing on existing context-aware applications and frameworks, and we deeply looked into what considerations needed to be handled to realize pragmatic context-aware systems.

The rest of the paper is organized into following sections. In section 2, we explain a general process in context-aware systems to understand the concepts and functionalities of general context-aware systems and present the other design considerations of context aware systems and related issues. In section 3, we categorize existing context-aware systems into three groups according to their characteristics and explain several representative context-aware systems in each group. In section 4, we compare several representative context-aware systems and analyze the trend. In section 5, we conclude our work and suggest future work for better context-aware systems.

2. General Process in Context-aware Systems

Many context-aware systems have complicated architecture and have many sub-components that are responsible for representation, management, reasoning, and analysis of context information, and they provide their functionalities through the collaboration of the sub-components in the systems. Although there are various types of different context-aware systems, generally, a context-aware system follows four steps to process context-awareness.

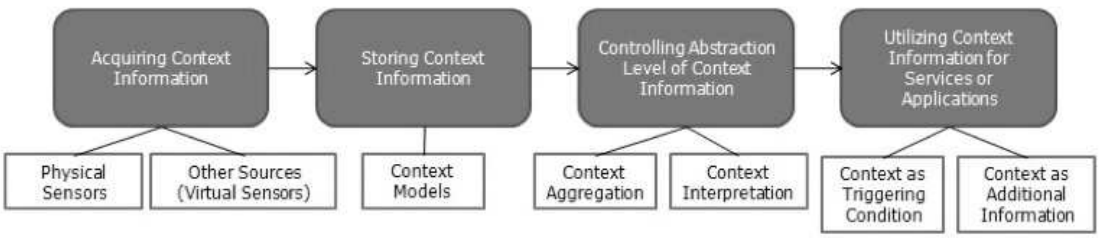

Fig. 1. General Process in Context-Aware Systems.

First step is acquisition of context information. By using physical and virtual sensors, the system can acquire various types of context-aware information. After acquiring context information, the system stores acquired context data into its repository. When storing context data, what kind of data model is used to represent context information is very important. Then, the system controls the abstraction level of context data by interpreting or aggregating context data. Finally, the system utilizes the abstracted context data for context-aware applications in many ways.

In the following sections, we explain how a general context-aware system processes each step in detail and discuss related issues. Figure 1 shows a summary of general process in context-aware system.

2.1 Acquiring Context Information

Because of the diversity of context information types, context information can be acquired in various ways such as using physical sensors, virtual sensors, sensor fusion, and so on. Firstly, physical sensors, which are hardware devices that convert physical analogue properties into computable digital data, can be used for context acquisition. According to the types of context information, many different physical sensors can be used. For example, to attain location information, GPS, Active Badge System, and IR sensors are available. Microphones for audio context and cameras for visual context can be used. Other various sensors such as motion sensors or pressure sensors can be applied in context-aware systems for a wide range of purposes. Baldauf et el. [4] summarized commonly used physical sensor types into the Table 1.

Table 1. Commonly used physical sensor types [4]

Type of Context	Available Sensors
Light	Photodiodes, colour sensors, IR and UV-sensors etc.
Visual context	Various cameras
Audio	Microphones
Motion, acceleration	Mercury switches, angular sensors, accelerometers, motion detectors, magnetic fields
Location	Outdoor: Global Positioning System (GPS), Global System for Mobile Communications(GSM); Indoor: Active Badge system, etc.
Touch	Touch sensors implemented in mobile devices
Temperature	Thermometers
Physical attributes	Biosensors to measure skin resistance, blood pressure

Assuming a context-aware application which recommends a music playlist based on user's preference, current location of users, and the weather condition of the location . In this situation, user's preference can be acquired by analyzing the user's music play log, and the weather conditions of current location can be attained by querying a web service provided by

a forecasting site. We can consider the software modules that perform context acquisition as virtual sensors. Just as physical sensors convert physical properties into context data, virtual sensors analyze data from diverse sources (e.g. user's previous log, Web service) and convert them into context data.

Fusion of multiple physical or virtual sensors is another way of acquiring context information. Sensor fusion can contribute useful extra information [5].

2.2 Storing Context Information

Most of context systems store acquired context data into their repository. Context models are closely related to context storing. Context information can be represented in various ways from simple data model like key-value model to sophisticated data model such as ontological model [6-12]. Many factors such as expressiveness, flexibility, generality, and computational cost depend on what kind of context model is used in the system. Table 2 summarizes a set of example context models.

Table 2. Context models

Context Model	Examples
Key-Value Model	Schilt's Approach, Context Toolkit
Logic-based Model	McCarthy's Approach
Mark-up Scheme Model	CC/PP, UAProf, CSCP, GPM
Object-oriented Model	Hydrogen
Graphical Model	Context extension of ORM, Vector space model
Ontology Model	SOCAM, CoBrA, CASS, CoCA

2.3 Controlling Context Abstraction Level

It is hard for context-aware systems to directly use the raw data provided by sensors. So, context-aware systems translate sensed signals into meaningful data so that they can understand and use context data more easily. Another benefit is that context-aware systems can reduce the number of context data and achieve better performance by controlling the level of context abstraction. If the context abstraction is separated from a context-aware application, the context-aware application does not have to know the details of sensors but

still can use the sensed context data by the sensors.

