DISSERTATION

Quality of Context in Pervasive Systems: Models, Techniques, and Applications

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften

unter der Leitung von

Prof. Dr. Schahram Dustdar
Distributed Systems Group
Institut für Informationssysteme
Technische Universität Wien

und

Prof. Dr. Do van Thanh
Department of Telematics
Norwegian University of Science and Technology

eingereicht an der

Technischen Universität Wien
Fakultät für Informatik

von

Atif Manzoor
Matrikelnummer: 0627638
Brigittenauer Lnde 224-228 /6539
A-1200 Wien, Österreich

Wien, November 2010

_____

Kurzfassung

Pervasive Computing Systeme zielen darauf ab, den Menschen bei der Durchfhrung alltglicher Aktivitten auf natrliche Weise zu untersttzen. Dieser Ansatz beruht auf der Verwendung von Rechenleistung und der Verarbeitung von Kontextinformationen zur flexiblen Anpassung seiner Umwelt. Diese Rechenleistung und Kontextinformationen werden von einer Vielzahl von Sensoren und Aktuatoren zur Verfgung gestellt, welche in dauerhafter kommunikativer Verbindung mit den Menschen und ihrer Umwelt stehen, weit ber konventionelle Mensch-Maschinen Interaktionen hinaus. Diese Gerte erfassen die menschliche Umwelt oft implizit, bewerten die aktuelle Situation und beliefern Anwender mit Kontextinformationen und Dienstleistungen welche besonders auf ihre Bedrfnisse zugeschnitten sind. Eine der zentralen Herausforderungen, denen Software-Ingenieure gegenberstehen um die Vision der Pervasive Systeme zu realisieren, ist mit der unzureichenden Qualitt von Kontextinformationen umzugehen.

Um dieses Problem zu adressieren, fhren wir das Konzept der Kontextqualitt (Quality of Context, QoC) ein. Diese QoC beschreibt den Wert von Kontextinformationen fr einen spezifischen Anwender. Darauf aufbauend prsentieren wir ein Modell zu Verarbeitung von QoC Metriken und beschreiben wie diese berechnet und kombiniert werden, um das Vertrauen in Kontextinformationen beziffern zu knnen. Dieser Vorgang berücksichtigt die Anforderungen bestimmter Anwender betreffend QoC. Das Vertrauen in Kontext beschreibt die Qualitt von Kontextinformationen unter Bercksichtigung verschiedenster Aspekte mit einer einzelnen Metrik welche auf die Anforderungen eines bestimmten Anwenders zugeschnitten ist. Wir prsentieren auch Techniken zur Auflsung von QoC Konfliktsituationen in verschiedenen Schichten eines kontextbezogenen Pervasive Computing Systems. Diese Techniken knnen auf einer einzigen QoC Metrik, dem Vertrauen, basieren, welches aus einer Kombination von zwei oder mehr QoC Metriken abgeleitet wird.

Wir verwenden die oben genannten Modelle und Techniken um verschiedene Anwendungen in kontextbezogenen Pervasive Computing Systemen zu entwickeln. Unsere Experimente belegen die Wirksamkeit unseres Ansatzes der QoC Metriken, um verschiedenste Funktionen in solchen Systemen zu realisieren. Wir vertreten die Meinung, dass Konfliktlsungstechniken, die das Vertrauen in Kontext basierend auf der Kombination verschiedenster QoC Metriken verwenden, unter Bercksichtigung der Art der Aufgabe wirksamer sind, als Konfliktlsungstechniken die nur auf einer einzelnen QoC Metrik beruhen. Unsere Experimente zeigen auch, dass das Vertrauen in Kontext

verwendet werden kann, um den Wert und die Relevanz von Kontextinformationen fr einen spezifischen Anwender erfolgreich zu beziffern.

Abstract

Pervasive computing systems aim to facilitate human beings in carrying out their usual activities in a natural way by dynamically adapting the situation to their requirements using computing power and context information available in the environment. This computing power and context information is provided with the assistance of plethora of sensing and actuating devices continuously communicating with the human beings and their environment beyond the conventional means of man-machine interaction. These devices implicitly sense the human environment, comprehend the current situation, and furnish consumers with context information and services especially tailored to meet their needs. One of the core challenges that software engineers face to achieve the vision of pervasive systems is to deal with the inadequate quality of context information.

To address this problem we introduce a novel approach to perceive Quality of Context as the worth of context information for the specific context consumers and present our model to process and use QoC metrics. We describe and evaluate QoC metrics and present a new technique to combine different QoC metrics to infer the value of confidence on context considering the requirements of a particular context consumer regarding quality of context information. Confidence on context presents the quality of context information from different aspects by a single metric tailored to the needs of a specific context consumer. We also present QoC based conflict resolving techniques to resolve the conflicting situations at different layers of a context-aware pervasive computing systems. These techniques can be based on a single QoC metric or confidence on context, inferred from a combination of two or more QoC metrics.

We use the aforementioned models and techniques to develop different applications in context-aware pervasive computing systems. Our experiments demonstrate the effectiveness of our approach to use QoC metrics in performing different functionality in such systems. We find that conflict resolving techniques that use confidence on context based on the combination of different QoC metrics selected considering the nature of task are more effective than conflict resolving techniques using a single QoC metric. Our experiments also demonstrate that confidence on context can be used to successfully depict the worth and relevance of context information for a specific context consumer.

Acknowledgements

A number of people helped me while writing this dissertation, both directly and indirectly. First, I would like to thank my advisor Prof. Schahram Dustdar for giving me an opportunity to undertake the research work for this dissertation at the Distributed Systems Group. I would also like to thank him for his continuous support, encouragement, and inspiration throughout the period of my Ph.D. candidature.

I would like to thank my examiner Prof. Do van Thanh for his valuable comments to improve this dissertation. I would particularly like to acknowledge the support of Hong-Linh Truong for his collaboration with me during my Ph.D. candidature and providing me with his feedback on my research. His suggestions helped me a lot to make many improvements to this dissertation.

I would like to thank my colleagues, Lukasz Juszczyk, Shariq Bashir, Christoph Dorn, Ahmed Kamran, and Harald Psaier, who worked with me in carrying out different research tasks. I am grateful to the EU project Opportunity consortium, especially Prof. Gerhard Trster, Daniel Roggen, Claudia Villalonga, and Alberto Calatroni, for providing me the data set to conduct the experiments presented in this dissertation.

I would like to thank all other colleagues at Distributed System Group, especially Daniel Schall and Martin Treiber, and all my fellow Pakistani scholars in Vienna for all the laughter, advice, and information that they shared with me throughout the period of my Ph.D. candidature. I am also thankful to Florian Skopik for helping me in the translation of the abstract of this dissertation to the German language.

I would like to express my deepest gratitude to my family for their endless love, encouragement, and support that was always a motivational force for me to work on this dissertation. I would also like to thank them for all the compromises they made during my stay in Austria. Last but not the least, I would like to thank Higher Education Commission Pakistan for the financial support to work on this dissertation.

Atif Manzoor
November, 2010
Vienna, Austria

Dedication

To my parents,

For their belief in me that inspired me to come as far as I have

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Mark Weiser while describing his vision of the computers of 21st century stated that they will disappear in the background by weaving themselves into the fabric of the everyday living and facilitate the human users by pervasively providing the computing power, information, and other services specifically tailored to their needs (Weiser, 1991). Recent advancements in the miniaturization of computing devices (e.g., wearable sensors embedded in the clothes), wireless communications capabilities, artificial intelligence techniques and algorithms, distributed system technologies, and proliferation of smart phones contributed to the progress towards Mark Weiser´s vision (Carlos and Paul, 2007). Consequently, many pervasive systems are developed to assist human users, such as easy living environments for physically and cognitively impaired persons (Aarts and Encarnao, 2008), remotely providing health care services to chronic patients (Bricon-Souf and Newman, 2007), and adaptive disaster response systems (Catarci, de Leoni, Marrella, Mecella, Salvatore, Vetere, Dustdar, Juszczyk, Manzoor, and Truong, 2008).

Context-awareness is the core element of any pervasive system and can be viewed as the process of sensing the environment, formulating sensor data as the knowledge about the current situation in the environment, reasoning with the available knowledge of the current situation considering different possible actions and system objectives, and fi-

nally taking actions to adapt the environment to fulfill the desired objectives (Baldauf, Dustdar, and Rosenberg, 2007; Cook and Das, 2007). However, quality of context (i.e., the quality of the knowledge about the current situation in the environment) may deteriorate before it is made available to context-aware applications and contrary to general assumption context information may be incomplete, inaccurate, and ambiguous (Henricksen, Indulska, and Rakotonirainy, 2002; Dey, Abowd, and Salber, 2001). Context-aware applications also has to perform extra effort to deal with the uncertainty of context information (Ranganathan, Al-Muhtadi, and Campbell, 2004). Inadequate quality of context information not only puts extra burden on the context-aware applications but may also result in unwanted actions that possibly will discomfort the human users.

Quality of Context (QoC) metrics that indicate the quality of context information from different aspects, such as the capability of sensing devices to collect sensor data, the precision and accuracy of the context processing algorithms to extract high level context, and the worth of context to perform the functionality of a specific application, may help to solve the problem of uncertainty and vagueness of context information. Context-aware applications may improve their performance if they are provided with usable QoC metrics that are evaluated considering their requirements regarding the collection, processing, and provision of context information. This dissertation presents a novel approach to process, evaluate, and provide QoC metrics tailored to meet the requirements of a specific context-aware application. We also introduce a technique to combine different QoC metrics to infer the value of confidence on context considering the requirements of a particular context consumer regarding the quality of context information. Furthermore, we present QoC based conflict resolving policies to resolve conflicts that may occur while performing different tasks in a context-aware system. Our experiments carried out in the domain of smart home demonstrate the effectiveness of our approach.

1.2 Motivation

Currently, the focus of the research and development in pervasive systems is on improving functionality, while reducing the size, cost, and power requirements of sensing and actuating devices but an equally important and difficult set of challenges revolves around context management (Franklin, 2001). To accomplish the vision of pervasive environments context information also needs to be of high quality. Traditional approaches to evaluate and manage context quality need to be adjusted to the requirements of pervasive systems. There may be many conflicting situations that a context-aware system can face while performing different tasks. QoC metrics, the confidence on context, and QoC-based conflict resolving policies may help to tackle those situations without human intervention, the most scare resource in pervasive systems. Here we present some of such situations that provide the motivation of this work.

Sensor Selection: Pervasive environments are characterized with the plethora of sensing devices that differ with each other considering the frequency of updates, the capability of a sensor to collect the context of an entity, the accuracy of a method that is used by sensors, representation format, and the cost of collecting that data (Cook, 2007; Chantzara, Anagnostou, and Sykas, 2006). Problems arise due to the mobility of sensors, computing devices, and entities in pervasive environments. This situation implies that a single sensor may not provide the highest quality data about a specific entity in the environment. For example, we cannot use the ambient sensors installed in the kitchen area to collect data about a user who is sitting in the living room. Considering such circumstances, there is a need of a strategy at the sensing layer that can dynamically select the best sensor to provide a specific piece of data at a certain instance of time. Evaluating and comparing QoC metrics for data gather by a specific sensor enable to select a sensor best appropriate to collect data about a particular event in the environment.

Context Distribution: Sensor data is provided to the algorithms to extract context information at different levels of abstraction, e.g., human activity can be classified as *"sitting"* at the lower level of abstraction and *"taking dinner"* at the higher level

of abstraction. Context is further provided to the context management layer to distribute it to different context-aware applications to perform their functionality accordingly. Context-aware applications may not be interested in every value of context generated at the lower level and may like to receive a context object having information worthwhile for their functionality. For example, Appliances Control (AC) and Ambiance Management (AM) are two typical context-aware applications deployed in a smart home to control actuators. The task of AC is the proactive anticipation of the future use of appliances in the home kitchen and switch them on or off to support the user. When a user wants to start cooking, the AC should switch on and preheat the oven. The AC also switches appliances off when they are no longer in use to save power and avoid any injuries. The AC should switch off the oven when no cooking activity is currently detected or expected to happen in near future. The AM controls the home ambiance by tuning temperature, light, and background music. For example, the AM decreases luminosity and the volume of background music when the user is relaxing. Both of the applications heavily depend on the users' current activity to effectively perform their functionality. It will be convenient for the applications with optimal use of resources if they notified only when a users' activity relevant to their functionality is performed. QoC metrics evaluated according to the worth of a specific application can help to select context relevant for a specific application and enhance their efficiency.

QoC Metrics Processing: QoC metrics indicate the quality of context information from different aspects by different metrics, such as reliability, timeliness, completeness, and significance. Context objects presenting the same context information may have high or low quality considering different quality dimensions. Different applications may also have different quality requirements regarding different quality dimensions. The decision to select a single context object can only be made combining all those QoC metrics as a single metric according to the needs of a specific application. Simple strategies to combine QoC metrics, such as taking average of all QoC metrics, may present insufficient information. Situation demands a strategy that not only effectively combine all QoC metrics but also provide a simple mechanism for context consumer applications to give

their input in this process. Inference of confidence on context using fuzzy logic meritoriously deal with such circumstances considering all the requirements.

1.3 Contributions

The main contributions of this thesis are:

Novel QoC Metrics and Their Evaluation Methods: To better perceive the quality of context information we extend the existing list of the QoC metrics by introducing the QoC metrics presenting worth of context information for a specific context consumer. We have also developed the techniques to evaluate QoC metrics from sensor characteristics, measurement context, and context consumer requirements.

A Novel Technique to Infer The Confidence on Context: Confidence on context present the quality of context information from different aspects by a single metric tailored to the needs of a specific context consumer. In this thesis we present a novel technique to combine different Quality of Context (QoC) metrics to infer the value of confidence on context. Our technique also considers the requirements of a particular context consumer regarding QoC metrics while confidence inference. Confidence on context is further provided to the context consumers to select high quality context and use the confidence in their functionality.

QoC Based Conflict Resolving Techniques: We present QoC based policies resolving techniques to resolve the conflicting situations at different layers of a context-aware pervasive computing system. These policies can be based on a combination of one or more QoC metrics. We also illustrate how these policies can be used to resolve different conflicting situations in context-aware pervasive computing systems.

Applications Using QoC: We also used the aforementioned models and techniques to develop different different applications in context-aware pervasive computing

systems. Our applications demonstrate the effectiveness of our approach to use QoC metrics in performing different functionality in such systems.

1.4 Publications

Quality of Context: We presented the state of the art of the existing work on Quality of Context in pervasive environments and their comparison with our work in a book chapter published in The Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications (Manzoor, Truong, Malik, and Dustdar, 2010). We presented the QoC metrics and their evaluation algorithms in Third European Conference on Smart Sensing and Context (EUROSSC 2008) (Manzoor, Truong, and Dustdar, 2008). Preliminary work on QoC metrics developed into a new definition of QoC and QoC processing model that are discussed in an article published in Knowledge Engineering Review Journal (Manzoor, Truong, and Dustdar, 2010). We discuss these issues in detail in Chapter 2, Chapter 3 and Chapter 4.

Confidence on Context: We presented the concept of Confidence on Context in The Fifth International International Symposium on Web and Mobile Information Systems (WAMIS 2009)(Manzoor, Truong, and Dustdar, 2009a). We also devised a context inference system based on fuzzy logic presented in 2010 Asia Pacific Services Computing Conference (APSCC 2010). We presented the details of issues related to the inference of confidence on context in Chapter 5.

QoC Based Conflict Resolving Techniques Different QoC based policies that can be used to resolve conflicts at different stages of a context-aware pervasive computing system are presented in The First International Workshop on Quality of Context (QuaCon 2009) (Manzoor, Truong, and Dustdar, 2009b). We discuss conflicting situations in context-aware pervasive computing systems and QoC based conflict resolving policies in Chapter 6.

QoC Based Applications: We present a context aggregation system that have used QoC to remove inconsistent, duplicate, and conflicting information in The Fifth

International Symposium on Web and Mobile Information Systems (WAMIS 2009)(Manzoor, Truong, and Dustdar, 2009a). We discuss and solve the design time conflicts occurring while selecting sensor data from different sensing modalities to extract a particular type of context in The Fifth European Conference on Smart Sensing and Context (EUROSSC 2010)(Manzoor, Villalonga, Calatroni, Truong, Roggen, Dustdar, and Tröster, 2010). These applications are presented in Chapter 7

Other publications that are not directly included in the thesis but discuss different aspects of context-aware pervasive computing systems are:

- Truong H.-L., Juszczyk L., Manzoor A., Dustdar S. (2007). ESCAPE - An Adaptive Framework for Managing and Providing Context Information in Emergency Situations . 2nd European Conference on Smart Sensing and Context (EuroSSC'07), 23. - 25. October 2007, Kendal, Lake District, England.

- Catarci T., de Leoni M., Marrella A., Mecella M., Vetere G., Salvatore B., Dustdar S., Juszczyk L., Manzoor A., Truong H.-L. (2008). Pervasive and Peer-to-Peer Software Environments for Supporting Disaster Responses. IEEE Internet Computing, January 2008.

- Truong H.-L., Juszczyk L., Bashir S., Manzoor A., Dustdar S. (2008). Vimoware - a Toolkit for Mobile Web Services and Collaborative Computing . Special session on Software Architecture for Pervasive Systems, the 34th EUROMICRO Conference on Software Engineering and Advanced Applications, 3. - 5. September 2008, Parma, Italy.

- Truong H. -L., Manzoor A., Dustdar S. (2009). On Modeling, Collecting and Utilizing Context Information for Disaster Responses in Pervasive Environments. Workshop on Context-Aware Software Technology and Applications (CASTA 2009), August 24, 2009 Co-located with ESEC/FSE 2009, Amsterdam, The Netherlands.

- Juszczyk L., Psaier H., Manzoor A., Dustdar S. (2009). Adaptive Query Routing on Distributed Context - The COSINE Framework . International Workshop on

the Role of Services, Ontologies, and Context in Mobile Environments (ROSOC-M). 10th International Conference on Mobile Data Management (MDM'09), 18. - 20. May 2009, Taipeh, Taiwan.

- Ahmad Kamran Malik, Atif Manzoor, and Schahram Dustdar. Context-Aware Privacy and Sharing Control in Collaborative Mobile Applications. (Accepted for Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications, ISBN: 978-1615206551, to be published by IGI Global on May 2010).

1.5 Thesis Outline

This thesis is structured as follows: Chapter 2 presents background information on pervasive environments, context-aware computing, a and historical perspective on the issue of the deficiency of quality of context information in such computing systems. Furthermore, it also presents state of the art of the existing work on QoC in pervasive computing systems. Chapter 3 summarizes the problem statement, discuss the limitations of the existing research works, and present our novel approach to solve the problem.

Chapters 4 to 6 present the main contribution of this thesis. Chapter 4 presents our view of QoC as the worth of context for a specific application. It also discusses QoC metrics that are evaluated later in the chapter. It also present a context model that incorporates the QoC metrics.

Chapter 5 presents our confidence inference system that combines the QoC metrics and infer the value of confidence tailored to the needs of a specific context consumer. Confidence on context presents the quality of a context information from different aspects for a specific context consumer.

Chapter 6 discusses the conceptual layers of a context-aware computing system and different conflicting situations that can occur on those layers. Later this chapter presents our QoC based conflict resolving policies that can be used to resolve those conflicts.

Chapter 7 presents the smart home data set that we used in our experiments. It also presents our experiments preformed to evaluate our approach to use QoC and confidence on context based policies to resolve conflicts in context-aware pervasive computing systems. It also presents the results of our experiments. Finally, we discuss our results and conclude our thesis in Chapter 8.

Chapter 2

Background and Related Work

2.1 Overview

In this chapter, we present the background knowledge of the thesis and discuss the related work. We briefly discuss the existing definition of context and building blocks of a context-aware system in pervasive computing environment. We also discuss different techniques, tools, and technologies that have been used to design and develop those building blocks. Later in the chapter we present the historical perspective of the problem of imperfection of context information in pervasive computing systems and different approaches that have been used to solve this problem. We also present a comparison of these approaches to our approach.

2.2 Background

2.2.1 Context

The American Heritage Dictionary of the English Language defined context as "the part of a text or statement that surrounds a particular word or passage and determines its meanings" or "the circumstances in which an events occurs; a setting". Lately context is used in a wide range of applications in different computing domains, such

as natural language processing, databases, communications, electronic documentation, and vision to solve a multitude of problems. (Brézillon, 1999). Consequently, many definitions of context considering the requirements of a specific domain appeared in research literature (Bazire and Brzillon, 2005). One of the most widely used definition of context from the prospect of pervasive computing systems is given by Dey and Abowd (Dey and Abowd, 1999) as

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves."

Schilit, Adams, and Want (1994) categorized computing environments context as user context, computing context, and physical context. Chen and Kotz (2000) have enhanced the list by adding temporal aspects of the context. All aspects of information considered in the aforementioned categories of context are usually not related to perform the functionality of a specific application. The definition of context given by Dey and Abowd (1999) implies that context is limited to information that is relevant for a specific interaction between a user and an application. For example, a user context includes her current location, identity, activity, mood, and who is around her. But for a security control application that automatically opens the door of a building for a person considering her rights to enter the building is only concerned with the identity of the user. So for that particular application context is only the identity of the approaching user. Considering this we can assert that context is any information that is relevant to perform the functionality of an application according to prevailing situation to facilitate a user.

2.2.2 Context-Aware Systems

Context-awareness is the capability of the system to sense the continuously emerging situation and to adapt its services according to the changed circumstances without any direct intervention from the user of the system (Harter, Hopper, Steggles, Ward, and Webster, 2002). Context and context-awareness are the essential building blocks of a pervasive computing system (Chen, Finn, and Joshi, 2003). Since the publication of the pioneer paper of Weiser (Weiser, 1991) on pervasive systems, a lot of research

efforts have been undertaken to design and develop context-aware pervasive computing systems (Chen and Kotz, 2000; Baldauf, Dustdar, and Rosenberg, 2007; Yi Hong, ho Suh, and Kim, 2009). Different techniques and approaches, depending upon the special requirements of a system, nature of sensing devices, number of system users, or the facility of further extension of system, have been used to design and build such systems.

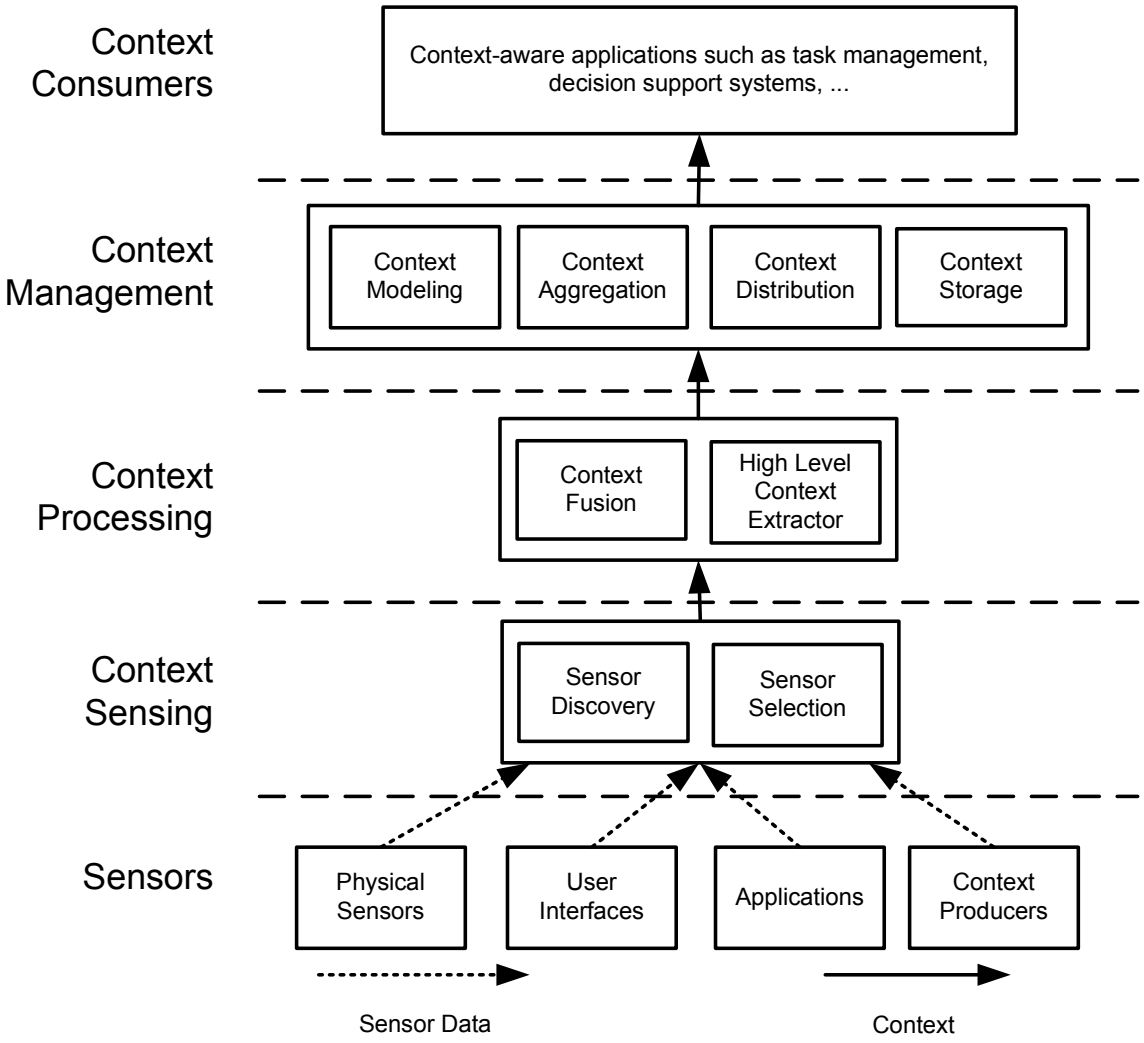

Figure 2.1: Conceptual layers of a context-aware system

Chen (Chen, 2004) presented direct access to sensors, using a middleware framework, and acquiring context from a context server as the three different alternatives

to collect context information for a context-aware systems. However, designing and developing middleware solutions to work as intermediatory between context producers and context consumers became more popular. Recent advancements in the software engineering techniques, such as Separation of Concerns (Hursch and Lopes, 1995), also contributed to rise the number of context-aware systems developed as a layered middleware framework with context-aware applications lying at the top of stack. Figure 2.1 shows the different conceptual layers of a generic context-aware system for collecting processing, managing, and using context information and the building blocks of those layers as depicted in different designs (Baldauf, Dustdar, and Rosenberg, 2007). Here we will give a brief introduction of some of these components that are related to this thesis.

2.2.2.1 Sensors

An important aspect of context is that it is not provided as an explicit input to the computing systems (Gray and Salber, 2001). Sensors — weaved into the fabric of daily living — gathering data in pervasive environments constitute the first layer of a context-aware system. Sensors in pervasive environments are not limited to physical or hardware sensors but also include virtual sensors as shown in the sensing layer of Figure 2.1. Physical sensors consist of hardware based sensing modalities, such as ambient sensors (van Kasteren, Noulas, Englebienne, and Kröse, 2008; Logan, Healey, Philipose, Tapia, and Intille, 2007), vision based sensors (Lepri, Mana, Cappelletti, Pianesi, and Zancanaro, Lepri et al.; Maekawa, Yanagisawa, Kishino, Ishiguro, Kamei, Sakurai, and Okadome, 2010), environmental sensors and wearable sensors (Zappi, Lombriser, Stiefmeier, Farella, Roggen, Benini, and Tröster, 2008). These sensors collect raw sensor data from the environment.

Virtual sensors are software based sensors and are embedded in different applications to collect useful data. For example, the current location of a human user not only be extracted from data collected by the physical or hardware sensors but also be extracted from data collected by the virtual sensors by browsing her electronic calendar, a travel-booking systems, emails etc. Virtual sensors also include the software modules or other nodes available in the environment that process the raw sensor data from the physical sensors and extract high level context information, e.g., a classification

algorithm process the wearable sensors acceleration data to extract that human user is walking. Data collected by physical and virtual sensors is processed by the higher layers and is provided to context consumer applications to adapt the environment according to the current situation in the environment.

2.2.2.2 Context Sensing

The context sensing layer is responsible for the retrieval of raw sensor data and high level context information available at other nodes in the environment. There may be redundant sensors available in the environment that provide same information. These sensors may also have different capabilities to observe the environment regarding their range, precision, resolution, and accuracy. These characteristic may affect the quality of the context extracted from sensor data. One of the tasks of the context sensing layer is to select most suitable sensor available to collect data about a specific event in the environment.

Proliferation of wireless capability in pervasive computing devices is also continuously increasing the mobility of sensors (Patwardhan, Perich, Joshi, Finin, and Yesha, 2006). Many of these sensors — carried by humans or installed on mobile devices — continuously change their location. These sensors are not only limited to physical sensing systems, such as human wearable health monitoring sensors (Jovanov, Milenkovic, Otto, and de Groen, 2005), but also include virtual sensors, such as software sensors embedded in a user's mobile device.

Mobility of the sensing systems implies that a single sensor may not always be available to provide data. Similarly it also implies that a single sensor may also not be believed to provide highest quality data about a certain entity in the environment. For example, a flood rescue worker providing information about the current flood situation near an important bridge in the city may not be the best choice to collect information, once she moved away from the bridge. Considering this situation, it is also one of the important tasks of the context sensing layer to discover the sensors available in the environment and select the best one to provide a specific piece of information.

2.2.2.3 Context Processing

The context processing layer of a context-aware system is responsible for extraction of context information from raw sensor data or low level context information collected at the lower layer. Context information may be extracted at different levels of abstraction. For example, raw sensor data from human body wearable sensors can be used to extract human activity context that "user is walking". Similarly low level context information like "user is standing", "user is using plates", and "user is using cooking range" can be combined to extract high level context information that "user is preparing dinner".

State of the art machine learning algorithms are used for this purpose (Roggen, Calatroni, Rossi, Holleczek, Föorster, Tröoster, Lukowicz, Bannach, Pirkl, Ferscha, Doppler, Holzmann, Kurz, Holl, Chavarriaga, Creatura, and del R. Millàn, 2010). The algorithms are first trained with sample data before extracting high level context. Accuracy of context extraction process is heavily dependent upon the amount of training data and nature of a specific classification algorithm. Different algorithms may have different accuracy to predict particular type of context. Context information from two or more sensors may also be combined to increase the accuracy of the predictions.

2.2.2.4 Context Management

The middleware for a context-aware system in pervasive computing environments works as an intermediator between plethora of sensing systems and context-aware applications interacting with actuating systems. Lower layers of the middleware collect raw data from the sensors, extract high level context information, and provide it to context management layer. It is the responsibility of the context management layer to efficiently aggregate and store all context information according to a well designed context model and distribute context information to all concerned context consumers. In this section we provide a brief description of the tasks preformed at context management layer and an overview of different techniques used to perform there functionality.

Context Modeling: A well designed context model to collect, store, and share the context information among different interacting applications is a key to the success of context-aware pervasive computing systems. Different techniques ranging

from the key-value models (Schilit, Adams, and Want, 1994), markup schemes models (Indulska, Robinson, Rakotonirainy, and Henricksen, 2003), graphical models (Henricksen, Indulska, and Rakotonirainy, 2003), object oriented models (Schmidt, Aidoo, Takaluoma, Tuomela, Laerhoven, and de Velde, 1999), logic based models (Ghidini and Giunchiglia, 2001), and ontology based models (Strang, Linnhoff-Popien, and Frank, 2003) has been used to present the context information. Strang and Linnhoff-Popien (Strang and Linnhoff-Popien, 2004) analyzed these techniques considering their ability to accommodate distributed composition, partial validation, richness and quality of information, and level of formality and concluded that ontology based models are best to model context information. Ontology based models are also becoming popular to model context information because of their additional capabilities to reason about the information (Niu and Kay, Niu and Kay).

