

# Reconfigurable Middleware for Sensor Based Applications

Peizhao Hu and Jadwiga  
Indulska\*  
School of ITEE  
The University of Queensland  
Brisbane, QLD 4072, Australia  
{peizhao, jaga}@itee.uq.edu.au

Ricky Robinson  
Queensland Research Laboratory  
National ICT Australia Limited  
300 Adelaide Street, Brisbane, QLD 4000,  
Australia  
Ricky.Robinson@nicta.com.au

## ABSTRACT

The pervasive computing paradigm introduced context-aware applications that adapt themselves to their surrounding environment based on context information. Context information may be of different types including sensed context information gathered from a variety of sensors. In pervasive computing user intervention/interactions with their application should be minimised. On the other hand, pervasive computing environments are very dynamic environments - context information that supports adaptation decisions may not always be available due to user mobility and potential network and/or sensor failures. As context information supports autonomous adaptation of applications, the provision of context information itself has to be managed by an autonomic system able to both recognise and recover from context delivery failures. This paper presents the architecture of a context management system that is reconfigurable with regard to sensor and/or network failure and can support dynamic discovery and replacement of sensors. One of the components of this middleware architecture, the model and management of sensor descriptions, is described in more detail as it plays an important role in sensor discovery and replacement.

## Categories and Subject Descriptors

H.3.4 [ Systems and Software]: Distributed systems

## Keywords

context-awareness, pervasive computing, autonomic computing

## 1. INTRODUCTION

In the past, the fields of logic and artificial intelligence created a fundamental framework for computer intelligence

\*The authors are also affiliated with the Queensland Research Laboratory, National ICT Australia Limited, Brisbane, QLD 4000, Australia.

introducing applications which can reason based on both existing knowledge and understanding of situations. In some cases, particularly when dealing with logical tasks, computers perform better than humans, such as in mathematical calculation and reasoning over knowledge base. However, humans are better at adapting to the changing context of their tasks or surrounding environment. Such an adaptation forms the fundamental principle of context-aware computing, in which computing applications/services adapt to the changing computing environment and/or changing user tasks. The adaptation is triggered by evaluation of context information gathered from a variety of sources, including sensing devices. Context-awareness is identified as a milestone towards “invisible computing” or “computer disappearance” [10]. Context information is defined as any information that can be used to characterise the situation of an entity that is relevant to an interaction between user and application, such as person, place, or object [4]. Some approximation of the current application’s context can be captured from user provided information, hardware sensors or monitoring software. Context information is managed by the context management system which gathers and evaluates context information and either responds to applications’ queries about context information or sends applications notifications about context changes. A context management system is a component of a middleware that supports distributed, context-aware applications.

Current applications/services are often distributed over heterogeneous computing devices connected by heterogeneous networks. Such computing environments are prone to node failures or disconnections and this can be exacerbated by user mobility. Computing devices (servers, sensors, etc) or services that participate in the application, can fail due to a number of reasons [2], including low battery power, physical damage, network disconnections or Quality of Service (QoS) problems, and Byzantine failures brought by premeditated attacks on the application.

Existing approaches to context-aware applications mainly focus on how context information is gathered and processed to support application adaptation - problems of dynamic changes in the environment like user mobility and node and communication failures that affect context information gathering, have not been fully addressed. Frequent interactions between human and computer for configuring, tuning and maintenance are infeasible in such systems. As the environment/situation changes in such computing systems are quite frequent, it is essential that context management systems are also context-aware and self-adapting. Therefore, there

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MDS '06, November 27-December 1, 2006 Melbourne, Australia  
Copyright 2006 ACM 1-59593-418-9/06/11 ...\$5.00.

is a need to develop advanced and fault tolerant context management systems that ensure availability and reliability of their services.

Advanced approaches to designing context-aware applications use high level, abstract models of context information with no direct binding to particular sensors. For example, a context model may show that a particular application uses location information, however the model does not specify what kind of location sensor produced the location reading and whether the information had to be processed before being delivered to the context management system. On the other hand, a disconnection from a context source or its failure requires dynamic, run time matching of an abstract model of context information to a potential source of this information. For such models, it is necessary to develop a context management system that is able to map, at run time, the high abstraction of context information into new sources of this context information.