Many context-aware systems perform context abstraction in two ways – Context aggregation and context interpretation. Context aggregation means that the system aggregates many low-level signals (raw data) into manageable number of high level information. For example, a context-aware system converts thousands of temperature signals into several keywords (e.g. 'hot', 'cold', 'moderate', 'cool', 'warm') by context aggregation. Context interpretation is another method that interprets context information and gives higher level of semantics. For example, a context-aware system can interpret GPS signal into street name.

2.4 Utilizing Context Information for Services or Applications

Utilizing acquired and abstracted context information as useful information for services or applications is the final step of the general context-aware system process. Context information can be used for two purposes – context information as triggering condition and context information as additional information.

Firstly, a context-aware system can use context information as action triggering condition. In this case, the system monitors context status and triggers actions when the context satisfies a specific situation.

Secondly, context information can be used as additional information for services or applications to enhance the quality of service of application. For example, user's current context information can be used as additional information for better results of the search queries. These two kinds of context information usage can be combined together.

There exist many types of context-aware application [15]. Schilit et el. [14] describes four categories of context-aware applications: proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions.

In a similar way, we present our context-aware applications categories below. We extend the personalization and recommendation(suggestion) concepts into Schilt's categories:

(a) Context-Aware Personalization Providing personalized contents or information based on user's current context information (e.g. tour guide service)
(b) Context-aware Suggestion Providing suggestions to users behavior based on user's current situation (e.g. warning dangerous situation)
(c) Automatic Device Configuration Automatically setting up device's configuration according to user's current situation (e.g. screen brightness of PDA)
(d) Context-aware User Interface Optimizing user interface based on user's current context (e.g. emphasize icons that user may select)

2.5 Design Considerations of Context-Aware Systems

When designing context-aware system, we need to consider many aspects of context-aware systems. Context-aware systems can be implemented in many different ways and can have different structure, depending on the development focus of the system. In this section, we discuss several design considerations, which should be considered when designing context-aware systems, such as architecture style, handling dynamicity, privacy protection, and performance & scalability.

Architecture Style Context-aware system's representative architecture styles can be categorized into three - Stand-alone, Distributed, Centralized Architecture. Because each style has its characteristics, advantages and disadvantages, it is very important to choose appropriate architecture style when designing context-aware systems. Many aspects of context-aware system may depend on its architecture style. Figure 2 shows the simplified architecture diagram of each architecture style. We explain characteristics of each style below.

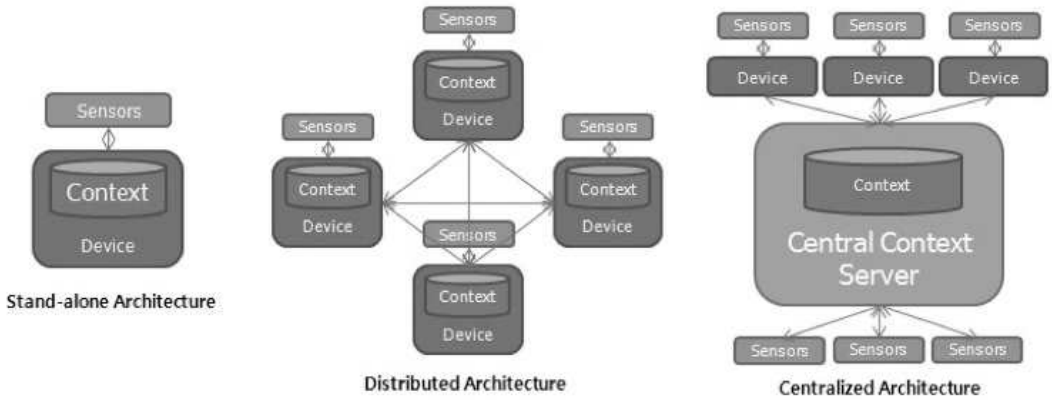

Fig. 2. Architecture styles of context-aware systems

Stand-alone Architecture, which is a basic architecture that directly accesses sensors, does not consider context sharing of devices. This architecture can be relatively easily implemented but has limitations due to the fact that it can't process device collaboration. This architecture is appropriate for small, simple or domain-specific applications.

Distributed Architecture Context-aware systems, which have distributed architecture, can store context information in many separated devices, and there is no additional central server. Each device is independent with other devices, thus, context-aware system can ignore some less important devices that have congestion problem and still continue context-aware

operations. Each device manages its own context information and share context information with other devices by communicating with other devices, as a result, ad-hoc communication protocols are required. However, it is hard for devices to know overall situation of every device when using ad-hoc communication protocols. Usually mobile devices lack resources and computation power, so, distributed architecture are with limitations in dealing with computationally intensive applications.