W4H is also an important technique to identify context concepts in a particular domain (de Freitas Bulcao Neto and da Graca Campos Pimentel, 2005). W4H: who (identity), where (location), when (time), what (activity), and how (device profiles) (Truong, Abowd, and Brotherton, 2001) exploit an interaction from five different aspects. These aspects can be described by answering the following questions as presented in (de Freitas Bulcao Neto and da Graca Campos Pimentel, 2005).

- Who are the participants of the interaction?
- Where does the interaction take place?
- When does the interaction take place?
- What does the interacction describes?
- How is context captured and accessed in the interaction?

Granular context model is also used in context modeling to store and distribute context at different levels of detail (Dorn, Schall, and Dustdar, 2006). Using this technique every object in a context model is stored at different levels of detail. For example, the location of a person can be shared at the level of details describing his current country, city, building, floor, or room. Granular representation of

context allow to share context with different peers at different levels of detail considering their access rights and requirements to use that piece of context.

Context Storage and Aggregation: Efficient storage, aggregation, and retrieval of context is key to the success of any context-aware application. Context can be stored at a central location in the network or can also be distributed at different locations. Different technologies have also been used to store context information in context-aware systems. Henricksen et al. (Henricksen, Indulska, McFadden, and Balasubramaniam, 2005) used a relational database system distributed at different locations in the network. They used JDBC[1] to manage the database and provided interface based on Java RMI and XML Web services. Resource Description Framework (RDF) and Web Ontology Language (OWL) also provide a standard way to semantically represent and store the context data. Gu et al. (Gu, Pung, and Zhang, 2005) used RDF / OWL based ontology to store context information and provided a XML Web services based interface to interact with it. Truong et al. (Truong, Juszczyk, Manzoor, and Dustdar, 2007) used XML to save context information at multiple locations in a context-aware system developed to have an adaptive collaborative rescue effort in response to a natural disaster. They also provided XML based Web services to access context information. Lately, RDF / OWL based ontology is becoming a popular technology to model, store, and aggregate context information because of their semantically rich formalism to present real-life entities to machine readable constructs and their ability to reason with existing knowledge to to deduce implicit information.

Context Distribution: The context distribution component is responsible for making the making the context available to other nodes in the environment. Different tools and technologies have been used to implement the context distribution functionality in a context-aware system middleware. RCSM (Reconfigurable Context-Sensitive Middleware) (Yau and Karim, 2004) is a middleware supporting context sensitive applications based on the object model. The JCAF (Java Context Awareness Framework) (Bardram, 2005) is a Java RMI (Remote Method Invocation) based framework to provide infrastructure and programming support

[1] http://java.sun.com/products/jdbc/

to develop context-aware applications. The GAIA project is a CORBA (Common Object Request Broker Architecture based) middleware supporting active space applications (Román, Hess, Cerqueira, Campbell, and Nahrstedt, 2002). Due the use of open standards and protocols, technology independence, lose coupling, and better machine-to-machine interpretability independent of any software application or operating system Web services are becoming a popular mean to implement the functionality of a context management middleware. A detailed survey of context-aware systems that used Web services in presented in (Truong and Dustdar, 2009).

2.2.2.5 Context Consumers

Context consumers make the top layer of any context-aware system. Context-consumers are not limited to adapt the environment to facilitate users but also include adapting work flow in many collaborative works. Context can also be used to optimize the performance of different task in a context management middleware.

2.3 Related Work

In this section we discuss the state of the art of the research efforts that defined and evaluated QoC metrics to improve the performance of context-aware systems and optimized the utilization of the scare resources in pervasive environments. First we discuss research that realized the problem of the imperfection of the context information and tried to model meta-data about the quality of context along with context information. We also discuss the concept of QoC and different QoC metrics that have been presented in the literature. Later, we analyze the approaches, algorithms, and mechanisms that are used to evaluate various QoC metrics. We also present an overview of the research works that used QoC metrics to evaluate confidence on context. Finally, we discuss how different context-aware systems took advantage of QoC metrics in performing the tasks to acquire and provide the context information to context-aware applications. We also present a comparison of these approaches with this work.

2.3.1 Imperfection of Context

Quality of context information has been considered unsatisfactory since the early days of research in context-aware systems and it is recognized that there can be errors in sensing data and processing context information (Schmidt, 2000). Dey et al., in describing context information model of Context Toolkit, also asserted that in contrast to general assumption, context information is usually inaccurate and ambiguous (Dey, Abowd, and Salber, 2001). Judd and Steenkiste stated, in describing the Aura system, that context information is dynamic and typically has uncertainty associated with it (Judd and Steenkiste, 2003). Apart from above mentioned works Gray and Salber (Gray and Salber, 2001), Castro and Muntz (Castro and Muntz, 2000), and Ranganathan et al. (Ranganathan, Al-Muhtadi, and Campbell, 2004) also recognized the problems of imperfection of context information as shown in Table 2.1. In these circumstances it is necessary to collect some additional information about the quality of context information and present it along with context information. Here we discuss the works that tried to present that additional information as meta-data. A summary of these works is shown in Table 2.1.

Different Stages in Realization of Imperfection of Context Information	*Works That Realized Imperfection of Context Information*
Considered imperfection of context information	Schmidt (2000), Castro and Muntz (2000), Gray and Salber (2001), Dey et al. (2001)(Context Toolkit), Ranganathan et al. (2004)(Gaia),
Modeled imperfection of context information as meta-data	Lei et al. (2002), Henricksen and Indulska (2004), Hönle et al. (2005)
Modeled imperfection of context information as QoC	Buchholz et al. (2003), Krause and Hochstatter (2005), Razzaque et al. (2005)

Table 2.1: Stages in the realization of the imperfection of context information

Gray and Salber emphasized that sensing context information from environment is a lot more complex task than explicit input to a system (Gray and Salber, 2001). They proposed the term of sensed context and defined it as *"properties that characterize a phenomenon are sensed and that are potentially relevant to the tasks supported by an application and/or the means by which those tasks are performed"*. They also

proposed a model for sensed context information that aims at handling those complex issues. Along with the information about a particular sensed data, they also suggested to collect meta-information about properties of sensed context. This meta-information includes forms of representation, information quality, sensory source, transformation, and actuation. They include properties like coverage, resolution, accuracy, repeatability, frequency of sample rate, and timeliness as information quality attributes. They also discussed how the nature of sensed context and meta-data can be used for application design and development and presented global view of their architecture for handling context information.

Ebling et al. discussed different design issues faced by a context-aware system (Ebling, Hunt, and Lei, 2001). These issues include protecting privacy of users, scalability and extensibility of systems, and synchrony of operations. Apart from those issues, they also discussed quality of context information (QoI) as the extent to which it corresponds to real world. They identified source of context information as important factor in evaluating quality of context information. They discussed freshness and confidence as important QoI metrics and stressed on the need of work to address the issues related to QoI. Castro and Muntz discussed about QoI and presented it by having a measure of accuracy and a measure of uncertainty in the most likely value of query variable (Castro and Muntz, 2000).

Dey et al. discussed the issues related to the acquisition and representation of context information and privacy of context information (Dey, Abowd, and Salber, 2001). They also asserted that in contrast to general assumption, context information is usually inaccurate and ambiguous. They suggested that there are three approaches to deal with the issue: (i) pass ambiguity on to applications (ii) attempt to disambiguate data automatically (iii) attempt to disambiguate data manually. They also suggested that an application should mention its accuracy requirements to context acquisition and provisioning frameworks and context should be provided to that applications according to those requirements. They suggested to fuse data from multiple sources to improve the accuracy of data. But they only mentioned the accuracy of sensor data.

These early works clearly recognized the problem of the imperfection of context information but mostly they concentrated to resolve this problem by doing probabilistic reasoning by the applications using context information. This undue burden on the

context-aware applications affected their capability to concentrate on their main task to adapt to emerging situations in pervasive environments. These works also suggested some parameters to represent the quality of context information. Later works have recognized more parameters that can be used to indicate quality of context information and used those parameters as meta-data with context information. In the next section we will discuss about those works.

2.3.1.1 Imperfection Presented as Meta-Data

Lei et al. presented the design of a middleware infrastructure for context collection and dissemination, realized as a context service (Lei, Sow, Davis, Banavar, and Ebling, 2002). They emphasized that privacy, quality of information, and extensibility are very critical issues in any context acquisition and dissemination system. Castro and Muntz extended the idea of QoI as the extent to which data corresponds to the real world and emphasized that sources of context data should be allowed to express the inaccuracy and uncertainty of data (Castro and Muntz, 2000). They recommended to use timestamp indicating the freshness of context data and confidence asserted by the data source as QoI metrics.

Henricksen et al. presented a scenario to emphasize the importance of context information in particular situations (Henricksen, Indulska, and Rakotonirainy, 2002). They discussed different characteristics of context information and emphasized that context information can be static and dynamic. They characterized the context information as imperfect and stated that *"information may be incorrect if it fails to reflect the true state of the world it models, inconsistent if it contains contradictory information, or incomplete if some aspects of context are unknown"*. Finally they also tried to associate the quality measures of freshness, accuracy, and certainty of context information in a context model.

Henricksen and Indulska stressed the fact that contrary to general assumption about the quality of context information being perfect, context information can be unknown, ambiguous, imprecise, and erroneous (Henricksen and Indulska, 2004). They claim that imperfect context information presents a significant obstacle to the success of context-aware applications that is commonly overlooked. They stressed that this conflict in context information should be resolved early using conflict resolution techniques. They

also categorized the context information as sensed, static, profiled, and derived and discussed sources that are used to obtain context information, quality issues, and sources of inaccuracy that are associated with them. Finally, this work also presented a model for context information.

Hönle et al. presented a context model integrated with meta data (Hönle, Kappeler, Nicklas, Schwarz, and Grossmann, 2005). They emphasized that meta data, giving additional information about data, improves the operational value of data and can be used for resource finding, enhanced data selection, trust and data quality issues, and sensor fusion. They categorized meta data as system generated, technically measurable, technical restrictions, authorship, and cost and asserted that this data can be used to derive quality metrics associated with data such as reliability, precision, consistency, age, and access control.

The above mentioned works indicated the factors that affect the quality of context information. These factors include the type of context information that can be sensed, static, profiled, or derived, the characteristics of source of context information, and features of environment where context information is gathered. Many parameters indicating the quality of context information are also defined and modeled with context information. These works also suggested that conflicts in context information should be resolved in the early stages of context distribution chain. Consequently, the term QoC was defined to indicate the quality of context and resolve the issues associated with conflicting situations in context usage and distribution in pervasive computing systems. In the next section we discuss the works that recognized the concept of QoC and indicated metrics to present QoC.

2.3.2 Quality of Context

The term QoC was first defined in (Buchholz, Küpper, and Schiffers, 2003) and various context models also strived to present QoC along with context information. Different metrics were also considered to present QoC information. Here we discuss different definitions of QoC, metrics proposed by different research works, context models that supported QoC, and different techniques to evaluate QoC metrics.

Quality Concept	*Suggested Metric*	*Description*	*Suggested By*
Temporal	Timeliness	Range of measure in time	Gray and Salber (2001)
	Up-to-datedness	Age of context information	Buchholz et al. (2003)
	Staleness	Out of time for use	Henricksen and Indulska (2004)
	Refresh rate	How often receive a new measurement	Huebscher and McCann (2004)
	Age	How old is data	Hönle et al. (2005)
	Frequency	Sample rate	Gray and Salber (2001)
Correctness	Accuracy	Information is measured correctly	Gray and Salber (2001)
	Probability of correctness	Probability that information is correct	Buchholz et al. (2003)
Observation level	Resolution	Smallest perceivable element	Gray and Salber (2001)
	Precision	Exactness of measurement	Buchholz et al. (2003)
Information amount	Coverage	Amount of sensed context	Gray and Salber (2001)
	Completeness	All aspects of context are known	Kim and Lee (2006a)
Trust on Sensor	Reliability	Degree of confidence on sensor	Gray and Salber (2001)
	Trustworthiness	How likely is that sensor provided correct information	Buchholz et al. (2003)
	Authorship	Information about sensor	Hönle et al. (2005)
Privacy	Access control	Extent to which context is restricted	Kim and Lee (2006a)
Worth of context	Significance	Worth of context for a specific context consumer	Manzoor et al. (2010)

Table 2.2: QoC concepts and different representations used for QoC metrics

Buchholz et al. defined Quality of context as *"Quality of Context (QoC) is any information that describes the quality of information that is used as context information. Thus, QoC refers to information and not to the process nor the hardware component that possibly provide the information"* (Buchholz, Küpper, and Schiffers, 2003). They presented precision, probability of correctness, trust-worthiness, resolution, and up-to-datedness as important QoC metrics. They also compared QoC with Quality of Service (QoS) that gives the information about the performance of a service and Quality of Device (QoD) that characterize technical properties and capabilities of a device. They emphasized that although these three quality metrics are different from each other, they can influence each other. They also presented scenarios on how context providers can cooperate with Context-Aware Service (CAS) providers by sensing the context from the environment, refining context information by doing reasoning on it, and finally providing it to context provider who used this context information to adopt the behavior of context-aware services. They has also discussed scenarios where and why we need the QoC. Finally, they compared their work with early works that also considered quality of context information in context-aware systems.

Krause and Hochstatter presented the necessity of QoC metrics, analyzed a general context provisioning process, and derived requirements for QoC-handling (Krause and Hochstatter, 2005). They also gave a new definition of QoC as *"QoC is any inherent information that describes context information and can be used to determine the worth of the information for a specific application. This includes information about the provisioning process the information has undergone (history, age), but not estimations about future provisioning steps it might run through."* They identified the sources of QoC parameters as the characteristics of sensor, situation of specific measurement, value expressed by the context information object itself, and granularity of representation format.

Razzaque et al. analyzed different existing approaches to model context information. These approaches include set theory, directed graph, first-order logic, and preferences (Razzaque, Dobson, and Nixon, 2005). Later they discussed the dependency relationship which is a special type of relationship that exist between context entities and attributes. They also stressed that Quality of Context Information (QoCI) should also be modeled as part of context information models and user of context information

or applications will also be provided with this quality of context information.

2.3.2.1 QoC Metrics and Evaluation

Although there are many efforts that tried to model QoC with context information, they lack consistent terminology for QoC metrics. Different QoC metrics have been used to show same concepts. For example, time resolution of context information has been recognized as up-to-datedness (Buchholz, Küpper, and Schiffers, 2003), timeliness (Gray and Salber, 2001), staleness (Henricksen and Indulska, 2004), refresh rate (Huebscher and McCann, 2004), and age (Hönle, Kappeler, Nicklas, Schwarz, and Grossmann, 2005). Table 2.2 shows the summary of QoC concepts and different form of representations that have been used for those concepts. These works also did not provide the QoC metrics in a form that can show the worth of context information for an application and to allow those parameters to be used by an application. We extended those concepts to include the worth of context information for a specific context consumer and introduced the QoC metric significance. Table 2.3 shows different research works that have defined metrics considering different QoC concepts. Though QoC metrics have been indicated and context information models have been designed to accommodate these metrics with context information, few works have tried to evaluate these parameters. Here we discuss the works that proposed different QoC metrics and evaluated some of them.

Kim and Lee proposed accuracy, completeness, representational consistency, access security and up-to-datedness as QoC parameters and presented statistical method to calculate accuracy of sensor data (Kim and Lee, 2006a). However, their method to measure accuracy is more appropriate in those cases where sensors get continuous data around some average value, e.g., data from temperature sensors. They also measured the completeness of a context object as the ratio of available attributes to total number of attributes for a specific context object.

Sheikh et al. presented five QoC parameters as precision, freshness, spatial resolution, temporal resolution and probability of correctness and tried to quantify these parameters (Sheikh, Wegdam, and van Suinderen, 2008). In this process they considered different options that can be used to interpret and represent QoC parameters for different type of context information. For example, they discussed that boolean,

Research Works That Defined QoC Metrics Based on Different QoC Concepts	*Quality Concepts Base of Different QoC Metrics*						
	Temporal Attribute	*Correctness of Context*	*Observation Level*	*Amount of Context*	*Capability of Sensors*	*Privacy of Context*	*Worth of Context*
Gray and Salber (2001)	X	X	X	X	X	-	-
Buchholz et al. (2003)	X	X	X	X	X	-	-
Henricksen and Indulska (2004)	X	-	-	-	-	-	-
Huebscher and McCann (2004)	X	-	-	-	-	-	-
Hönle et al. (2005)	X	-	-	-	-	X	-
Kim and Lee (2006a)	-	-	-	-	-	X	-
This dissertation	X	X	X	X	X	-	X

Table 2.3: Comparison of research works that defined QoC metrics

numeric, complex types with an incremental structure, and unordered complex types can be used to present precision of different type of context information. They also discussed the options that can be used for other types. Tang et al. presented a context quality model based on OWL-DL and used a function that is based on a specific application to evaluate QoC value of certain context (Tang, Yang, and Wu, 2007). They also illustrated with different scenarios that how the value of QoC parameters can be changed with the change in current situation and applications.

Toivonen et al. used quality attributes to calculate trust in range [0..1] (Toivonen, Lenzini, and Uusitalo, 2006). They used different formulas for evaluating trust in different situations. This system evaluates trust in two steps. The first step is the traditional calculation of trust using quality attributes, e.g., using recommendations. The second step adjusts the trust value calculated in the first step by using the context attributes. The adjustment function uses context based predicate and weights. If the context predicate condition is true then the weight for this context attribute is increased otherwise decreased. Different weights can be assigned by the user to increase or decrease the context attribute values in different conditions. For example, a user wants to select a web application which requires less memory, he can assign increasing and decreasing values for relative memory requirement.

To calculate trust Neisse et al. used Subjective Logic which expresses trust with a triple belief, disbelief and uncertainty (Neisse, Wegdam, van Sinderen, and Lenzini, 2007). The results of these functions are mapped on a set {VT, T, U, VU} that describes very trustworthy, trustworthy, untrustworthy and very untrustworthy respectively. If the belief is higher than disbelief, the result is trustworthy, if it has uncertainty not lower than 1/3 and very trustworthy otherwise. But if belief is not higher than disbelief, it is considered untrustworthy if it has uncertainty not lower than 1/3 and very trustworthy otherwise. The used a recommendations manager to establish indirect trust with new entities about which user is not already aware. This trust is based on information received from other entities.

As compared to the above works we relate QoC to the worth of context information for an application (Manzoor, Truong, and Dustdar, 2008). We classified QoC into QoC sources and QoC metrics. QoC sources are the information about the sensors that collect context information, the environment where that context information is collected, and the entities about which the context information is collected. We have evaluated QoC metrics , such as, up-to-datedness, trustworthiness, completeness, and significance using QoC sources.

The research works presented above have evaluated very few QoC metrics as shown by the Table 2.4. Different data representations have also been used to quantify the QoC metrics. Sheikh et al. have shown different options that can be used to quantify QoC metrics and used the numbers in range [0..3] to use QoC metrics in a scenario to enforce privacy (Sheikh, Wegdam, and van Suinderen, 2008). We evaluate QoC metrics as a real number having value in rang [0..1], 0 present the lowest value and 1 the highest value (Manzoor, Truong, and Dustdar, 2008). These metrics are presented to context-aware applications to optimize the functionality of their tasks. In the nest section we discuss works that combined QoC metrics to present confidence on context information.

2.3.3 Context Confidence

As the quality of context information is considered imperfect, different QoC metrics, such as reliability, timeliness, and completeness, are also evaluated and attached to

Research Works That Evaluated QoC Metrics Based on Different QoC Concepts	*Quality Concepts Base of Different QoC Metrics*						
	Temporal Attribute	*Correctness of Context*	*Observation Level*	*Amount of Context*	*Capability of Sensors*	*Privacy of Context*	*Worth of Context*
Schmidt (2006)	X	-	-	-	-	-	-
Kim and Lee (2006a)	-	X	-	X	-	X	-
Toivonen et al. (2006)	-	-	-	-	X	-	-
Neisse et al. (2007)	-	-	-	-	X	-	-
Brgulja et al. (2009)	-	X	-	-	-	-	-
This dissertation	X	X	X	X	X	X	X

Table 2.4: Comparison of research works that evaluated QoC metrics

context objects. But those QoC metrics do not prove sufficient to provide context reasoning applications with high quality context information. Selection of a context object on the base of high value of single QoC parameter such timeliness may result in using a context object with low values of other QoC parameters such as trustworthiness and completeness. Simple mathematical methods to accumulate QoC metrics, such as average, also cannot present the true state of the quality of a context object. Context reasoning applications also need to know about the range of numerical values of all QoC metrics at the design time to mention their requirements about QoC. Such applications also cannot manage to model all QoC metrics in their logic to take advantage of the quality of a context object. These situations compel the applications to work with low quality context information that heavily affects the their capabilities to recognize current situation.

Different research efforts also proposed to attach context information with confidence metadata to enable the applications in context-aware systems to select and use context with high value of confidence (Pietschmann, Mitschick, Winkler, and Meißner, 2008; Bu, Gu, Tao, Li, Chen, and Lu, 2006). Most of the time, however, only a single quality parameter measured confidence or uncertainty of context information. Bu et al. proposed to indicate confidence on context information based on the time of generation of context object (Bu, Gu, Tao, Li, Chen, and Lu, 2006). Schmidt also used age of context information to indicate confidence and select among conflicting context

information (Schmidt, 2006). In this case a most recently collected context object is always be selected and can result in ignoring context objects with highly reliability and accurate information generated at earlier stage in favor of less quality context object generated later.

Ranganathan et al. used precision of sensor measurement to indicate uncertainty of context information (Ranganathan, Al-Muhtadi, and Campbell, 2004). Korpipaa et al. used probability of correctness of context information to present confidence on context information (Korpipaa, Mantyjarvi, Kela, Keranen, and Malm, 2003). Mostly confidence is not modeled as a metric that presents different QoC metrics collectively. However, McKeever et al. proposed a model that combines different QoC metrics to have the value of confidence that can be used at the application level but they do not foresee any mechanism to combine QoC metrics as required by the context consumer (McKeever, Ye, Coyle, and Dobson, 2009). Similarly, Brgulja et al. also proposed to combine different QoC metrics to calculate confidence on context information without considering the context consumer requirements (Brgulja, Kusber, David, and Baumgarten, 2009).

Compared to these works, we have combined different QoC metrics to indicate confidence on context objects according to the requirements of application using context information (Manzoor, Truong, Dorn, and Dustdar, 2010). We have also presented a simple mechanism for higher level applications to indicate their requirements for quality of context objects. With the help of confidence metric, services can successfully select context objects without compromising the quality of context information from any aspect. The confidence value at the application layer in context-aware systems also allows them to model quality in their logic and enhance their performance. In the next section we discuss about the works that have used QoC metrics to perform different tasks in context-aware systems.

2.3.4 Context Management Systems Using QoC

QoC metrics not only indicate about the timeliness, completeness, precision, and significance of context information but also provide the information about the reliability of the source of context information. Context information management systems can

take advantage of QoC metrics in performing different tasks, such as, source selection in acquiring context from sensors, reasoning on raw context data to extract high level context information, aggregating high level context information, context query routing, and privacy enforcement while context sharing. In this section we discuss how different systems have used QoC parameters to enhance their efficiency while providing the context information to high level applications and other peers in pervasive environment.

Research Works That Used Different QoC Concepts	*Applications that used different QoC concepts*						
	Conflict Resolving	*Context Aggregation*	*Privacy Enforcement*	*Context Selection*	*Event Generation*	*Confidence Inference*	*Access Control*
Huebscher and McCann (2004)	-	-	-	X	-	-	-
Chantzara et al. (2006)	-	-	-	X	-	-	-
Bu et al. (2006)	X	-	-	-	-	-	-
Neisse et al. (2007)	-	-	X	-	-	-	-
Breza et al. (2007)	-	-	-	X	-	-	-
Pawar et al. (2007)	-	-	-	X	-	-	-
Da Rocha et al. (2008)	-	-	-	-	-	-	-
Sheikh et al. (2008)	-	-	X	-	-	-	X
Toivonen et al. (2006)	-	-	-	-	-	-	X
This dissertation	X	X	-	X	X	X	-

Table 2.5: Research works that used QoC metrics in different applications

Chantzara et al. presented an approach that used quality of information for evaluating and selecting the information to be used as context information (Chantzara, Anagnostou, and Sykas, 2006). They calculated a utility function based on QoC attributes. But in that work QoC attributes are completely provided by the context sources that can bias the decision for source selection.

Huebscher and McCann used QoC parameters in their adaptive middleware for context-aware applications in smart homes (Huebscher and McCann, 2004). They have also used QoC parameters to perform different tasks in their middleware, such as context provider selection. But their work is based on the assumption that context

providers are able to estimate QoC metrics and provide them to their middleware. Bu et al. considered the QoC metrics such as delay time, context correctness probability, context consistency probability, and correlation among those metrics (Bu, Gu, Tao, Li, Chen, and Lu, 2006). They also calculated another metric relative frequency and used it to resolve inconsistency among various context objects.

Sheikh et al. quantified the QoC metrics, such as precision, spatial resolution, temporal resolution, freshness, and probability of correctness to have the value in range [0..3] and used those metrics to enforce their privacy policy in a health information system (Sheikh, Wegdam, and van Suinderen, 2008). The owner of context information can specify the quality of context information that can be accessed by caregivers. For example, only city of the patient is shared in normal situations while in emergencies complete location information including city, street, postcode, and house number is shared.

Breza et al. used QoC for source selection and providing autonomic behavior in wireless sensor network (Breza, Anthony, and McCann, 2007). QoC requests are made by the context requester. Different sensors send their QoC information to the requester. These values are evaluated at the context requester node to select a sensor that fulfills QoC requirements. As soon as an acceptable sensor has been found, the requester sends a message to other sensors to stop sending QoC values. As QoC metrics change over time, they also suggested an autonomously managed system for QoC metrics.

Pawar et al. proposed a context distribution framework in which context sources are selected on the basis of QoC metrics (Pawar, van Beijnum, Peddemors, and van Halteren, 2007). When a new context source is registered in the service directory, the registration information also include the information about the capabilities of context source. Information about the capabilities of context sources is defined by the information about source, information about the entity about which context is collected, context type, and QoC metrics about context information. This capability information is used to make selection among different sources of context information.

Toninelli et al. have taken QoC metrics into account to make access control decisions (Toninelli, Corradi, and Montanari, 2009). Neisse et al. have presented a trust model to achieve the goals of privacy enforcement and service adaptation (Neisse, Wegdam, van Sinderen, and Lenzini, 2007). They used entities like user, context owner,

context provider, service provider, and identity provider. Each of these entities should have trust in each other to provide and consume services from each others, so they have different trust relationships. Trust of these entities is dependent on each other. For example if a provider is not trustworthy then its provided services are also not trustworthy. User trust on context provider affects the trust on context information.

We use QoC metrics to define the policies to resolve the conflicts in context information (Manzoor, Truong, and Dustdar, 2009b). We define the conflict resolving policies on the bases of QoC metrics, such as reliability, timeliness, completeness, and significance. These policies can be used individually or in combination with each other depending upon the context of use of information and the requirements of a specific application. We also demonstrate the effectiveness of these policies in a context aggregation and context selection systems. Later we use the confidence on context in (Manzoor, Truong, Dorn, and Dustdar, 2010) to generate the events of interest for a specific context consumer.

2.4 Summary

In this chapter, we presented the conceptual layers of a context-aware system in pervasive environments and briefly described different tasks performed at those layers. Later, we presented the state of the art of the existing research efforts undertaken to solve the problem of the imperfection of context information. We also presented a comparison of existing work with this dissertation. As compared to existing work this dissertation presents subjective view of QoC and introduces novel metrics to consider quality of context information as the worth of context information. This dissertation also evaluates those metrics using a novel approach to precess QoC. Finally, we presented the analysis of existing work with this dissertation considering the context-aware application using QoC metrics in their functionality. Our approach to infer the value of confidence on context by combining QoC metrics tailored to the need of a specific context consumer proved more effective than existing approaches. In the next chapter we present our model to process QoC metrics and our approach to use QoC metrics to perform different tasks in a context-aware system.

Chapter 3

Novel Quality of Context Processing Model

3.1 Overview

In this chapter we present our novel approach to process QoC metrics. First we discuss the limitations of existing approaches to use QoC metrics. Then we present our perception of the concept of QoC. We also present our definition of QoC that depicts our view of the concept. Later, we present our model to precess and use QoC metrics in a context-aware pervasive computing system.

3.2 Limitations of Existing Approaches

Considering the historical perspective of the problem of the imperfection of context information, different research efforts defined the term Quality of Context (QoC) and identified QoC metrics. However, these works did not provide QoC metrics in a form that can indicate the worth of context information for a specific application and allow those metrics to be used by that application without any further processing. They also did not make any distinction between the QoC indicators which can be used to calculate QoC metrics and high level QoC metrics presented to context-aware application. While

context models designed to accommodate QoC metrics only few works tried to evaluate some of basic metrics, such as timeliness, completeness, and accuracy.

Many works also proposed having a single metric presenting the confidence on context information, but most of the time evaluation of confidence is limited and involves only a single QoC metric. Existing models proposed to combine QoC metrics to infer the value of confidence did not consider the context consumer requirements that also result in the failure of the confidence on context to indicate the quality of context information considering the needs of a specific context consumer concerning the quality of context information.

Conflict resolving policies in context-aware system are also mostly designed to solve the problem of context selection. Even these policies are limited to very simple strategies, such as involving the user in mediation process, resolving the conflicts by using some predefined static policies based on user preferences, discarding all the conflicting context, discarding the last received, or discarding the first received context objects. These strategies may also slow down the process of decision making, distract users, or discard some important context objects as well. Moreover, context conflicts cannot be resolved at design time and need a strategy that can dynamically handle them at runtime without user intervention.

3.3 A Novel Definition of QoC

As we discuss in Chapter 2 quality of context is considered unsatisfactory since the start of research in context-aware systems. Early context-aware systems have also tried to collect and model additional data with context. Subsequently, more research efforts were undertaken to explore the problems associated with the imperfection of context and term Quality of Context (QoC) was coined.

QoC was first defined by Buchholz et al. (2003) and later by Krause and Hochstatter (2005) as discussed in Chapter 2. Both of the definitions consider QoC as an objective term that is independent of the situation of the use of context information and consumer requirements for that context information. However, general quality literature has taken quality as both absolute and relative term and defined quality as “freedom from

errors", "conformance to specifications", and "features of product that meet customer's needs" Juran et al. (1999). These definitions describe quality as a twofold concept that has an absolute quality, showing that the end product is free of errors and a relative quality that shows how much the end product meets customer's needs. Similarly QoC should also inform about the quality of context information in both of these aspects. First, QoC considers the limitations of sensors in collection of context information and situation of specific measurement, e.g., measurement errors, collection of partial information. This aspect of QoC shows the absolute quality of context information that depicts that how much context information is free of error and describes the current situation in the environment. Second, QoC also considers the fact that different context consumers may have different requirements about quality of context information. This aspect of QoC shows the relative quality of context information that depicts that how much context information meets the requirements of a specific consumer to use it for a specific purpose.

For example, a context information service that provides information about the location of a person uses a GSM (Global System for Mobile Communications) method and can provide location information with the granularity of the current district of a person. Another context information service that uses a GPS (Global Positioning System) method provides the location information of that person with the granularity of the current street of the location of a person. As these two services are collecting context information with high accuracy, they will have same objective quality. Context information provided by these services will also have same subjective or relative quality for a context consumer service that is only providing information to tourists about interesting places to visit in a city. But the first service will have a low subjective quality for a context consumer service that makes an optimal plan for visiting all sites in a city, as this service will need the location information of a person with higher granularity, i.e., at least with the detail of the current street of the location of that person. The second context information service that provides location information with higher granularity will have higher subjective quality for this context consumer.

This example shows that the quality of context information may vary with different context consumers and QoC cannot be measured independently of a context consumer and the intended purpose. Context that is appropriate for use with one application

may not be suitable for use by another application. Therefore QoC must also consider the requirements of context consumer and the intended use of context information. This aspect of QoC will show that how much context information is suitable for use by a specific context consumer for intended purpose.

