Activity Awareness in Context-aware Systems using Software Sensors

Thesis submitted for the degree of

Doctor of Philosophy

at the University of Leicester

by

Kamran Taj Pathan

Department of Computer Science

University of Leicester

October 2013

Dedicated to

Mr & Mrs Taj Muhammad Pathan,
Saima Kamran,
Maryam, Abdullah & Azlan

Abstract

Context-aware systems being a component of ubiquitous or pervasive computing environment sense the users' physical and virtual surrounding to adapt their behaviour accordingly. To achieve activity context tracking devices are common practice. Service Oriented Architecture is based on collections of services that communicate with each other. The communication between users and services involves data that can be used to sense the activity context of the user. SOAP is a simple protocol to let applications exchange their information over the web. Semantic Web provides standards to express the relationship between data to allow machines to process data more intelligently.

This work proposes an approach for supporting context-aware activity sensing using software sensors. The main challenges in the work are specifying context information in a machine processable form, developing a mechanism that can understand the data extracted from exchanges of services, utilising the data extracted from these services, and the architecture that supports sensing with software sensors. To address these issues, we have provided a bridge to combine the traditional web services with the semantic web technologies, a knowledge structure that supports the activity context information in the context-aware environments and mapping methods that extract the data out of exchanges occurring between user and services and map it into a context model. The Direct Match, the Synonym Match and the Hierarchical Match methods are developed to put the extracted data from services to the knowledge structure.

This research will open doors to further develop automated and dynamic context-aware systems that can exploit the software sensors to sense the activity of the user in the context-aware environments.

Declaration

The studies outlined in this dissertation were undertaken in the Department of Computer Science, University of Leicester and supervised by Dr. Stephan Reiff-Marganiec and Prof. Reiko Heckel. I hereby declare that this submission is my own work and is the result of work done during the period of registration. To the best my knowledge, it contains no previously published material written by another person. Any use made within this thesis of the work of other authors in any form is properly acknowledged at the point of use. This thesis contains work which has not previously been submitted for a degree at any other institution.

Parts of the research work presented in some sections have already been published, such as the Context Modelling and Reasoning (Chapter 3) in Reaching Activities by Places in the Context-aware Environments using Software Sensors [46] and Mapping (Chapter 5) in Mapping for Activity Recognition in the Context-aware Systems using Software Sensors [47]. There are some other publications of the author which are referenced in the appropriate chapters.

Kamran Taj Pathan
Leicester, UK. October 2013

Acknowledgements

‘The most Merciful, taught Quran, created Man, and taught Eloquence’, and I am very much thankful to Allah to choose Dr. Stephan Reiff-Marganiec as my supervisor who has always been very helpful and cooperative. He encourages whenever I feel low. His significant support and guidance enabled me to complete my research work successfully. With much respect, I also thank Prof. Reiko Heckel, being my second supervisor to provide very concrete opinions on my work to help me complete.

I like to say thank you to all my colleagues in the Department of Computer Science, University of Leicester specially Mr. Yi Hong, Dr. Hong Qing Yu and Dr. Monika Solanki for their help and support to feel as home away from home. In development, the discussion with Mr. Yi Hong, Mr. Noman, and Mr. Majid and various programming blogs helped a lot.

Acknowledgements are always there for my Employer, University of Sindh Jamshoro for awarding me the scholarship to complete my doctoral degree.

Finally, I am indebted to thank from my heart to my wife Saima, my family and specially my father and mother Mr. and Mrs. Taj Muhammad Pathan who taught me what I am today.

Thank you!
Kamran Taj Pathan

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CANTEXT	Computer Activity And Time Entity Location Tracking
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
she	A user of the context-aware systems
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
UML	Unified Modelling Language
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WWW	World Wide Web
XML	Extensible Markup Language

Chapter 1

Introduction

> *Prototype tabs, pads and boards are just the beginning of ubiquitous computing. The real power of the concept comes not from any one of these devices—it emerges from the interaction of all of them [84].*

Weiser [84] in 1988 founded the term ubiquitous and defines it as the integration of daily life's unseen devices. Since then it has seen much interest and in addition to the ubiquitous the term pervasive has become common. One of the areas of pervasive (ubiquitous) computing is context-aware computing. Mobility of devices and use of services make context-aware systems a popular research field these days. Context in the physical world, which involves a number of important concerns related to the use of data provided by sensors to the context-aware computing platform, which includes: what to sense for the particular type of context, how to acquire the information needed and how to apply reasoning to that information to infer the context of a user. It is highly needed that programs and services should respond according to the user's situation and behave the way she wants these to be, i.e. Services and systems should be more dynamic. Such systems would be effective for businesses and individual user. Businesses can fulfil the need of the users and the users can enjoy the system with less redundancy. To identify the context the information which we require can be captured in a number of ways, for example from user profile information, network (to sense location, time, nearby objects etc.), sensors (for activity) and other sources.

The *Active Badge Location System* [69] was very first context-aware system that allows to determine the location of the employees who wear the badge. This system also directs the phone calls to a telephone nearest to that member of staff. In late 1990s some other context-aware systems e.g. [24], [52] and [42], were also primarily concerned with exploiting the location information.

Many researchers have defined context as per their understanding in an effort to consider a more general concept of context. Schilit and Theimer [14] used the term context-aware in 1994 and described as identities, location, objects and nearby people. In 1996 Brown [65] defined context as the elements that surround a user which a computer can identify. An often cited and quite generic definition of context is that by Dey and Abowd [11]: "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 applications themselves." The context is useful and people have worked on it, focussing on location mostly, although the situation of an entity might consist time, location, activity and the surrounding factors that might affect the activity of an entity.

The term context has been classified into two categories (external and internal) by Prekop and Burnett [64] and Gustavsen [71], and physical and logical by Hofer et al. [78]. Physical context can be determined by hardware sensors and logical context is either given by the user's input or by capturing her interactions with the services available. For example through observing or analysing the user's profile, activities, working routines, typing activity etc. Most research in this area makes use of physical sensors for light, movement, sound, temperature, touch, and of course location. The logical sensors (*Watson Project* [35] and the *IntelliZap Project* [50]) however provide related information by reading user's information from opened web pages and other documents [52]. Google [26] analyses user’s data (interactions) and based on those interactions targets advertising.

While much hardware and software is already installed at the organizational level and billions of machines are connected to each other, much attention is given to the installation of additional hardware sensors e.g. [24] and [23], when gathering information about user’s activity context is concerned.

Currently the computers are no longer dependent on the desk, rather they are appearing in many forms, often handheld and mobile. Web 2.0 has given a rise to the interactivity of the user with the web applications where web pages are not being maintained by a single user, but by participation and collaboration of all users who have an interest in the content. Everything is being shared, corporates have opened back end systems to their partners for better supply chain management, customers have more direct access to their data, governments are interacting with their citizens via the web, and personal users use services such as Facebook [22] or Twitter [82] to share personal information and their current thoughts and activities. We have entered an area of information availability and sharing that can be described by “anytime with anybody”. With everyone using services for so many activities the information needed to sense activities is already communicated across networks and only needs to be exploited.

The installation and configuration of hardware sensors is time consuming and represents an extra cost. For example, physical sensors have been used to sense activities, such as whether a user is walking or sitting, or whether they are sitting at the desk and typing. Eye-tacking techniques have been explored to see what users are looking at.