Centralized Architecture (Context Server) Sensors and devices are connected to a centralized context server that has rich resources and computational power, and context information can be stored in both a central server and client devices. If a device needs to get other device's context information, the device queries the central server and gets the result. In this architecture, every communication is performed by querying the context server, so the communication protocol can be relatively simple than distributed architecture. By using a computationally powerful device as a central server, many applications which require high resources and cost can be processed. However, there is a disadvantage of this approach that it can be critical if the central server fails or congestion occurs.

Handling dynamicity Handling dynamicity is one of important considerations to make a context-aware system possible to process sophisticated context-aware applications. Entities (people, devices, sensors, etc) varying from simple sensors, resource-poor mobile devices to central server with high performance participate to process context-aware applications. At the same time, connections and disconnections of many entities may dynamically occur. A context-aware system should be able to discover and deal with dynamically changing heterogeneous entities and resources.

Privacy Protection Privacy protection is another important consideration to step forward to successful implementation of context-aware systems. Context-aware systems autonomously gather information from the users, so some of the users may feel uncomfortable in that the system can use or open their information without any notice [16]. Lack of appropriate ways privacy protection mechanism is a barrier for realizing practical context-aware systems. Thus, a context-aware system should let users to express their privacy needs. Context-aware systems are responsible for protecting user's context information from illegal accesses.

Performance & Scalability Many operations for context-aware applications have to be processed in real time, and some context-aware systems have need of reasoning and inference functionalities, which require high computational cost and resource, to process intelligent functionalities. However, resource-poor mobile devices mainly participate in context-aware applications in most cases. Thus, context-aware system developer should consider how to manage resources for achieving acceptable performance and scalability. Also communication protocol must scale adequately to deal with a large number of communicating devices.

3. History of Context-Aware Systems

From domain specific context-aware systems or location-based systems to general and extensible systems, there have been a wide variety of context-aware applications and frameworks. To understand how context aware systems have been evolving since they were introduced, we present a review of several existing context-aware systems. We grouped existing context-aware systems into three categories based on their characteristics and the time when they were introduced, and we named each category the 1st, 2nd, 3rd generation of context-aware systems. We explain each generation of context-aware systems in the following sections.

3.1 The 1st Generation of Context-Aware Systems

Since early 90s, many of earlier context-aware system researchers have developed location-based service systems such as The Active Badge System [17], Cyberguide [18], Sumi et el.[19], Cheverst et el.[20], and Cricket Compass [21].

They share common characteristics in that they provide information according to user's current location. In this period, many researchers focused on how to achieve user's location. Many of the 1st generation context-aware systems depend on specific domain such as call forwarding [17], tour-guide [18][19][20], location information providing [21], and shopping assistant [22]. The components in these systems are often integrated into a single domain, so they can be optimized by sacrificing the system's flexibility and generality. Followings are the brief overviews of The Active Badge Location System (a location-aware system) and Cyberguide (a domain specific context-aware application)

The Active Badge Location System

Active Badge Location System is one of the representative location-aware systems. As explained above, the system mainly focuses on how to achieve user's location information. By using this system, the authors implemented a call forwarding application which forwards phone calls to user's nearest phone.

Active Badge Location system uses particularly designed tags, which emit unique identifications for approximately a tenth of a second every 15 seconds. Sensor network installed in a building gathers the signals from the tags that users hold, and a master station polls the sensors and makes context data available to clients. The system supports for predefined operations (e.g. FIND, WITH, LOOK, NOTIFY, HISTORY), and the command interpreter processes the operations requested by clients.

Cyberguide

Cyberguide is a tour-guide application which provides information, which may be useful for users, based on user's current location and time. To achieve user's current location, Cyberguide combines two mechanisms – GPS (for outdoor) and IR sensors (for indoor). The system has graphic user interface, which indicates user's current location on the map, and it keeps the history of places tourist have visited.

3.2 The 2nd Generation of Context-Aware Systems

Although the 1st generation context-aware systems explained in the previous chapter consider only limited types of context and focus on sensing context information, 2nd generation context-aware systems focus on achieving more generality and supporting for various types of context information. In addition, context-aware frameworks that can be used for development of application in many domains started to be introduced.

One of early attempts of framework to support building context-aware applications is Context Toolkit [23] proposed by Dey and Abowd in 2001. Context Toolkit can support various domains of context-aware applications, and it provides context widgets that can be used as reusable software components for accessing and interpreting context data while hiding details.

In 2002, Hofer proposed another early context-aware framework called Hydrogen [24] that can deal with various types of context information. Context Toolkit and Hydrogen both are distributed 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.

Roman's Gaia Project [25][26] in 2003 is different from other prior existing approaches in that it is similar to traditional operating system. It consists of core services and a framework for building context-aware applications and uses 4-ary predicates in DAML+OIL as its context data model.