Considering this objective and subjective nature of QoC we have defined it as,

"Quality of Context indicates the degree of conformity of the context collected by sensors to the prevailing situation in the environment and the requirements of a particular context consumer".

The objective view of QoC considers those features of context that are independent of any requirements of a context consumer or the situation of the use of context. These objective characteristics portray how much context is free from errors, i.e., degree of conformity of context to the prevailing situation in the environment. Information about the sensor characteristics that have collected context and situation of specific measurement will be used to determine objective QoC metrics. The subjective view of QoC shows the characteristics of context that illustrate how much a piece of context meets the requirements of a particular consumer to use it for a specific purpose, i.e., degree of conformity of context to the requirements of a particular context consumer. Information about the intended use and the consumer requirements will be used to determine the subjective metrics of QoC.

3.4 Novel QoC Processing Model

We propose a novel QoC processing model as shown in Figure 3.1. Our model consists of different layers for processing QoC information. The lowest layer is the QoC source layer that consists of data used by higher layer to evaluate QoC metrics. This layer consists of the characteristics of sensors that collect context information, i.e., *Sensor Characteristics*, situation of a specific measurement, i.e., *Measurement Context*, and the information about the requirements of a context consumer and the detail of the context of the use of information, i.e., *Specifications and Consumer Requirements*. QoC metrics evaluated from the data at QoC source layer lie at the next higher layer and are divided in *Objective QoC Metrics* and *Subjective QoC Metrics*. *Objective QoC*

Metrics show the quality of context as an independent quantity and their calculation will involve *Sensor Characteristics* and *Measurement Context*. *Subjective QoC Metrics* will show the quality of context for use by a specific context consumer for particular purpose. These metrics will also involve *Specifications and Consumer Requirements* for their calculations. In chapter 4 we will give the detail of the each building block of our QoC processing model.

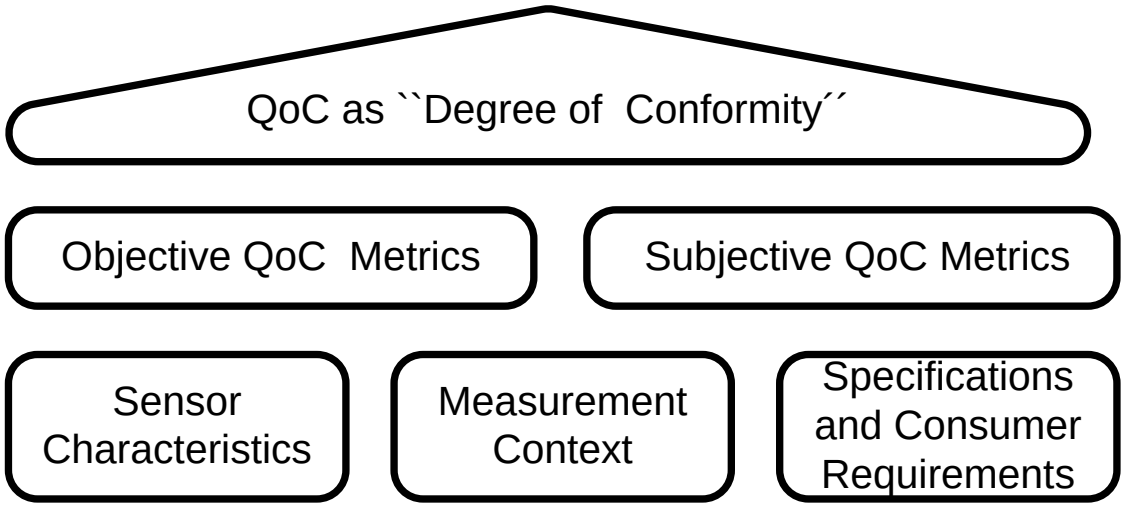

Figure 3.1: QoC processing model

3.5 Framework Design

Deficient quality of context information and conflicting situations at different layers of a context-aware systems are the principal challenges to tackle in the design and development of pervasive environments. Human attention being one of the most scare resource in such environment also enforce that this solution should also be independent of any run time input from them. We solve this problem by evaluating QoC metrics for the context information, inferring the value of confidence on context, and introducing the conflict resolving policies that enclose all the layers of context-aware pervasive computing system as shown in Figure 3.2. *QoC Evaluator* evaluates the QoC metrics using sensor characteristics, measurement context, specifications, and context

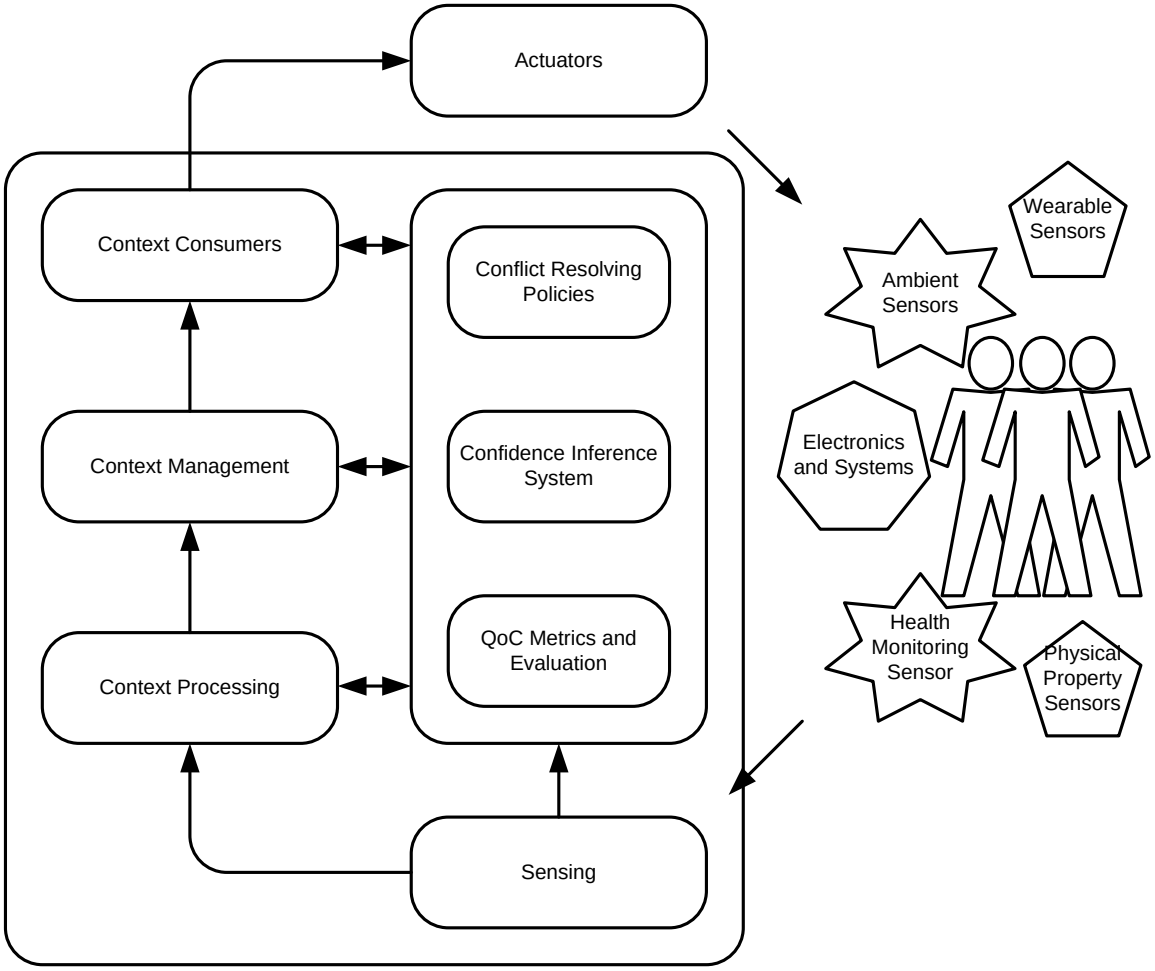

Figure 3.2: Framework in which we implemented our novel approach

consumer requirements as described in Chapter 4. QoC metrics are combined in different combinations depending upon the context consumer requirements and situation of the use of context to infer the value of confidence on context presented in Chapter 5. Different policies are also devised based on the individual and different combinations of the QoC metrics as described in Chapter 6. Components of the context-aware pervasive computing systems, as discussed in Chapter 2 and shown in Figure 3.2 can use QoC metrics, confidence on context, and conflict resolving policies to resolve conflicts at different levels of a systems. We have affectively demonstrated the effectiveness of our approach in selecting sensing modalities, extracting features to drive high level

context information, context aggregation, and generating relevant events in Chapter 7.

3.6 Summary

In this chapter we discussed the limitations of the existing approaches to process QoC metrics and presented our novel approach to deal with QoC metrics. We also presented our definition of QoC and indicates our perception of the concept of QoC. We presented our model to process and use QoC metrics in a context-aware pervasive computing system. In the next chapter we describe different components of our QoC processing model in more detail. We also present the process to evaluate QoC metrics and present those metrics along with context information.

Chapter 4

Quality of Context Metrics and Evaluation

4.1 Overview

The lack of information about the Quality of Context (QoC) can degrade the performance of context-aware systems in pervasive environments. Context-aware systems can take advantage of QoC if context producers also provide QoC metrics along with context information. In this chapter, we classify and define QoC metrics. We evaluate QoC metrics considering the capabilities of sensors, circumstances of specific measurement, requirements of context consumer, and the situation of the use of context information. We also illustrate a context model that incorporate QoC metrics along context information.

4.2 Sensor Characteristics

Sensor characteristics are different properties of sensors that are used to evaluate QoC metrics. These characteristics will include the information about the accuracy, granularity, precision, time period, sensor state, and sensor range. Table 4.1 presents a brief description of sensor characteristics. There may be different ways to calculate the

values of sensor characteristics for physical and virtual sensors. Here we describe each sensor characteristic and the detail of the ways to work out its value for both physical and virtual sensors.

Metric	*Definition*	*Collection Method / Origin*
Accuracy	Extent to which data is correct and free of errors	Sensor data sampling or sensor specifications
Precision	Degree of exactness with which context is collected	Sensor data sampling or sensor specifications
Resolution	Degree of detail with which context is collected	Measurement unit or sensor specifications
Time Period	Time interval between two readings of context	Sensor configuration
Sensor State	Physical state of sensor	Sensor configuration
Sensor Range / Span	Maximum distance for which sensor can collect context	Sensor Specification

Table 4.1: Brief description of metrics in sensor characteristics

4.2.1 Accuracy

Accuracy is the degree of the correctness of context or the capacity of sensor to measure a quantity close to actual value. Inaccuracy or absolute error of a physical sensor can be calculated by taking the difference the measured and true value of a quantity (Webster, 1999), that is

$$E = M - T \tag{4.1}$$

where E is the error in the measurement, M is the measured value, and T is the true value of measured quantity. Accuracy of the physical sensors is calculated by taking the relative error and normalizing it to one, that is

$$Accuracy = 1 - \frac{|E|}{T} \tag{4.2}$$

where E is the value of absolute error as calculated in Equation 4.1 and T is the true value of the measured quantity. Fraction $\frac{E}{T}$ gives the relative value of the error.

Accuracy is calculated by subtracting the relative error from 1.

Accuracy of virtual sensors, such as a classification algorithm, is calculated as the rate of the number of instances that have been correctly classified to the total number of instances, that is

$$Accuracy = \frac{Number\ of\ correctly\ classified\ instances}{Total\ number\ of\ instances} \tag{4.3}$$

4.2.2 Precision

Precision is the degree of the exactness of a measurement. It indicates the capacity of a sensor to give the same reading when measuring the same quantity under the same circumstances. Contrary to accuracy of a sensor that presents the closeness of a measurement to the true value, precision presents the closeness of the successive sensor readings of the same quantity under same circumstances. Precision is a necessary but not the sufficient condition for the accuracy of a sensor. It is possible for a sensing system to be accurate but not precise, precise but not accurate, neither, or both.

Precision of a physical sensor can be measured by repeating the experiments for a number of times and examining the variation in data (Taylor and Thompson, 1998). Precision of a virtual sensor, such as classification algorithm, can be measured by taking the ratio of the total number of true positives, i.e., the instances that have been correctly recognized as positive, and the total of true positives and false positives, i.e., the instances that are incorrectly recognizes as positives. Equation 4.4 is used to evaluate precision.

$$Precision = \frac{number\ of\ true\ positives}{total\ number\ of\ true\ positives\ and\ false\ positives} \tag{4.4}$$

Confusion matrix that presents the predicted classes against true classes as shown in Figure 4.1 can be used to get the values of true positives and false positives (Villalonga, Roggen, Lombriser, Zappi, and Tröster, 2009). True positives are presented on the matrix diagonal. The off-diagonal elements show the instances that have been confusedly predicted as other classes. These off-diagonal elements can be summed up along the true class rows to present false negatives for that class and along the predicted

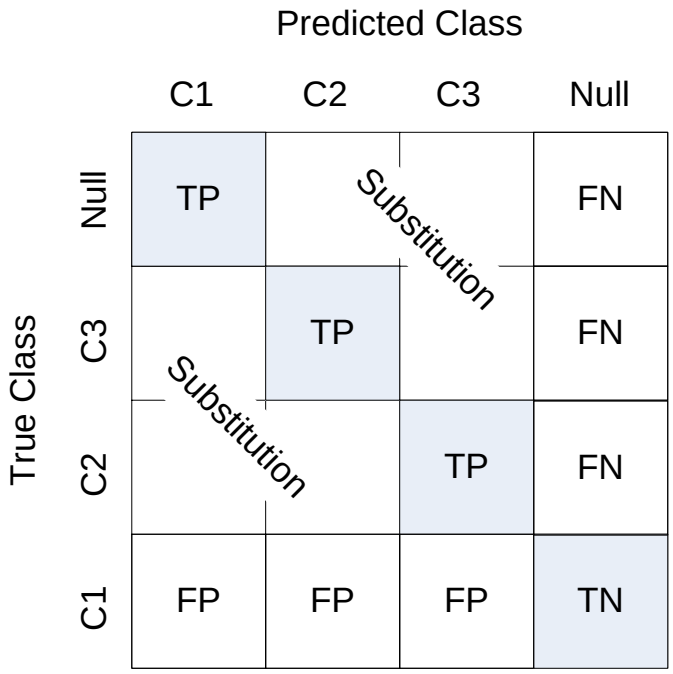

Figure 4.1: Confusion Matrix

class column to present the false positives for that class as shown in Figure 4.1.

4.2.3 Resolution

Resolution indicates the degree of the detail with which a sensor can collect data or the smallest possible detectable change in the environment that can be expressed in the sensor output (Carr and Brown, 2001). For example, a mercury thermometer marked in degree centigrade can measure the temperature with smallest possible change of 0.2 °C while a digital thermometer can measure the temperature to the nearest of 0.1 °C. It will mean that the digital thermometer can measure the temperature with high value of resolution. Fine granularity or higher value of resolution indicates that information is expressed with more detail.

4.2.4 Time Period

Time Period indicates the time interval between two measurements. For example, if a location sensor collects the location information of a person after every one minute then *Time Period* of this information will be one minute.

4.2.5 Sensor State

Pervasive environments include both static and dynamic sensors. Static sensors are the sensors fixed at a specific place. Such sensors are embedded in a specific device or environment to monitor state change, e.g., temperature sensors, device on / off sensors. Dynamic sensors include sensors that can change their position, e.g., sensors worn on human body, sensors embedded in a mobile device. *Sensor State* indicates whether the source of information is static or dynamic.

4.2.6 Sensor Range

Sensor Range is the maximum distance for which a sensor can collect a context object. Every sensor will have a different value of *Sensor Range*. For example, images of a disaster stricken site can be collected by a satellite and an ordinary camera. But the value of *Sensor Span* will be very high for satellite camera as compared to a camera carried by a field worker on disaster site. Similarly, cameras with different capabilities will have different values of sensor span.

4.3 Measurement Context

Measurement context shows the information related to a specific measurement. This information will include the *Measurement Time, Sensor Location, Information Entity Location and Available Attributes* for a specific type of context object. Table 4.2 presents the brief description and the collection methods of measurement context information. In the remaining section will describe them in detail.

Metric	*Definition*	*Collection Method / Origin*
Measurement Time	Time of collection of context information	Time stamp at the time of context collection
Sensor Location	Location of sensor when context information is collected	GIS for static sensor and GPS embedded in dynamic sensor
Information Entity Location	Location of the real world entity about which context is collected at the time of collection of context	GIS for static sensor and GPS embedded in dynamic sensor
Available Attributes	Number of attributes that have a value for that context object	Context object

Table 4.2: Brief description of metrics in measurement context

4.3.1 Measurement Time

Measurement Time is the time at which a specific event happens in the environment and is observed by the sensor. Time stamp is also attached with sensor data before forwarding it to other nodes that collect sensor data and extract and disseminate the higher level context information. *Measurement Time* is one of the most important quality attribute of the context information and is used to evaluate the timeliness of context information.

4.3.2 Source Location

Source Location is the geographical location of the sensor that collects a context object. As we have already discussed in Section 4.2.5 that pervasive environments include both static and dynamic sensors and in Section 4.2.6 that these sensors can reliably collect information about an event up to a certain distance called sensor range. In these circumstances sensor location becomes important to evaluate the ability of a sensor to capture information about a certain entity. The most common system for sensing outdoor location is the Global Positioning System (GPS). Different techniques, such as infrared (Want, Hopper, Falcao, and Gibbons, 1992), radio frequency identification

(RFID) (Ni, Liu, Lau, and Patil, Ni et al.), and wireless local area network (Smailagic and Kogan, 2002) can also be used to capture indoor location with different accuracy and precision (Hazas, Scott, and Krumm, 2004).

4.3.3 Information Entity Location

Information Entity Location is the geographical location of the entity whose situation is represented by a context object. *Source Location* along with *Information Entity Location* will be representing the space resolution. They help to decide about the reliability of a sensor to collect that context object. For example, if we have more than one context object, representing the same entity in the environment, the context object collected by the sensor closest to the entity will get maximum value of reliability, provided that all the sources are collecting the information with the same accuracy.

4.3.4 Access Level

Access Level is also an important contextual characteristic of context. The owner of a context object can set the level of granularity with which the context object can be shared with other context consumers to protect his / her privacy. For example, the location of a person can be expressed at the level of granularity of country, city, street, and building. The owner of the context can set the access level so that information up to the level of current city of his / her location should be shared with other context consumers. Context consumers may mentions their access level in the context subscription request. The context distributor checks the context consumer access level against the context accessing policies set by context owner to allow context consumer to access context at certain level of granularity.

4.3.5 Available Attributes

Available Attributes are the number of attributes available for a context object. Available attributes are particularly important to measure the completeness of a context

object for a specific context consumer. *Available Attributes* can be measured by looking at a particular context object.

4.4 Specifications and Consumer Requirements

Context consumer specify their requirements about the quality of context information. These requirements will be used with other information to calculate the subjective metrics of QoC. Table 4.3 shows the brief description of specifications and consumer requirements.

Metric	*Definition*	*Collection Method / Origin*
Validity Time	Maximum length of time for which a specific type of context information is stable	Context data model
Required Attributes	Number of attributes that are required to have a value for that type of context information	Context data model
Critical Value	Level of importance of context information of a specific type	Context data model
Access Level	Information about the rights of context consumer to access certain type of information	Context subscription

Table 4.3: Brief description of metrics in specifications and consumer requirements

4.4.1 Validity Time

Validity Time indicates the length of time for which the value of context remains stable and valid. *Validity Time* will have a different value for each type of information and context request. For example, the location of a fast moving vehicle changes very rapidly and has lower value of *Validity Time* as compared to the location of a walking man. Similarly, stable data, such as profile of an agent in collaborative working environment, does not change very often and has higher value of *Validity Time*. Context consumer mentions the validity time of every type of context object in context models.

4.4.2 Critical Value

Critical Value of context information will indicate that this information is crucial in a specific scenario. This concept particularly affects quality of context information in scenarios where it will be used in emergency tasks. For example, in disaster response scenario, context object having information about the people caught in the low lying area of the city will be of high critical value.

4.4.3 Total Attributes

Total Attributes is the number of attributes for which a context object can have any value. The value of the total number of attributes can be gathered by parsing the context data model. Context consumer may also mention the required attributes in his subscription request. In that case number of attributes required by a context consumer are compared with available attributes to evaluate completeness of specific context object for a particular context consumer.

4.5 QoC Metrics

QoC metrics are derived from the combination of sensor characteristics, contextual characteristics, and input from specifications and consumer requirements. These metrics will include objective as well as subjective metrics. Objective metrics are calculated independent of the requirements of any context consumer and show that the context information is collected free of error and is suitable to use at an instance of time. Objective metrics include *Reliability*, *Timeliness*, and *Completeness* of context information. Subjective metrics are calculated as quality of context information compared to user requirements for use for a specific purpose. Some of these metrics are *Significance*, *Access Right*, and *Representational Consistency*. Table 4.4 also present the brief description of objective and subjective QoC metrics.

QoC Metrics	*View*		*Description*	*Calculation Method*
	Obj.	*Sub.*		
Reliability	X	-	Indicates the extent to which context can be considered credible	Combination of span reliability and accuracy
Timeliness	X	X	Indicates validity of context to use considering its freshness	Ratio of age and time period or validity time of context depending upon subjective or objective view
Completeness	X	X	All aspects of phenomenon in the environment have been shown	Ratio of available number of attributes to total or required number of attributes of a context object depending upon subjective or objective view
Significance	-	X	Critical value of context information for a specific application	Ratio of critical level of context to maximum critical level that type of context can have
Usability	-	X	Indicates suitability of for use for an intended purpose	Comparison of level granularity of context with level of granularity required by context consumer
Access Right	-	X	Indicate the extent to which owner of context allows the context consumer to access context	Comparison of access level of context allowed by context owner to access level of context consumer
Representation Consistency	-	X	Extent to which context representation format is consistent to consumer requirements	Comparison of representation formats

Table 4.4: Evaluation criteria for QoC metrics

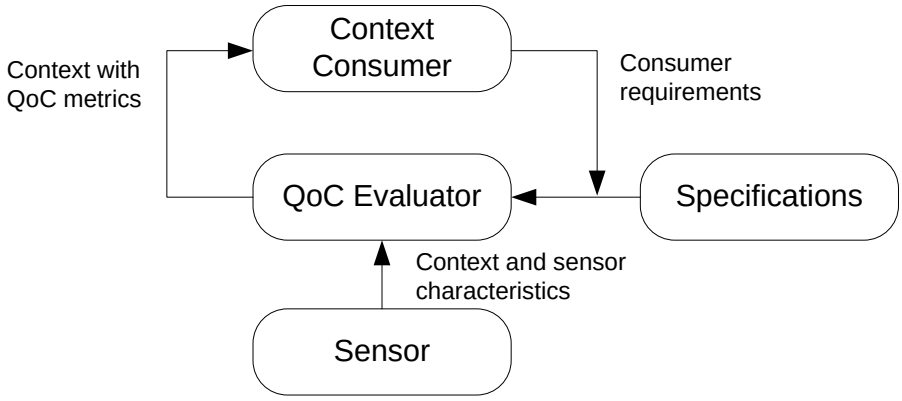

Figure 4.2: QoC Evaluator evaluating QoC metrics

Figure 4.2 shows the process of the evaluation of QoC metrics. The QoC Evaluator gets the sensor characteristics from the sensors. Context consumer mention some of their requirements for the quality of context information in data model provided to the QoC Evaluator. The QoC evaluator evaluates the QoC metrics and provides them to context consumer with context information. The first step to present QoC metrics in a usable form to the context consumer is their quantification, i.e., presenting QoC metrics in a numerical form that is understandable for the context consumer. As we will evaluate QoC metrics against context consumer requirements, it will be appropriate to measure QoC metrics which can have value in range [0..1]. The maximum value 1 means that QoC metric is in complete compliance to the given requirements while the minimum value 0 means total nonconformity to requirements. The sources for the evaluation of QoC metrics are classified as sensor characteristics, contextual characteristics, and specifications and consumer requirements. Table 4.1, Table 4.2, and Table 4.3 show the possible methods that can be used for the assessment of sensor characteristics, measurement context, and specification and consumer request attributes. In the remaining section we will describe how QoC sources are used to evaluate different QoC metrics.

4.5.1 Reliability

Reliability indicates the belief that we have in the correctness of information in a context object. It is an objective QoC metric and is calculated independent of the context

request. Reliability of a context object is particularly useful in making selection from different source of same context object, as this metric can tell how reliable was a sensor to collect a particular context object.

The *Reliability* of a context object is evaluated from the information provided about the sensor that collects that context object, such as accuracy and precision of measurement. Virtual sensors, such as classifiers, may also have their probability of correctness of context extracted from the sensor data. We have also considered the range of sensor to reliably collect the context information to evaluate the reliability of context information. Following equation is used to evaluate the reliability of a context object $\mathcal{O}$.

$$Reliability(\mathcal{O}) = \begin{cases} (1 - \frac{d(\mathcal{S},\mathcal{E})}{d_{max}}) * (n * \frac{\prod_{i=1}^{n} r_i}{\sum_{i=1}^{n} r_i}) & : \quad if \quad d(\mathcal{S},\mathcal{E}) < d_{max} \\ 0 & : \quad otherwise \end{cases} \tag{4.5}$$

where $d(\mathcal{S},\mathcal{E})$ is the distance between the location of the sensor and the location of the entity. d_{max} is the maximum distance for which we can trust on the observation of this sensor. Every type of sensor will have different value for d_{max}. We have used r_i to indicate all the sensor characteristics that contribute towards the reliability of sensor to collect a particular context object. These sensor characteristics include to accuracy and precision with which a sensor collects the sensor data and probability of correctness with which a virtual sensor extract the high level context information. As shown by Equation 4.5 reliability of a context object is directly proportion to sensor characteristics to collect or extract a context object and is inversely proportion to distance between sensor and entity about which context information is collected.

4.5.2 Timeliness

Timeliness is a quality measure that indicates the degree of the freshness of a context object at a given time. As the situation in pervasive environments changes very rapidly, applications using a context object without having any knowledge about timeliness of that context object may take undesired actions that can result in loss of resources

and frustration on the part of user. The value of *Timeliness* and hence the validity of context object decreases as the age of that context object increases.

Therefore, the context consumer can look at the value of *Timeliness* to be sure of the validity of information contained by that context object. For example, this metric can help to more confidently combine the information contained in that context object with the existing information in a context store to provide the current situation of flood in the city to teams participating in rescue work. Context objects, having low value of timeliness, may have misleading or wrong information and can be ignored. If we have static information, e.g., a smart home layout or profile of the people living in a smart home, we can set the lifetime of that information infinite so that its age will not affect the value of *Timeliness* and it will always be maximum.

Timeliness of a context object can be measured in both objective and subjective ways. To measure the timeliness of a context object independent of consumer requirements we consider the age of context object and time period, i.e., time interval after which sensor takes the new readings. The age of context object $\mathcal{O}$ is calculated by taking the difference between the current time, t_{curr}, and the measurement time of that context object $\mathcal{O}$, $t_{meas}(\mathcal{O})$ as shown by Equation 4.6.

$$Age(\mathcal{O}) = t_{curr} - t_{meas}(\mathcal{O}) \tag{4.6}$$

We take the ratio of the age of a context object with the time period for that context object to calculate the objective view of timeliness of that context object. We will also normalize the value of timeliness to have its value in rang [0..1] as shown in Equation 4.7

$$Timeliness(\mathcal{O}) = \begin{cases} 1 - \frac{Age(\mathcal{O})}{TimePeriod(\mathcal{O})} & : \quad if \quad Age(\mathcal{O}) < TimePeriod(\mathcal{O}) \\ 0 & : \quad otherwise \end{cases} \tag{4.7}$$

To get the subjective view of the timeliness of a context object $\mathcal{O}$, validity time of context object mentioned by the context consumer is considered in spite of time period

as shown in Equation 4.8.

$$Timeliness(\mathcal{O}) = \begin{cases} 1 - \frac{Age(\mathcal{O})}{ValidityTime(\mathcal{O})} & : \quad if \quad Age(\mathcal{O}) < ValidityTime(\mathcal{O}) \\ 0 & : \quad otherwise \end{cases} \tag{4.8}$$

The value of timeliness and hence the validity of context object $\mathcal{O}$ decrease as the age of that context object increases and approaches zero when age of context object become equal to time period or validity time of context object considering objective or subjective view of timeliness. Context objects, having low value of the timeliness, may have misleading or wrong information and can be ignored.

4.5.3 Completeness

Completeness indicates the amount of information available in a context object. The maximum value of the completeness of context object means that all aspect of the situation in the environment has been contained in that context object. Lower value of the completeness of a context object will mean ambiguous situation and can result in an undesired action by a context consumer application. *Completeness* of context object is also significant in making selection among context objects presenting information about the same event in the environment.

Completeness can be measured in both objective and subjective ways. We evaluate the objective view of the completeness of a context object as the ratio of the sum of the weights of available attributes of context object to the sum of the weights of the total number of attributes of that context object. The objective value of the completeness of a context object $\mathcal{O}$ is evaluated by the following equation.

$$Completeness(\mathcal{O}) = \frac{\sum_{j=0}^{m} w_j(\mathcal{O})}{\sum_{i=0}^{n} w_i(\mathcal{O})} \tag{4.9}$$

In Equation 4.9 m is the number of available attributes of context object $\mathcal{O}$, n is the total number of attributes that a context object of that type can have as indicated in the context model, and w presents the weights of different attributes. As different attributes can have different worth we have used their weights. Subjective view of a

context object $\mathcal{O}$ can be calculated by taking the ratio of the sum of the weights of available attributes of a context object $\mathcal{O}$ to the sum of the weights of all attributes of a context object $\mathcal{O}$ required by a specific context consumer applications. Context consumer applications may also mention the weights of different attributes of context object according to their preferences.

4.5.4 Significance

Significance indicates the worth or the preciousness of context information in a specific situation. The value of *Significance* is of particular importance in scenarios that involve life threatening situations for humans. Its value for a context object will increase if that context object contains information which needs immediate response or attention, e.g., a context object that contains information about a break in and a broken window in a smart home scenario will have a high value of significance as this situation indicates the home security breach and needs immediate action. The significance of context object can be used to generate events of particular interest for a specific context consumer application.

A context object that is very significant for one context consumer may not be important for another one. For example, in the aforementioned security breach illustration that context object is very significant for the application responsible for the safety of people living in the house but is completely irrelevant for another application that is responsible for home ambience control. Considering this nature of the significance of context object we only have the subjective view of the significance of a context object that indicates the worth of a context object for a specific context consuming application.

We evaluate the *Significance* of a context object considering the critical value of the context information contained by a context object to the maximum critical value that a context object of that type can have. As the levels of critical value are provided by context consumer, significance provides the subjective view of QoC. Significance of context object $\mathcal{O}$ is evaluated by the following equation.

$$Significance(\mathcal{O}) = \frac{CV(\mathcal{O})}{CV_{max}(\mathcal{O})} \tag{4.10}$$

where $CV(\mathcal{O})$ is the critical value of the context object $\mathcal{O}$. $CV_{max}(\mathcal{O})$ is the maximum critical value that can be assigned to a context object of the type that is represented by $\mathcal{O}$.

4.5.5 Usability

Usability of context information depicts how much that piece of context information is suitable to use for the intended purpose. It considers the level of granularity of the collected context object and the level of granularity required by an applications. For example, a context consumer needs the information about the location of a person at the level of granularity of street of his location. But if a context information service provides information only about the current city of his location, that context information will have lower value of usability.