Service oriented architecture gives an idea of “software as a service” which is developed and updated at a single site and users are using these services over a networked infrastructure to accomplish their day to day tasks. This leads to the question of whether such services and the information exchanged can be used in addition to hardware sensors or on their own to provide context information. We postulate that by capturing the exchanges that occur between the user and the services, we can gather data that is helpful to infer what a user is doing. The hypothesis to be investigated in this thesis is that software sensors based context-aware systems can also provide a suitable platform for activity sensing.

Analysing the problem in more detail, we can see a number of problems that need to be solved before the envisioned software sensors and their use become reality. For a start, it is required that data from sensors can be stored in structured ways, so that it can be exploited by query and reasoning techniques—semantic web technologies provide an excellent basis for this. However, Web Service communications are not usually conducted in a semantic setting, so the gap between syntactic elements that can be obtained (the details of which are also necessary to study) from service communications need to be mapped into the semantic data structures in order to create and update context elements. In this work we are addressing these questions to define the operation of software sensors and showing that they can help to reduce the effort and cost of building context aware systems (that exploit hardware sensors and therefore need tracking devices etc. an aid to the value of the context-aware systems) by using the software that is already in place. The potential is that rather than forcing a new system and way of doing things onto a user, the proposed system supports the user in what they are doing in a ubiquitous, smart and personal way..

We will introduce software sensors to monitor an employee's activity by collecting data from his interactions with the help of a Calendar Service (i.e. Title, Time and Location), Weather Service, and Profile Service.

1.1 Research Challenges

Although a number of context-aware systems have been developed over the previous years, the applications of these systems remain in early stages for various reasons. The main reason is that the field is very large and solutions heavily depend on hardware sensors being available for sensing activity context of the user.

We propose to build context-aware systems that sense activity and are based on software sensors. To build a bridge between traditional web services and semantic web technologies, we identify three main research challenges and consider some sub-issues out of these challenges. These are presented below:

1) How to acquire the context data related to the user's activities using software sensors?

- Which architecture supports and allows to capture the user's context with the use of software sensors?
- Where to place the software sensors to accomplish the task?
- What data to sense?

Service-oriented architecture is the answer to adopt the architecture to fulfil the needs of data relevant with the user's context. The structured collection of discrete software modules does not only provide users the facility to interact to do their regular tasks using standards but can also be exploited further to obtain the context by placing sensors at the right place. To acquire the context of the user software sensors are placed in between the user and the services in the service-based architecture. These sensors will monitor exchanges of messages between users and services and extract the relevant information automatically; it is worth to note that much of the information is sent across open protocols such as SOAP where one can see at the network level what operations are invoked on services, and what arguments are passed to the same. Such data relates directly to the activity of users. The context related data of the user can be used to provide means to test the research along with the utilization in the future for the general purposes to make applications context-aware.

2) Which data model adopts the semantic interoperability?

- How to layout the structure for the data?
- Which model supports reasoning rules to infer the context?
- Which model might be reused and extended for future context-aware systems?

To store access and manipulate the data acquired, the need is to present a data model. The model should provide a way to represent the context data with the logical operations that can be performed on that data. We present a Semantic based RDF model (the context model). It supports the research questions regarding a high-level context model, for example extensibility, reusability, multidimensional support for relationships, the knowledge structure and last but not the least the reasoning rules to infer the context out of the knowledge base. Raw data from the sensors can then be enriched by using reasoning tools to create context information.

3) How can this raw data become information?

- How to transform syntactic context from sensors to semantic information?
- How to map that data into the existing knowledge structure meant for context-aware systems?

To bridge a gap between traditional web services and the semantic web technologies a mapping methodology is proposed. After acquiring the data from sensors, it is essential to map it into the semantic structure so that further reasoning rules can be applied. To transform the hierarchical data (in the form of XML) into the triple format (Subject-Predicate-Object (OWL)); the challenges are to match the element node and value for the instance into the structure, to match the synonyms for that element against the instance, and to consider the knowledge hierarchy of the tag.

1.2 Thesis Statement

Sensing user activity is a major research challenge. This thesis shows that software sensors which extract activity data from the exchanged messages, storing it in a general purpose context base and reasoning about such data allows to sense user's activity.

1.3 Research Methodology and Contributions

In the field of pervasive computing one of the challenges is to make applications context-aware. A key issue for this is sensing activity so that a system adapts its behaviour according to the situation. To acquire the context data different methods have been examined to sense the activity of the user using logical or virtual sensors in the context-aware environment. An architecture based on service oriented architecture is proposed (see Chapter 4), which utilises the user's data through exchange of web services to undertake the research which can provide us the user's context, raw or otherwise.

Services carry data of the user using special protocols, to extract that data for our system use, we have developed software sensors which are going to extract data exchanges between user and services. These travel through SOAP [58] protocol.

A high-level context model is needed to store this data. We have proposed a semantic model for software sensors in a service based context-aware environment which makes use of software sensors. For further details please see Chapter 3.

The data model (Context Model) which has been modelled for context-aware systems based on software sensors provides semantic interoperability, extendibility, reusability etc. and specifically build for our case study, supports common schemas to be shared between different taxonomies because context-aware systems require data exchange between different technologies to facilitate user and services. Ontology based modelling is the approach which uses proper knowledge management, avoids inconsistency and allows application of reasoning rules. The beauty of this approach is that in the future context sources become reusable and extendable. For software sensors to work without any conflict these features play a vital role.

To put the sensed data into an existing knowledge structure is a challenge; we have proposed a methodology which not only creates the new instances but also places the data according to the existing knowledge, which is somehow already managed/structured (i.e. the Context Model). Explanation is provided in Chapter 5.

The data model for the gathered data has been presented for knowledge management and rules engine. This will be helpful for the acquired data to classify and to infer in a specific domain. This requires the reasoning rules applied to the data acquired and classified and also requires general rules to which extent the common user wants his data to be executed automatically.

While carrying out a complete study we have presented that software sensors can play a major role in context-aware systems especially in Activity Sensing. While using web services we send the context to different services at that particular time this is only a relevant data of the user but by combining these bits logically can lead to useful information and further facts can be derived out of it. In future the proposed infrastructure might be extended to different scenarios; we have attempted to make one as an example. We have provided the scenario starting from sensing the data to make it into information and further extended by knowledge management.

We have shown that traditional web services can be helpful to provide user's context and when combined with the semantic web technologies can be useful to sense the activity of the user in the context-aware environments.

This dissertation describes to develop a system, which will have the ability to sense the activity of the user using virtual or logical sensors by following the user's own record, its situational data, exchanges of messages between a service and the user. After mapping that data into the context model reasoning rules can be applied to that data to achieve the suitable results. This is aimed to relieve the burden from the hardware sensors (to sense the activity context) which might not only be costly but may also consume more time, if implemented in the future as a full.

1.4 Thesis Outline and Summary

This chapter has introduced the concepts and research questions that are answered to allow sensing activity context through software sensors.

The remaining chapters of this thesis are organized as:

Chapter 2 describes the research background and related work. We introduce the basic concepts and notions used in the area, highlight shortcomings in existing work and related this to the issues that we address. We will also introduce some more advanced issues of direct relevance to our work in more detail.

Chapter 3 presents the context model (a semantic model designed for software sensors) to store and process the data acquired from the exchanges between user and web services.

Chapter 4 develops the first major contribution of this work, namely the principles and details of the architecture for software sensors.