CASS [27] is another middleware system developed to support context-aware applications in mobile devices. The system aims to provide context data abstraction and separate context based inferences and application behaviors. It uses database table as its context model. We do not explain details of CASS system in this paper due to the fact that the system shares similarities with the other explained systems.

Context Toolkit

The Context Toolkit is one of the early attempts to develop a context-aware framework that aims to develop reusable software components to make development of context aware applications easier, inspired by the success of GUI development kits.

The Context Toolkit has a distributed architecture, but there still needs to be a central discoverer that is used for discovering available widgets, aggregators, interpreters. It uses attribute-value tuple as its context data model, and stores context data and its history. The Context Toolkit provides a basic access control for privacy protection of users by providing context ownership concept.

The Context Toolkit consists of context widgets, interpreters, and aggregators. Context Widgets are distributed sensor units encapsulating the complexity of actual sensors from applications. They can be used as reusable building blocks for context sensing. Context Interpreters interpret context information to abstract low-level sensed information into high-level information to suit the expected needs. Context Aggregators are responsible for composing context information by subscribing to context information provided by Context Widgets and Context Interpreter so that it can hide even more complexity about the context.

Hydrogen

Hydrogen is another open and extensible framework to support context-awareness on mobile devices. Hydrogen is similar to Context Toolkit in that the aim of Hydrogen is also providing reusability of software components. It differentiates between a remote and a local

context and allows context sharing among devices so that sensed context information can be provided for more than the device connected to the sensors. Hydrogen does not support resource discovery and persistent storage functionalities.

Hydrogen has a there layered architecture – Adaptor Layer, Management Layer, Application Layer. All layers are located on one device, and there is no central server. Adaptor Layer is responsible for separating context storing, sensing from other layers and provides the same context information to multiple applications. Management layer performs context sharing with other devices using peer to peer communication via WLAN, Bluetooth.

Hydrogen adapted object-oriented context model as its context model. Since the context model can take advantage of inheritance, various context objects (e.g. Location Object, Device Object) can be supported by inheriting the super-class Context Object.

Gaia

Gaia is an agent based ubiquitous software platform project which is aiming for providing user-centric, resource-aware, multi-device, context-sensitive mobile services in active spaces, extending context-awareness to traditional operating system concept. Gaia coordinates heterogeneous software entities and networked devices.

Gaia represents context data using 4-ary predicates in the format of Context (<Context Type>, <Subject>, <Relater>, <Object>). Context Provider acquires context information from sensors or external sources, and Synthesizer infers high-level context data from low-level context data. The inference is based on predefined first order logic rules. (E.g. Context (Number of people, Room 2401, >, 4) AND Context (Application, PowerPoint, is, Running) => Context (Social Activity, Room 2401, Is, Presentation))

In the Gaia system, Component Management Core manipulates creation, destruction, uploading of components. As Gaia is an extension of traditional operating system to include context-awareness, Gaia provides several kernel services in the system to support context-aware application in active spaces. Especially, Presence Service identifies entities (E.g. application, service, device, and person) present in an active space. Context File System incorporates context into the traditional file system so that Gaia can store context information in repository. The file system attaches context to files and directories, so that important data in current context can be easily retrieved. To adapt to their environment, applications can query or register context information from Context Service. Gaia does not focus on scalability and privacy protection.

3.3 The 3rd Generation of Context-Aware Systems

W3C announced the OWL (Web Ontology Language) in 2003. OWL is a semantic markup language 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, since it's flexible, expressive, and suitable for knowledge sharing. Since OWL is announced by W3C, Many of 3rd generation context-aware systems have adopted ontology as their context-model (e.g. CoBrA [28][29], SOCAM [30][31] , Enhanced CoCA [32][33] , etc. Some other systems in this generation tend to focus on supporting for privacy protection (e.g. CoBrA, Context Fabric [34]) and achieving acceptable scalability and performance of the systems (e.g. Enhanced CoCA, SOCAM)

Chen's CoBrA (Context-Broker Architecture, 2003) is an agent-based context-aware system and the first context-aware system which uses OWL to model ontological context model named SOUPA [29] and privacy policy. In CoBrA system, a centered 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.

SOCAM (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 [31]; however, SOCAM adopted hierarchical approach for designing the context ontology. The ontology used in SOCAM consists of upper ontology and domain-specific ontologies.

The Context Fabric proposed by Hong is 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.

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. In 2008, Ejigu proposed Enhanced CoCA (Collaborative Context-Aware Service Platform) that combines ontology approach and relational database approach [35] to achieve achieves acceptable performance of system.

CoBrA

CoBrA (Context Broker Architecture) is an agent based architecture that enables context-aware services in intelligent spaces. A centralized resource rich context server named Context

Broker manages a shared model of context and controls agents and devices that are participating in context-aware services in the intelligent space. The centralized context broker CoBrA enables knowledge sharing among agents and maintains consistent contextual knowledge and protects the privacy of users. Since CoBrA is a centralized architecture, congestion of the central server may occur; however, CoBrA deals with the problem by broker federation.