Usability of context information will be measured by comparing the granularity of context information presented by the context object and the granularity of context object that is required by a context consumer. Granularity of a context object is assigned a level in the context model considering the detail of information. For example, the location information of a person is assigned the highest level if it gives the information about the country, city, street, building, and room of the current location of the person and will be assigned the lowest granularity level if context object only gives the information about the country of the current location of that person. The following equation shows the evaluation of this metric.

$$Usability(\mathcal{O}) = \begin{cases} 1 & : \quad if \quad GranularityLevel(\mathcal{O}) >= GranularityLevel(\mathcal{CR}) \\ 0 & : \quad otherwise \end{cases} \tag{4.11}$$

As shown by the above equation usability of context object will be equal to 1 if the granularity level of the context information presented by context object $\mathcal{O}$ is greater than the granularity level of of context information requested by context consumer. Otherwise it will be 0.

4.5.6 Access Right

Access Right of context information is also a subjective QoC metric and will vary depending upon context consumer who is going to access that context information. *Access Right* of a context object will be calculated by comparing the access level of granularity of context information that is allowed by the owner of context to the access level that is required by the context consumer. Access right of a context object $\mathcal{O}$ is calculated by the following equation.

$$AccessRight(\mathcal{O}) = \begin{cases} 1 & : \quad if \quad AccessLevel(\mathcal{O}) >= AccessLevel(\mathcal{CR}) \\ 0 & : \quad otherwise \end{cases} \tag{4.12}$$

As shown by the above equation, access right will be 1, i.e., context consumer will be allowed to access that context object if access level of context object $\mathcal{O}$ allowed by the context owner is greater than or equal to access level requested by context consumer.

4.5.7 Representation Consistency

Representation Consistency of a context object depends upon the amount of effort that is needed to transform that context object according to the data model presented by context consumer. If similar data formats are used by sensors and context consumer then representation consistency will have maximum value. Otherwise the value of representation consistency decrease with increase in effort needed to transform the context object according to the requirements of context consumer as shown by the following equation.

$$RepresentationConsistency(\mathcal{O}) = \frac{k}{TransformationEffort} \tag{4.13}$$

As shown by the above equation, representation consistency of a context object $\mathcal{O}$ will be inversely proportional to the effort that is needed to transform that context object according to context consumer requirements. k is a constant that is used to normalize the value of representational consistency in range [0..1] and its value depends upon the maximum and minimum effort that a context consumer perform for the transformation of data.

4.6 QoC Evaluator

QoC Evaluator uses sensor characteristics, measurement context, context specifications and context consumer requirements to evaluate QoC metrics in range [0..1], where 0 indicate the lowest level of quality and 1 indicate highest level of quality. Algorithm 1 illustrates the steps used to evaluate QoC metrics. Algorithm 1 takes sensor characteristics, measurement context, context specifications, consumer requirement, and QoC criteria as input and calculate QoC metrics as output.

Line 2 starts a loop that runs for every QoC metric included in the *QoC Criteria* and evaluate its value. Line 3 initialize the *qocSource* with a null value. *qocSource* will contain all the information that is needed to evaluate QoC metrics. Line 4 start a loop to check every QoC source in sensor characteristics whether it is needed to evaluate that specific QoC metric, if yes include it to the *qocSource* in the body of the loop. Line 9 starts the same loop for measurement context passed to the algorithm and add all attributes required to evaluate the concerned QoC metric in *QoCSource*. Similar loop also run for specifications and consumer requirements on Line 14 and Line 19 respectively. The algorithm passes *qocSource* and the concerned QoC metrics to a function to calculate QoC metric and add that metrics to *qocMetric*. First loop starting at Line 2 when it evaluates all the required QoC metrics. Finally, the algorithm returns the evaluated QoC metrics back at Line 26.

4.7 Enrichment of Context Information Model with QoC

Developing a model to represent context information is of foremost importance in a context-aware system. Without a suitable model it is impossible to provide context information. The main considerations to model context information are: What are the system requirements? What is the purpose of collecting information? How is the information going to be used? Who will use this information? Another factor in the provisioning of context information is to deliver the user with only necessary information considering his current situation.

Algorithm 1 QoC metrics evaluation

INPUT: SensorCharacteristics *sc*, MeasurementContext *mc*, Specifications *sp*, ConsumerRequirements *cr*, QoCCriteria *qocCriteria*

OUTPUT: QoCMetrics *qocMetrics*

```
1: QoCMetrics qocMetrics ← null
2: for each QoCMetric qm_i in qocCriteria do
3:     QoCSources qocSources ← null
4:     for each attribute sc_j in sc do
5:         if sc_j is required to calculate qm_i then
6:             qocSources.sc_j ← sc.sc_j
7:         end if
8:     end for
9:     for each attribute mc_j in mc do
10:        if mc_j is required to calculate qm_i then
11:            qocSources.mc_j ← mc.mc_j
12:        end if
13:    end for
14:    for each attribute sp_j in sp do
15:        if sp_j is required to calculate qm_i then
16:            qocSource.sp_j ← qm.qm_j
17:        end if
18:    end for
19:    for each attribute cr_j in cr do
20:        if cr_j is required to calculate qm_i then
21:            qocSource.cr_j ← qocSource.cr_j
22:        end if
23:    end for
24:    qocMetrics.qm_i ← calculateMetric(qm_i, qs_i)
25: end for
26: return qocMetrics
```

To achieve these requirements, our model to provide context information is based on W4H (de Freitas Bulcao Neto and da Graca Campos Pimentel, 2005) and granular

context model (Dorn, Schall, and Dustdar, 2006). W4H model present the five semantic definitions of a context model: identity (who), location (where), time (when), action (what), and device profiles (how) (Abowd, Mynatt, and Rodden, 2002; Truong, Abowd, and Brotherton, 2001). *Who* dimension of the context model present the entities in the environment. These entities include human users, sensors, and actuators that have an interaction with the environment. Apart from the identities of these entities context models can also take advantage of their roles in the environment. For example, in a smart office environment different personnel have different tasks to do, such as finance manager, human resource manager, and marketing manager. Context-aware applications adapt the environment and behavior according to their roles. *Where* dimension of a context model deals with the spatial attribute of a pervasive environment and represent the location of an interaction or context capture. As pervasive environments contain both static as well as dynamic entities, *Where* dimension is very significant to understand the current situation in the environment.

When dimension of a context model present the temporal aspect of different actions taking place in a pervasive environment. It also include the information about intervals between different events. Such information help the context-aware applications in proactive anticipation of different events in pervasive environments. *What* dimension is related to the details of the event that have actually happened in the environment. It can be a very low level context collected by a sensor, such as room temperature, or high level context extracted after combining data gathered from different sensors, such as human user is cooking. *What* dimension present the information necessary for the working for any context-aware application and is the core of a context model. *How* dimension of a context model deal with the source of context information. It presents the hardware and software that have been used to collect the context information. This dimension includes not only the physical sensors that have been used to collect sensor data but will also include the algorithms and techniques that have been used to extract high level context information from sensor data.

Pervasive environments characterized by continuously evolving uninterrupted supply of sensor data. A very little amount of these avalanches of sensor data may be relevant to perform the functionality of a specific application. Applications may also desire to have a specific type of context at a certain level of detail. For example, human

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:complexType name ="QoCMetricsType">
    <xs:sequence>
        <xs:element name="Reliability" type="tns:QualityDecimal"/>
        <xs:element name="Timeliness" type="tns:QualityDecimal"/>
        <xs:element name="Completeness" type="tns:QualityDecimal"/>
        <xs:element name="Significance" type="tns:QualityDecimal"/>
        <xs:element name="Usability" type="tns:QualityDecimal"/>
        <xs:element name="AccessRight" type="tns:QualityDecimal"/>
        <xs:element name="RepresentationConsistency" type="tns:QualityDecimal"/>
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="QualityDecimal">
    <xs:restriction base="xs:decimal">
        <xs:minInclusive value="0"></xs:minInclusive>
        <xs:maxInclusive value="1"></xs:maxInclusive>
    </xs:restriction>
</xs:simpleType>
```

Figure 4.3: (Simplified) XML schema representation of QoC metrics

activity may described as “making breakfast” at higher level of abstraction and “open drawer”, “get plate”, “cut bread” at lower level of abstraction. Transferring only necessary relevant context may also increase an application responsiveness and decrease the cost of context, such as network bandwidth. To achieve these objectives we also use the concept of granular context in our model and present context at different levels of detail. Every context object in our context model is annotated with QoC metrics to enable the context consumers to be aware of the quality of context information. Figure 4.3 shows the XML representation of the QoC-metrics. Every QoC-metric is of type QoC-Decimal which is defined as the restricted type over decimal to have the value in range [0.0, 1.0].

4.8 Summary

In this chapter we presented our novel QoC metrics that present the objective and subjective view of the QoC as the worth of context information for a specific context consumer. We also presented the process of the evaluation of QoC metrics. Later, we presented our context model enriched with QoC metrics. In the next chapter we describe our confidence inference system. Confidence inference system infer the values of confidence on context information considering different QoC metrics tailored to the needs of a specific context consumer.

Chapter 5

Confidence Inference System

5.1 Overview

Confidence inference system infers the value of confidence on context by combining QoC metrics according to the requirements of a particular context consumer. Figure 5.1 shows different components of confidence inference system. Context sensor services that present the virtual sensors gather sensor data from physical sensors and extract the high level context information. High level context information is provided to QoC evaluator along with sensor characteristics and context of the sensor data collection and information extraction. QoC evaluator combines this information with context consumer requirements to evaluate QoC metrics. Confidence inference system further uses QoC metrics to infer the value of confidence on context information. We discussed the detail of sensor services, QoC sources, QoC metrics, and QoC evaluator in Chapter 4. Here we will describe the concept of confidence on context, context consumer service, and the detail of confidence inference system.

5.2 Confidence on Context

Confidence on context information is a multidimensional quantity that is used to present quality of context information from various aspects. Confidence on context

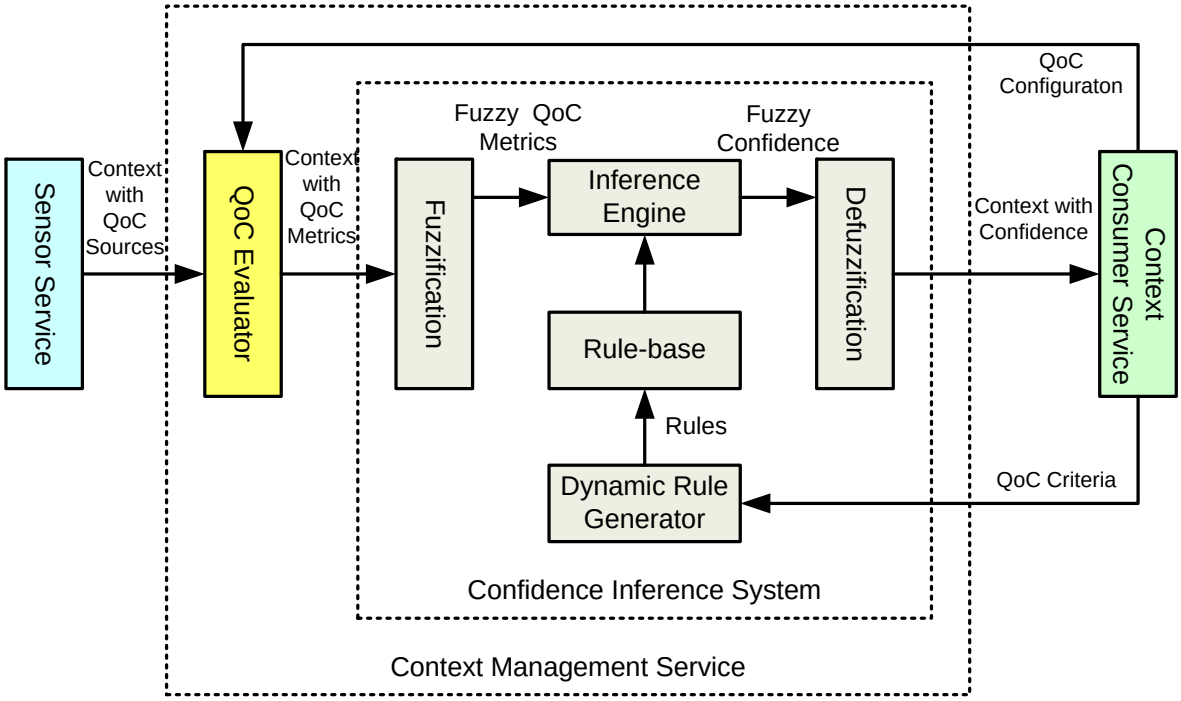

Figure 5.1: Context Management Service with QoC Evaluator and Confidence Inference System

indicates by a single metric that how much a context objects is free of errors, valid to use, and relevant to perform a specific task by a particular context consumer and liberates the context consumers from the extra effort to reason about the context or QoC metrics. Different QoC metrics such as reliability, timeliness, completeness, significance, and usability are combined to provide the value of confidence on context.

These QoC metrics can be combined according to the requirements of a particular context consumer to infer the value of confidence on context information. In this chapter we will describe the sources of confidence on context, i.e., QoC metrics and the context consumer requirements, the confidence inference system that uses fuzzy logic to infer the value of confidence, and the dynamic rule generator that generates the rules according to context consumer requirements. These rules are used by our confidence inference system to infer the value of confidence on context.

5.3 Context Consumer Request

Every context consumer has different sets of requirements considering QoC metrics. For example, a context consumer application that has been installed in a smart home to deal with emergency situations is much more concerned about the reliability of context information than another application that maintains the home entertainment system. Therefor, considering the requirements of a specific consumer regarding QoC are indispensable to infer confidence. There must also be a simple mechanism for context consumer to specify their requirements. In our system context consumers mention their requirements by selecting the linguistic values of a QoC metric based fuzzy variable. For example, we defined the linguistic values of reliability based fuzzy variable as *Very High*, *High*, *Medium*, *Low*, *Very Low* and context consumers can indicate their requirements regarding reliability of context using any of these linguistic values. Context consumers can also optionally mention the threshold value for confidence on context. If a context consumer mentions the threshold value of confidence then context distribution system only forward the context objects that have the confidence on context higher than the threshold value. Otherwise a context consumer will receive all context objects with value of confidence on context. Figure 5.2 shows a context consumer request for the context of type user's activity and QoC requirements *VeryHigh*, *Fresh*, and *Vital* for reliability, timeliness, and significance respectively, and threshold value of 0.8. The requirements are used to dynamically generate the rules to infer confidence on context information as described in Section 5.4.3 and Section 5.4.2.

5.4 Confidence Inference System

We have used fuzzy logic (Zadeh, 1996) to infer the value of confidence on context as shown by the *Confidence Inference System (CIS)* in Figure 5.1. The first step to use any fuzzy inference system is to define fuzzy variables corresponding to all input base numerical variables and the membership functions that maps the numerical value of base variables to linguistic values of fuzzy variables. We define fuzzy variables and their membership functions for every QoC metric that are used by the *Fuzzification* to map QoC metrics provided as an input to the system to their fuzzy values. The

Rule-base holds the knowledge in the form of a set of rules to make the inference. These rules are generated by the *Dynamic Rule Generator* considering the requirements of the concerned application. Fuzzy values of QoC metrics are provided to the *Inference Engine* that uses the rules in the *Rule-base* with generalized modus ponen for making inference from input fuzzy variables QoC parameters to output fuzzy variable confidence. Defuzzification is the process of converting the fuzzy variable value back into a numerical value. The context management system passes the value of confidence along with context object to the context consumer. In the remaining section we will discuss each component of confidence inference system in detail. In the remaining section we will discuss all the steps to infer confidence on context considering an example context consumer request shown in Figure 5.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<QoCCriteria contextType = ‘‘UserActivity’’>
    <QoCRequirement>
        <reliability>VeryHigh</reliability>
        <timeliness>Fresh</timeliness>
        <significance>Vital</significance>
    </QoCRequirement>
    <threshold>0.8</threshold>
</QoCCriteria>
```

Figure 5.2: An example of QoC criteria

5.4.1 Fuzzification

Fuzzification is the process of converting the base variables to fuzzy values for those variables. Numerical values of QoC metrics work as the base variables in our *CIS*. The first step in fuzzification of a base variable is to define fuzzy variable corresponding to every base variable that is used to infer output value. An example of QoC criteria to evaluate confidence on context is shown in Figure 5.2. QoC criteria has *Reliability*, *Timeliness*, and *Significance* as the QoC metrics that should be used to evaluate confidence. We define the corresponding fuzzy variables as for *Reliability* that can

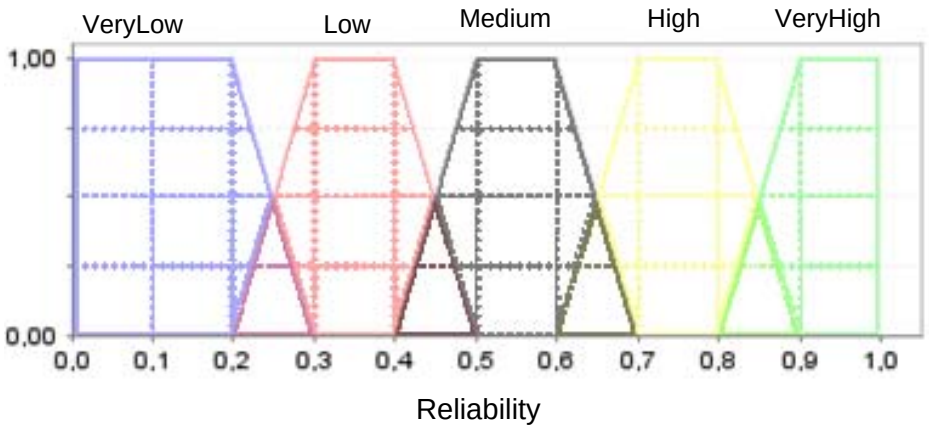

Figure 5.3: Membership function for fuzzy variable Reliability

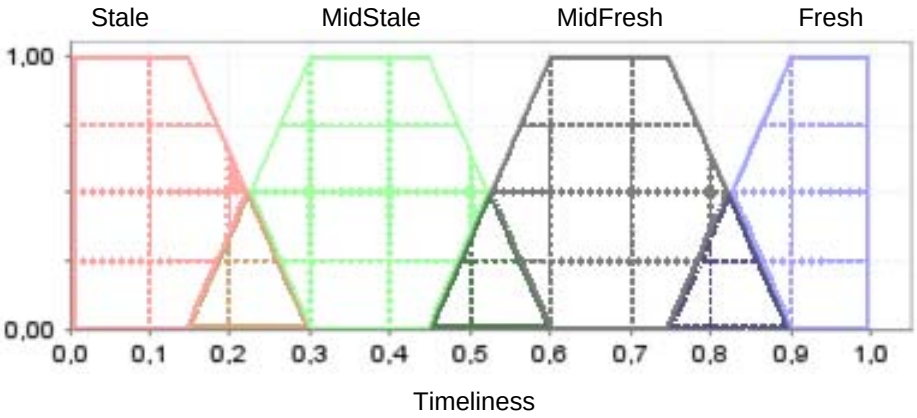

Figure 5.4: Membership function for fuzzy variable Timeliness

have the the fuzzy values *VeryHigh*, *High*, *Medium*, *Low*, and *VeryLow* as shown in Figure 5.3. Fuzzy variable *Timeliness* can have the fuzzy values *Stale*, *MidStale*, *Mid-Fresh*, and *Fresh* and fuzzy variable *Significance* can have the fuzzy values *Negligible*, *Mid Negligible*, *Mid Vital*, and *Vital* as shown in Figure 5.4 and Figure 5.5 respectively.

Fuzzy membership functions are used to convert base numerical variables to to fuzzy values of corresponding fuzzy variables. For example, Figure 5.6 shows the conversion of numerical values of base variables *Reliability*, *Timeliness*, and *Significance* to the fuzzy values of their corresponding variables. Contrary to crisp set membership function fuzzy membership function allows a fuzzy variable to have membership of more than one value to some extent. For example base variable *Reliability* have numerical value

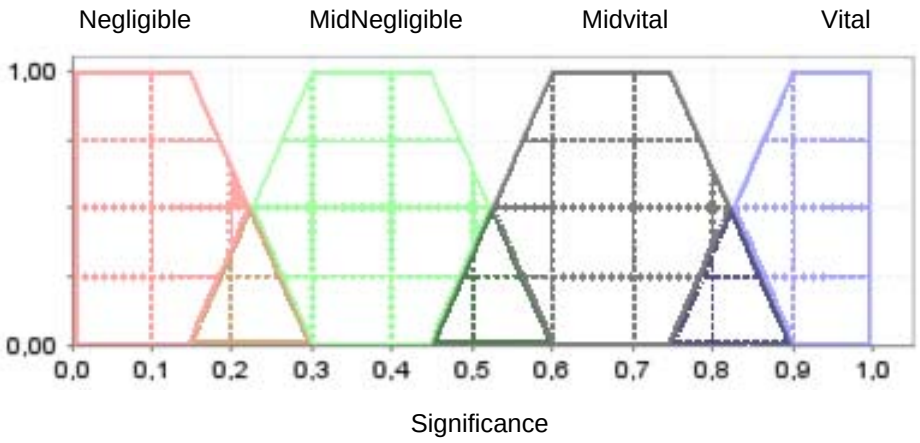

Figure 5.5: Membership function for fuzzy variable Significance

0.85 has membership to fuzzy values *VeryHigh* and *High* up to the extent of 0.5 as shown in Figure 5.6. Similarly numerical value 0.79 of base variable *Significance* has membership of two fuzzy values *MidVital* and *Vital* to some extent. Numerical value 0.65 of base variable *Timeliness* has full membership of the fuzzy value of *MidFresh* of fuzzy variable *Timeliness*.

5.4.2 Rule-Base and Dynamic Rule Generator

The *Dynamic Rule Generator* takes QoC criteria, discussed in Section 5.3, defined by a context consumer as an input and dynamically generates the rules. These rules are added in the *Rule-base* and used by the *Inference Engine* to infer the value of confidence on context. Algorithm 2 provides the procedure for generating the rules. The algorithm starts with a nested *for* loop at Line 2 that will run for each fuzzy value of QoC metrics indicated in QoC requirements. The first block of Algorithm 2 at Lines 4-5 checks whether the current fuzzy value of a particular QoC metric is greater than or equal to required fuzzy value of that particular QoC metric. If the condition is true then that fuzzy value of QoC metric is mapped to maximum value of confidence on context. For example, for QoC requirement indicated in Figure 5.2 we will add the following rules to map significance of context to confidence.

IF `Significance is Vital` **THEN** `Confidence is VeryHigh`.

The second block of Algorithm 2 starting from Line 7 checks whether the current fuzzy value of a QoC metric is minimum for that QoC metric or not. If this condition is true then the algorithm generates the rule to map the minimum value of QoC metric to the minimum value of confidence on context. For example, the algorithm will generate the following rule to map the minimum value of significance of context to minimum value of confidence on context.

IF `Significance is Negligible` **THEN** `Confidence is VeryLow`.

The third and final block of Algorithm 2 starting from Line 10 maps all the remaining fuzzy values of QoC metrics to fuzzy values of confidence on context. These fuzzy values of QoC metric will be less than the QoC requirement set by a context consumer and greater than the minimum fuzzy value that can be assigned to that QoC metric. The algorithm gets the index of fuzzy value of confidence as the ratio of the difference between maximum and minimum fuzzy value of confidence to difference of current fuzzy value of a QoC metric value and fuzzy value of that QoC metric that has been assigned in QoC requirements. For example, for QoC requirements indicated in Figure 5.2, following rules will be generated to map significance of context information to confidence on context.

IF `Significance is MidVital` **THEN** `Confidence is High`

IF `Significance is MidNegligible` **THEN** `Confidence is Low`

Similarly, the algorithm generates the rule to map fuzzy values of all QoC metrics mentioned in QoC requirements. These rules are added to the *Rule-base*. The *Inference Engine* uses them to infer the value of confidence on context by combining the QoC metrics. As these rules are generated considering the QoC requirements set by a context consumer, the confidence value reflects quality and relevance of context for each particular context consumer. Figure

5.4.3 Inference Engine

Once fuzzy QoC variables are assigned the values based on base numerical QoC metrics, the inference engine uses the rules available in the rule base to infer the fuzzy value of the output fuzzy variable confidence on context. Figure 5.6 shows the triggered rules

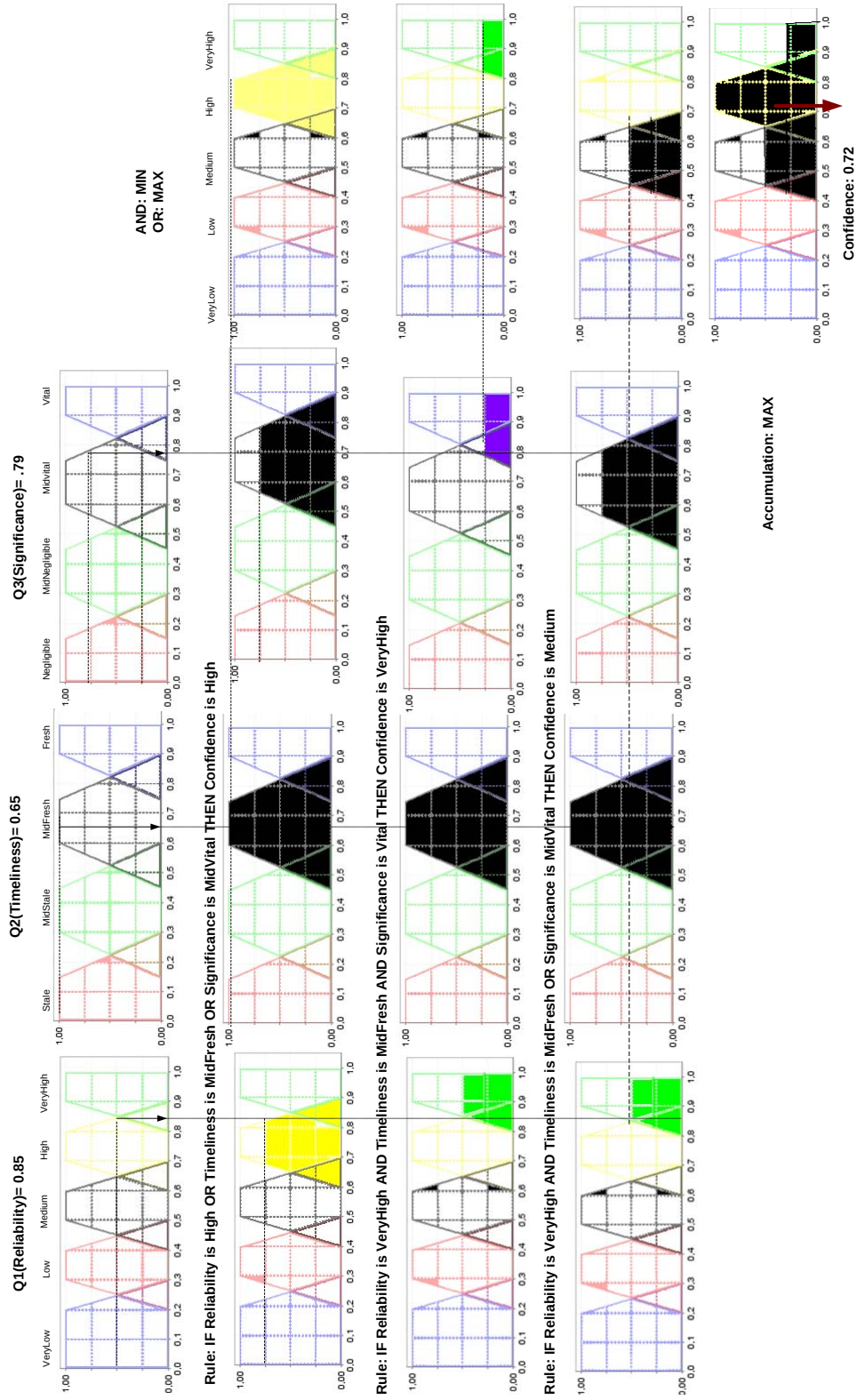

Figure 5.6: Fuzzy process to infer confidence on context from QoC metrics

Algorithm 2 Rule generation according to QoC requirement set by a context consumer

INPUT: QoCRequirement qocReq
OUTPUT: GeneratedRules gRules

1: GeneratedRules $gRules \leftarrow null$
2: **for** each QoCMetric qc_i in QoCRequirement **do**
3: **for** each FV (Fuzzy Value) fv_j in QoCMetric qc_i **do**
4: **if** $qc_i.fv_j >= qocReq.qc_i.FV$ **then**
5: gRules.addRule(IF qc_i is fv_j THEN Confidence is Confidence.FV.MAX)
6: **else**
7: **if** $qc_i.fv_j$ is MINIMUM **then**
8: gRules.addRule(IF qc_i is fv_j THEN Confidence is Confidence.FV.MIN)
9: **else**
10: i$\leftarrow \frac{Confidence.FV.MAX.Index - Confidence.FV.MIN.Index}{qocReq.qc_i.FV.Index - qc_i.FV.Index}$
11: gRules.addRule(IF qc_i is fv_j THEN Confidence is Confidence.FV[i-1])
12: **end if**
13: **end if**
14: **end for**
15: **end for**
16: **return** $gRules$

and the process of converting the fuzzy values of input variables *Reliability*, *Timeliness*, and *Significance* to fuzzy value of *Confidence*. *MIN* and *MAX* are used for the *AND* and *OR* operators respectively. *MAX* operator is used while accumulating the fuzzy values of confidence.

The inference engine uses the rules in the rule-base with generalized modus ponen for making inference from input fuzzy variables QoC parameters to output fuzzy variable confidence. Classical modus ponen rule of inference is defined as if it is known that a statement A is true and $A \rightarrow B$ is also true, then it can be inferred that B is true. Fuzzy logic generalized classical modus ponen to generalized modus ponen. Generalized modus ponen allows $\acute{A}$ and $\acute{B}$ to be little different from A and B. Consequently, if $\acute{A}$ is true and $A \rightarrow B$ is also true, then it can be inferred that $\acute{B}$ is true. After evaluating all the rules and getting the results from the rules, finally the inference engine accumulates these results in a final fuzzy value of confidence.

5.4.4 Defuzzification

The Center of Gravity (COG) is a popular technique to determine the numerical value from the fuzzy value and it is used to get the value of confidence as follows.

$$C = \frac{\int_{min}^{max} x\mu_C(x)dx}{\int_{min}^{max} \mu_C(x)dx} \tag{5.1}$$

where C is the numerical value of confidence that we will get after the process of defuzzification, x is the output variable and $\mu_C(x)$ is the confidence membership function for corresponding value of xi.

5.5 Summary

This chapter presented our confidence inference system that infer the value of confidence on context by combining QoC metrics according to the requirements of a particular context consumer. We have used fuzzy logic to facilitate context consumer by providing the easiness of mentioning his QoC requirements in linguistic terms. Confidence on context indicate the quality of context information from different aspects, such as reliability, timeliness, completeness, and significance of context information, considering the needs of specific context consumer for that metrics. In the next chapter we present the conflict resolving policies that can be used to resolve conflicts at different conceptual layers of a context-aware system. These policies can be based on a single or more QoC metric based on the requirement of a specific context consumer.

Chapter 6

QoC based Conflict Resolving Techniques

6.1 Overview

Context-aware systems in pervasive environments face many conflicting situations while collecting sensor data, processing sensor data to extract consistent and coherent high level context information, and disseminating that context information to assist in making decisions to adapt to the continuously evolving situations without diverting human attention. These conflicting situations pose stern challenges to the design and development of context-aware systems by making it extremely complicated and error-prone. QoC metrics can be used to cope with these challenges. In this chapter, we discuss the conflicting situations that a context-aware system may face at different layers of its conceptual design and present the conflict resolving policies that are defined on the basis of the Quality of Context metrics.

6.2 Conflicting Situations in Context-Aware Systems

As we discuss in Chapter 2 mostly context-aware systems are conceptually designed as a layered framework. These layers are assigned the task of collecting sensor data, extracting high level context information from this data, aggregating and storing context information after eliminating redundant and inconsistent context, providing this information to interested applications and users, and finally applications and users take actions to adapt themselves to this information. Different conflicting situations can arise during the execution of the aforementioned tasks. These conflicting situations strongly affect the capability of context-aware systems to adapt to the evolving situation in pervasive environments. In this section we discuss the conflicts that can take place at different layers of a context-aware system in performing different tasks. Table 6.1 provides the summery of these conflicts corresponding to each layer.