Chapter 5 consist of the second core contribution of the thesis by proposing the mapping methodology that helps to transform the syntactic data sensed by the sensors into semantic information that is stored in the context model.

Chapter 6 details the case study.

Chapter 7 presents the implementation and evaluation of the approach.

Chapter 8 provides concluding remarks and identifies future directions.

Chapter 2
Research Background and Related Work

In Chapter 1, we laid a foundation by discussing the research aims and objectives, challenges, research methodologies and contributions. This chapter will discuss the research background and related work.

In the first overview of the Context-aware Computing is presented and the following sections depict the overview of Service-oriented Architecture, Semantic Web, Sensors, Context Modelling, Mapping Methodology and Context-aware Applications.

Parts of this chapter are already published in [46], [47] and [49].

2.1 Overview of Context-aware Computing

Ubiquitous computing [84] refers to the collective use of computers available in the physical environment of the users, perhaps embedded in a form invisible to users. It is a vision for planting computers into our everyday life and environment, instead of representing an everyday living environment into computers.

Pervasive Computing can be viewed as a combination of mobile computing and computers embedded in the situation and so can be understood as another term for ubiquitous computing as described by IBM Chairman Lou Gerstner.

Sentient computing [05] refers the systems “using sensors and resource status data to maintain a model of the world which is shared between users and applications”. As such systems try to build a model of a part of the world from sensory information about the users’ circumstances and the environment, the idea is very much suggestive of, if not synonymous with context-aware computing, but with an emphasis on the world model [72].

Computers are producing intelligent behaviour and surrounding the user is a current concern in the future computing. Context-awareness is one of the features of ubiquitous computing.

2.1.1 Context and Context-awareness

Merriam-Webster dictionary [56] defines context as “the interrelated conditions in which something exists or occurs”. This definition can be more specified in the field as to whether those conditions are assertion in logic, a person, or a computer.

Schilit, Adams et al. [15] divided context into three categories from the perspective of distributed and mobile computing:

- **Computing context**: This can be described as connectivity, cost and bandwidth of communication and nearby resources such as workstations, displays, printers etc.
- **User context**: For example entity’s profile, location and the current situation of the user.
- **Physical context**: It can be defined as lighting, temperature and traffic conditions.

And Chen and Kotz [23] have proposed a fourth category:

- **Time context**: Such as time, week, month, and season.

The most practiced definition for the context by Dey and Abowd [3] is:

"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 applications themselves".

Talking about context in literature, at least four types are agreed to represent a context are:

- Entity
- Time
- Location
- Activity

For example An Entity performs an Activity in a certain Time and Location defines its context. Looking at the tracking devices (hardware sensors) perspective the above definitions fit fine but when we are talking about the software sensors then before creating a data model we have to take care of *Place* (not only Location) completely where the other attributes for example the place information, purpose, related data are important. For this reason, we have proposed the *Place* instead of *Location* (explained in Chapter 3), which is also part of Place's information that can limit the activity done surrounding that area or place hence affects the overall activity of the user as far as situational data is concerned.

2.1.2 Context-aware Systems

Context-aware systems are the systems that define: what to sense, way to acquire sensor data, and reasoning to infer context. Any information which can be obtained using sensors can be termed as the context, including the emotional states and movements of a user [72].

“Context aware systems are concerned with the acquisition of context (e.g. using sensors to perceive a situation), the abstraction and understanding of context (e.g. matching a perceived sensory stimulus to a context), and application behaviour based on the recognized context (e.g. triggering actions based on context)” [9]

Dey and Abowd [26] have classified as:

- Services that present information refers to the applications which either present context information to a user directly or use context to trigger or do an appropriate action(s) to the user.
- Services that execute automatically define the kind of applications which by detecting any context changes trigger a command, or reconfigure the system on behalf of the user.
- Services that attach context information for later process or retrieval states the applications that tag the data captured with the relevant information of context [12].

In the light of literature review we can conclude that the data we sense using the context-aware architecture (Chapter 4) needs the context model (Chapter 3). The context model should incorporate/consider the terminology related to context-aware systems (i.e. Entity, Place, Time & Activity), because we are dealing here with the data fetched using software sensors.

2.2 Overview of Service-oriented Architecture

OASIS [87] defines service-oriented architecture as: “A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations”. To design and develop ‘software as a service’, the service-oriented architecture (SOA) provides with the set of principles. SOA requires interoperability between different platforms using a communication protocol that depends on the messages (SOAP). SOA has three main components. The first one is client (service requester), service itself, and service repositories. The communication between a service and a client is facilitated by the service repository. Services encapsulate the description of the software; Web Service Description Language (WSDL) [86] is an XML format for describing services as a set of end points. To invoke the services, clients interact with the services in the form of messages that are transmitted using SOAP protocol for exchanging structured information. This information is in the form of XML and carries the arguments client sends to perform its task on web services. This is where we can sense the user’s context in the form of raw data only. Following sections will unleash to make this data usable in our proposed system.

2.3 Overview of Semantic Web

The inventor of WWW, Tim Berners-Lee presents an idea on how to make the existing web intelligent? this is billions of pages and increasing with no semantics at all to help with the users’ need, so is the dream of context-aware systems, therefore considered as a supporting testbed for the context-aware systems that utilize software as the sensors.

Berners-Lee et al. [73] wrote in the year 2001, “The Semantic Web will enable machines to comprehend semantic documents and data, not human speech and writings”. This is not a new web but the extension of the current web. Automatic reasoning is the answer to some of the challenges like vastness, uncertainty and inconsistency in the semantic web although, semantic web uses RDF while most web sites use HTML.

To take a journey from traditional web services to the semantic web technologies three of the technical standards of the semantic web are defined below:

The data model of semantic web to store the information is represented by RDF (Resource Description Framework) [60]. SPARQL (SPARQL Protocol and RDF Query Language) [20] is used to query data of the semantic web. OWL (Web Ontology Language) [19] is the schema language of the semantic web to represent the knowledge and is explained in section 2.5.

Semantic web services combine web services with ontology-based data about data (metadata) and provide semantic interoperability to the syntactic operations and data. In our system we intercept the messages between a user and web services and therefore are interested to make the semantic model out of that interaction. Semantic web technologies have leaps in the last decade in the standards. The giant companies (e.g. Google [26] and Facebook [22]) are moving to the open graph/knowledge graph. This motivates us to use this technology which has a bright future ahead. Next section continues the chapter with the overview of the sensors.

2.4 Overview of the Sensors

To reach the context-aware systems we add an element of software sensors with the SOA and Semantic web technologies. Sensors are almost as varied as their applications. According to Klein (1999) [51], a sensor is a transducer (front-end hardware) that “converts the energy entering its aperture into lower frequencies from which target and background discrimination information is extracted in the data processor." or "A sensor is a device that converts a physical phenomenon into an electrical signal. As such, sensors represent part of the interface between the physical world and the world of electrical devices, such as computers. The other part of this interface is represented by actuators, which converts electrical signals into physical phenomena" [41]. A sensor might be hardware as well as software, or the combination of both, whichever acquires context related data. As a broad definition the devices which return context information are considered as sensors. For example clock of the computer is accessed with the help of operating system or a tracking device [72].