CoBrA has four main components – Context Base, Context Inference Engine, Context Acquisition Module, and Privacy Management Module. Context Acquisition Module can acquire context information from various sources like devices, sensors, user behaviors etc. Since the CoBrA uses ontological context model, CoBrA also can cooperate with Semantic Web and use them as external source of context information. Context Base manages ontological context information of entities, and Context Inference Engine infers high level information from existing context information.

CoBrA provides an ontology-based privacy policy language Rei so that it allows users to define their privacy policy to control the sharing of their context information. Privacy Management Module protects user's privacy from illegal accesses.

SOCAM

SOCAM (Service-Oriented Context-Aware Middleware) is another context-aware framework which adapted ontology as its context model. It has a centralized architecture and includes a set of independent services that support for acquiring various contexts from different context providers, interpreting contexts through context reasoning, and delivering contexts.

Context Provider acquires heterogeneous context information from various internal and external sources and convert them into ontological representation. Then, Context Reasoner in the Context Interpreter performs logic reasoning to preprocess context information and maintain consistency of context information.

Context ontologies and past contexts are stored in context database, and Context KB (Knowledge Base) provides a set of APIs so that context-aware applications can query, add, delete, and modify stored context information. In addition, SOCAM has a discovery mechanism called Service Locating Service where context providers and the context interpreter can advertise their presence.

Although SOCAM and CoBrA are very similar in that both use ontological context model and perform context information abstraction and management, SOCAM's context ontology design is different from CoBrA's. SOCAM separated its context ontology into two layers - Common upper ontology & domain-specific ontologies. Common upper ontology represents general concepts for every application, and domain-specific ontology is for representing concepts for sub-domains. Separating the context ontology into two layers rather than using a large-sized ontology can reduce the system's the burden of context processing, and the system can achieve better scalability.

Context Fabric

Context Fabric provides architecture for privacy-sensitive context-aware applications and several customizable privacy mechanisms. It is concerned with protecting user's privacy rather than context sensing and processing. The objective of Context Fabric is to share the right context information, with the right people and services, at the right situation.

Context Fabric manages context information in several infospaces. Infospaces contain context tuples that are basic units of context storage, represented in XML format. A context tuple contains several attributes for context information and privacy tags that expresses how the data should be handled when they flow to other infospaces.

Context Fabric provides a set of privacy related methods and operators that can be used when programming privacy sensitive context-aware applications which are able to control accesses to context tuples in infospaces. A privacy sensitive context-aware application supported by Context Fabric controls the sharing of context information, referring to privacy tags that each context tuple contains.

Enhanced CoCA

In 2008, Ejigu introduced enhanced CoCA (Collaborative Context-Aware Service Platform) based on hybrid context management model. The platform represents context data and performs reasoning, aggregation, and interpretation on context data. CoCA triggers actions and decisions based on the context data and supports neighborhood collaboration for sharing computational resources among devices.

Although processing and storing context information costs high resources, mobile devices have limited computational resources. Also, current ontology engines cannot process computationally intensive high-level inferences, thus, computationally expensive

characteristics of context reasoning process has become a bottleneck for the development of context-aware systems.

To overcome the problem, enhanced CoCA represents context using EHRAM (Entity, Hierarchy, Relation, Axiom, Metadata) model and uses hybrid context management (HCoM) to manage context data. It uses semantic ontology approach to manage context semantics and uses relational database approach to manage context data. It processes them separately and combines the results together for better reasoning and decision support. Additionally, enhanced CoCA uses heuristics to load only relevant data to minimize the size of the reasoning space so that it can scale better.

CoCA provides a neighborhood collaboration method so that mobile devices can share their computational resources to perform expensive processes. It adapted JXTA protocols as a supporting technology to enable mobile devices to advertise and discover their resources in the CoCA service platform.