6.2.1 Sensing

Sensing is the lower most layer of a context-aware system and is responsible for collecting sensor data. The different sensors and sensing modalities in the pervasive environments, collecting data with different capabilities at different times, generate many conflicting situations to select a suitable sensor and sensing modality to collect sensor data. Here we discuss conflicting situations that can occur at sensing layer in performing the tasks of sensor selections and sensing modalities selection.

6.2.1.1 Sensor Selection

Sensor data collected by different sensors may differ with each other considering the frequency of updates, the capability of a sensor to collect the context of an entity, the accuracy of a method that is used by sensors, representation format, and the cost of collecting that data (Cook, 2007; Chantzara, Anagnostou, and Sykas, 2006). Problems also arise due to the mobility of sensors, computing devices, and entities in pervasive environments. Users — carrying portable computing devices and wearing sensors on

Conceptual Framework Layers	*Conflicts*	*Examples*
Sensing	Sensor selection (Cook, 2007; Chantzara et al., 2006) , sensing modality selection	Different type of sensors, such as wearable sensors and ambient sensors, can be used to extract context about the current activity of a user
Context extraction	Feature selection, classifier selection, conflicts in context extracted from different classifiers(Wun et al., 2007; Wu et al., 2003; Huebscher et al., 2006)	Different classifiers have different accuracy in classifying different type of features
Context management	Context aggregation, Context storage (Perich et al., 2004)	Redundant and inconsistent data reaching a node from different routs
Context consumer	Conflicting interests of applications (Park et al., 2005)	Different preferences set by two users present in living room

Table 6.1: Conflicting situations at different layers of a context-aware system

their limbs — not only move among different smart environments, such as smart home, smart car, smart office, and smart conference rooms but also at many different places inside these environments, such as living room, kitchen, and bedroom in a smart home. Different sensing devices and context extraction computing resources will be available during these transitions. More than one sensor may also be providing information about the same event in the environment. We cannot permanently rank a sensor to collect the context of a particular entity. For example, we can use the ambient sensors installed in the kitchen area to collect data about a user who is sitting in the living room. So there is a need of a strategy that can dynamically decide which sensor is more reliable to collect the context of a certain entity at some specific time. QoC metrics, such as reliability, that not only consider the capabilities of sensor to collect data but also consider its suitability to collect data about a certain event at a given instance of time can be used to resolve conflicts in such situations and effectively select a suitable sensor.

6.2.1.2 Sensing Modalities Selection

Sensor data can be collected from different sensing modalities to extract high level context information. For example, context information about the current user activity can be extracted using the wearable sensors and ambient sensors. Wearable sensors may give better results that involve more movements of human limbs, such as walking, while ambient sensors may also give better results in activities that involve a lot of interaction with objects available in the environments, such as making dinner. Selecting from different available sensing modalities to extract a particular type of context is one of the conflicting situation that can be faced at design time of a context-aware pervasive computing system. Such conflicting situation can be resolved to use QoC metrics, such as reliability of context extracted from different sensing modalities.

6.2.2 Context Processing

Sensor data cannot be presented directly to applications. It needs to be filtered, fused, correlated, and translated to extract the higher level context data and detect the emergent events (Wun, Petrovi, and Jacobsen, 2007). Context processing layer is responsible for extracting high level context from the sensor data. Figure 6.1 shows the process of the extraction of high level context information. At first stages features are extracted from the acceleration based sensor data. These features are further selected and provided to machine learning algorithms, such as classifiers, to extract high level context information. Here we discuss conflicting situations that can occur at context processing layer while performing the task of feature selection and classifier selection.

6.2.2.1 Feature Extraction

Classifiers use the features extracted from sensor data to classify high level context, such as human activity. Different set of features can be used to extract high level context information, for example walking, used plates, and used coffee machine are different type of features that can be used to extract human activity at higher level of detail. Feature extraction is the process to select most optimal list of features for high level context extraction. Another conflicting situation can be faced while deciding about the

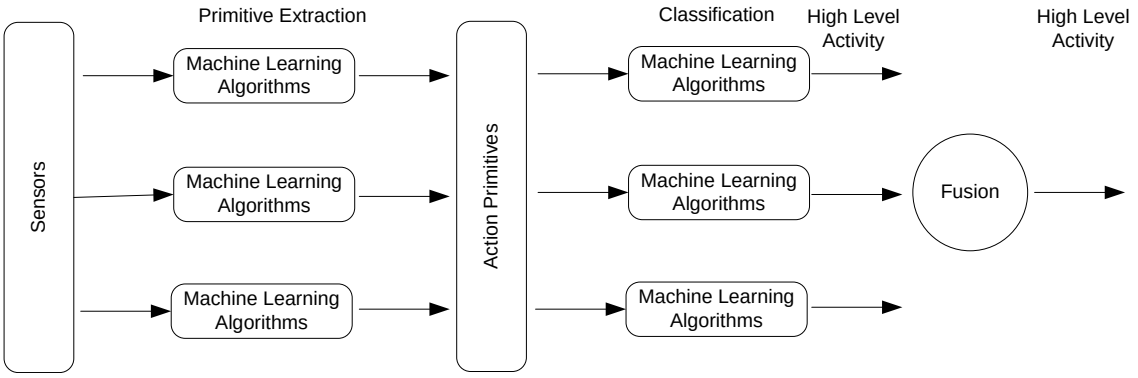

Figure 6.1: High level context extraction from sensor data

size of time window that can be used to collect those features. For example, features can be extracted from sensor data of the window of five seconds, ten seconds, or sixty seconds. Depending upon different choices in aforementioned situations, classifiers may not only extract high level context with different reliability but will also take different amount of time. QoC metrics, such as reliability and timeliness, can be used to resolve conflicts in these situations.

6.2.2.2 Classifier Selection

Classifier distinguish among themselves on the basis of using training data, time required for training the classifier, time required to classify a particular instance, and reliability of the classifier to classify correctly. One of the most conflicting situation is to select a particular classifier on the basis of its attributes that suits to use that classifier in a particular situation. QoC metrics can be used to rank the classifiers for different situations and select the best one at a particular instance of time.

6.2.3 Context Management

Once the context is extracted at the lower layers of a context-aware system, it is provided to context management layer that is responsible for aggregating and storing the context according to an effective context model and disseminate it to different context consumers. Here we discuss the conflicting situations that can occur at the

context management layer while performing aforementioned tasks.

6.2.3.1 Context Aggregation and Storage

The high mobility of sensors, unreliable wireless connections, and the nature of tasks in pervasive environments result in the acquisition of a lot of redundant and conflicting context. For example, context about the same event in the environment may be extracted at different nodes and disseminated in the environment. Different nodes may have used different sensors and classifiers to extract this context. Resultantly different context objects may not only have redundant but also conflicting information. This redundant and conflicting context not only results in the wastage of scare resources, such as memory, but also can lead to undesired behavior of context-aware applications. Sensor data used to extract high level context may also be collected at different times. These circumstances make it a tough decision to make selection among two context objects presenting the same entity in the real world. This is one of the conflicting situations that context-aware pervasive computing systems can face at run time. Simple conflict resolving policies, such as drop first, drop all, can result in deleting some valuable information. In critical situation, such as a context-aware ubiquitous home for patients Kim and Lee (2006b) and telehealth applications Kara and Dragoi (2007), loss of information can result in severe situations for the people using it. Decisions can better be made to discard or keep a context object on the basis of policies defined using these QoC metrics that depict the quality of context from different prospects, such as reliability of sensors, timeliness of context, and completeness of context.

6.2.3.2 Context Distribution

Different context consumers subscribe to context information management services to get context information. Pervasive environments generate avalanches of sensor data. Sensor data is collected and transformed to high level context information. Lower layers of a context-aware system also produce different type of context and provide it to the context management layer for the distributing it to different context consumers. All this context is not relevant to the functionality of a specific context consumer. For example, an application responsible to maintain a comfortable temperature in a smart

home with optimal power consumption will only be interested to get information about the current temperatures at different places of a smart home and location of people living in that home. This application will not be interested in other type of context, such as whether the lights of the living room are on or off. While another application that is responsible for maintaining the house ambiance with optimal power consumption will be very much interested in the later discussed type of context.

Subjective QoC metrics, such as significance of context that indicate the critical value of context information for a specific application can be used to generate events of interest for specific application. Policies based on such QoC metrics can also save the context consumers from performing many tasks such as filtering of context to get relevant information.

6.2.4 Context Consumers

Context-aware applications use context information to adapt their behavior to user needs and changes in the environment. If conflicts are not resolved in context information at the earlier stages, the applications that take actions on the basis of that context information get in conflict while making decisions. Decisions made on the basis of unreliable and conflicting context can also be the cause of inconvenience for the user of the system. For example, switching on or off the lights at wrong moments can be quite irritating for people living in the home. QoC metrics can be used to resolve such conflicting situations and provide an easy living environment for the users with optimal use of resources. Similarly, resolving conflicts at the earlier stages can release the context-aware applications from this additional liability to do reasoning about the quality of context information and applications can concentrate on their task to make prompt actions with change in context.

6.3 QoC Based Conflict Resolving Policies

Earlier systems have used some simple conflict resolving strategies such as drop all, drop last, drop first (Xu, Cheung, Chan, and Ye, 2007), involved user to resolve conflicts

(Dey and Mankoff, 2005), or do the mediation on the basis of some predefined static policies (Park, Lee, and Hyun, 2005). These strategies may slow down the process of decision making, distract user, or discard some important context objects as well. Amount of context generated in pervasive environments also make it impossible for the human user to be the part of any meditation process (Cook, 2007). Most of the context conflicts can also not be resolved at design time(Capra, Emmerich, and Mascolo, 2003) and need a strategy that can dynamically handle them at runtime without distracting user. Quality of context (QoC) metrics can be used to devise the policies to resolve the conflicts at different layers of a conceptual framework of context-aware system.

The main consideration of these policies is to resolve the conflicts in such a way that the decision should have been taken in the favor of context object that contains the context information of the highest quality. This quality of context information is characterized by QoC metrics. Chapter 4 discusses the QoC metrics and the methods that have been used to evaluate these QoC metrics in range [0..1]. These policies can be based on a single or more QoC metric. In case of more QoC metrics confidence on context can be evaluated as described in Chapter 5. Context consumer can use these policies to select a single best context object or he can also mention a minimum threshold level so that all the context objects having quality more than that threshold should be selected. Appendix A shows XML schema to create an XML instance of conflicting resolving entity. We have also taken into account the user centered design of context-aware systems and tried that human users of the system should not be distracted during the execution of these policies. In this section we discuss the policies that can be based on different QoC metrics and the situation where these policies can be useful.

6.3.1 Reliability Based Policy

Reliability of the context information not only depicts the accuracy and the precision of the sensors and the context classification methods but also indicate the information about the suitability of a particular sensor to collect that type of context about a particular entity in the environment. Process of the evaluation of reliability of context information has been described in Chapter 4. These attributes of this QoC metrics

make it particularly useful in resolving the conflict when we have more than one sensor collecting the context of same entity or event. For example, we have temperature sensors at different places in the living room of a smart home. The sensors that are installed near the electric radiator heater will be sending the higher value of the temperature of living room as compared to the sensors in the other places in the living room. To provide a comfortable temperature in the room we rely more on the readings of the sensors, that are closer to the sitting area than the sensors in the far off corners of the living room and sensor near the radiator. Reliability of the context collected by the sensors will also have higher value as we also involve distance of the sensor from the concerned entity to evaluate it as described in Chapter 4. Reliability of context information can also play a significant role in selecting suitable sensing modalities for extracting a specific type of context information. Reliability can also be a part of policies to select the classification algorithms to extract high level context information. Figure 6.2 shows a conflict resolving policy based on the reliability of context information. Fuzzy value of the reliability of context information has been set as *VeryHigh* and threshold value is set as 0.90.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConflictResolvingPolicy >
    <QoCCreteria>
        <Reliability>
            VeryHigh
        </Reliability>
    </QoCCreteria>
    <Threshold>
        0.90
    </Threshold>
</ConflictResolvingPolicy>
```

Figure 6.2: An instance of reliability based conflict resolving policy

6.3.2 Timeliness Based Policy

Timeliness of context information is the most important characteristic of context information that indicates the degree of rationalism to use a context object at a specific instance of time. This metric can be useful in resolving the conflict in those context objects that changes its value very rapidly, e.g., location of a fast moving vehicle. In this case, it will be more suitable to use the context object with the highest value of timeliness. Whereas, timeliness will not have much role in the case of conflict in static information that have been profiled in the system, e.g., information about the structure of a smart home. Figure 6.3 shows a simplified XML representation of the conflict resolving policy based on timeliness of context information. This policy provides *MidFresh* as the minimum requirement for the timeliness of context information. Timeliness can also be very useful in resolving conflicts when used in combination with other QoC metrics, such as reliability of context information.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConflictResolvingPolicy >
    <QoCCreteria>
        <Timeliness>
            MidFresh
        </Timeliness>
    </QoCCreteria>
    <Threshold>
        0.75
    </Threshold>
</ConflictResolvingPolicy>
```

Figure 6.3: An instance of timeliness based conflict resolving policy

6.3.3 Completeness Based Policy

Completeness of a context object is particularly important to get the complete picture of the current situation of real world. According to this policy decision is made in the favour of that context object which has more complete information about current

situation. Completeness of context information is particularly important to resolve conflicts at higher level of context-aware systems to perform tasks such as context aggregation, context storage, and context distribution that use context information at high level of abstraction. Completeness of context information can also be used to resolve conflicts between two context objects while providing context to context consumers. Completeness is particularly important in those cases when two context object have same values for other QoC metrics, such as timeliness and reliability of context information. Figure 6.4 shows the XML representation of a conflict resolving policy based on completeness of context information.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConflictResolvingPolicy >
    <QoCCreteria>
        <Completeness>
            Ample
        </Completeness>
    </QoCCreteria>
    <Threshold>
        0.95
    </Threshold>
</ConflictResolvingPolicy>
```

Figure 6.4: An instance of completeness based conflict resolving policy

6.3.4 Significance Based Policy

Significance of a context object is particularly important in resolving conflicts while performing context dissemination task. Significance can be used to decide whether a context object contain worthwhile information for a specific type of application or not. Significance based policies can resolve the conflict to decide whether a particular context object should be forwarded to a particular application or not. For example, if smoke sensors detect heavy smoke in bedroom, it is the information of high significance for emergency response application that can start water sprinkles and set off fire alarm.

This metric can also be used to generate events that need prompt actions from the applications. Applications can specify that the context objects with high value of significance should be reported on priority basis. Figure 6.5 shows XML representation of a conflict resolving based on significance of context information.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConflictResolvingPolicy >
    <QoCCreteria>
        <Significance>
            Vital
        </Significance>
    </QoCCreteria>
    <Threshold>
        0.85
    </Threshold>
</ConflictResolvingPolicy>
```

Figure 6.5: An instance of significance based conflict resolving policy

6.3.5 Usability Based Policy

Usability of context object is more suited while resolving conflicts in the selection of a context object for usage by a high level. Usability of context is very important metric for the prospective of the targeted application as the context object will not be of any use for that application if it does meet the usability criteria set by that application even though that context object has high value for other QoC metrics, such as reliability, timeliness, and significance. Figure 6.6 shows XML representation of a conflict resolving policy based on usability of context information.

6.3.6 Access Right Based Policy

A context consumer can have rights to access context at different levels of granularity of context information. Similar to usability of context information if a context consumer

```
<?xml version="1.0" encoding="UTF-8"?>
<ConflictResolvingPolicy >
    <QoCCreteria>
        <Usability>
            Appropriate
        </Usability>
    </QoCCreteria>
    <Threshold>
        1.00
    </Threshold>
</ConflictResolvingPolicy>
```

Figure 6.6: An instance of usability based conflict resolving policy

cannot have the right to access context information at the desired level of granularity he will not be interested to know about other QoC metrics of the context information, such as reliability, timeliness, and completeness. Figure 6.7 shows XML representation of access right based policy.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConflictResolvingPolicy >
    <QoCCreteria>
        <AccessRight>
            Authorized
        </AccessRight>
    </QoCCreteria>
    <Threshold>
        1.00
    </Threshold>
</ConflictResolvingPolicy>
```

Figure 6.7: An instance of access right based conflict resolving policy

```
<?xml version="1.0" encoding="UTF-8"?>
<ConflictResolvingPolicy >
    <QoCCreteria>
        <RepresentationConsistency>
            MidCompatible
        </RepresentationConsistency>
    </QoCCreteria>
    <Threshold>
        0.95
    </Threshold>
</ConflictResolvingPolicy>
```

Figure 6.8: An instance of representation consistency based conflict resolving policy

6.3.7 Representation Consistency Based Policy

Representation consistency of a context object, in combination with other QoC metrics, can play an important role to effectively select a context object for a specific context consumer. Figure 6.8 shows the XML representation of a conflict resolving policy based on the representation consistence of a context object.

6.3.8 Confidence Based Policy

Apart from the above mentioned fundamental policies, policies can also be defined based on two or more QoC metrics depending on the requirements of a particular application. For example, a policy can also be defined by combining QoC parameters, such as *Timeliness* and *Reliability*. In such policies value of confidence based on mentioned QoC metrics is inferred to make decision in case of a conflict. For example, if a context aggregator uses a policy based on the combination of *Timeliness* and *Reliability* of a context object with the threshold value of 0.8, then all the context objects having the value of confidence on context more than 0.8 will be selected. Context consumer can mention the value of threshold according to their requirements considering the perspective of the use of context information. Figure 6.9 shows a conflict resolving policy based on the confidence of context information inferred from the QoC metrics