Recent work has utilized sensors to observe patterns in real life. For example, in a study by Clarkson [17], data was collected using two cameras, a microphone, and an orientation sensor worn by a user for 100 days. Scene segmentation and quantitative analysis techniques were then applied on the data to extract patterns in daily life, demonstrating that the life of a person can be segmented into distinct situations distinguished by location and activity at that location and that a "person's life is not an ever-expanding list of unique situations", there is a regularity in daily life's situations.

2.4.1 Context sensing

Context sensing refers a method the context-aware applications apply for context information collection Ferscha et al. [04] define two types of context sensing mechanisms:

- **Event-based sensing**: Event based sensing is a continuous watchdog method that monitors the environment for occurrence of events and It sends the notification to the applications immediately.

- **Continuous sensing**: Continuous sensing explores the real world information over time, continuously. It filters the streams with respect to data volume and importance with respect to applications.

The data which is obtained out of those techniques is called: continuous data and event-based data. Example of continuous data sources are light conditions, temperature etc. and an example of event based data is when a continuous source changes or triggers to any event like change in the context information. This occurs when an event is generated based on information from a continuous data sources changes. For example a user enters in the meeting room and the document based services trigger an event to retrieve all the documents that are required in that room to facilitate the attendees, in response to the time written in the calendar that shows the activity of the user as Meeting with her colleagues.

2.4.2 High-level Context Sensing

Parallel with the contextual information like location, time profile, we also consider activity of the user as a high level context. One approach is computer vision that is based on the combination of input devices e.g. cameras. But as we are dealing here with sensors from software side then the other approach might be to sense the user's interaction with the services (e.g. calendar service) to find out the activities of the user for a particular time. It is true that the user's calendar might not always be that much updated either intentionally or otherwise, or sometimes the user might be reluctant to write all her activities in the calendar [08].

2.4.3 Sensing Changes in Context

Displacement (change of Place) of the entity needs to be updated every time the context changes at the context-aware system level. One of the types of context (i.e. *location*) can be dynamic for a person but very less changeable for a computer device in a room hence affects the behaviour of the change in context. Brewington and Cybenko [16] estimated change rates by observing previous information and tested on web pages. This is also one of the examples of sensing with software. Web usage mining allows collection of web access information and can be collected via access logs. However, the proposed approach advances by using service-oriented architecture and captures data that is being exchanged for the services to interact with the user and can be utilized to the more personal devices i.e. smartphones than the personal computers where we are not certain about the actual user's availability. We sense the messages that are travelling from a user to the services to perform certain tasks. Making a time interval to think what to sense, when is a good time to sense, either we would set a time interval or would check all the patterns of usual life of a user etc. Because when we look at the types of context which reflect on context completely then all the types are dynamic. The techniques to sense context need to be reliable and robust, which is very hard sometimes for example if an active badge [69] is not worn by the user but placed at the table for certain amount of time, then the location determined by the system will be false. In such a way we should build certain systems which are not only reliable but also robust in their output by not leaving any side un-reachable. In this regard the other gap can be filled with applying software sensors so that whenever a user uses any service from other sources we can sense the context of her.

2.5 Overview of Context Modelling

A context model helps data to process or store easily in the context-aware system. To compare context modelling approaches we need to differentiate various modelling techniques. We portray here the existing models used to represent, store and exchange context data. We have reviewed a classification of models based on the used data structures. We elaborate different models for processing of data for the context and present our own for the context-aware systems where software sensors play a major part. To model the context information (i.e. location) Schilit et al. [15] used key-value pairs. Markup Scheme models are based on a hierarchical data structure which consists of markup tags with attributes and content. They are usually based upon XML [74] type languages such as RDF/S [60] and have the advantage of easy tool access, but lack of formality and expressiveness. Unified Modelling Language (UML) [54] is a language to model the context using UML diagrams. Bauer [34] and Henricksen et al. [44] and [45] modelled in-air traffic management and a context model which is graphics oriented. Like any object oriented approach the object oriented models provide the features of encapsulation and reusability. Like object oriented paradigms object level details are encapsulated (hidden) to other components. A first order logic based context modelling approach has been published by McCarthy et al. [37]. They introduced contexts as abstract mathematical entities which are helpful in artificial intelligence. Ontology is a mechanism to specify concepts and their interrelations [81]. Context models based on Ontologies have been first proposed by Otzturk and Aamodt [63].

They first proposed the concept by analyzing psychological studies in combination with contextual information. They normalized and combined the knowledge from various domains. The basis to propose an ontological model was because of the sound knowledge in the field of formalization; the knowledge is evaluated using a Reasoner. CoBrA system [28] uses broker-centric agent architecture and is another approach based on ontologies. It provides ontologies to characterize persons, objects and places in the runtime, e.g. intelligent meeting rooms.

A different view is proposed by context models based on ontologies, which have been first proposed in [63]. These models allow the knowledge management by normalizing the information from different domains thus being better positioned for reuse and extend. The CoBrA system [28] characterizes entities (i.e. places or persons) by providing the ontological concepts into the context. Most of the existing work focuses on describing people and their locations; it also considers time. However, it usually does not consider the activity or the motivation (the why).

Early models were mainly modelled for specific domains or even if it is in general area then it is lacking for software sensors. Generic models are of interest because number of applications can take benefit from those models and can share knowledge across different platforms but they are very hard to make. Model-oriented approach supports extendibility and reusability. Context reasoning is usually based on semantic web technologies.

Gu et al. [76] have modelled context based on an ontology-oriented approach but this lacks the upper ontology by not making it more general according to software sensors, which may affect context reasoning in the domain specific ontologies where physical sensors are not that much involved. While creating the semantic model we have broadened their work fulfilling the shortcomings and present our ontology-based model for context that try to stick with the agreed types of context and addresses these limitations which occur when we deal with software sensors' kind of systems (explained in Chapter 3). We have emphasized more on activity context because to describe a rich source of information for advanced adaptation in collaboration like inContext [31], activity context plays a major role. Considering different models we can conclude that the most promising approach to context modelling for context-aware environments related to requirements in this section is ontology based models.

Along with many other advantages ontology based models offer structure, extensibility, reusability, formality and simplicity which are the paramount need of context-aware systems.

We have chosen Web Ontology Language (OWL) because it fulfils all the requirements we need on systems like structure, expressiveness compared to other ontology based languages, it has the capability to be distributed across different platforms, scalability, compatibility, accessibility, openness and extensibility, automated reasoning, and of all a W3C [85] recommendation. Also being a Web language it is an obvious choice to be used in connection with services. Other approaches may be suitable for the various environments but this approach is better for software sensors where knowledge management is at the priority.

Ontology-based models provide formalization for specifying the concepts and sub concepts; and provide the relationship and constraints between these. Through this approach knowledge sharing and reuse in context-aware environments is easy and standardized. Further it is evaluated using a Reasoner. These are the core reasons that help us to choose this technique and propose a system based on software sensors. To model for software sensors is a key to context-aware systems. Early models addressed mainly for the other part of the story i.e. hardware sensors or for the specific domain. Context models make any application build easier if addressed to the wider use. Most of the models are designed for the user's current situation, others for the locations and time. This study aims to design the context model that covers the main features of the context-aware systems, for querying the knowledge base and applying reasoning rules to facilitate the exchange of context, where data comes from the software sensors.

2.6 Overview of the Mapping Methodology

As of now, we have managed to extract soap messages for the activity context of the user. These messages are written in XML [74] which is a very flexible format designed for large-scale electronic publishing for exchange of data on the web and other places. The important thing to understand here is that the data we achieved is in a form which lacks in conceptualization hence makes mapping a necessary step.