In this paper, we address the challenges of developing fault tolerant context management systems. The emphasis is on the architecture of the context management system and on developing an appropriate model of sensor descriptions. The model has to support (i) dynamic discovery of sensors, (ii) composition of components processing sensor information in order to produce high level abstraction of context information, and (iii) mapping from high level abstraction of context information to particular sensors. The structure of the paper is as follows. Section 2 discusses various design challenges for developing adaptive context management systems. Our approach to describing sensors, partially based on emerging standards, is presented in Section 3, while Section 4 presents our loosely coupled architecture of the adaptive and reconfigurable context management system. Section 5 reviews the existing works on reconfiguration of context management systems. In Section 6, we conclude the paper and discuss our future work on reconfigurable middleware for context-aware applications.

## 2. DESIGN CRITERIA

In this section we describe requirements and research challenges of extending middleware with reconfigurable context management systems.

### 2.1 Application Level Context Models

Context models provide a basis for developing adaptive, context-aware applications[6, 9]. High level abstractions of context information used in such models allow sharing context information between many applications and allow runtime replacement of sensors by sensors providing the same sensing phenomena. For example, location information can be obtained from a variety of hardware and software sensors. Representing location information in the context model without tightly binding it to a particular sensor, allows replacement of the sensor at runtime. On the other hand, the lack of information on the source of context information has to be compensated by the context management system when the current source of location information fails at run time. Dynamic reconfiguration of the current set of sensors used by a particular context-aware application requires sensor characteristics and capabilities information, as well as information regarding their current situation (their current context like location and connectivity). It is therefore necessary to investigate suitable sensor description mod-

els.

### 2.2 Association & Interoperation

In pervasive environments, a variety of devices differing in terms of their types, processing power, screen size, etc; may cooperate as part of a distributed application. Physical and computational differences create bootstrapping problems for system integration and deployment. In addition, behaviour of computing devices and computer networks are quite unpredictable in such systems. Devices may leave or fail and network links may experience QoS problems or fail which creates association problems. Context management systems are required to handle these problems, preferably without user intervention, if they affect context information delivery. The problems can be divided into communication bootstrapping, discovery of devices/sensors within the system administrative domain, their appropriate configuration, and establishment of communication for accessing or providing their services. The communication bootstrapping problem can be resolved by incorporating the same communication medium on devices, however, this introduces complexity to large system deployment (particularly on resource constrained devices that have limited communication interfaces and resources) and restricts hardware devices that could be used in pervasive systems. Another challenging issue is that pervasive computing may involve computing devices from a variety of domains. Mobile devices can move across administrative domains. Therefore a solution that scales to large systems with a number of domains can improve the reconfiguration performance.

### 2.3 Ubiquity

The distributed nature of both context sources and context-aware applications requires a decentralised context management system with some degree of redundancy built into the system. This would prevent single point of failure and allow discovery of information from redundant components. Distribution and redundancy of high level context models and context information can support discovery and dynamic association of context information sources.

## 3. SENSOR DESCRIPTION

Existing sensor based applications were mostly built for specific sets of sensors, and replacing them after their failures requires manual reconfiguration according to sensor specifications. This is because existing systems are not able to perform the tasks of obtaining and interpreting sensor specifications automatically without user intervention. In pervasive computing, where situations change rapidly, complete sensor descriptions and their runtime information (e.g., context information about sensors including their location and connectivity) can support runtime discovery and dynamic replacement of sensors supporting context-aware applications.

Designing middleware for sensor-based applications brings a number of challenges[3]: (1)*Integration of idiosyncratic sensors* - sensors used in context-aware computing are not typical in their construction and programming interfaces and require additional information about sensor capabilities. For example, an active keyboard of a desktop can provide information about the user's location and can therefore be used as one type of location sensor. This would require some description of this particular functionality in the sen-

sensor description. (2) *Abstracting from sensor data* - abstraction of context information is required by applications to avoid tight coupling with specific individual sensors; and (3) *Sensor outputs may need to be further processed* - e.g., sensor data may require fusion and/or interpretation before it can be used by context-aware applications. In this section, we present our solutions for developing comprehensive sensor descriptions. To meet the above requirements our approach combines two emerging sensor description standards and adds a sensor context model that captures sensor runtime information with regard to the environment.

### 3.1 Sensor Description Standards

To support interoperability and reconfiguration of sensor-based applications that use a variety of sensors over heterogeneous networks, the sensor descriptions should be based, to the extent possible, on standard descriptions of sensors. There are two emerging sensor description standards which can greatly support sensor discovery and also dynamic matching of context information to sources of this information.