```
<?xml version="1.0" encoding="UTF-8"?>
<ConflictResolvingPolicy >
    <QoCCreteria>
        <Reliability>
            VeryHigh
        </Reliability>
        <Timeliness>
            MidFresh
        </Timeliness>
        <Completeness>
            Ample
        </Completeness>
        <Significance>
            Vital
        </Significance>
    </QoCCreteria>
    <Threshold>
        0.8
    </Threshold>
</ConflictResolvingPolicy>
```

Figure 6.9: An instance of confidence based conflict resolving policy

reliability, timeliness, completeness, and significance and has the threshold value of confidence on context as 0.8.

6.4 Summary

In this chapter we presented conflict resolving policies based on QoC metrics. These conflict resolving policies can use a single or more QoC metric to define a conflict resolving policies. A threshold value for the selection of more context objects meeting a minimum criteria for the quality of a context object can also be mentioned. Selection of particular QoC metrics and threshold value in a conflict resolving policies is completely dependent upon the functionality and the requirements of a particular

context consumer. In the next chapter we present the experiments performed to show the effectiveness of our approach.

Chapter 7

Experiments and Evaluation

7.1 Overview

In this chapter we perform the experiments to evaluate our approach to calculate QoC metrics as the worth of context information for a specific application. First we describe the data set that we use in these experiments. This data set is collected in a smart home environment to predict human activities of daily living (ADL). Later, we extract the high level context, i.e, ADL, from sensor data and evaluate QoC metrics for that context. We further use QoC metrics in performing different tasks of a context-aware system in our experiments.

7.2 Data Description

In our experiments we used the smart home data set gathered in the EU project OPPORTUNITY[1] for the machine recognition of human activities of daily living. Here we briefly describe the feature of data set that are related to our work. Interested readers may see the detailed description of the data set in (Roggen, Calatroni, Rossi, Holleczek, Föorster, Tröoster, Lukowicz, Bannach, Pirkl, Ferscha, Doppler, Holzmann, Kurz, Holl, Chavarriaga, Creatura, and del R. Millàn, 2010).

[1] http://www.opportunity-project.eu/

Activities	*Description*	*Duration (seconds)*
Idle	Not performing any activity	583
Relaxing	Go outside and have a walk	157
Early morning	Move around in the room and casually check the objects	276
Coffee time	Prepare coffee with milk and sugar using coffee machine and drink it	129
Sandwich time	Prepare sandwich with bread, cheese, and salami using bread cutter, various knives, and plates and eat it	375
Clean up	Put objects used to original place or dish washer and cleanup the table	183

Table 7.1: Activities and their duration during a single run

Data set about the naturalistic human activities is collected in a sensor rich environment: a room simulating a studio flat with kitchen, deckchair, and outdoor access where subjects performed daily morning activities. Fifteen networked sensor systems with seventy two sensors of ten modalities were deployed in the environment, embedded in the objects, and worn on the body. Opportunistic data sensing environment in which many sensors collected data about the same event makes the data set ideal for evaluating QoC for context extracted from data collected by different sensing systems, analyzing the effectiveness of different sensing modalities regarding the extraction of a specific type of context, and performing our experiments to evaluate the effectiveness of our approach to process QoC metrics and use them in resolving conflicts at different levels of a context-aware system.

Twelve subjects executed activities of daily living in this environment, yielding an average of 2 hours of effective data per subject, for a total twenty five hours of sensor data. According to estimations over 11000 interactions primitives with objects and over 17000 interactions primitives with environment have been recorded. This makes data set highly rich in gesture primitives and largest for the purpose of multimodal activity recognition. Table 7.1 shows the short description of those activities and their duration for a single run. Table 7.2 shows the different action primitives that are used in the experiments. These action primitives are extracted from the annotations of

the data sets. These annotations are performed by experts using the videos of all the proceedings during data collection process and identified all the actions performed by the subject during his activities.

Action Primitive Category	*Description*	*Primitive Values*
Locomotion	Basic human movements	walk, run, stand, lie, sit, stairs up, stairs down
Left arm locomotion	Left arm movements	reach, move, release, lock, unlock, open, close, stir, sip, bite, clean, cut, spread
Right arm locomotion	Right arm movements	
Left arm object	Left hand interaction with objects	fridge, dishwasher, drawer1 (top), drawer2 (middle), drawer3 (lower), door1, door2, switch, table, cup, chair, glass, spoon, sugar, knife salami, knife cheese,salami, bottle, plate, cheese, bread, milk, lazy chair
Right arm object	Right hand interaction with objects	

Table 7.2: Brief description and values of action primitive categories

Locomotion primitives are extracted from data collected by the sensors worn the subject body. Locomotion include action primitives such as walking, sitting, lying. Arm locomotion data is extracted from data collected by the sensors worn on the arms of the subject and include the action primitives such as cut, spread, release. Object data is collected from the interaction of sensors embedded in the arms and objets. Primitives extracted from this data present whether a particular object is used or not at a specific instance. Multiple sensors of different modalities collecting information about the activities performed in the environment made this data set ideal to perform activity recognition in an opportunistic environment and observe the effectiveness of different sensing modalities. We have used five runs of three subjects of this data set.

7.3 Analyzing Time Window Length for Feature Extraction

First step to extract high level context information from sensor data, i.e, current activity being performed by the human subject, is to extract features over a time window. These features indicate the number of times a particular sensor is fired in that interval of time or not. For example, if the sensor embedded in plates is fired seven times feature "plate used" has the value seven for that period of time. We pass the feature vectors to classification algorithms to classify the subject activity during this interval of time. We use the time windows length one, five, ten, thirty, and sixty seconds. We use the data collected by five activities of daily living (ADL) runs of three different subjects. So we have fifteen different of ADL run data. To separate the data segments into a test and training sets we use an approach known as "leave one ADL run data out". In this approach one ADL run data is used for testing purpose while the remaining ADL runs data is used for the testing purpose. We repeat this process for all the fifteen ADL runs. The weighted average of the evaluation metrics is taken as the end result. We assign the weights to different activities according to their number in the data set.

QoC Srouce Metrics	*Time Window Length (seconds)*				
	01	05	10	30	60
True positive rate	0.831	0.843	0.855	0.824	0.805
False positive rate	0.045	0.048	0.038	0.052	0.056
Precision	0.839	0.852	0.866	0.846	0.835
Recall	0.831	0.843	0.855	0.824	0.805
F-Measure	0.830	0.842	0.856	0.818	0.804
ROC Area	0.880	0.891	0.901	0.871	0.867

Table 7.3: Metrics using IBK Classifier for different time window lengths

We use the classification algorithms J48, IBK, and Bayesian network for the purpose of classification of high level activities. J48 is the WEKA (Hall, Frank, Holmes, Pfahringer, Reutemann, and Witten, 2009) implementation of C4.5 decision tree (Quinlan, 1993). C4.5 decision tree chooses one attribute of the data that most effectively splits its set of samples based on the criterion of normalized information gain (difference in entropy). The attribute with the highest normalized information determines the de-

cision at each node. IBK implements k-nearest neighbor, an instance-based learning algorithm that generates classification prediction using only specific instances (Aha and Kibler, 1991). Bayesian network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (Bouckaert, 2004). We have chosen these classification algorithms considering different learning strategies used in these algorithms, such as decision tree and instance based learning.

QoC Srouce Metrics	*Time Window Length (seconds)*				
	01	05	10	30	60
True positive rate	0.819	0.852	0.869	0.815	0.779
False positive rate	0.050	0.043	0.035	0.062	0.089
Precision	0.817	0.852	0.876	0.828	0.797
Recall	0.819	0.852	0.869	0.815	0.779
F-Measure	0.814	0.847	0.868	0.811	0.775
ROC Area	0.897	0.914	0.922	0.881	0.853

Table 7.4: Metrics using J48 Classifier for different time window lengths

QoC Srouce Metrics	*Time Window Length (seconds)*				
	01	05	10	30	60
True positive rate	0.740	0.520	0.782	0.824	0.805
False positive rate	0.070	0.084	0.061	0.052	0.056
Precision	0.752	0.657	0.807	0.846	0.835
Recall	0.740	0.520	0.782	0.824	0.805
F-Measure	0.732	0.537	0.779	0.818	0.804
ROC Area	0.942	0.893	0.952	0.881	0.866

Table 7.5: Metrics using Bayes Net Classifier for different time window lengths

Table 7.3 shows the QoC source evaluation metrics for the classification algorithm IBK. Table 7.4 shows the QoC source evaluation metrics for the classification algorithms J48. Table 7.5 shows the QoC source evaluation metrics for the classification algorithms Bayesian net. These metrics have been calculated using the confusion matrix as described in Chapter 4 Figure 4.1. The QoC source metrics are further used to evaluate the reliability of the classification algorithms to classify the subject activity. Figure 7.1 shows the reliability of the classification algorithms IBK, Bayesian network

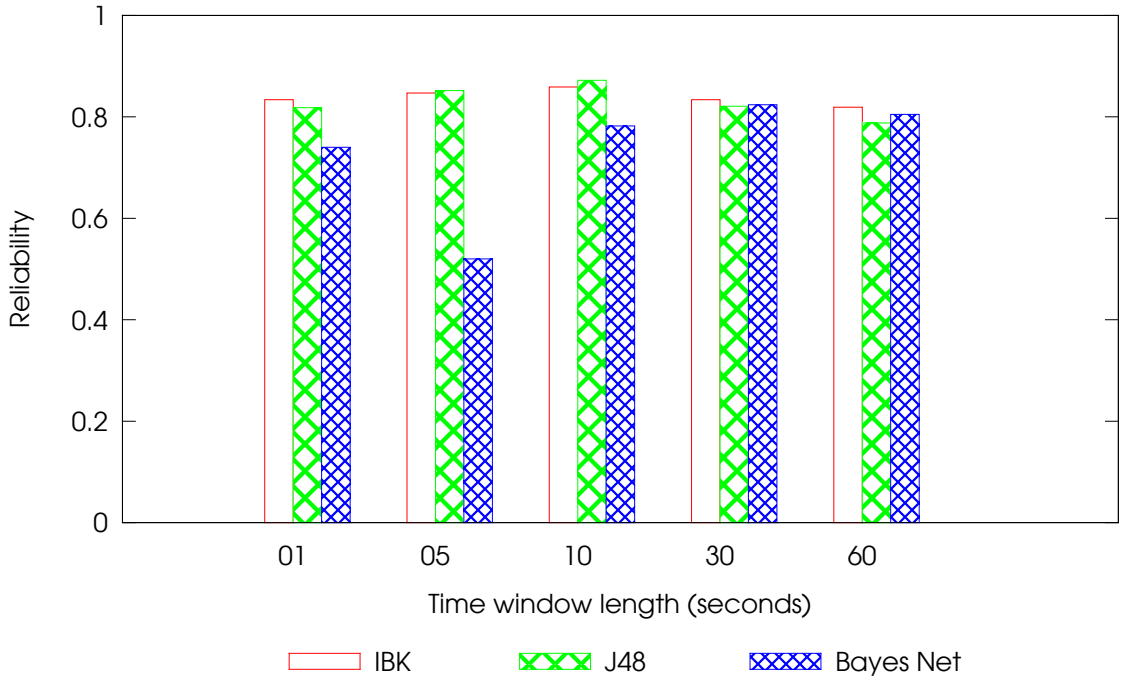

Figure 7.1: Reliability with classifiers IBK and J48 for different time slice lenghts

and J48 for different time window lengths. We observe that classification algorithms have shown better performance for the window length of ten seconds than others. Reliability starts to decrease the window length is increased beyond ten seconds. For the subsequent experiments we use the time window length of ten seconds.

7.4 Identifying Important Action Primitives

In this experiment we analyzed the impact of different combinations of action primitives on high level activity recognition. For this purpose we divided action primitives in seven different combinations. First, we conduct experiments for every individual activity with all primitive sets. Later, we have also observed the impact of primitive sets considering all activities collectively. In this section we discuss the different primitive sets, impact of those sets on each activity, impact of the sets on all activities, and finally we will discuss the results and present our recommendations.

7.4.1 Action Primitive Sets

Table 7.6 shows the action primitive sets and the categories of action primitives that have been used in those sets. In the S1 we use all the action primitive categories described in Table 7.2. In the S2 we excluded the left arm object movement action primitives and the right arm object movement action primitives. In the S2 we used the action primitives extracted from wearable sensors only. These primitives indicate subjects's body and arm motions, e.g., moving, reaching, and releasing an object. In the S3 we also excluded all the wearable sensors that give us information about the locomotion of human limbs. In this set we are left only with action primitives of which values are about subject body actions like walk, sit, and stand. In S4 we use action primitives with both arms locomotion. In S5 and S6 we use left and right arm locomotion respectively. In S7 we use only object sensors, i.e., we only have information about the use of a specific object and we do not have any information about whether concerned person was sitting, standing, lying, or walking. Similarly, we do not have any information about which hand is used to handle an object. We use these different combinations of sensors with classification algorithms IBK, J48, and briefly described in Section 7.3. Table 7.1 presents the detail of the values of these primitives.

Action Primitive Set	*Categories of Action Primitives*
S_1	locomotion, left arm movements, right arm movements, left arm object, right arm object
S_2	locomotion, left arm movement, right arm movement
S_3	locomotion
S_4	left arm movement, right arm movement
S_5	right arm movement
S_6	left arm movement
S_7	object movement

Table 7.6: Different sets of action primitives

7.4.2 Evaluating Action Primitives Impact on Recognizing Activities Separately

In this section, we examine the impact of these primitive sets on each activity separately. Figure 7.2 shows the true positive rate of all primitive sets for activity *Idle* that are used with classification algorithms J48, IBK, Bayesian network . While being *Idle*, the subject is not performing any activity. The classifiers show good performance in recognizing this activity with all action primitive sets. J48 shows marginally better performance with action primitive sets S1 and S2 while IBK shows marginally better performance with action primitive sets S6 and S7. This result is not surprising as when the subject is idle, she is neither interacting with any object nor making movements. Consequently, most of the time action primitive has null value during this activity. So action primitives extracted from any particular type of sensing modalities are not particularly significant in case of recognizing this activity. Action primitives extracted from any sensing modality can easily detect whether the subject is idle or not.

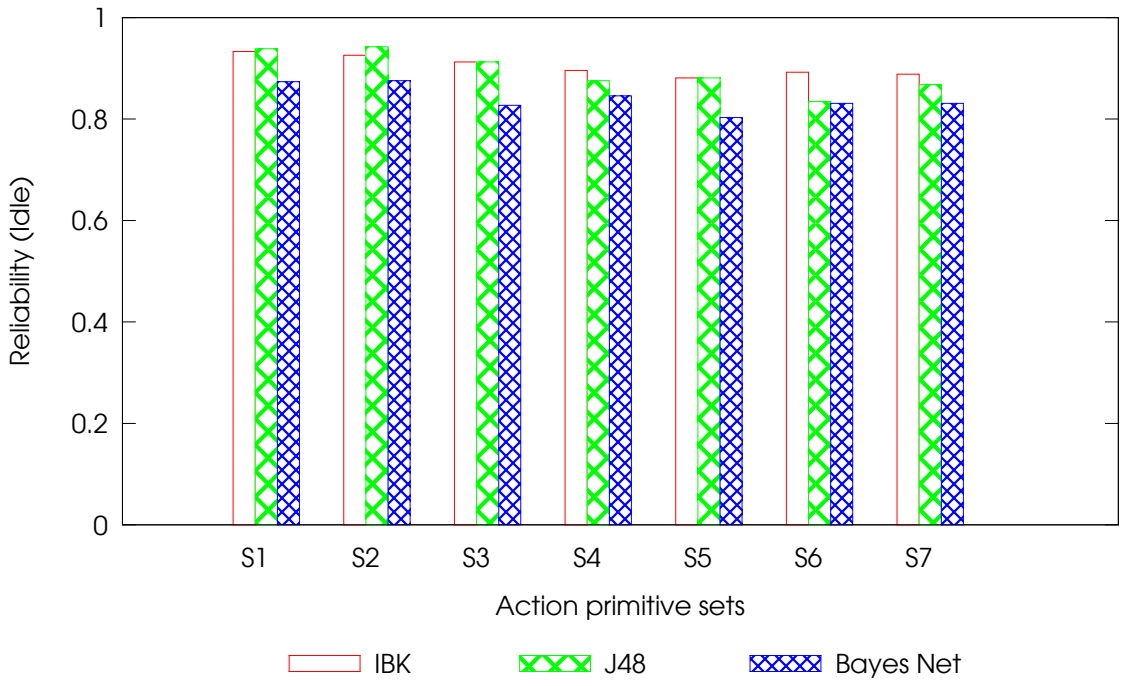

Figure 7.2: True positive percentage of activity *Idle*

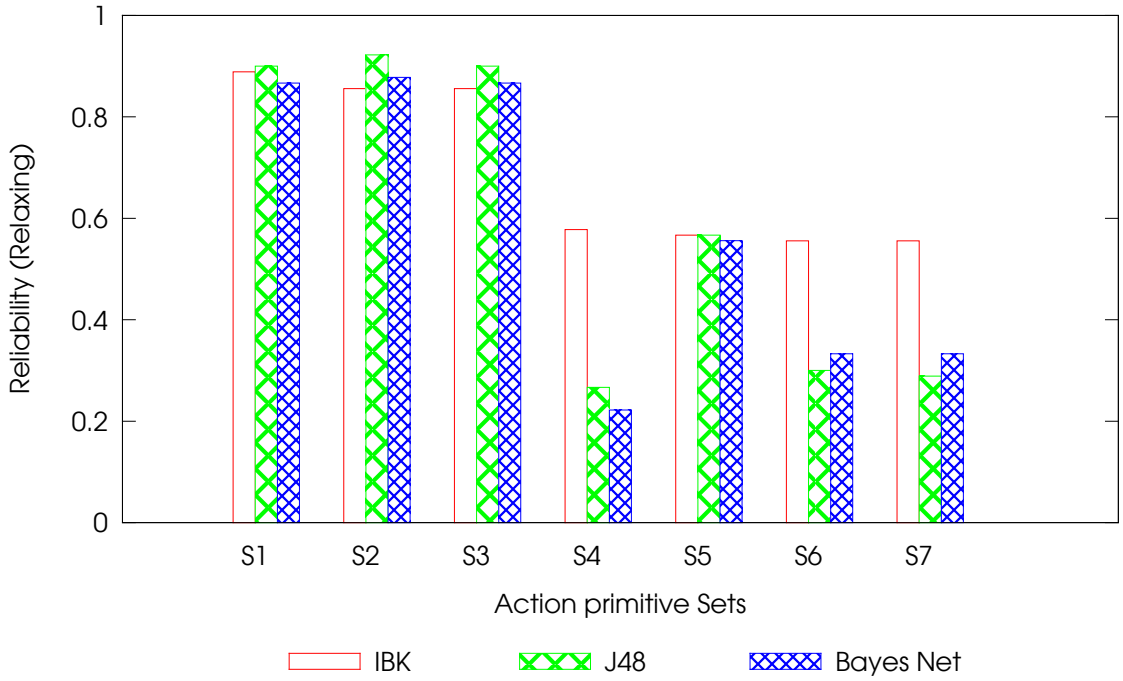

Figure 7.3: True positive percentage of activity *Relaxing*

Figure 7.3 shows the true positive rate of all action primitive sets for activity *Relaxing* that are used with classification algorithms J48, IBK, and Bayesian network. This activity proved to be most difficult one to recognize. During this activity subject was either taking rest or casually moving around the building. She has not been particularly involved in any activity. She has not also been interacting with any object in the environment.

Looking at the true positive rate of this activity it can clearly be seen that the classifiers perform better while using action primitive sets S1, S2, and S3 than while using action primitive sets S4, S5, S6, and S7. Primitive set S1 contains action primitives that have been extracted from all sensor modalities. Primitive set S2 contains action primitives that have been extracted from wearable sensors worn on subject body and limbs. Primitive set S3 contains action primitives that have been extracted from wearable sensors worn on human body that present action primitives about subject body locomotion, such as walking, siting, lying. Human body locomotion action primitives in all the three sets, i.e., S1, S2, and S3, that show better performance to recognize

activity *Relaxing*.

The main reason is that during this activity the subject is not interacting with any object. So most of the time action primitives extracted from sensors embedded in objects have null value. This is the reason why action primitive sets from those sensors are not very successful to recognize this activity. The classifiers get almost the same feature for this activity as for the *Idle* activity. Comparatively the high number of idle activity overwhelmed the classifiers decision and classifiers got confused to make a distinction between *Idle* and *Relaxing*. Classifier detected most of the *Relaxing* activities as the *Idle* activity. Primitive sets S1, S2, and S3 show better performance because those sets include action primitives, such as walking and standing, extracted from wearable sensors worn on subject body. But when action primitive sets exclude action primitives extracted from wearable sensors worn on subject body recognizing *Relaxing* activity becomes very difficult. Consequently wearable sensors giving information about human locomotion action primitives proved vital for recognizing this activity.

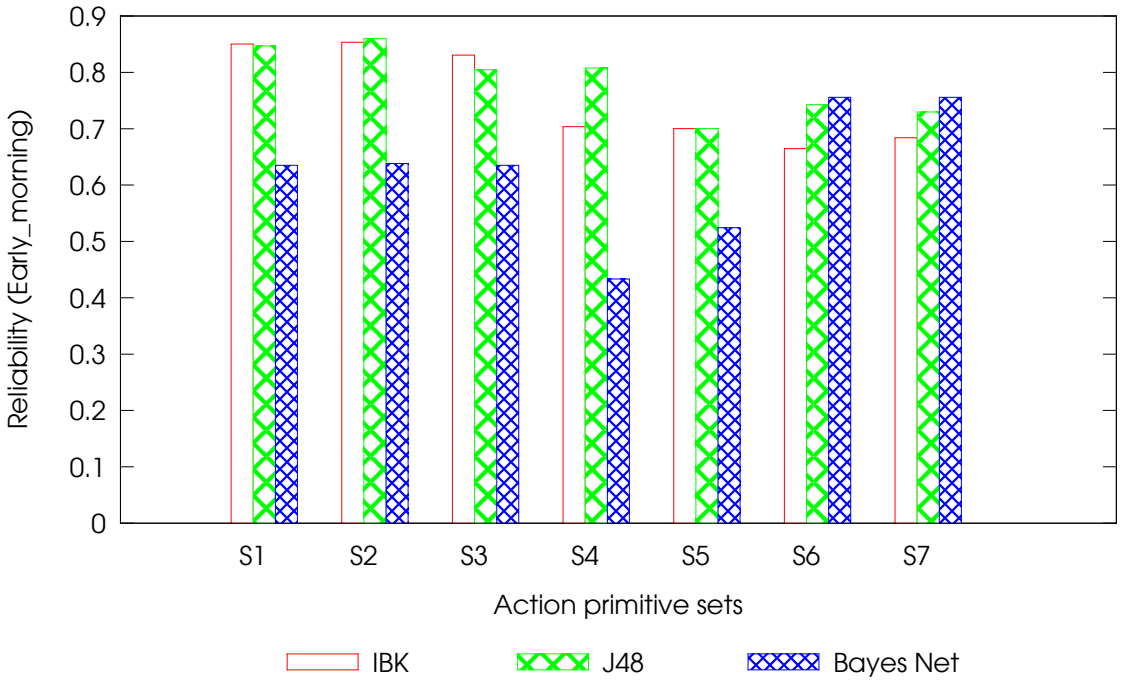

Figure 7.4: True positive percentage of activity *Early_morning*

Figure 7.4 shows the true positive rate of all action primitive sets for activity *Early_morning* that are used with the classifiers J48, IBK, Bayesian network. During this activity the subject moved in the room and randomly checked some objects in the drawers and on the shelf. Although primitive sets S4, S5, S6, and S7 showed better performance in recognizing this activity as compared to recognizing *Relaxing* activity, action primitive sets S1, S2, and S3 that contain action primitives extracted from wearable sensors worn on the subject body won in this case too.

The reason for their better performance was that during this activity the subject spends a lot of time in physical activities. Again in this case too she has not interacted with objects available in the environment for much time. Object sensors had been able to recognize this activity when the subject had not interacted with some of the objects. Action primitives extracted from wearable sensors worn on human limbs are also not very helpful in recognizing *Early_morning* activity as these sensors also provide information about human interaction with objects available in the environment. Wearable sensor providing human locomotion primitives again proved vital in this case.

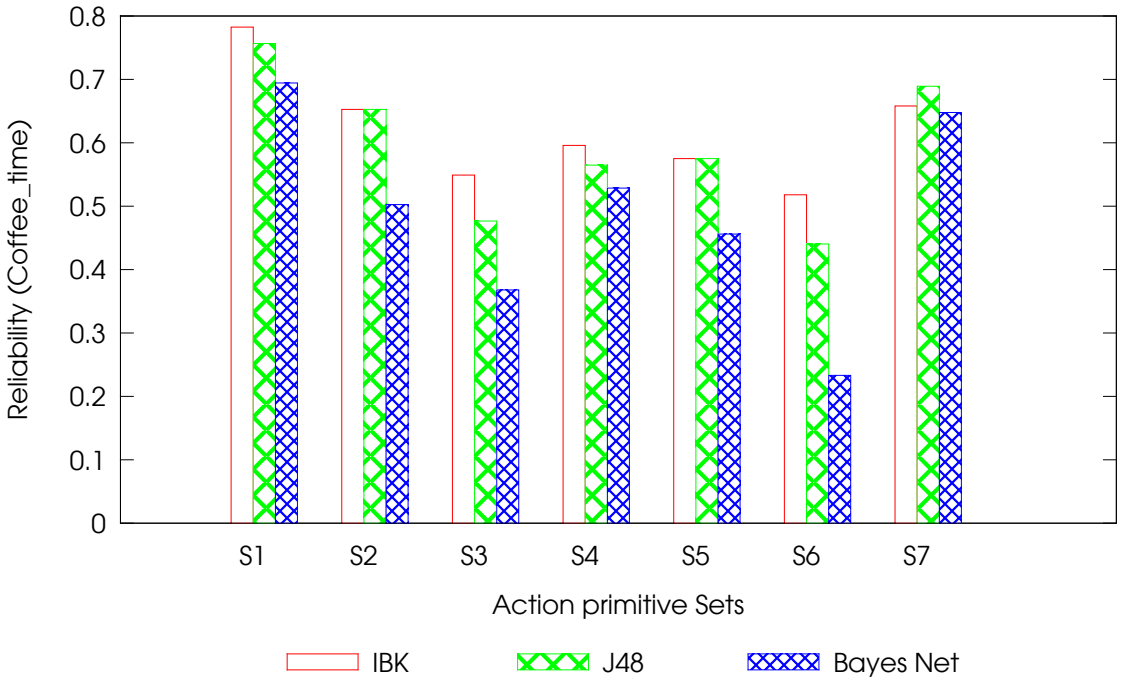

Figure 7.5: True positive percentage of activity *Coffee_time*

Figure 7.5 shows the true positive rate of all primitive set for activity *Coffee_time* that are used with classification algorithms J48, IBK, and Bayesian network. During coffee_time the subject prepared coffee with milk and sugar by using a machine, took sips of coffee and also interacted with different objects in the environment. As evident from the activity description this activity is more distinctive on the basis of objects that have been used during this activity as compared to human action primitives. Subsequently action primitive sets S1 and S7 that contain object usage action primitives performed better than other action primitive sets.

Action primitive set S2 that contains action primitives extracted from wearable sensor showed better performance when all those action primitives are used together. Right arm locomotion action primitives show better performance than the subject body locomotion action primitives and left arm locomotion primitives as shown by the values of reliability for action primitive sets S2, S5, and S6. Wearable sensors worn on right arm show better performance because most of the time subject may have been interacting with objects with dominant limbs. In our opinion if we have more rich wearable sensors that can provide human locomotion primitives like sip, wearable sensors can considerably increase the true positive rate of this activity.

Figure 7.6 shows the true positive rate of all primitive sets for activity *Sandwich_time* that are used with classification algorithms J48, IBK, and Bayesian netwrok. During this activity the subject interacted with different objects in the environment like bread, cheese, and salami, and had also used bread cutters, various kind of knives, and plates to prepare the sandwich. Later the subject ate that sandwich. Contrasting to *Idle* activity when the subject was motionless most of the time and has interacted with few objects, in this activity the subject has not only performed many low level physical activities, like cutting the bread, but has also interacted with various objects in the environment.

As a result, all primitive sets had also performed good in case of this activity as compared with other activities. Human body action primitives that can be extracted from wearable sensors data provide a better rate of true positives as compared to object usage primitives. But in case of this activity a clear winner is the primitive set that use a combination of all action primitives extracted from wearable sensors and object sensors. Combining the action primitives from wearable sensors with primitives

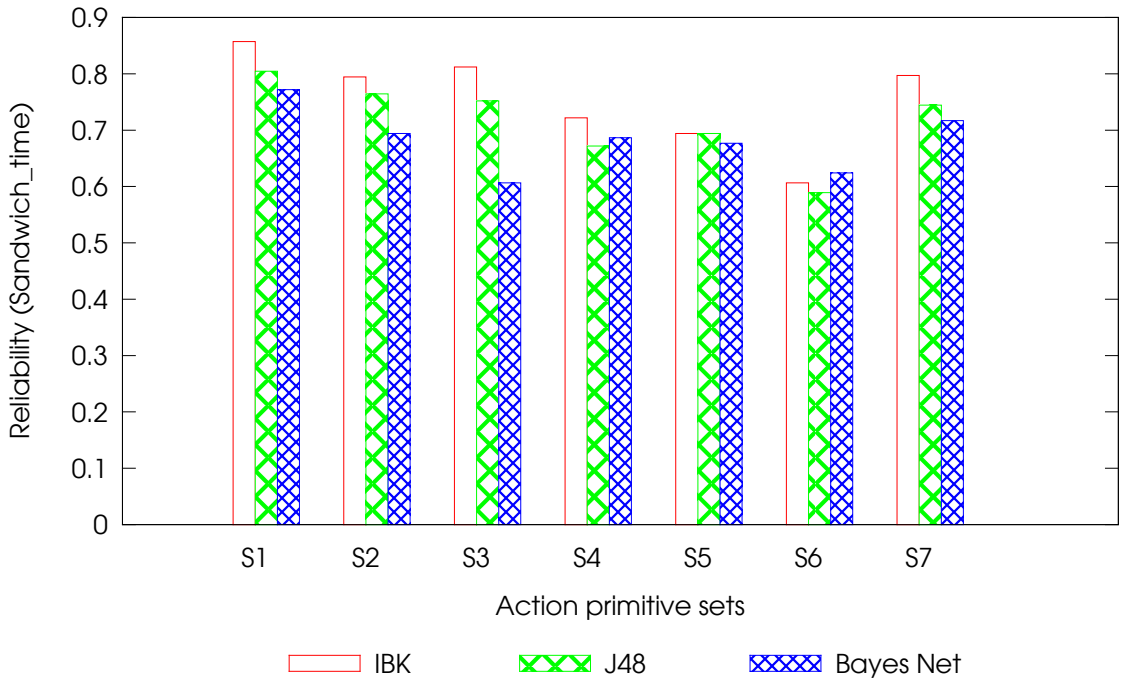

Figure 7.6: True positive percentage of activity *Sandwich_time*

about the usage of objects available in the environment provided a clear evidence about *Sandwich_time* activity as indicated by the high reliability of algorithms using sensor set S1 in Figure 7.6.

Figure 7.7 shows the true positive rate of all primitive sets for activity *Cleanup* that are used with classification algorithms J48, IBK, and Bayesian network. *Cleanup* is the final activity in the ADL drill run for data collection. During this activity the subject puts all objects used to original places or dish washer and clean up the table. Classification algorithms could not show as good reliability to recognize this activity as other activities. Body locomotion primitives, such as walk, sit, and stand, failed when those action primitives have been used alone to detect *Cleanup* activity. However, limbs locomotion primitives, such as reach, move, and release, showed comparatively better performance for the sensors used with right arm than the sensors used with left hand. If we compare the performance of all action primitive sets when these action primitive sets have been used alone, action primitives extracted from object usage sensors and action primitives extracted from sensors worn on right limbs give the best

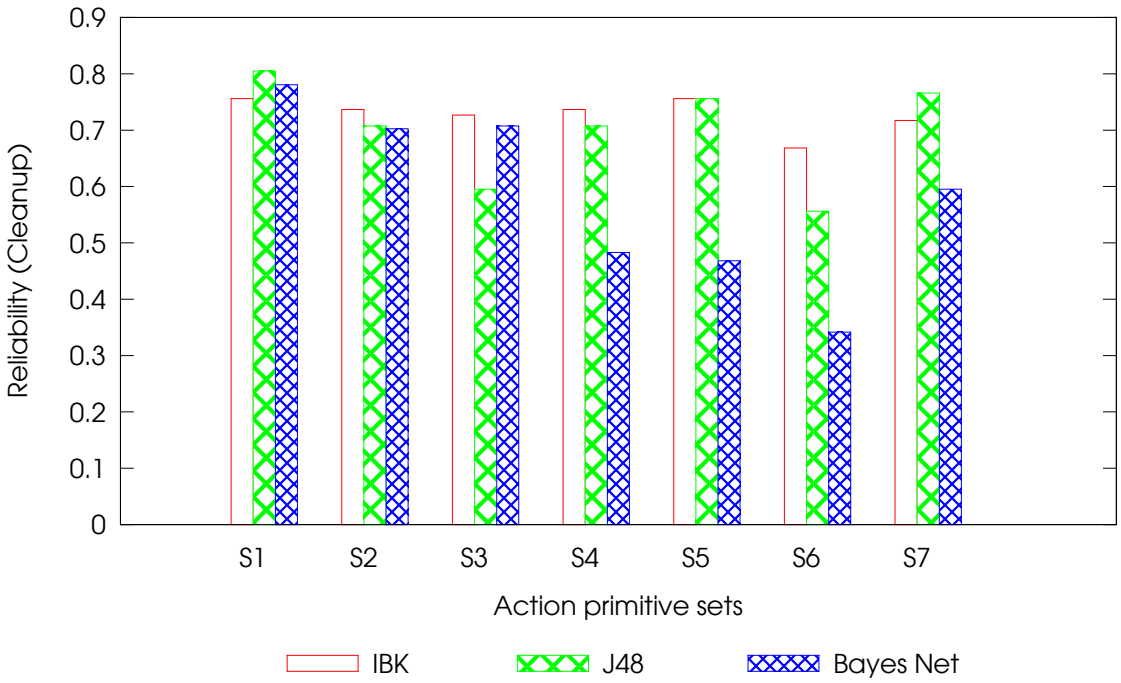

Figure 7.7: True positive percentage of activity *Cleanup*

performance, as shown in Figure 7.7. Overall primitive set that used combination of human locomotion primitives, limbs locomotion primitives, and object usage primitives showed the best performance to detect *Cleanup* activity.

7.4.3 Evaluating Action Primitives Impact on Recognizing Activities Collectively

Figure 7.8 shows the weighted average of true positive rate, false positive rate, precision, recall, and f-measure of all activities classified using J48 classification algorithm. Figure 7.9 shows the weighted average of true positive rate, false positive rate, precision, recall, and f-measure of all activities classified using IBK classification algorithm. Figure 7.10 shows the weighted average of true positive rate, false positive rate, precision, recall, and f-measure of all activities classified using Bayes Net classification algorithm. These metrics are used to calculate the value of reliability of the classifiers to accurately recognize the activity of human subject using different primitive sets as

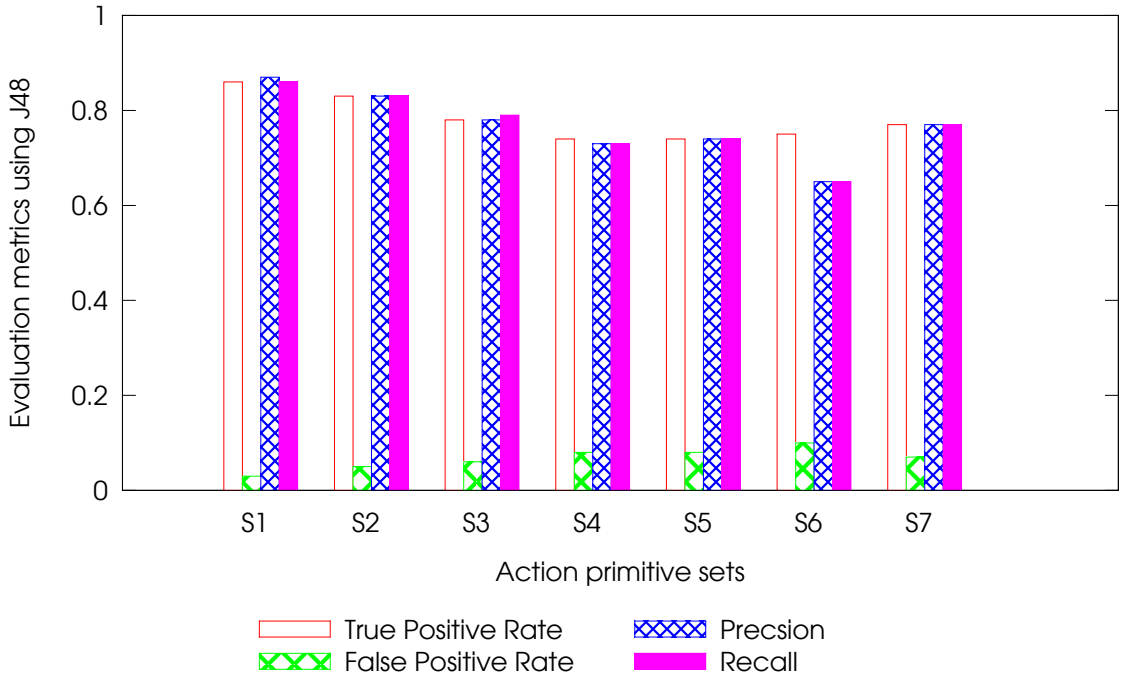

Figure 7.8: Evaluation metrics for different primitive sets using decision tree

shown in Figure 7.11.

Action primitive set S1 that includes action primitives extracted from all the sensing modalities available in the data set showed the best true positive rate with a comparatively low value of false positives. Subsequently we also have good values of other metrics for primitive set S1. Although the use of wearable sensors to recognize high level activities is very rare action primitives extracted from wearable sensors also show comparatively good performance as depicted by the evaluation metrics of primitive set S2 that contain only primitives extracted from wearable sensor data.