Web ontology language (OWL) [19] has been designed in a way to let applications and machines to process the information instead of representing it only for humans to understand. OWL provides interpretability of web content for machines along with formal semantics than XML, RDF and RDF Schema.

After identifying the importance of mapping, let us review that what different kinds of methods are already available.

Ferdinand et al. [56], Bohring and Auer [27], and Ghawi and Cullot [67] presents different methods to generate OWL ontology from an XML data source. Rodrigues et al. [76] presents the technique which manually maps XML schema documents to the existing OWL ontologies and automatically transform XML instances documents into individuals of the mapped ontology, without looking if same meaning of the word (a tag name in XML) is presented in the concept, resulting the repetition of the word exists in the instance that does not justify to the mappings for the instances to place properly. Especially in software sensors related data where we are more interested about the instances to fit appropriately. The shortfall of the available techniques is that if an ontology developer has put a synonym (from a lexical database which is also valid move) instead of the actual word into the ontology then these techniques will create a new class or instance by disturbing the structure. In the same way, if the hierarchy of the classes (as per lexical database) is not properly modelled then it will create a new class without checking the proper hierarchy suggested by the lexical database hence does not support conditional mapping.

Every other mapping technique try to solve the puzzle of trees with labelled nodes of XML to the representation of triples in the OWL, tags with the resources or literals (for details please see Table 2 of Section 5.1); but these skip the importance of instantiation at the proper place which finally affects the outcome of reasoning support for web engineering. Various techniques overlook the details of the tag name itself, and the value attached to it, explained in Chapter 5.

We are taking care of the instances as well as structure of the knowledge which makes this mapping (explained in Chapter 5) novel to support the field of activity sensing in the context-aware systems using software sensors.

2.7 Overview of Context-aware Applications

The context once known can be used in different applications to meet the user’s own convenience for the purposeful object. In the proposed system however the inference rules are specific to the system but the instantiated data can be used in different applications where the system needs activity context. We have reviewed the number of context-aware applications like Dey [11], Chen x Kotz [23] and Baldauf et al. [55], these are presented below:

Active Badge Location System [69]

Context: *Location*.

One of the first location-aware system that exploits the location of the badge wearer and sends information to the receptionist. The system helps to forward the phone calls at the user's convenience e.g. nearby telephone according to the current location.

Teleporting – making applications mobile [21]

Context: *Location of the User and Workstations*.

Teleporting, called as "followme" computing makes use of an application to follow the user by mapping the user interface from the location of the user to the nearby resources which are in the same circle.

Mobisaic Web Information System [25]

Context: *Location and Time*.

Mobisaic Web Information System refers context information in the dynamic URLs by the authors. This system helps with active documentation through which the web pages change automatically by looking at the change of other variables e.g. environmental.

Shopping Assistant [1]

Context: *Location of the Customer within the Store.*

Shopping assistant provides help with the position of the items, and states any sale of the items to the shoppers through the store. This system stores and maintains the users' profiles which is a privacy concern.

Cyberguide [24]

Context: *Location and Time of the Tourist*.

This system provides location and direction information of the tourist on the interactive map. Diary is also maintained automatically to suggest other places that might be catchy to the tourist. Global Positioning System (GPS) and an Infrared (IR) Positioning System detect the outdoor and indoor location respectively.

Conference Assistant [12]

Context: *Location of the Attendee, Time, Plan of Presentations*.

This system examines plans of the conference, presentation topic, location, and research interest for the specific user to suggest that one should attend the presentation. It further takes hold of the conference by recording the presentation and comments for later use.

People and Object Pager [65]

Context: *Location of the User, Objects and the People Nearby*.

This system sends a notification to a badge wearer who has not worn any pager through nearby visitor. Another approach of the system is whenever any book is needed the message is broadcasted to all so that notification may be sent afterwards.

Fieldwork [6]

Context: *Location of the User and Time*.

Fieldwork provides assistance in the data collection and observations by fieldworkers. The system records data regarding environment, location and time tagged on the map.

Adaptive GSM phone and PDA [7]

Context: *Activity of the User, Level of Light, Proximity of other People.*

In this system font size becomes larger or normal according to the user's activity i.e. walking or stationary respectively on the PDA. This adopts the environmental conditions and ringing patterns of the phone also.

Office Assistant [30]

Context: *Activity of the Office Owner and Schedule*.

Whenever any visitor comes at the office door and places their feet on the pressure sensitive mats, the assistant (an agent) interacts with them to manage the schedule of the office owner, should he is busy, free or otherwise, by looking at the visitor's ID. Such systems would be more helpful if scheduled information of the visitor could be gathered, to help automatic appointment system like inContext [31].

ComMotion [57]

Context: *Location and Time*.

This project developed at MIT creates messages considering time and location the recipient arrives at a place. The system can also send and read voice messages on the screen for the user. CybreMinder [10] at Georgia Tech reminds according to the location of nearby people, and current weather conditions.

The use of context-aware systems varies from application to application. We can easily understand from the literature that till now most of the work is done for the *location context* and less work is done in the *activity context*. Activity context is sensed with the help of hardware sensors and the activities like standing, walking, sitting etc. are being sensed using hardware sensors which are at some stage useful but many activities of the users are still in infancy.

Hardware sensors can best answer the questions: what the user is doing? (e.g. *typing*); but may be less advantageous when the questions like the following occur: *what a user is doing with typing?,* where software sensors can answer more appropriately. Software sensors can also sense the way a user is playing with the computer, so that a change of software should occur accordingly to a user’s current context as per her needs and criteria which is the cornerstone to the field of context-aware systems.

This research has proposed to work on software sensors that sense the user’s activity by looking at the exchanges occurred between user and services, this sensed data is then mapped to the structured knowledge to further infer the context. We believe this solution will aid researchers to develop more into software sensors based systems.

2.8 Summary

In the first we have started with an origin and overview of the field of context-aware computing, and then reviewed the literature of service-oriented architecture, semantic web, sensors, context modelling, mapping, and context-aware applications, to support our research along with a description where it is helpful with the proposed system.

Next chapters provide how we proceeded with the proposed system to accomplish the task of sensing activity context with the use of software sensors in the context-aware environment.

Chapter 3
Context Modelling and Reasoning

A formal context model will provide the structure needed to reason about stored content; it will also allow for sensed data to be converted into richer information by adding semantics and relations between data items. As we are looking at larger heterogeneous systems, it is meaningful to define the model in detail, as it will act as a bridge between sensor technologies and the users of context information.

To capture the user's activity context the characteristic of the approach is not only to represent the model for the high-level context to low-level sensor data to be used for hardware sensors but for the software sensors as well. When we consider the virtual data then we need to document the resources of the environment (resources of the place where activity is performed) which might affect the overall activity of the user. This chapter presents the proposed context model (Figure 2) that addresses these issues and provides the context model for software sensors.

Parts of this chapter are already published in [46], [47] and [49].