One of these standards is IEEE 1451 which comprises a family of open standards defining common interfaces for transducers (sensors and actuators) through which they can be accessed and queried about their functionality. Along with network independence supported by the Network Capable Applications Processors (NCAP), TEDS (Transducer Electronic Data Sheet) is a key element for enabling sensor discovery and interoperability. Among others within the standard family, IEEE 1451.4 further extends the idea of TEDS by defining encoding/decoding template for each type of transducer, and at the same time allows manufacturers to create customised templates based on the standard command sets in order to meet their growing product range. The template encoding structure supports comprehensive description of transducers which can be stored in a memory efficient manner and therefore it is suitable for memory constrained devices. Transducer's configuration information is encoded in a compact form according to assigned templates, and stored either in Electronic Erasable Programmable ROM or in a Virtual TEDS<sup>1</sup> file. This allows a variety of devices/sensors such as microphones, accelerometers, etc. to be dynamically discovered and automatically configured for use.

The limitation of TEDS is that it describes the basic sensor functionality (hardware, sensor calibration, sensing phenomenon, quality of sensor reading) but it cannot capture all the additional descriptions needed by logical/virtual sensors. TEDS is also not able to describe the processing of sensor data (e.g., fusion, interpretation) that sometimes has to be applied to change the data into the type of context information required by context-aware applications. These two additional requirements are addressed by the Sensor Web Enablement initiative which is an ongoing activity carried out by the Open Geospatial Consortium (OGC)<sup>2</sup> with the aim of allowing sensors and sensor data to become discoverable and accessible in real time over the Web. A collection of sensor properties for reconfiguration is modelled in the process metadata group, which includes *sml:identification*, *sml:classification*, *sml:capabilities* and *sml:characteristics*. In a similar way to the IEEE 1451 TEDS, *sml:identification* and *sml:classification* are used to

assist sensor discovery functionality. *sml:identification* provides information such as the manufacturer ID, model number and serial number, while *sml:classification* provides information concerning sensor types, applications, and supported sensing phenomena. *sml:capabilities* and *sml:characteristics* provide detailed technical specifications which can be used for autonomic sensor reconfiguration. This comprehensive sensor information modelled in SensorML description is represented in the XML format.

As future sensors equipped with either embedded or virtual TEDS descriptions according to the IEEE 1451 standard, in the future, the SensorML description related to sensor identification, classification and calibration can be automatically generated from the TEDS descriptions discovered from sensors. SensorML can also model detailed sensor information as processes with inputs, outputs, parameters and methods. SensorML processes can be composed into process chains in order to model the required processing. For a system to use idiosyncratic sensors with extraordinary phenomenon types, there is a sensor classification definition that describes the correlation of sensor types and corresponding phenomenon types. In this way, idiosyncratic sensors can be defined virtually by making use of simple sensors together with data processing. Obvious examples are cameras and desktop computers located in the same office which can be utilised as location sensors when appropriate software components are used. There already exist approaches that specify such preprocessing of sensor data as a composition of interfaces of the processing components, but proprietary composition specification languages are used for this purpose, such as *iQL* (as described in the related work). However using SensorML for this purpose increases interoperability.

In our approach we use the integration of TEDS and SensorML to provide a description of physical and virtual sensors that supports discovery of new sensors and/or matching of high level context information to sensors when sensor replacement is necessary. Our current implementation is able to decode and extract sensor descriptions from either virtual or embedded TEDS and use them to automatically generate complete SensorML descriptions as mentioned earlier. The implementation is developed in Java with SensorML description framework support. In addition, we use SensorML as a means to describe composition of processes which process/interpret sensor data to change it to the required context information.

### 3.2 Sensor Context Model

Applications adapt their behaviour autonomously based on context information sensed from a variety of sensors in their operating environment. Sensors, however, are also in a dynamic environment: sensors may be mobile, their connectivity may change, their energy level may change, they can fail, etc. This is runtime information that describes a current situation of a particular sensor in the environment. In other words, it is context-information about sensors. Neither the previously described standards nor the composition of interfaces (e.g. *iQL*) are suitable tools to capture this dynamically changing information. This information is of the same kind as context information used by context-aware applications and has to be gathered, evaluated and managed by the context management systems. We therefore use the same modelling approach which is used to model context

<sup>1</sup><http://www.ni.com/teds>

<sup>2</sup><http://www.opengeospatial.org>

information of context-aware applications to model context information of sensors.