4. Comparison and Trend Analysis

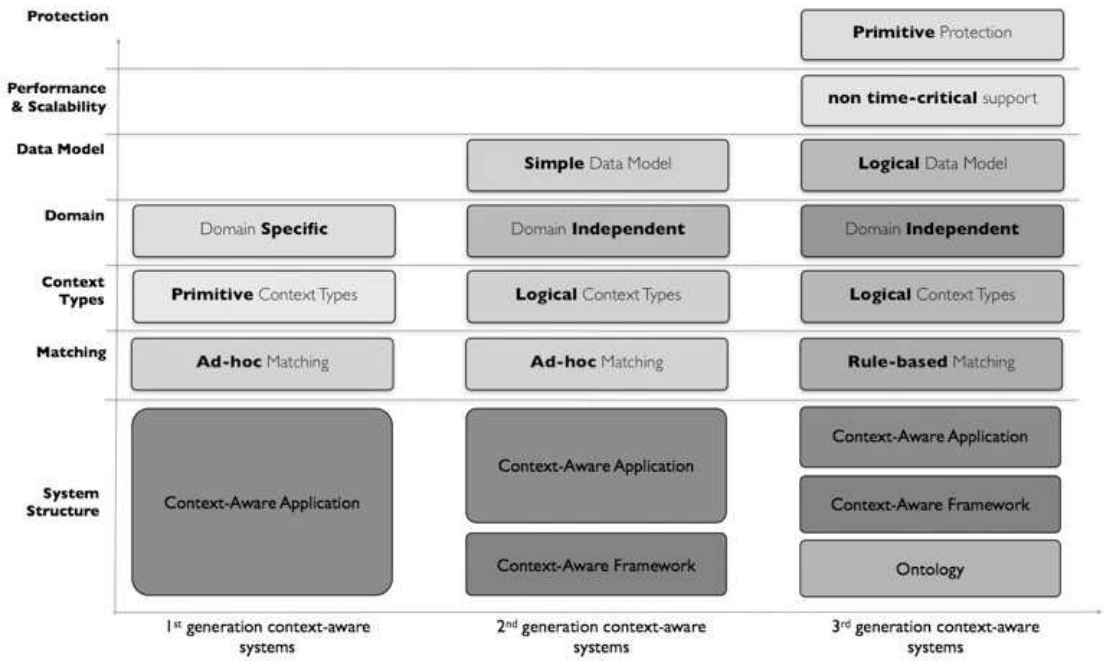

Fig. 3. Trend Summary

In Figure 3, we summarized the trend of the discussed context-aware systems in the previous sections. Most of the systems in the 1st generation focus on domain-specific application (e.g. Cyberguide) that mainly use location of users (e.g. The Active Badge System), but later systems in the 2nd and 3rd generation are frameworks that can support for

implementing a wide range of context-aware applications in many domains. They tend to be more general than the 1st generation context aware systems in that they can process more various types of context information acquired not only from sensors but also other external sources.

Simple context-data models are used in the earlier systems. Context Toolkit uses attribute-value model, and Gaia uses 4-ary predicates for representing context information. In the later generation, many of 3rd generation of context-aware systems have adopted more logical models such as semantic markup language OWL so that they can insure better formality and flexibility.

Using ontological context model has another advantage that it allows to take advantage of ontology reasoning engines that can be used for context abstraction, inference, rule-based matching. CoBrA, SOCAM, CoCA use ontological context model and take advantage of the reasoning engines. However, other systems like Context Toolkit that use simple context model requires a lot more programming effort to enable intelligent context processing. Most systems offer encapsulation of sensors to hide details of context processing.

Most systems in 1st and 2nd generation lack privacy protection module. Context Toolkit can manage and control the context ownership, but it's limited due to the fact that user's privacy needs require more than just context ownership. Currently, CoBrA has the most sophisticated method for protecting user's privacy by providing a privacy policy language named Rei. Users can express their privacy policy in Rei. The Context Fabric mainly focuses on proving architecture for privacy sensitive context-aware applications. It offers a set of privacy sensitive operations and tags that can be attached to context tuples. However, many privacy and security protection related problems are still remaining.

Another important consideration often disregarded in earlier generations is performance and scalability. In the 1st and 2nd generation, many systems lack consideration for achieving better performance and scalability; however, several systems in the 3rd generation of context-aware systems started to focus on performance and scalability. SOCAM offers separation of upper and domain ontology to release the burden of context processing, and CoCA provides a hybrid approach of ontology and relational databases. The comparison of context-aware systems and frameworks explained in this paper is depicted in the Table 3.

Table 2. Comparison Table

	System Name	Architecture Style	Context Acquiring	Context Model	Context Repository	Context Abstraction	Handling Dynamicity	Privacy Protection	Performance & Scalability
1st generation	The Active Badge Location System	Stand-alone Architecture	Active Badge	-	-	-	-	-	-
	Cyberguide	Stand-alone Architecture	GPS, IR	-	-	-	-	-	-
2nd generation	Context Toolkit	Distributed Architecture	Context Widget	Attribute-Value	-	Context Interpreter, Context Aggregator	Central Discoverer	Context Ownership	-
	Hydrogen	Distributed Architecture	Adaptor Layer	Object-oriented Model	-	-	-	-	-
	Gaia	Distributed Architecture	Context Provider	First Order Logic4-ary predicate	Context File System	Synthesizer Context Provider	Presence Service	-	-
3rd generation	CoBrA	Centralized Architecture	Context Acquisition Module	Ontology Model	Context Knowledge Base	Context Inference Engine	Context Broker	*Rei*: privacy policy language	-
	SOCAM	Centralized Architecture	Context Provider	Ontology Model	Context Database	Context Interpreter	Context Service Locating Service	-	Upper/Domain-specific Ontology
	The Context Fabric	Distributed Architecture	Infospace	XML Context Tuple	Infospace	-	Infospace	Privacy Tag	-
	CoCA	Distributed Architecture	Data Source Module	Ontology Model	Data Modeling & Storage Layer	Pre-processing Layer	Collaboration Layer	-	Hybrid Approach – Ontology & RDBMS