The reasons for their good performance is the comprehensive nature of the action primitives extracted from those sensors. These sensors not only provide information about the action primitives like walk, sit, and stand but also indicate that an object available in the environment is used. These primitives also proved very helpful in recognizing activities like *Idle*, when subject is not performing any activity, *Early_morning*, when subject is walking around and handling different objects, and *Relaxing* when subject is siting or lying. However, classifiers get confused when they have to detect

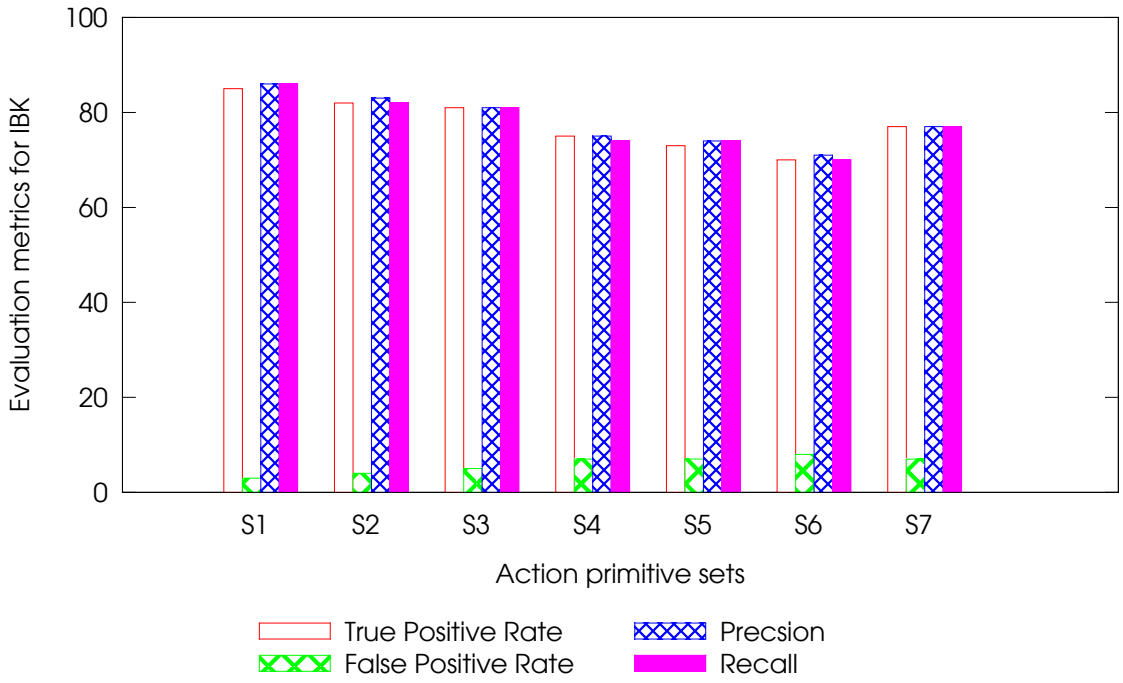

Figure 7.9: Evaluation metrics for different primitive sets using IBK

a single activity among ones that used same objects. Reasons is that there have not been any information about which object is used as shown by the high value of false positive rate and low value of precision as compared with sensor set S1 where we used all the sensors.

Primitive set S3 consists of only locomotion action primitives that inform about low level human actions. This sensor set had shown comparative performance with sensor set S7 that consists of primitives from object sensors. The weighted averages of these sensors are almost equal to each other. The main reason is that locomotion primitives are better in recognizing activities like *Relaxing* and *Early_morning* while object usage primitives are proved good in detecting activities in which the subject has higher number of interaction with different objects, such as *Sanwich_time*. These sensor sets got confused in recognizing other activities as evident by their high false positive rate and low value of precision. Primitive set S4 used primitives extracted from both arms. This set has not performed as good as the locomotion primitive sets or the object usage primitive set. Primitive set S5 used only action primitives extracted from

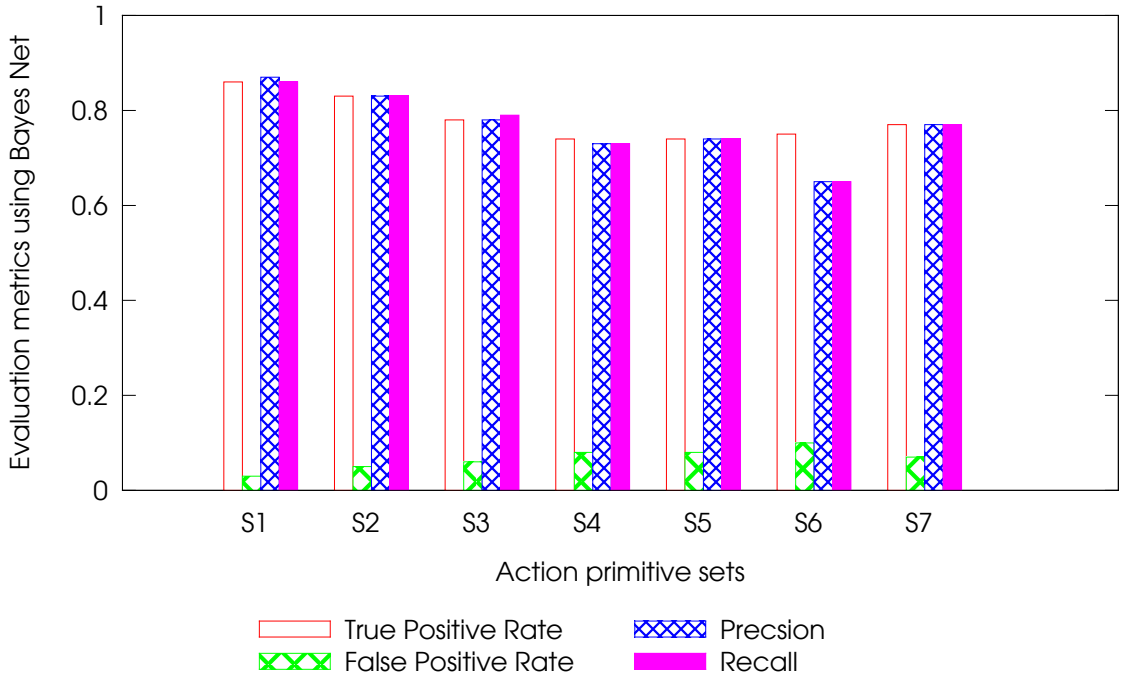

Figure 7.10: Evaluation metrics for different primitive sets using Bayes Net

right arm and primitive set S6 used only primitives extracted from left arm. Although both of these primitive sets showed good quality for some of the activities, they could not show good overall accuracy when they are used alone. Primitive set S7 showed better performance comparatively. Clearly primitive set S1 that used combination of object sensors with wearable sensors proved best to recognize human activities in the smart home environments.

7.4.4 Discussion and Recommendations

Considering the results of this experiments we observed that wearable sensors that have been ignored in recognizing human activities in smart home environments played a significant role in improving the performance of human activity recognition systems. Although when different types of action primitives have been used alone, object or environment usage action primitives gave better results, human body locomotion and limbs locomotion primitives also proved vital in recognizing some of human activities, such

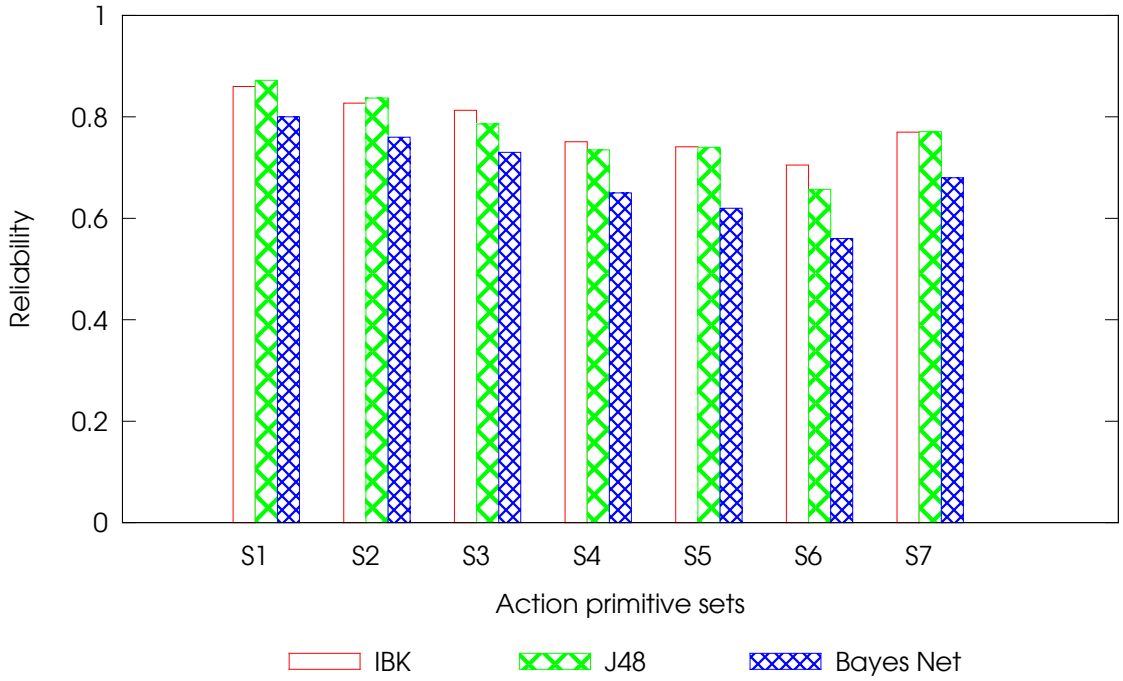

Figure 7.11: Reliability of different classifiers using different sensor sets

as relaxing, idle, and sleeping. Object and environmental usage primitives completely failed in recognizing these activities. Limbs locomotion primitives, like reach, cut, and touch, also proved significant in recognizing those activities that include not only using but also performing actions on different objects. Examples of the actions performed on objects include cutting bread, applying bread spreads. Wearable sensors that can be used to extract action primitives like sip or bite are also important in correctly distinguishing activities like drinking coffee or eating a sandwich. Object or environmental usage sensors are very important to install in areas, such as kitchen, where human are expected to have greater interaction with those objects to recognize human activities. Wearable sensors are significant in recognizing activities during which human does not interact much with environment, such as *Relaxing*. Sensors used with dominant limbs are more reliable in recognizing human activities than sensors used with other limbs. As wearable sensors in combination of object sensors clearly outperformed only object sensors in recognition of all activities, it is indispensable to use wearable sensors in smart environments to improve their performance.

7.5 Evaluating the role of confidence on context

In this experiments we evaluated the confidence on context information considering the requirements of two context-aware applications Appliance Control (AC) and Ambience Management (AM) in smart home environments. Later we have used the confidence on context information to improve the functionality of context dissemination. In the following section first we describe the scenario of our experiments and experiment settings. Later we evaluate QoC metrics, infer the value of confidence on context, and illustrate how confidence on context can be used to improve the functionality of different tasks in context-aware applications.

7.5.1 Scenario Description

Appliances Control (AC) and Ambiance Management (AM) are two typical context consumer services (CCS) deployed in a smart home to control actuators. In our scenario, The task of AC is the proactive anticipation of the future use of appliances in the home and switch them on or off to support the user. For example, when a user wants to start cooking, the AC should switch on and pre-heat the oven. The AC also switches appliances off when they are no longer in use to save power and avoid any injuries. For example, the AC should switch off the oven when no cooking activity is currently detected or expected to happen in near future. The AM controls the home ambiance by tuning temperature, light, and background music. For example, the AM decreases luminosity and the volume of background music when the user is relaxing.

Both of the CCSs heavily depend on the users' current activity to effectively perform their functionality. Examples of typical activities in a home environment include relaxing, coffee time, and sandwich time. Virtual sensor services (VSS) collect the data from the environment and various Context Management Services (CMS) subscribe to those VSSs to obtain sensor data. Ultimately AC and AM subscribe to CMS to get the required context (user activity). AC and AM are interested only in a subset of the user's activities. AC is interested in activities that the user performs in the kitchen area

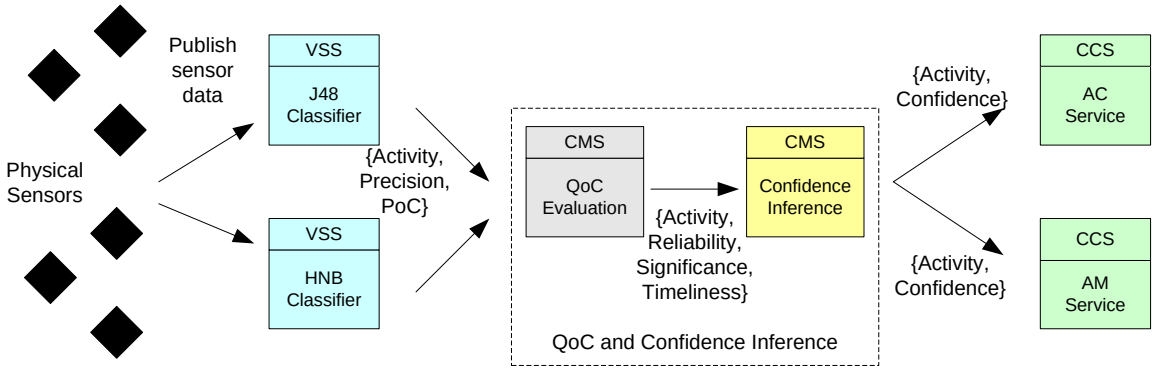

Figure 7.12: Experiment scenario

to keep the appliances ready for use. In contrast, AM is more interested in activities performed in other parts of the house.

Large number of irrelevant context objects of inadequate quality increase the application's burden to reason on QoC metrics or context information itself and infer whether an underlying piece of context is of sufficient quality and relevant to carry out their task. This extra effort affects the main functionality of the services, i.e., to adapt to dynamically changing situations in the smart home. CMS infer the value of confidence on context information from QoC metrics. CMS further uses the confidence on context information to select context objects worthy to perform the functionality of a specific application.

7.5.2 Experiment Setting

Figure 7.12 depicts the scenario of our experiment. The environment is embedded with many physical sensors. They sense the environment and collect the data as described in Section 7.2. Physical sensors provide this data to virtual sensors. Virtual sensors classify the current user activity using machine learning classification algorithms. In our experiment, the virtual sensors use machine learning algorithms J48 and Hidden Naive Bayes for the purpose of classification of user activity as shown in Figure 7.12. J48 is an algorithm that implements a C4.5 decision tree (Quinlan, 1993). Hidden naive Bayes (HNB) is an extended form of naive Bayes and accommodates the attribute dependencies (Zhang, Jiang, and Su, 2005). We use their implementations available

in WEKA Hall et al. (2009). Virtual sensors used the classification models of the aforementioned algorithms that have been trained with the sensor data. Precision and recall of classification methods to recognize user activity were also calculated to evaluate the accuracy of classification methods. VSSs provide the classified context (user activity) to the context management services that evaluate the QoC metrics and confidence on context and provide them to CCSs along with the context information items (user activity). We performed our experiments on a system having Intel Core 2 T5500 @1.66 GHz CPU and 2 GB RAM.

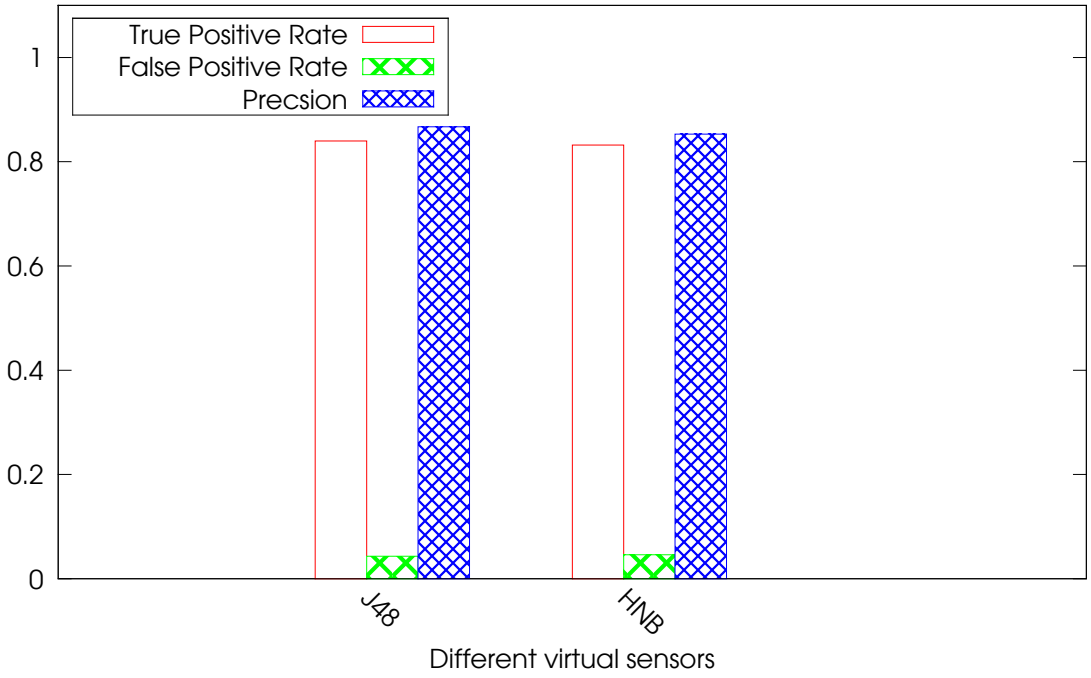

Figure 7.13: Precision of different virtual sensors

7.5.3 Evaluation of QoC metrics

In the first part of the experiment we have evaluated the QoC metrics of the context (user activity) recognized by the virtual sensors. The first QoC metric that we have evaluated was the *Reliability* of context. *Reliability* is defined as the belief in correctness of context information contained by a context object is evaluated combin-

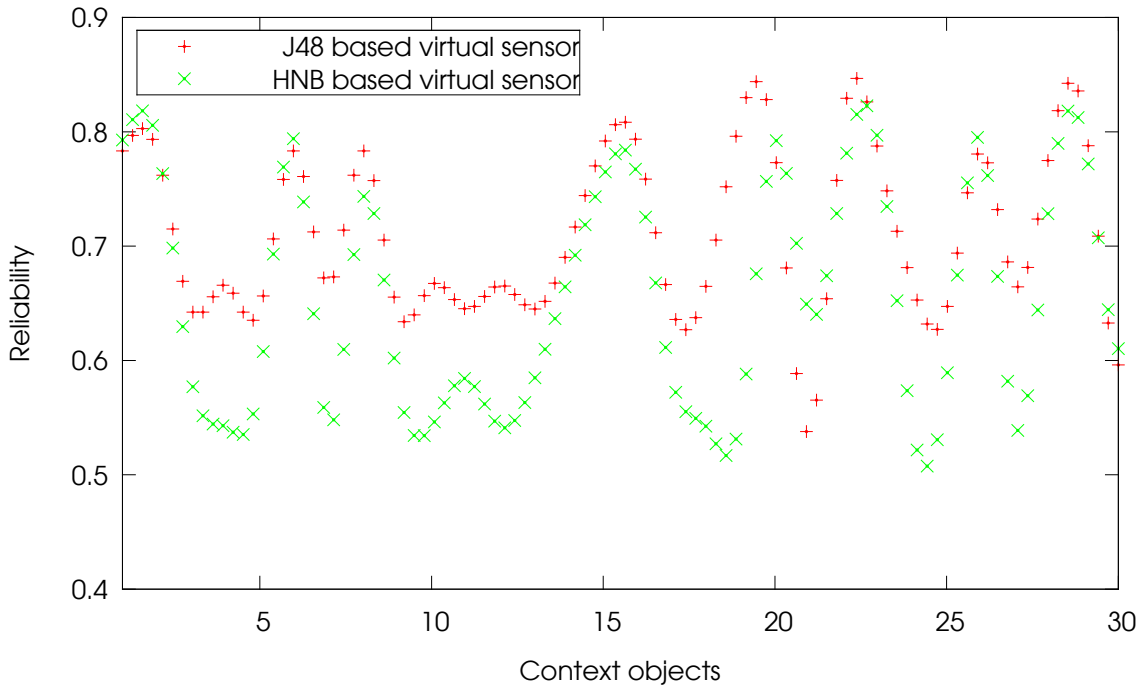

Figure 7.14: Reliability of context

ing *Precision* and *Probability of Correctness (PoC)* of the context extracting process. WEKA implementation of J48 and HBN also provide PoC of context extracted from sensor data. *Precision* is the exactness of the context extracted from the context extraction process and is calculated as the ratio of true positives to total number of positives. Figure 7.13 shows the true positives, false positives and precision of context (activity) recognition chains using J48 and HBN classification algorithms. *PoC* is the probability of correctness of the predication made by a context (activity) recognition chain. We have combined *Precision* and *PoC* of context to calculate *Reliability* using following Equation 4.5. The *Reliability* of context extracted from two different context recognition chain is shown in Figure 7.14.

The *Significance* of context is evaluated as the ratio of critical level of a particular context to maximum critical level that type of context can have. Applications configure the critical level of different types of context according to their requirements as shown in Figure 5.1. As we mentioned before AC and AM services are interested in different values of context (user's activity). AC—that switch appliances on or off

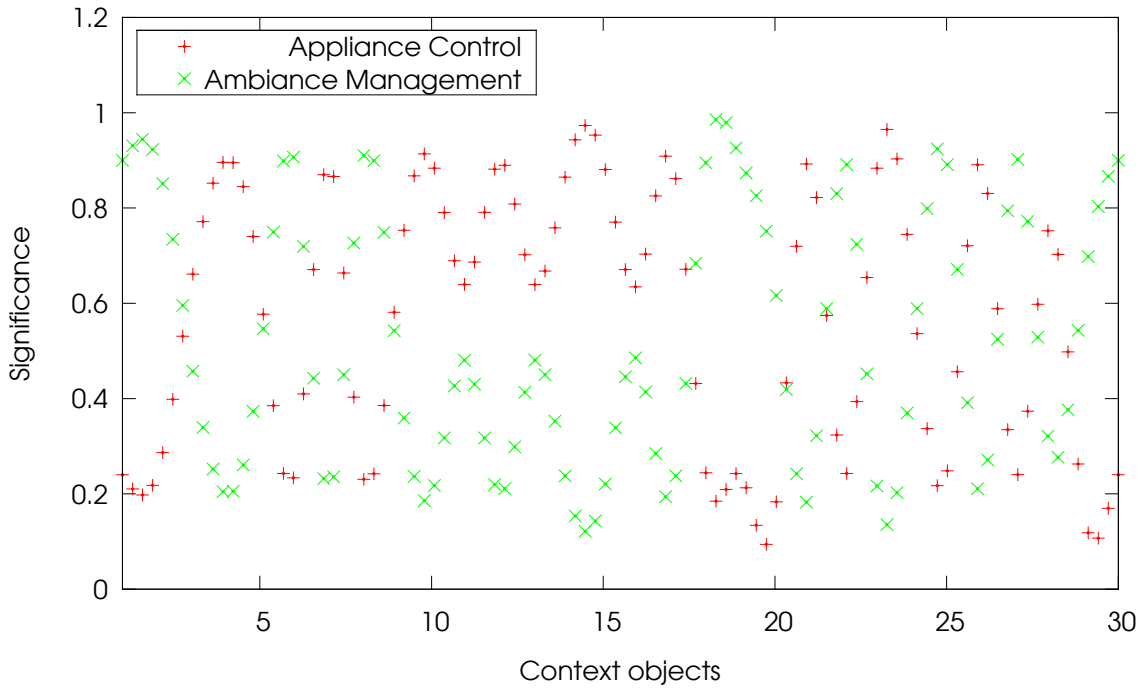

Figure 7.15: Significance of context

depending whether there is a chance that user will like to use these appliances— has assigned kitchen activities a higher critical value, so that when a context (user's activity) is recognized as one of kitchen activities, such as *Coffee_time*, *Sandwich_time*, and *Cleanup*, it has high value of significance for the AC service. Similarly, the AM service is more interested in user's activities that have been performed outside kitchen area activities, such as *Idle*, *Relaxing*, and *Early Morning*, which have higher significance for this service. Figure 7.15 shows the significance of context for the different services.

7.5.4 Evaluation of confidence

In our experiment QoC metrics *Reliability*, *Significance*, and *Timeliness* are used to evaluate confidence on context. QoC metrics having numerical value serve as the base variables for fuzzy QoC variables. Both CCSs specified QoC requirement values of *Reliability*, *Significance*, and *Timeliness* as *VeryHigh*, *Fresh*, and *Vital* respectively. We passed QoC metrics that have been evaluated in our experiments as described above

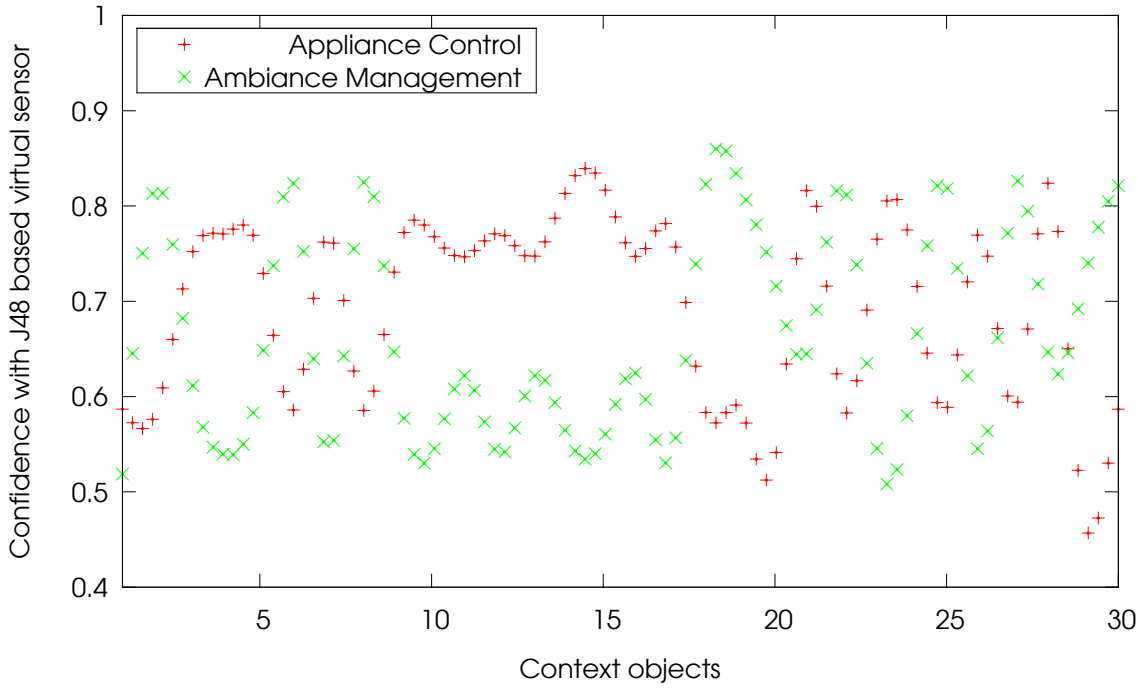

Figure 7.16: Confidence on context using J48

to the *Confidence Inference System* that used the rules generated by the *Rule Engine* to infer the value of confidence for each context object considering the requirements of both applications. Figure 7.16 and Figure 7.17 show the confidence on context considering the requirements of both applications from context extraction chains using J48 and HBN classification algorithms. A careful examination of the values of confidence for different context objects in both of Figure 7.16 and Figure 7.17 depicts that the confidence on context is distinctively different for the two services for most of context objects. For example, context object 15 in Figure 7.16 and Figure 7.17—user activity *Coffee_time*—has higher value of confidence for AC than for AM. Considering higher value of confidence we can decide that this particular context object is more important to AC than to AM to perform their functionality. This observation demonstrates that the confidence on context successfully indicates the quality and relevance of context to both services.

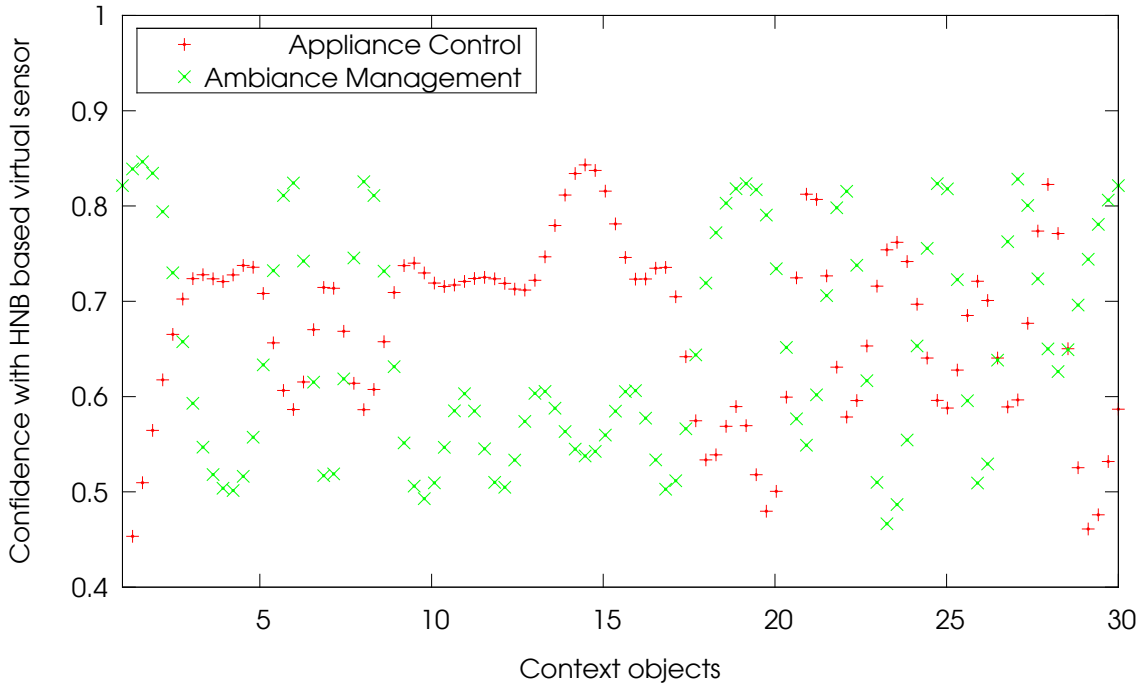

Figure 7.17: Confidence on context using HNB

7.5.5 Context consumers using confidence

In our experiments we have also used the confidence on context information to filter the context objects that have been forwarded to CCSs. These CCSs can also provide the threshold value of the confidence on context information for interested context objects, i.e., CMS should only forward those context objects to CCSs that meet the required value of confidence. Figure 7.18 shows the number of context objects that have been provided to both CCSs using different threshold levels. The number of context object decreased as we increase the threshold level of confidence. Figure 7.19 shows the precision — as defined in information retrieval to indicate the relevant context objects — with rise in the threshold level of confidence. Precision is calculated as the proportion of true positives, i.e., objects that a particular application should receive, to total number of context objects received by an application. We can observe in Figure 7.19 that the precision of the objects received increases with the rise in threshold level for confidence on context information. Directly proportional relationship between the precision and the value confidence threshold depicts that the number of relevant

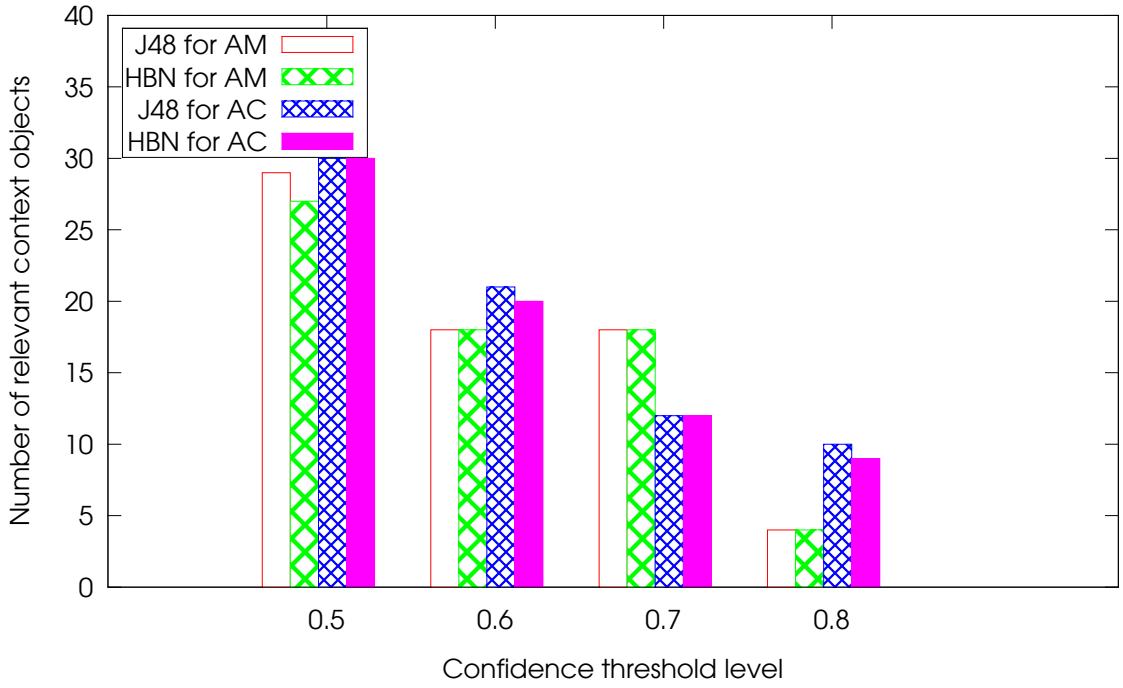

Figure 7.18: Number of context objects selected

context object received by a particular CCS increase with increase in the value of confidence threshold.

Figure 7.20 shows the recall with increase in threshold level of confidence on context information. Recall is the fraction of context objects that are relevant to the functionality of a particular application and they have been retrieved successfully. Recall is calculated as the ratio of relevant context objects received by an application to the total relevant context objects generated. We can observe in Figure 7.20 that the value of recall remains constant up to certain values of confidence and then it starts to decrease. This is because with low values of confidence threshold applications have been receiving all the context objects that have been generated. Increase in threshold decreased the number of context objects receive as shown in Figure 7.18 that eventually also decreased the number of relevant context objects received by the services. The service-specific confidence threshold value serves as a tradeoff between receiving all relevant but also unsuitable context items, and receiving most relevant context items and missing some of them.

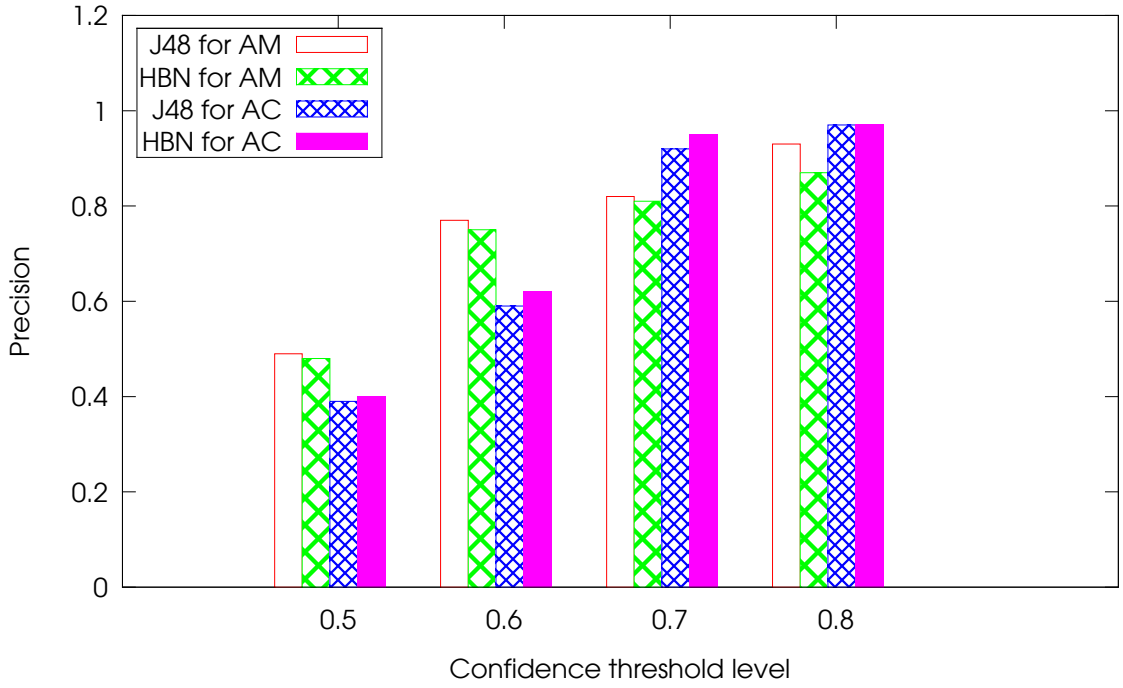

Figure 7.19: Precision of context selection

Figure 7.21 shows the time consumed by the context chain from sensor data generation to delivering the context object to context consumer services (CSSs) with different number of context objects in single burst and different number of CSSs subscribed for the context. In this experiment, we have taken CCSs that have different set of QoC requirements and context is distributed depending upon the confidence on context considering the requirements of a specific CCSs. Figure 7.21 shows that the context chain can accommodate a burst of two hundred and fifty context objects with CCSs of five different types of QoC requirements to distribute context within one second. In case we have more than two hundred and fifty context objects in a single burst and five CCSs with context delivery requirements of less than one second we also need more than one CMSs and better load balancing techniques.

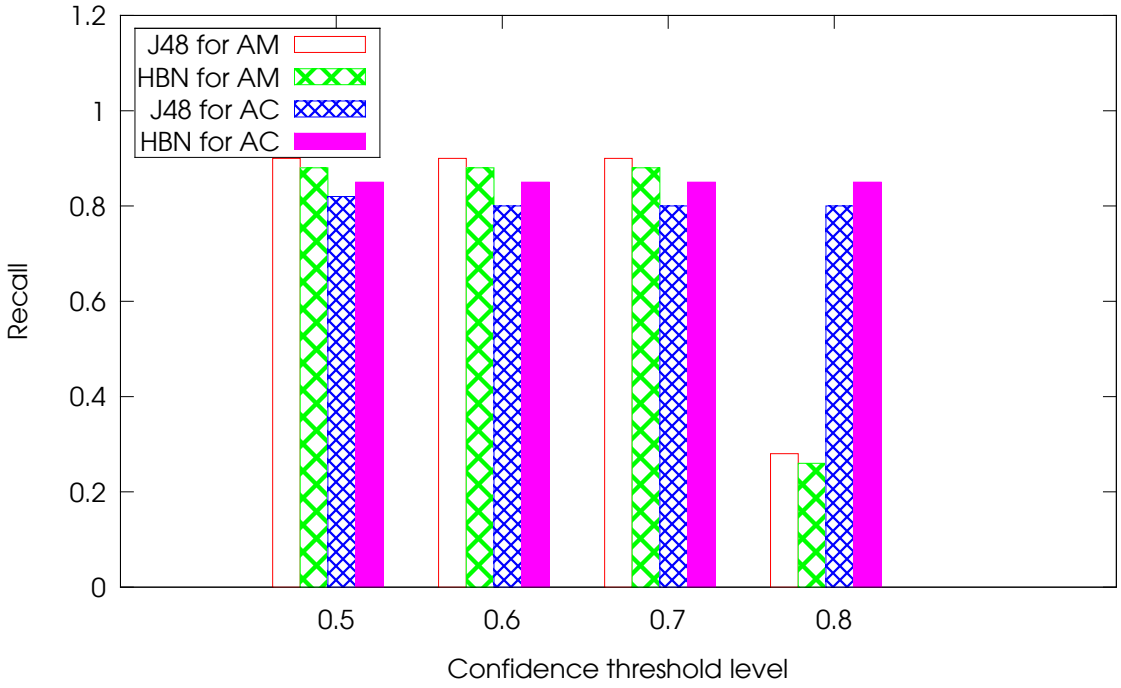

Figure 7.20: Recall of context selection

7.6 Summary

In this chapter we have performed the experiments to evaluate our approach to calculate QoC metrics as the worth of context information for a specific application and use those metrics to enhance the performance of different tasks in a context-aware system. We used QoC metrics evaluated for high level context information, i.e., human activity of daily living, to decide about the time window length for feature extraction. The time window length of ten seconds is best suited to extract features from sensor data. Later we used the time window length of ten seconds for the remaining experiments.

We also used reliability of context extracted from sensor data with WEKA implementation of classification algorithms J48 and IBK to identify important action primitives to recognize different activities of daily living. We used the action primitives extracted from wearable sensors worn on subject body and limbs and sensors embedded in objects and environment. We observed that action primitives extracted from wearable sensors that have so far been neglected to recognize human activities of

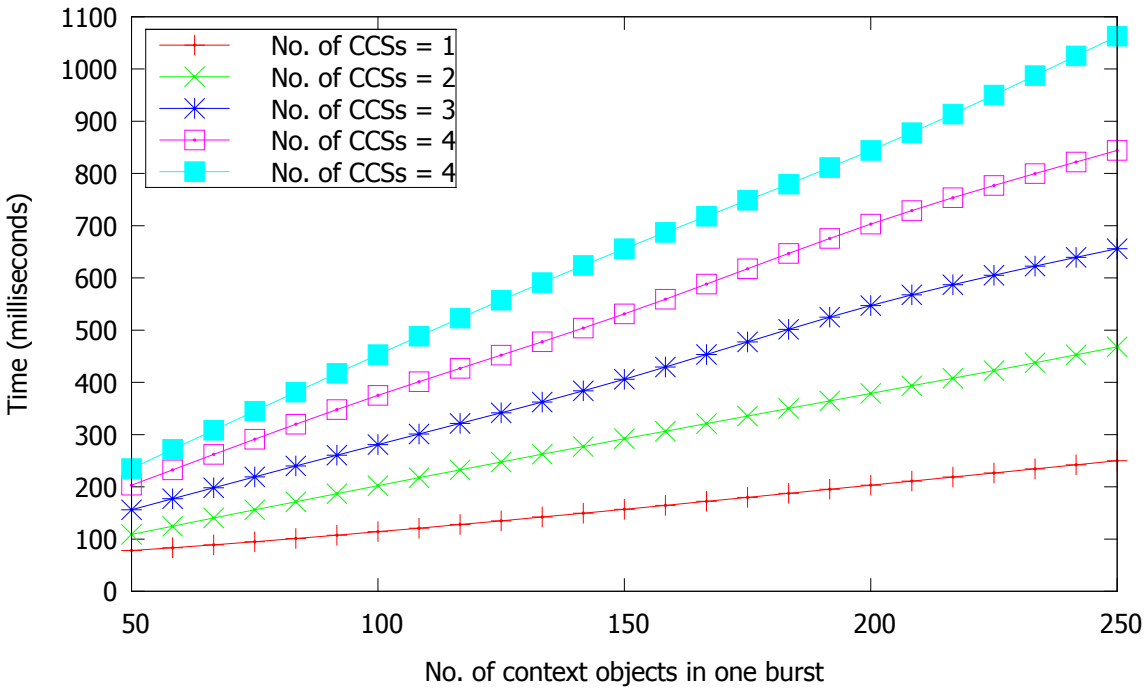

Figure 7.21: Time consumed in context delivery

daily living are quite crucial in recognizing some of the activities. Overall action primitive sets that used combination of action primitives extracted from wearable sensors and object sensors performed best.

Later we inferred the value of confidence on context information from QoC metrics evaluated for high level context information. Confidence on context is inferred tailored to the needs of two context-aware applications in the smart home environment. The experiments demonstrated that confidence on context successfully depicts the worth of context for a specific context-aware application considering quality and relevance of context for that particular application. Our experiments also showed that confidence on context can be used to select the important context objects for a specific context-aware application to perform its task. The threshold value of confidence to select context objects can also be optimized as a tradeoff between precision and recall of relevant context objects. In the next section we present the conclusion of the thesis.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This work presents a novel approach to perceive Quality of Context (QoC) as the worth of context information for specific context consumers and presents a model to process and use QoC metrics. Our model discusses sensor characteristics, measurement context, and context consumer requirements as QoC indicators that have been processed evaluating high level QoC metrics to represent subjective and objective view of QoC. Objective view of QoC presents absolute quality of context independent of any context consumer and subjective view of QoC presents quality of context relative to the requirements of a specific context consumer.

QoC metrics are further combined to present confidence on context information tailored to the QoC requirements of a particular context consumer. We use fuzzy logic to infer the value of confidence on context that also enables us to incorporate context consumer input the process of confidence inference. Later, this work discusses the conflict resolving policies that can be used to deal with conflicting situations that may arise in performing different tasks in context of a context management system to collect sensor data, extract high level context information, and disseminate context to different nodes. These conflict resolving policies can be based on a single QoC metric or confidence on context inferred from two or more QoC metrics.

We use a data set collected in a smart home environment to perform our experiments

to evaluate QoC metrics and analyze the affectiveness of QoC and confidence on context based conflict resolving policies to deal with conflicting situations in a context-aware pervasive system. We use these policies to resolve conflicts in the selection of different sensing modalities that can used to extract high level context information. We have also used these policies to resolve conflict in the decision to decide a time window size for feature extraction. Later, confidence on context based conflict resolving policy is used to resolve conflicts in performing the task of context distribution to select context objects worthwhile for a specific context consumer.

We discover that the subjective view of QoC to evaluate QoC metrics as the worth of context information for a specific context consumer resolve context consumer to perform extra burden to reason about the quality of context information before using it. We find that QoC based conflict resolving policies are very effective to deal with conflicting situations in performing different tasks at different layers of context-aware pervasive system, such as sensor selection, context storage and aggregation, and context distribution. We find that conflict resolving policies defined on the basis of the confidence on context inferred from two or more QoC metrics are more successful than conflict resolving policies defined on a single QoC metrics. We notice that the selection of QoC metrics in a conflict resolving policy is completely dependent upon the nature of the conflict and the task to be performed by a specific context consumer.

8.2 Future Work

For future work, we plan to use confidence on context and QoC based conflict resolving techniques to do more sophisticated reasoning while extracting high level context information. We also plan to enhance the quality of context information by combining the context information and QoC metrics from more than one context objects. Furthermore, we plan to study the interdependence of QoC, Quality of Data, and Quality of Service and enhance the evaluation of our QoC metrics.

We also plan to model the knowledge gained about the significance of different kinds of action primitives in recognizing high level human activities of daily living. Moreover, we plan to use this knowledge to develop an adaptable human activity recognition chain

that can dynamically select the sensors available in the environment, extract relevant action primitives, and recognize human activities of daily living with optimal cast in terms of time and energy.

Bibliography

Aarts, E. and J. Encarnao (2008). *True Visions: The Emergence of Ambient Intelligence*. Springer Publishing Company, Incorporated.

Abowd, G. D., E. D. Mynatt, and T. Rodden (2002, jan.). The human experience [of ubiquitous computing]. *Pervasive Computing, IEEE 1*(1), 48 – 57.

Aha, D. and D. Kibler (1991). Instance-based learning algorithms. *Machine Learning 6*, 37–66.

Baldauf, M., S. Dustdar, and F. Rosenberg (2007). A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput. Volume 2*(4), 263–277.

Bardram, J. E. (2005). The java context awareness framework (jcaf) - a service infrastructure and programming framework for context-aware applications. In *Pervasive*, pp. 98–115.

Bazire, M. and P. Brzillon (2005). Understanding context before using it. In A. Dey, B. Kokinov, D. Leake, and R. Turner (Eds.), *Modeling and Using Context*, Volume 3554 of *Lecture Notes in Computer Science*, pp. 29–40. Springer Berlin / Heidelberg.

Bouckaert, R. R. (2004). *Bayesian network classifiers in Weka*. Working paper series. University of Waikato, Department of Computer Science. No. 14/2004. Hamilton, New Zealand: University of Waikato.

Breza, M., R. Anthony, and J. McCann (2007). Quality of context driven autonomicity. In *2nd International Workshop on Engineering Emergence in Decentralised Autonomic Systems*.

Brézillon, P. (1999). Context in problem solving: a survey. *Knowledge Engineering Review 14*(1), 47–80.