Figure 1 shows a sensor context model in Context Modelling Language(CML)[6]. CML is an extension of Object Role Modelling (ORM), in which associations between entities are modelled as fact types (e.g., sensor has power). ORM was extended to capture types of context information (static, profiled, sensed, derived), temporal fact types, dependencies between types of information, and quality meta-data. Detailed explanations of CML are presented in [6].As shown in Figure 1, there are a few static fact types that link the *sensor* to the sensor description in SensorML and also show its sensing phenomena; the model also captures information about whether a sensor is an instance of the Fixed or Mobile sensor class. Other fact types represent runtime context information (i.e., location, power, connectivity).

The explanation of the types of information captured by the sensor context model is as follows. Devices/sensors may provide sensing that is different from their default functionalities, and this is captured in the model as a sensing phenomenon of the sensor/device. In addition, we model location information differently for fixed and mobile sensors. For fixed sensors, location information is modelled as a profiled fact, while for mobile sensors, the snapshot-sensed fact type is used. The snapshot-sensed fact type is used for modelling mobile sensor location because applications are interested in device location at a particular time. In real world scenarios, mobility of sensors may result in ambiguous location information (e.g., several, possibly conflicting, readings of location information). A quality metric, certainty, is assigned to the location information of a mobile sensor to support resolution of ambiguous information. Security and privacy control is supported in our current design - as shown in the sensor model, an ownership relation is modelled to support discovery of malicious behaviour.

Matching of context information to a new source of the information is divided in two steps. The first step provides matching of the context information with the sensor context models to find sensors which can provide the required sensing phenomenon and meet other matching constraints like location and power level. The second step involves retrieving SensorML descriptions of the selected sensors for a detailed evaluation of which of the preselected sensors is the best match with the required context information. *SensorML description* in the sensor context model is used in the second step of matching to identify the SensorML description of the sensor/device. This two step process can accommodate searching through both static and dynamic sensor information and also increases search performance as the number of SensorML description matchings is reduced.

#### 4. RECONFIGURATION ARCHITECTURE

In this section, we present an architecture of context-aware systems that includes a dependable context management system capable of dynamically finding and replacing sources of context information supporting context-aware applications. Figure 2 illustrates this architecture as composed from three layers:

- context-aware applications which consume context information and base their adaptation decisions on the information;
- a reconfigurable context management system at the

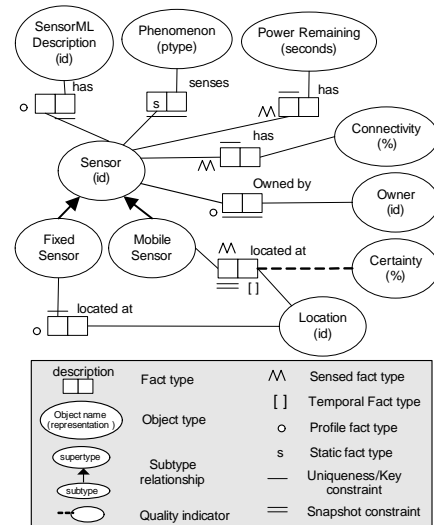


Figure 1: An example of Sensor Context Model

middleware layer, built from several components, that stores and evaluates context information according to the context models of context-aware applications and disseminates this information to the applications through responses to queries and/or through notifications about context changes; and,

- a context sensing layer which includes context sources (sensors) which produce context data; this layer may include additional processing components of context information (if required). The latter may be needed to change the raw sensed data into the type of context information required by the applications.

The loose coupling of context information sources and context-aware applications eliminates the burden of context information gathering and processing by resource constrained devices and also supports reuse of context information. In addition, it hides context information access and low-level sensor operations from applications, and therefore allows dynamic reconfiguration of the set of sensors providing context information to the applications.

The middleware layer has several components which can be classified in two groups:

- components which support context-aware applications (Context Model repository, Context Facts repository); and,
- components which support reconfiguration of the set of sensors providing context information for particular applications (Reconfiguration manager, Context Source manager, Sensor Description Repository).

In this section we describe the functionality of the Context Source manager and the Reconfiguration manager.