5. Conclusion and Outlook

To better understand how context-aware systems have evolved, we categorized the systems into three generations and analyzed the trend. We found out that most 1st generation context-aware systems focused on implementing location-aware domain-specific applications, on the other hand, 2nd and 3rd generation context-aware systems have their similarity in that they achieved more generality and flexibility. They support various domains and more types of context information. Recently, ontological context model is widely adapted in many systems, because it provides formality and intelligent processing methods. Protecting user's private context information and achieving performance and scalability have become important issues in recent systems.

Context-aware systems require intelligent processing such as reasoning and inferences for many reasons for context abstraction or consistency management of context information, so, semantic technology and intelligent processing is becoming important more and more as time passes. It seems obvious that ontological context model and reasoning engines are useful tools for context-awareness. However, we should be acknowledged that current ontology reasoning engines and semantic technology have limitations to perform resource intensive processes, thus, balancing performance, scalability and intelligence is very important to realize practical context-aware systems. Future context-aware systems should be able to achieve acceptable performance and scalability, not giving up too much intelligence.

Based on our study on previous researches, we found out that most of existing researches are only concerned with providing sets of operation for context-aware applications. SOCAM and CoCA supports for building context-aware services; however, they are still at the early stages.

A context-aware service is not just an application that uses context information as input. It can be defined as cooperation and interaction of applications and entities for a specific objective. A future context-aware system should offer easier and more intuitive ways for service providers to add new context-aware services that they want to provide to users and for users to subscribe services that they want to use. A service description language and an interpreter that interprets service descriptions may be a candidate solution. Historical context information and logs in service level should be managed by the system so that it can use them as feedbacks for better service quality, although they are disregarded in most existing systems.

Many current context-aware systems lack of consideration on utilizing virtual sensors mentioned in the section 2.1 and only focus on utilizing physical sensors, although virtual sensors can be also very useful context information source. We conclude that utilizing many already available data such as logs, user profiles, and so on can be a good support for limitation of physical sensors.

In conclusion, we believe that a service centric context-aware system, which can process higher level of services rather than just several context-aware applications, with acceptable performance, scalability and reliable privacy protection mechanisms will be a key driver for realizing practical and useful context-aware systems in the future.

6. Acknowledgement

This research was supported by Seoul R&BD Program(WR080951).

References

[1] G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith, and P. Steggles (1999) 'Towards a Better Understanding of Context and Context-Awareness', Handheld and Ubiquitous Computing, Springer-Verlag, pp. 304-307.

[2] Coutaz, J., Crowley, J. L., Dobson, S., and Garlan, D. (2005) 'Context is key', Communications of the ACM

[3] Chen, G. and Kotz, D. (2000) 'A Survey of Context-Aware Mobile Computing Research', Technical Report. UMI Order Number: TR2000-381., Dartmouth College

[4] M. Baldauf, S. Dustdar, F. Rosenberg (2007) 'A Survey on Context-Aware Systems', International Journal of Ad Hoc and Ubiquitous Computing, Vol.2, Number 4, pp.263-277.

[5] A. Schmidt, M. Beigl, H.W. Gellersen (1999) 'There is More to Context Than Location', Computers & Graphics

[6] T. Strang, C. Linnhoff-Popien (2004) 'A Context Modeling Survey', *Workshop on Advanced Context Modeling, Reasoning and Management, UbiComp*

[7] J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen (2003) 'Experiences in Using CC/PP in Context-Aware Systems', M.-S. Chen et al. (Eds.): Mobile Data Management 2003, LNCS 2574, pp. 247–261.

[8] F. Fuchs, I. Hochstatter, M. Krause, M. Berger (2005) 'A Metamodel Approach to Context Information', *PERCOMW*

[9] M. Melucci (2005) 'Context Modeling and Discovery using Vector Space Bases', *Proceedings of the 14th ACM international conference on Information and Knowledge Management*

[10] XH. Wang, DQ. Zhang, T. Gu, HK. Pung (2004) 'Ontology Based Context Modeling and Reasoning using OWL', *Pervasive Computing and Communications Workshops*

[11] H. Chen, F. Perich, T. Finin, A. Joshi (2004) 'SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications', Mobile and Ubiquitous Systems: Networking and Services

[12] T. Strang, C. Linnhoff-Popien, K. Frank (2003) 'CoOL A Context Ontology Language', LNCS 2893

[13] E. Christopoulou, A. Kameas (2005) 'GAS Ontology: An ontology for collaboration among ubiquitous computing devices', International Journal of Human-Computer Studies

[14] B. Schilit, MM. Theimer (1997) 'Context-Aware Applications - From the Laboratory to the Marketplace', Personal Communications, IEEE

[15] B. Schilit, N. Adams, R. Want (1994) 'Context-aware computing applications', *Workshop on Mobile Computing Systems and Applications*, pp. 85-90 .