3.1 Context Modelling

Context modelling specifies the context related entities and relationships and constraints between each other. Context models provide the structure and

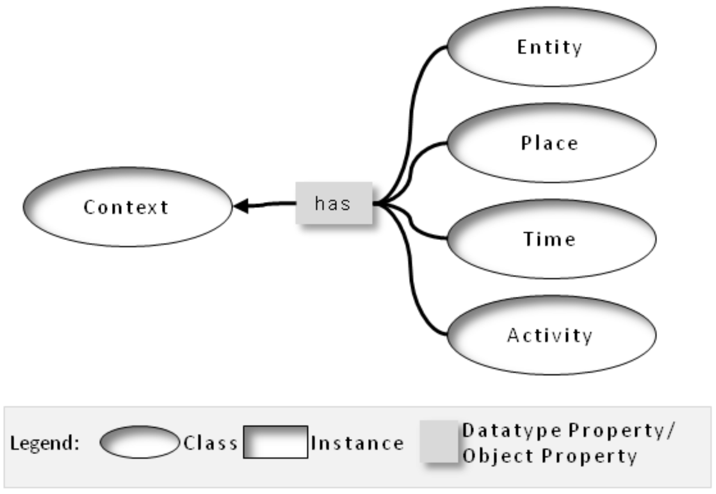


Figure 1: Upper Ontology of the Context Model

methods to process the context data. To bridge the gap between acquiring context data acquired from traditional methods and using/reusing it further, a clear need for a well-designed model exists. This allows system to which follows the definition and infers information, and will also allow for different systems to interact in meaningful ways.

We have presented a novel semantic model for software sensors in a service-based context-aware environment. It provides general upper level classes, properties and relations between them so that lower ontologies can easily adopt for particular domains. If a generic model of context can be formed then it is beneficial for systems or services which are interacting with each other and the end users, who ultimately tries to accomplish their daily tasks in the context-aware environments utilizing software sensors. The use of standard representation languages will make sharing, reuse and extension of context models and context data easy. In addition, a more formal context model provides the ability to infer new facts from gathered context information.

The model of context presents the context in a form which provides a general level upper classes, properties and relations of Place and Resources etc. hence considers the aspects which are needed to model the context information for the use of software as well as hardware sensors (in addition to the previous designed models for later category only), so that lower ontologies can extend them for specific domains. The model is intended to allow storing data at a higher level of abstraction than that gained from software sensors, in some sense one could refer to this as a situation.

3.2 Design Considerations

The semantic web [55] provides standards for structural, syntactic, meta-data and ontology languages. We have selected the Web Ontology Language (OWL) [19] to develop our context model because it provides the required functionality and is a widely accepted standard (a W3C [85] recommendation) which also has good tool support. OWL allows interoperability between systems and services and supports inference mechanism by providing additional vocabulary, defines taxonomies, properties, relations and constraints.

In our work we have used literature to define four semantic types (*entity, place, time* and *activity*) which reflect the W4H [11] concept (explained in Chapter 2): for context descriptions (*who, where, when* and *what*) respectively. *How* (which refers to devices) has been implemented as a subclass of entities, accounting for the fact that devices nowadays might act on their owners' behalf.

We will now consider these four elements in the context of software sensors in some more detail:

Entity (who and how): An entity that performs some actions and can be an actor, an organization or an object (machine or non-machine etc.).

Place (where): physical location in a pervasive environment. The place also holds some information which affects the overall situation of the user (e.g. its category, location etc.). The location is not static but dynamic and can change with the passage of time. Some places have purpose to be built and if we know where the interaction takes place; we can design the system architecture.

Time (when): The *when* addresses the time of that interaction in systems or services. The simple approach is registering everything or relevant events indexed according to the time they occurred hence an important one to the event notification services.

Activity (what): This describes an activity performed in a pervasive environment. To obtain the information regarding the activity of the entity is a challenge. The traditional systems exploited hardware sensors to solve the problem but this model utilises software sensors to solve the same problem. For systems that utilize the software sensors must record and structure all the information relevant to the former types of context.

Context-aware computing needs information to be exchanged between entities, which might be users or services, and the context model should support that interoperability in a semantic way (users and services might use different terminology). Taking an example from the above Scenario a user's activity can easily be understood from his profile, calendar, timetable and email services.

According to the definition given by Dey and Abowd [11] context has a great variety. Out of that variety we are mostly concerned about activity context here, which obviously relates to the other aspects of context. Context information is interrelated as per the above definition. For example in our scenario (described later) if a user has updated her calendar for a trip but because of weather condition is unable to fly then we will have to consider this factor also.

In this discussion we have come to know that ontology based modelling is the approach which uses proper knowledge management, avoids inconsistency, and applies reasoning rules. The beauty of this approach is that in future context sources become reusable and extendable. For software sensors to work without any conflict these features will help to widespread the knowledge and play a vital role.

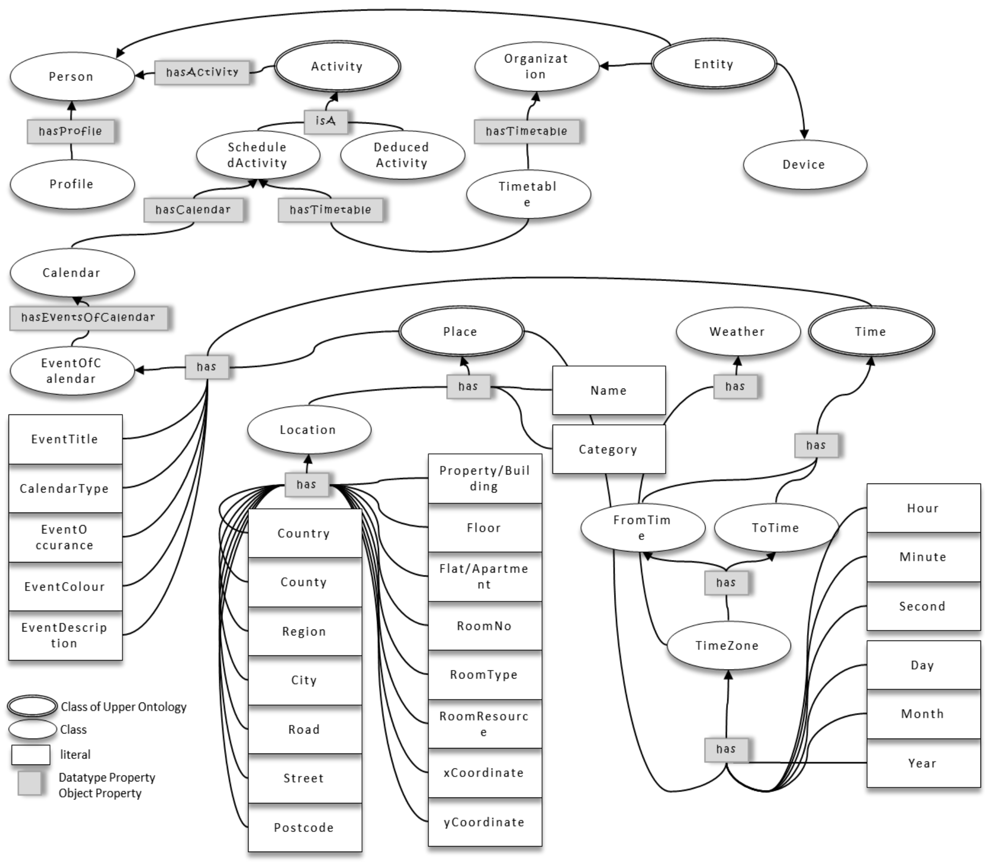


Figure 2: An Overview of the proposed Context Model for Activity Sensing using Software Sensors in the Context-aware Environments

3.3 Proposed Context Model

The ability to store and retrieve context information in a structured form is essential. Suitable structures can be defined in a number of ways, but most commonly a context model is employed. An overview of the proposed context model based on literature and definitions of context is presented in Figure 1. The inspiration of our context model is based on the ontology by Gu et al. [76], that provides a vocabulary for representing knowledge about a domain and for describing specific situations in a domain. The ontology addresses taxonomies that are related with the more traditional physical sensors hence provides a gap for our research. The inspiration is taken from Gu et al. [76], however the ontology is designed by following the literature, context types related with the situation of the user and considering software sensors mainly to sense the activity of the user. Because the structure based on hardware sensors take input as present activities captured from tracking devices etc. For example standing, walking, sitting, listening to music, and entering the kitchen etc.. However, using software sensors our studies not only cover present time but also through scheduled and deduced activities cover future dimensions and concrete activities like meeting, teaching, stuck in the traffic etc. To deal with such situations, we have proposed that for the clear semantics all related knowledge should be structured. Some of the ontology-based models are already available which either support more to hardware sensors (e.g. CONON [72]) or limited in its domain for example inContext [31].