#### 4.1 Context Sources manager

The Context Sources manager is responsible for handling interactions with context information sources in the context management system, including reconfiguration of the interactions if context delivery failures happen. It cooperates

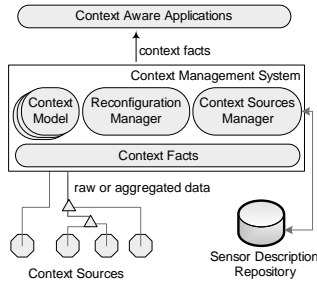


Figure 2: Reconfiguration Architecture

with an external component, the sensor description repository, to provide services for managing context information sources for multiple context-aware applications. The Context Source manager is built from the following components:

**Sensor Discovery service:** Sensors must have the same communication medium (i.e., USB, Infra, Bluetooth, WIFI, etc.) as the context management system in order to become discoverable. We exploit sensor widgets for various communication interfaces operating at the boundary of the system to bootstrap physical interactions with sensors. This is a typical approach in other existing solutions. Once communication is established, sensors self-introduce themselves to the system and provide their TEDS description in encoded format, then the context management system decodes/extracts TEDS information according to the assigned template. The Context Sources manager verifies security and privacy rights according to the sensor’s TEDS description. Using the TEDS user data field, reliable security mechanisms, such as public key authentication, can be supported.

**Common Sensor Knowledge Base:** The Common Sensor Knowledge Base (CSKB) is a repository of sensing rules, which are defined by domain experts in order to provide intelligence to the system for accurate configuration and derivation of advanced sensing capabilities of sensors. When new sensors are discovered, CSKB is consulted to determine any advanced sensing capabilities they can achieve, possibly with additional data processing.

**SensorML Engine:** The SensorML engine is a dedicated service for creating/accessing SensorML descriptions in the context management system. The Context Sources manager supplies the SensorML engine with the TEDS description and a description of advanced sensing processes in order to create the SensorML description of a particular sensor. Additional information about the sensor (e.g., location, environment factors) is also described in the SensorML description to support sensor configuration.

## 4.2 Reconfiguration manager

The role of the Reconfiguration manager is to monitor and reconfigure/redefine mappings between context fact types and context sources, and to perform dynamic reconfiguration of information sources for context-aware applications. We outline three services that the Reconfiguration manager provides as part of the reconfiguration process.

**Mappings Monitoring service:** Current mappings of context sources and context facts are actively monitored in order to detect context delivery failures. Monitoring is done depending on the context fact type specification and the context source type. Proactive monitoring is used for frequently

used/updated context facts.

**Context Source Matching service:** Once broken mappings are detected, discovering appropriate context sources is a fundamental step towards reconfiguration. This is a two step process as described in 3.2. Matching of context sources uses matching restrictions in the form of “SELECT *phenomena* FROM *locations* [WHERE {*conditions*}]”, where corresponding content of each field is modelled in the sensor context model. These restrictions need to be created at the design time of context-aware applications. In the cases of context-aware applications demanding specific sensors, detailed specification requirements can be described as *conditions* in the restrictions.

**Mapping Reconfiguration service:** When the best matching sensor is selected the next step is to define how data sensed from the sensor will be fed into the repository of context information (i.e., into a context fact). To ensure interoperability this mapping is based on the OGC Observation and Measurement standard for sensor observations and it has been discussed in our previous paper[7].

When the reconfiguration of the context source finishes, the Reconfiguration manager updates the list of current mappings between context fact types and context sources and starts to monitor the newly acquired sensors.

## 5. RELATED WORK

There exist numerous approaches to developing software infrastructures and programming toolkits which can be used by applications to obtain context information from sensors. Dey et al’s Context Toolkit[5] is the most well known solution, which wraps each sensor with a widget and allows the composition of aggregators and interpreters that process sensor data. This approach hides complex and physical sensor operations from the context-aware application. However, a crucial shortcoming of this solution is that it creates a tight coupling between applications, sensors, and intermediaries responsible for interpreting low-level sensor data. There is no provision for discovering sensors at run time and reconfiguring the processing. Moreover the context information is not shared across applications.

Other examples of advanced approaches to processing and managing context information are Solar[1], the context service together with context mediation[8] from the IBM autonomic computing initiative, and fault-tolerant approaches derived from the Gaia project. Solar is a middleware infrastructure that allows developers to specify the derivation of context information in the form of operator graphs, which include context information sources, sinks and channels. In this model, sensors are represented as data sources, and applications as data sinks. At run-time, operator graphs are instantiated and managed by the Solar platform. Within Solar, the complex context aggregation process is decomposed into a series of modular and re-useable operators, which can significantly reduce application overhead of managing the same or similar sensor data aggregation processes. Nevertheless, sensor configuration has to be specified in advance in the form of an operator graph, rather than providing real-time and dynamic sources discovery and reconfiguration. In addition, replacement of sensors with different specifications is not supported, even with some degrees of transformation applied.