[16] S. Lahlou, M. Langheinrich, C. Röcker (2005) 'Privacy and Trust Issues with Invisible Computers', Communications of the ACM, Vol. 48 March, pp. 59-60.

[17] R. Want, A. Hopper, V. Falco and J. Gibbons (1992) 'The Active Badge location system', ACM Transactions on Information Systems, 10(1):91-102.

[18] GD. Abowd, CG. Atkeson, J. Hong, S. Long, R. Kooper (1997) 'Cyberguide: A mobile context-aware tour guide', Wireless Networks

[19] Sumi, Y., Etani, T., Fels, S., Simonet, N., Kobayashi, K. and Mase, K. (1998) 'C-map: Building a context-aware mobile assistant for exhibition tours', Community Computing and Support Systems, Social Interaction in Networked Communities, Springer-Verlag, London, UK, pp.137–154.

[20] Cheverst, K., Davies, N., Mitchell, K., Friday, A. and Efs-tratiou, C. (2000) 'Developing a context-aware electronic tourist guide: some issues and experiences', *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM Press, New York, USA, pp.17–24.

[21] N. Priyantha, A. Miu, H. Balakrishnan, S. Teller (2001) 'The Cricket Compass for Context-Aware Mobile Applications', *International conference on Mobile computing and networking*

[22] A. Asthana, M. Cravatts, and P. Krzyzanowski (1994) 'An indoor wireless system for personalized shopping assistance', *In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pages 69-74.

[23] D. Salber, A. K. Dey, G. D. Abowd (1999) 'The Context Toolkit: Aiding the Development of Context-Aware Applications', *Proc. Conf. Human Factors in Computing Systems*

[24] Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J., and Retschitzegger, W. (2003) 'Context-Awareness on Mobile Devices - the Hydrogen Approach', *In Proceedings of Hawaii International Conference on System Sciences (Hicss'03)*

[25] R. Manuel, K. Christopher (2002) 'Gaia: A Middleware Infrastructure to Enable Active Spaces', IEEE Pervasive Computing

[26] A. Ranganathan, R.H. Campbell (2003) 'A Middleware for Context-Aware Agents in Ubiquitous Computing Environments', *Proceedings of ACM/IFIP/USENIX International Middleware* , Vol. 2672

[27] P. Fahy, S. Clarke (2004) 'CASS - Middleware for Mobile Context-Aware Applications', *Workshop on Context Awareness, MobiSys*

[28] Chen, T. Finin, A. Joshi (2003) 'An Intelligent Broker Architecture for Context-Aware Systems', *Adjunct Proceedings of Ubicomp*

[29] Chen, F. Perich, T. Finin, A. Joshi (2004) 'SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications', Mobile and Ubiquitous Systems: Networking and Services

[30] Gu, H. Pung, D. Zhang (2005) 'A service-oriented middleware for building context-aware services', Journal of Network and Computer Applications

[31] X. Wang, D. Zhang, T. Gu, H. Pung (2004) 'Ontology Based Context Modeling and Reasoning using OWL', *Pervasive Computing and Communications Workshops*

[32] D. Ejigu, M. Scuturici, L. Brunie (2007), 'CoCA: A Collaborative Context-Aware Service Platform for Pervasive Computing', Information Technology

[33] D. Ejigu, M. Scuturici, L. Brunie (2008) 'Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing', Journal of Computer

[34] Hong, J.I., Landay, J.A (2004) 'An architecture for privacy-sensitive ubiquitous computing', *In: 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys),* Boston

[35] D. Ejigu, M. Scuturici, L. Brunie (2007) 'Semantic Approach to Context Management and Reasoning in Ubiquitous Context-Aware Systems', Digital Information Management

Sangkeun Lee received the B.S. degree in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2005. He is currently a Ph.D. candidate at School of Computer Science and Engineering in Seoul National University, Seoul, Korea. His research interests include context-awareness, semantic technology, life-logging, and recommender systems

Juno Chang received the B.S. degree in Computer Science from Seoul Natioinal University, Seoul, Korea, in 1990, and the M.S. degree in Computer Science from Seoul National University in 1992. He received the Ph.D. in Computer Science from Seoul National University, and has worked for Sangmyung University since 2003. His research interests include digital contents, personalized search, and enterprise softwares.

Sang-goo Lee received the B.S. degree in Computer Science from Seoul National University at Seoul, Korea, in 1985. He also received the M.S. and Ph.D. in Computer Science from Northwestern University at Evanston, Illinois, USA, in 1987 and 1990, respectively. He is currently a professor in School of Computer Science and Engineering at Seoul National University, Seoul, Korea. He has been the director of Center for e-Business Technology at Seoul National University, since 2001. His current research interests include e-Business technology, database systems, product information management, and Ontology.