We have proposed a semantic model for software sensors which extends some of the terms (e.g. *Place's Category* defined in Place Ontology, *Non-Computerized Objects* etc.) which can add more meaning to the context and can be helpful for sensing context. This is a data model which describes how the data is stored and processed. The model defines a common vocabulary, structure of the context for sharing context related information between users and services to enable operations. It includes machine-processable definitions of the basic concepts in the domain and range and relations among them and reasoning becomes possible by explicitly definition. Since we use software as sensors these characteristics help to achieve our target. This model represents the concept of *activity*, *time*, *place* and *entity* and further those subclasses are derived out of the classes to justify the concept for the context-aware environments.

Reuse of existing ontologies has been considered where appropriate, to model

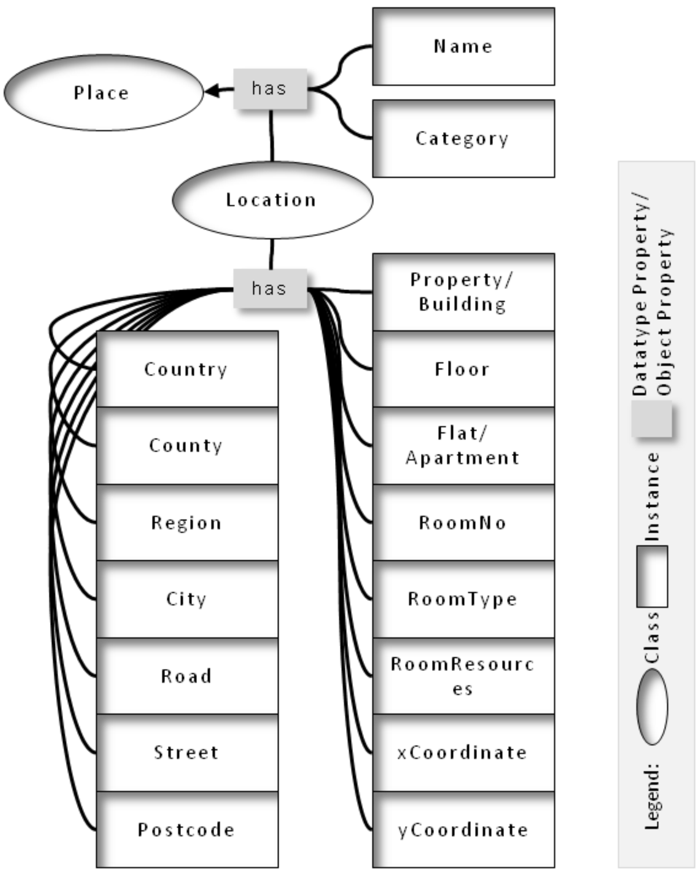


Figure 3: Place Ontology for the Context-aware Environments using Software Sensors

the context model in order to address different dimensions which could fit into the context-aware systems.

Our context ontologies are divided into upper ontologies and lower ontologies. The upper ontology models general terms of context about the physical world, keeping an eye on software sensors (e.g. place, object etc.) in context-aware computing environments. The lower ontologies model the details of general concepts and their properties (e.g. place type and category) in each sub-domain. With the change of an environment in the context-aware systems, the former may be used independent of specific domains (specific areas of context-aware systems) while specific needs might be covered in utilising the later one for specific concepts of relevance in a specific area.

3.3.1 Entity ontology (Actor, Object, Organization)

The main ontology from where the other ontologies connect to is *Entity ontology*. When we talk about user's context then it is the focal point to model the profile of the entities in a context-aware systems environment e.g. *Actor* and *Object*. This can be considered a type of context that is temporally quite stable in that it is stored once and does not get updated frequently. It may be stored for long or short period of time depending upon the application for which it is being used. Because a user's profile changes less frequently unlike the Activity she performs. We have modelled this ontology by keeping an eye on our proposed system that uses software sensors in which services are hierarchically represented as *Object<-Computer<-Services*. Service selection has to select services to help a user complete a specific activity. This selection is influenced by the service context and the user context. User context is usually gathered by use of hardware sensors and possibly by mining of past behaviour records. However, as we more and more use communicating software, especially Services, they can provide a lot of

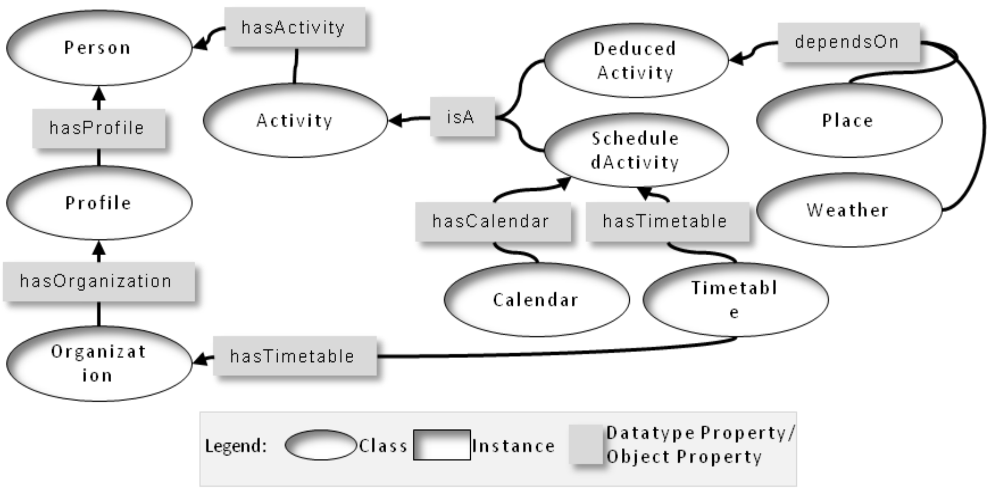


Figure 4: Lower Ontologies for Context-aware Scenario Using Software Sensors

information on the user activities. We have placed computer (machine) or non-computer (Whiteboard, Desks used to conduct a class) category in *Object* because we are dealing here with software sensors and due to that if some kind of computerized devices or non-computerized material are available at the work space might limit the activities. Rules are very useful and help with that aggregated data e.g. by combining person's *profile* and the nature of the *objects*, activity can be inferred that; what user would be doing at that particular time.