Brgulja, N., R. Kusber, K. David, and M. Baumgarten (2009). Measuring the probability of correctness of contextual information in context aware systems. *IEEE International Symposium on Dependable, Autonomic and Secure Computing*.

Bricon-Souf, N. and C. R. Newman (2007). Context awareness in health care: A review. *International Journal of Medical Informatics 76*(1), 2 – 12.

Bu, Y., T. Gu, X. Tao, J. Li, S. Chen, and J. Lu (2006). Managing quality of context in pervasive computing. In *QSIC '06: Proceedings of the Sixth International Conference on Quality Software*, Washington, DC, USA, pp. 193–200. IEEE Computer Society.

Buchholz, T., A. Küpper, and M. Schiffers (2003). Quality of context: What it is and why we need it. In *Proceedings of the 10th International Workshop of the HP OpenView University Association(HPOVUA)*. Hewlet-Packard OpenView University Association.

Capra, L., W. Emmerich, and C. Mascolo (2003). Carisma: Context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering 29*, 929–945.

Carlos, A. J. and M. Paul (2007). Ambient intelligence: Concepts and applications. *Computer Science and Information Systems 4*.

Carr, J. J. and J. M. Brown (2001). *Introduction to Biomedical Equipment Technology* (4th ed.). Prentice Hall.

Castro, P. and R. Muntz (2000, Oct). Managing context data for smart spaces. *Personal Communications, IEEE [see also IEEE Wireless Communications] Volume 7*(5), 44–46.

Catarci, T., M. de Leoni, A. Marrella, M. Mecella, B. Salvatore, G. Vetere, S. Dustdar, L. Juszczyk, A. Manzoor, and H. L. Truong (2008). Pervasive software environments for supporting disaster responses. *IEEE Internet Computing 12*(1), 26–37.

Chantzara, M., M. Anagnostou, and E. Sykas (2006). Designing a quality-aware discovery mechanism for acquiring context information. In *AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, pp. 211–216. IEEE Computer Society.

Chen, G. and D. Kotz (2000). A survey of context-aware mobile computing research. Technical report, Dartmouth College, Hanover, NH, USA.

Chen, H. (2004). *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. Ph. D. thesis, Department of Computer Science and Electrical Engineering, University of Marylang, Baltimore County.

Chen, H., T. Finn, and A. Joshi (2003). An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review 18*(03), 197–207.

Cook, D. J. (2007). Making sense of sensor data. *IEEE Pervasive Computing 6*(2), 105–108.

Cook, D. J. and S. K. Das (2007). How smart are our environments? an updated look at the state of the art. *Pervasive Mob. Comput. 3*(2), 53–73.

da Rocha, R. C. A., M. Endler, and T. S. de Siqueira (2008). Middleware for ubiquitous context-awareness. In *MPAC '08: Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing*, pp. 43–48.

de Freitas Bulcao Neto, R. and M. da Graca Campos Pimentel (2005, oct.). Toward a domain-independent semantic model for context-aware computing. pp. 10 pp.

Dey, A., G. Abowd, and D. Salber (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction Volume 16*, 97–166.

Dey, A. K. and G. D. Abowd (1999). Towards a better understanding of context and context-awareness. In *In HUC 99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pp. 304–307. Springer-Verlag.

Dey, A. K. and J. Mankoff (2005). Designing mediation for context-aware applications. *ACM Trans. Comput.-Hum. Interact. 12*(1), 53–80.

Dorn, C., D. Schall, and S. Dustdar (2006). Granular context in collaborative mobile environments. In *OTM Workshops (2)*, pp. 1904–1913.

Ebling, M. R., G. Hunt, and H. Lei (2001). Issues for context services for pervasive computing. In *Proceedings of the Advanced Workshop on Middleware for Mobile Computing*, Heidelberg, Germany. Springer.

Franklin, M. J. (2001). Challenges in ubiquitous data management. In *Informatics - 10 Years Back. 10 Years Ahead.*, pp. 24–33.

Ghidini, C. and F. Giunchiglia (2001). Local models semantics, or contextual reasoning=locality+compatibility. *Artificial Intelligence 127*(2), 221 – 259.

Gray, P. D. and D. Salber (2001). Modelling and using sensed context information in the design of interactive applications. In *EHCI '01: Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, London, UK, pp. 317–336. Springer-Verlag.

Gu, T., H. K. Pung, and D. Q. Zhang (2005). A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications 28*(1), 1 – 18.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten (2009). The weka data mining software: an update. *SIGKDD Explor. Newsl. 11*(1), 10–18.

Harter, A., A. Hopper, P. Steggles, A. Ward, and P. Webster (2002). The anatomy of a context-aware application. *Wireless Networks 8*, 187–197.

Hazas, M., J. Scott, and J. Krumm (2004, feb.). Location-aware computing comes of age. *Computer 37*(2), 95 – 97.

Henricksen, K. and J. Indulska (2004). Modelling and using imperfect context information. In *PerCom Workshops*, pp. 33–37.

Henricksen, K., J. Indulska, T. McFadden, and S. Balasubramaniam (2005). Middleware for distributed context-aware systems. In R. Meersman and Z. Tari (Eds.), *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*,

Volume 3760 of *Lecture Notes in Computer Science*, pp. 846–863. Springer Berlin / Heidelberg.

Henricksen, K., J. Indulska, and A. Rakotonirainy (2002). Modeling context information in pervasive computing systems. In *Pervasive '02: Proceeddings of the First International Conference on Pervasive Computing*, Volume 2414, pp. 79–117. Springer Berlin / Heidelberg.

Henricksen, K., J. Indulska, and A. Rakotonirainy (2003, January). Generating context management infrastructure from high-level context models. In J. Hjelm and Z. Tari (Eds.), *Proceedings of the Fourth International Conference on Mobile Data Management*, Melbourne, pp. 1–6. Monash University.

Hönle, N., U.-P. Kappeler, D. Nicklas, T. Schwarz, and M. Grossmann (2005). Benefits of integrating meta data into a context model. In *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, Washington, DC, USA, pp. 25–29. IEEE Computer Society.

Huebscher, M. C. and J. A. McCann (2004). Adaptive middleware for context-aware applications in smart-homes. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, New York, NY, USA, pp. 111–116. ACM.

Huebscher, M. C., J. A. McCann, and N. Dulay (2006). Fusing multiple sources of context data of the same context type. In *ICHIT '06: Proceedings of the 2006 International Conference on Hybrid Information Technology*, pp. 406–415. IEEE Computer Society.

Hursch, W. L. and C. V. Lopes (1995). Separation of concerns.

Indulska, J., R. Robinson, A. Rakotonirainy, and K. Henricksen (2003). Experiences in using cc/pp in context-aware systems. In *Mobile Data Management*, pp. 247–261.

Jovanov, E., A. Milenkovic, C. Otto, and P. de Groen (2005). A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and Rehabilitation 2*(1), 6.

Judd, G. and P. Steenkiste (2003). Providing contextual information to pervasive computing applications. In *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, Washington, DC, USA, pp. 133. IEEE Computer Society.

Juran, J. M., A. B. Godfrey, R. E. Hoogstoel, and E. G. Schilling (1999). *Quality Control Handbook* (5 ed.). McGraw-Hill handbooks. New York: McGraw Hill.

Juszczyk, L., H. Psaier, A. Manzoor, and S. Dustdar (2009). Adaptive query routing on distributed context - the cosine framework. In *Mobile Data Management*, pp. 588–593.

Kara, N. and O. A. Dragoi (2007). Reasoning with contextual data in telehealth applications. In *WIMOB '07: Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 69. IEEE Computer Society.

Kim, Y. and K. Lee (2006a). A quality measurement method of context information in ubiquitous environments. In *ICHIT '06: Proceedings of the 2006 International Conference on Hybrid Information Technology*, Washington, DC, USA, pp. 576–581. IEEE Computer Society.

Kim, Y. and K. Lee (2006b). A quality measurement method of context information in ubiquitous environments. In *ICHIT '06: Proceedings of the 2006 International Conference on Hybrid Information Technology*, pp. 576–581. IEEE Computer Society.

Korpipaa, P., J. Mantyjarvi, J. Kela, H. Keranen, and E.-J. Malm (2003). Managing context information in mobile devices. *IEEE Pervasive Computing 2*(3), 42–51.

Krause, M. and I. Hochstatter (2005). Challenges in modelling and using quality of context (qoc). In *MATA*, pp. 324–333.

Lei, H., D. M. Sow, J. S. Davis, II, G. Banavar, and M. R. Ebling (2002). The design and applications of a context service. *SIGMOBILE Mob. Comput. Commun. Rev. 6*(4), 45–55.

Lepri, B., N. Mana, A. Cappelletti, F. Pianesi, and M. Zancanaro. What is happening now? detection of activities of daily living from simple visual features. *Personal and Ubiquitous Computing*, 1–18.

Logan, B., J. Healey, M. Philipose, E. M. Tapia, and S. S. Intille (2007). A long-term evaluation of sensing modalities for activity recognition. In *Ubicomp*, pp. 483–500.

Maekawa, T., Y. Yanagisawa, Y. Kishino, K. Ishiguro, K. Kamei, Y. Sakurai, and T. Okadome (2010). Object-based activity recognition with heterogeneous sensors on wrist. In *Pervasive*, pp. 246–264.

Malik, A. K., A. Manzoor, and S. Dustdar (2010). *Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications*, Chapter Context-Aware Privacy and Sharing Control in Collaborative Mobile Applications. IGI Global.

Manzoor, A., H.-L. Truong, C. Dorn, and S. Dustdar (2010). Service-centric inference and utilization of confidence on context. In *Proceedings of The IEEE Asia-Pacific Services Computing Conference (APSCC)*. IEEE.

Manzoor, A., H. L. Truong, and S. Dustdar (2008). On the evaluation of quality of context. In *EuroSSC*, pp. 140–153.

Manzoor, A., H. L. Truong, and S. Dustdar (2009a). Quality aware context information aggregation system for pervasive environments. In *AINA Workshops*, pp. 266–271. IEEE Computer Society.

Manzoor, A., H. L. Truong, and S. Dustdar (2009b). Using quality of context to resolve conflicts in context-aware systems. In *QuaCon*, pp. 144–155.

Manzoor, A., H. L. Truong, and S. Dustdar (2010). Quality of context: Models and applications for context-aware systems in pervasive environments. *The Knowledge Engineering Review, Special Issue on Web and Mobile Information Services*.

Manzoor, A., H.-L. Truong, A. K. Malik, and S. Dustdar (2010). *Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications*, Chapter Quality of Context and Mobile Systems: Past, Present and Future. IGI Global.

Manzoor, A., C. Villalonga, A. Calatroni, H.-L. Truong, D. Roggen, S. Dustdar, and G. Tröster (2010). Identifying important action primitives for high level activity recognition. In P. Lukowitz, G. Kortuem, and K. Kunze (Eds.), *Proceedings of The Fifth European Conference on Smart, Sensing, and Context (EuroSSC)*, LNCS, Berlin Heidelberg, pp. 149 – 162. Springer Verlag.

McKeever, S., J. Ye, L. Coyle, and S. Dobson (2009). Using dempster-shafer theory of evidence for situation inference. In *Proceedings of the 4th European conference on Smart Sensing and Context*, pp. 149–162. Springer.

Neisse, R., M. Wegdam, M. van Sinderen, and G. Lenzini (2007). Trust management model and architecture for context-aware service platforms. In R. Meersman and Z. Tari (Eds.), *OTM Conferences (2)*, Volume 4804 of *Lecture Notes in Computer Science*, pp. 1803–1820. Springer.

Ni, L. M., Y. Liu, Y. C. Lau, and A. P. Patil. Landmarc: Indoor location sensing using active rfid. *Wireless Networks 10*, 701–710.

Niu, W. T. and J. Kay. Personaf: framework for personalised ontological reasoning in pervasive computing. *User Modeling and User-Adapted Interaction 20*, 1–40.

Park, I., D. Lee, and S. J. Hyun (2005). A dynamic context-conflict management scheme for group-aware ubiquitous computing environments. In *COMPSAC '05: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 1*, pp. 359–364. IEEE Computer Society.

Patwardhan, A., F. Perich, A. Joshi, T. Finin, and Y. Yesha (2006). Querying in packs: Trustworthy data management in *d hoc* networks. *IJWIN 13*(4), 263–274.

Pawar, P., B.-J. van Beijnum, A. Peddemors, and A. van Halteren (2007, july). Context-aware middleware support for the nomadic mobile services on multi-homed handheld mobile devices. pp. 341 –348.

Perich, F., A. Joshi, T. W. Finin, and Y. Yesha (2004). On data management in pervasive computing environments. *IEEE Trans. Knowl. Data Eng. 16*(5), 621–634.

Pietschmann, S., A. Mitschick, R. Winkler, and K. Meißner (2008). Croco: Ontology-based, cross-application context management. In *Proceedings of the Third International Workshop on Semantic Media Adaptation and Personalization*, pp. 88–93. IEEE Computer Society.

Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

Ranganathan, A., J. Al-Muhtadi, and R. Campbell (2004, April-June). Reasoning about uncertain contexts in pervasive computing environments. *Pervasive Computing, IEEE 3*(2), 62–70.

Razzaque, M. A., S. Dobson, and P. Nixon (2005, August). Categorization and modeling of quality in context information. In *Proceedings of the IJCAI 2005 Workshop on AI and Autonomic Communications. Edinburgh. Scotland*.

Roggen, D., A. Calatroni, M. Rossi, T. Holleczek, K. Föorster, G. Tröoster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, M. Creatura, and J. del R. Millàn (2010). Collecting complex activity data sets in highly rich networked sensor environments. In *In Proceedings of the 7th International Conference on Networked Sensing Systems, INSS 2010*. IEEE.

Román, M., C. Hess, R. Cerqueira, R. H. Campbell, and K. Nahrstedt (2002). Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing 1*, 74–83.

Schilit, B., N. Adams, and R. Want (1994, dec.). Context-aware computing applications. pp. 85 – 90.

Schmidt, A. (2000). Implicit human computer interaction through context. *Personal and Ubiquitous Computing 4*(2/3).

Schmidt, A. (2006). A layered model for user context management with controlled aging and imperfection handling. In *Modeling and Retrieval of Context*, Volume 3946/2006, pp. 86–100. Springer Berlin / Heidelberg.

Schmidt, A., K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. de Velde (1999). Advanced interaction in context. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, London, UK, pp. 89–101. Springer-Verlag.

Sheikh, K., M. Wegdam, and M. van Suinderen (2008). Quality-of-context and its use for protecting privacy in context aware systems. *Journal of Software Volume 3*, 83–93.

Smailagic, A. and D. Kogan (2002, oct.). Location sensing and privacy in a context-aware computing environment. *Wireless Communications, IEEE 9*(5), 10 – 17.

Strang, T. and C. Linnhoff-Popien (2004). A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*.

Strang, T., C. Linnhoff-Popien, and K. Frank (2003). Cool: A context ontology language to enable contextual interoperability. In *LNCS 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003). Volume 2893 of Lecture Notes in Computer Science (LNCS)., Paris/France*, pp. 236–247. Springer Verlag.

Tang, S., J. Yang, and Z. Wu (2007, Sept.). A context quality model for ubiquitous applications. *IFIP International Conference on Network and Parallel Computing Workshops, 2007. NPC Workshops.*, 282–287.

Taylor, J. R. and W. Thompson (1998). *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*, Volume 51. AIP.

Toivonen, S., G. Lenzini, and I. Uusitalo (2006). Context-aware trust evaluation functions for dynamic reconfigurable systems. In T. Finin, L. Kagal, and D. Olmedilla (Eds.), *MTW*, Volume 190 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Toninelli, A., A. Corradi, and R. Montanari (2009). *A Quality of Context Aware Approach to Access Control in Pervasive Environments*, pp. 236–251. Springer.

Truong, H. L. and S. Dustdar (2009). A survey on context-aware web service systems. *IJWIS 5*(1), 5–31.

Truong, H. L., L. Juszczyk, S. Bashir, A. Manzoor, and S. Dustdar (2008). Vimoware - a toolkit for mobile web services and collaborative computing. In *EUROMICRO-SEAA*, pp. 366–373.

Truong, H. L., L. Juszczyk, A. Manzoor, and S. Dustdar (2007). Escape - an adaptive framework for managing and providing context information in emergency situations. In *EuroSSC*, pp. 207–222.

Truong, H.-L., A. Manzoor, and S. Dustdar (2009). On modeling, collecting and utilizing context information for disaster responses in pervasive environments. In *CASTA '09: Proceedings of the first international workshop on Context-aware software technology and applications*, pp. 25–28.

Truong, K., G. Abowd, and J. Brotherton (2001). *Who, What, When, Where, How: Design Issues of Capture and Access Applications*, Volume 2201 of *Lecture Notes in Computer Science*, pp. 209–224. Springer Berlin / Heidelberg.

van Kasteren, T., A. K. Noulas, G. Englebienne, and B. J. A. Kröse (2008). Accurate activity recognition in a home setting. In *UbiComp*, pp. 1–9.

Villalonga, C., D. Roggen, C. Lombriser, P. Zappi, and G. Tröster (2009). Bringing quality of context into wearable human activity recognition systems. In *QuaCon*, pp. 164–173.

Want, R., A. Hopper, V. Falcao, and J. Gibbons (1992). The active badge location system. *ACM Trans. Inf. Syst. 10*(1), 91–102.

Webster, J. G. (1999). *The measurement, instrumentation, and sensors handbook*. Electrical engineering progress series. Berlin: Springer.

Weiser, M. (1991). The computer for the 21st century: Someday computers will be ubiquitous and largely unnoticeable. *SCIENTIFIC AMERICAN*. See also: Reaching for Weiser's Vision – IEEE Pervasive Computing Magazine 1 (Jan.-Mar. 2002).

Wu, H., M. Siegel, and S. Ablay (2003). Sensor fusion using dempster-shafer theory ii: static weighting and kalman filter-like dynamic weighting. In *Proceedings of the 20th IEEE Instrumentation and Measurement Technology Conference, 2003. IMTC '03*, pp. 907–912.

Wun, A., M. Petrovi, and H.-A. Jacobsen (2007). A system for semantic data fusion in sensor networks. In *DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, New York, NY, USA, pp. 75–79. ACM.

Xu, C., S. C. Cheung, W. K. Chan, and C. Ye (2007). On impact-oriented automatic resolution of pervasive context inconsistency. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pp. 569–572. ACM.

Yau, S. S. and F. Karim (2004). A context-sensitive middleware for dynamic integration of mobile devices with network infrastructures. *Journal of Parallel and Distributed Computing 64*(2), 301 – 317.

yi Hong, J., E. ho Suh, and S.-J. Kim (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications 36*(4), 8509 – 8522.

Zadeh, L. (1996, may.). Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems 4*(2), 103 –111.

Zappi, P., C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster (2008). Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection. In *EWSN*, pp. 17–33.

Zhang, H., L. Jiang, and J. Su (2005). Hidden naive bayes. In *Twentieth National Conference on Artificial Intelligence*, pp. 919–924. AAAI Press.

Appendix A

Conflict Resolving Policy Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://www.infosys.tuwien.ac.at/QoC/Policy"
    targetNamespace="http://www.infosys.tuwien.ac.at/QoC/Policy">
    <xs:complexType name="ConflictResolvingPolicy">
        <xs:sequence>
            <xs:element
                name="QoCCreteria"
                type="tns:QoCRequirement"
                minOccurs="0"
                maxOccurs="1">
            </xs:element>
            <xs:element
                name="Threshold"
                type="tns:QualityDecimal"
                minOccurs="0"
                maxOccurs="1">
            </xs:element>
```

```
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="QoCRequirement">
        <xs:sequence>
            <xs:element
                name="Reliability"
                type="tns:ReliabilityFLabel"
                minOccurs="0"
                maxOccurs="1">
            </xs:element>
            <xs:element
                name="Timeliness"
                type="tns:TimelinessFLabel"
                minOccurs="0"
                maxOccurs="1">
            </xs:element>
            <xs:element
                name="Completeness"
                type="tns:CompletenessFLabel"
                minOccurs="0"
                maxOccurs="1">
            </xs:element>
            <xs:element
                name="Significance"
                type="tns:SignificanceFLabel"
                minOccurs="0"
                maxOccurs="1">
            </xs:element>
            <xs:element
                name="Usability"
                type="tns:UsabilityFLabel"
                minOccurs="0"
                maxOccurs="1">
```

```
            </xs:element>
            <xs:element
                name="AccessRight"
                type="tns:AccessRightFLabel"
                minOccurs="0"
                maxOccurs="1">
            </xs:element>
            <xs:element
                name="RepresenationalConsistency"
                type="tns:RepresentationConsistencyFLabel"
                minOccurs="0"
                maxOccurs="1">
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="ReliabilityFLabel">
        <xs:restriction base="xs:string">
            <xs:enumeration
                value="VeryHigh">
            </xs:enumeration>
            <xs:enumeration
                value="High">
            </xs:enumeration>
            <xs:enumeration
                value="Medium">
            </xs:enumeration>
            <xs:enumeration
                value="Low">
            </xs:enumeration>
            <xs:enumeration
                value="VeryLow">
            </xs:enumeration>
        </xs:restriction>
```

```
</xs:simpleType>
<xs:simpleType name="TimelinessFLabel">
    <xs:restriction base="xs:string">
        <xs:enumeration
            value="Fresh">
        </xs:enumeration>
        <xs:enumeration
            value="MidFresh">
        </xs:enumeration>
        <xs:enumeration
            value="MidStale">
        </xs:enumeration>
        <xs:enumeration
            value="Stale">
        </xs:enumeration>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="CompletenessFLabel">
    <xs:restriction base="xs:string">
        <xs:enumeration
            value="Ample">
        </xs:enumeration>
        <xs:enumeration
            value="MidAmple">
        </xs:enumeration>
        <xs:enumeration
            value="MidDeficient">
        </xs:enumeration>
        <xs:enumeration
            value="Deficient">
        </xs:enumeration>
    </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="SignificanceFLabel">
    <xs:restriction base="xs:string">
        <xs:enumeration
            value="Vital">
        </xs:enumeration>
        <xs:enumeration
            value="MidVital">
        </xs:enumeration>
        <xs:enumeration
            value="MidNegligible">
        </xs:enumeration>
        <xs:enumeration
            value="Negligible">
        </xs:enumeration>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="UsabilityFLabel">
    <xs:restriction base="xs:string">
        <xs:enumeration
            value="Appropriate">
        </xs:enumeration>
        <xs:enumeration
            value="Inappropriate">
        </xs:enumeration>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="AccessRightFLabel">
    <xs:restriction base="xs:string">
        <xs:enumeration
            value="Authorized">
        </xs:enumeration>
        <xs:enumeration
            value="Unauthorized">
```

```
            </xs:enumeration>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="RepresentationConsistencyFLabel">
        <xs:restriction base="xs:string">
            <xs:enumeration
                value="Compatible">
            </xs:enumeration>
            <xs:enumeration
                value="MidCompatible">
            </xs:enumeration>
            <xs:enumeration
                value="MidIncompatible">
            </xs:enumeration>
            <xs:enumeration
                value="Incompatible">
            </xs:enumeration>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="QualityDecimal">
        <xs:restriction base="xs:decimal">
            <xs:minInclusive
                value="0">
            </xs:minInclusive>
            <xs:maxInclusive
                value="1">
            </xs:maxInclusive>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```