From the system architecture perspective, IBM’s context service acts as a context repository for managing the context

of various applications. It also handles interactions with actual context sources to provide the infrastructure level context collection and dissemination. Details of low-level sensor specific operations are encapsulated within the *Dispatcher*, which is a set of individual *Context Drivers* for different context sources. Applications request context information from context sources through an interface provided by each context driver (or through a composition of interfaces) according to the context aggregation and interpretation process. A template is used to specify metadata of context information such as quality of information (QoI), location, ownership and so on. The context service masks the heterogeneity of context sources, while context mediation handles context filtering, correlation and also rebinding of context sources. Application designers can specify the requirements of the context sources in the form of a *composer specification* using a language, called *iQL*. A composer specification is similar to the one most critical systems use for verifying system correctness, in which systems declare expected types for all inputs, outputs and arguments. These solutions combined together provide a reconfigurable infrastructure for context-aware applications. However, the approach is not based on an application level context model and therefore does not support sharing of context information nor evolution of context-aware applications. Application level context models provide an abstraction which relieves context-aware applications from a concern about how and where to gather context information, and in addition supports distributed management of context information. In addition, as discussed in this paper, adopting emerging sensor standards will support interoperability. Furthermore, with the availability of physical sensor descriptions, more realistic and robust real-time discovery and replacement of context sources are possible.

The Gaia project [2] employs redundancy of hardware and software components to provide fault tolerance of the Gaia context-aware system. This is a typical approach to achieve system fault tolerance and is fully applicable in Gaia, as this project is concerned with context-aware intelligent spaces for collaborative work. Such systems have a reasonably static infrastructure and are not affected to a large degree by mobility of users and multiple-domains. More dynamic systems require discovery of potential new context sources which can replace failed or disconnected sensors. The discovery and association of new sensors has to be carried out at run time.

## 6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a middleware architecture consisting of a context management system that is reconfigurable with regard to sensor and/or network failures. This architecture is suitable for context-aware applications that are built on high level abstraction models of context information. As sensor descriptions play an important role in the reconfiguration process, we have discussed in detail our approach to describing sensors. Our approach leverages emerging sensor description standards (TEDS and SensorML) and the CML language for context information modelling to produce a comprehensive description of static and dynamic features of sensors. Such a description supports: (i) run time discovery of static and mobile sensors, (ii) run time matching of high level context information to a sensor that can provide such information, (iii) dynamic composition of components providing a preprocessing of context informa-

tion, and (iv) run time monitoring of delivery of context information between sensors and the context management system.

We are currently refining the design of the described architecture and also building a prototype of the reconfigurable context management system based on the models and sensor descriptions presented in the paper. In addition to this work we plan to address other issues which can enhance the functionality of the reconfigurable context management system. This includes defining situations and preferences for the monitoring process to trigger dynamic reconfiguration of context sources.

## 7. ACKNOWLEDGEMENTS

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology, and the Arts; the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs; and the Queensland Government

## 8. REFERENCES

- [1] G. Chen, M. Li, and D. Kotz. Design and implementation of a large-scale context fusion network. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 246–255, 22–26 Aug. 2004.
- [2] S. Chetan, A. Ranganathan, and R. Campbell. Towards fault tolerance pervasive computing. *Technology and Society Magazine, IEEE*, 24(1):38–44, Spring 2005.
- [3] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed System: Concepts and Design*. Addison Wesley, 2005.
- [4] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. In *Proceedings of CHIA '00 workshop on Context-Awareness*, 2000.
- [5] A. K. Dey, D. Salber, and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2-4):97–166, 2001.
- [6] K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 77–86, 2004.
- [7] J. Indulska, K. Henricksen, and P. Hu. Towards a standards-based autonomic context management system. In *3rd International Conference on Autonomic and Trusted Computing (ATC) 06'*, 2006.
- [8] H. Lei. Context awareness: a practitioner's perspective. In *Ubiquitous Data Management, 2005. UDM 2005. International Workshop on*, pages 43–52, 4 April 2005.
- [9] T. Strang and C. Linnhoff-Popien. A context modelling survey. In *First International Workshop on Advanced Context Modelling, Reasoning And Management UbiComp 2004*, Nottingham, England, 2004.
- [10] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–10, September 1991.