Entity is categorized as different roles performed by a person, organizational profile, different groups, projects and person's profile. In the object category further to computerized and non-computerized objects that reflect the very nature of software sensors which covers everything from services data to inventory listing that can possibly affect the circumstances. For example, a building X is used for the classes only, the room 3 of that building is the lab for postgraduate students only. The Object describes computerized and non-computerized devices, which include a set of meta-data that models the features of different devices and resources regarding different platforms like: Services, Applications, Devices, Network, nearby things available and Agents. These platforms model the data about the devices e.g. what devices are available, what is the capabilities of I/O devices. So that inference can take place, if the user requirements are to conduct a class of 50 students and to demonstrate for that particular time the monitor and overhead projector are needed; whether that particular room having these descriptions fulfils the user demand and so the activity of user might be to conduct a class in that room.

3.3.2 Place ontology (Area, Position, etc.)

Dealing with software sensors, the availability of the user at different places (not the location only but the surrounding of it) affects the activities she performs. All those considerations are according to resources available at that place. For example if a user is in her own room does not states that she cannot teach. If all the resources are available in her room that can get across her message to the wider audiences (e.g. virtual class) can lead to a chance that she can teach. Places may have the *names*, *category* (e.g. restaurant), *location* (i.e. address, coordinates), and *resources* (computers, tables, chairs and TV etc.) that define the meta-data about that location therefore the upper class of *location*. Going further down location ontology describes the geographic coordinates (e.g. altitude, longitude, latitude) of the real world entities. This depends upon the placement/displacement of an entity, and will be changing in memory, if a displacement/relocation of entity occurs.

The user's activity become more effective by the place she is in. This does not only deal with the coordinates but also usual information like street, city, building, room etc. Locations are further divided into indoor and outdoor. Outdoor locations can best be located through the use of GPS systems, while talking about indoor locations then along with the other methods e.g. tracking devices (hardware sensors) have been the best approach so far. But this research tries to prove that those costly and time consuming methods may be lessened to go through the other options by looking at not only the location but also the activity of the user.

To know the exact location of a user, systems get help from hardware sensors, but as we are dealing here with software sensors, we can infer the location of the person from the messages he send and receive from the Services, mapped into a knowledge base and by applying rules on his current profile, places of interest, frequent visits etc.

3.3.3 Time ontology (Time, Day, Month, etc.)

For any event to occur there is a starting time (the time when that event starts) and ending time (the time when that event ends), no matter it takes this to happen even in the fraction of time in the real world. This may include time interval of activities (e.g. conducting a surgery class for 01 hour(s), 23 minute(s) and 17 second(s)) which is nearest to exact or might include dates (e.g. Monday, May 03, 2011). The most dynamic type of context which keeps changing in memory, even with a next instance is: Time Context.

Interval of events and the related information is covered by Time Ontology. A temporal instant is any one of the points on the universal timeline, enclosure of distinct and convex time instants makes a temporal interval [70]. While modelling some properties of event intervals and temporal relations between intervals and dealing with the services we believe that the time duration of activities are as important as the reaching time at that particular point. Looking at the available ontologies of Time we have modelled Time

Property	Domain	Range
isSupervisorOf	Researcher	PostGraduateStudent
isDirectorOf	Administrator	MastersProgram
hasStartTime	EventCalendar	Time
hasEndTime	EventCalendar	Time
isTeaching	Teacher	Class
isA	Faculty	Researcher
isA	Faculty	Administrator
isA	Faculty	Teacher
isAttending	Student	Class
hasPlace	Faculty	Place
hasLocation	Place	Location
hasCategory	Place	Category
hasResources	Place	Resources

Table 1: An Example of some of the Properties of the Lower Ontologies of Activity-aware Scenario

ontology according to our requirements to represent time in the occurrence of events in the context of the user.

3.3.4 Activity ontology (What users do?)

The Activity ontology describes 'What the user is doing?' of the model, the actions that a user does in the context-aware environment; describes situation of the user. This type of context is dynamic and will be changing inside memory every time a new or already working task changes. This has a high significance in research to make users free from feeding their changing tasks regularly. As our research mainly rounds between this category of the context, so as a result in this study the Activity Context is considered more. Activities are of two types: Scheduled and Deduced. The former one represents the activities which are planned according to date, time and location (e.g. project meeting), whereas the later represents the activities that occurs in an informal manner and can be inferred by combining scheduled context along with the other factors of the context and applying rules on the available data.

Combining all above referred data into one can be used to characterize the situation of an entity or context. In certain applications data may only be required as an input to an output device or sometimes the same data might be useful to store for long. In the next section the lower ontologies along with an example is defined to have a better understanding of the scenario.

3.3.5 Lower ontologies for activity-aware scenario

To demonstrate our scenario in conjunction with the model, lower ontologies are presented for the activity-aware scenario in the university domain. As we have already discussed, we are concentrating to make use of software sensors hence modelled our ontologies in such a way that can make a good use of the exchange of the messages. The nodes of the concept shown in Figure 4 represent the classes and subclasses relationship between concepts. Some of the properties and relationship between them are also presented in Table 1.

These taxonomies represent the extended model for the case study and are extended according to their properties only. This model classifies two disjoint activities a) ScheduledActivity and b) DeducedActivity that are subclasses of Activity. Considering the activity-aware university domain, Scheduled Activities are the activities which are updated either by user herself maintaining her calendar or by organization which publishes timetable. On the other hand Deduced Activities in the scenario include activities which are inferred by looking at the available data and applying inference procedures onto it.

The next section talks about the inference procedures to help infer the knowledge out of the existing facts.

3.4 Inference Procedures for Context Reasoning

To derive new facts by applying rules to existing facts in the knowledge base is called inference. Rules help us to reach at the results. To infer an activity context, scheduled activity can be known by defined activities but deduced activities can be inferred from singly or combining two or more context types. For example scheduled activity context i.e. teacher’s current activity can be queried from Faculty’s work calendar or Organization’s timetable, although deduced activity context can be inferred either from personal information, personal calendar, organization timetable, place, object and time individually or combining them all otherwise.

We have instances in the ontologies. A class may contain individuals, instances of the class and the instances may relate to any number of classes. The individual presents facts (statement) through graph; the reasoning rules might be applied on the existing facts to derive new facts.

As we are modelling for the activity-aware scenario so by considering this in a university domain some of the common scheduled activities are shown in *Activity Ontology*. This may apply to deduced activities through inference mechanism, because when we say deduced activities refer to the combination of scheduled and no. of other classes which might approach the final outcome and can change the situation of a user. If a person is physically present at

Leicester UK and person can have only one location physically, implies the person's physical location is Leicester UK (i.e. the standard rule for transitivity states that $A = B \wedge B = C \rightarrow A = C$). Activities performed in a class room (e.g. to conduct a class or a surgery) depend upon the role of the person who is either a teacher or a student.

The places along with a category and resources can describe what things are placed in that particular area which can affect the activity of the person. For example Room No. 02, Third Floor, Charles Wilson Building at University of Leicester, UK, can only be used as a *computer lab* not as a *group discussion* or *class lecture* infers the activities of those involved into that particular place.

Some of the semantic rules for the context model are mentioned here:

1. $hasPlace(?fm, ?pl) \wedge Time(?t) \rightarrow isTeachingClass(?fm, ?pc)$

where *fm* is a faculty member, *pl* is the name of a place or a building, *t* is the time zone and *pc* is a postgraduate class.

The above rule infers that a Faculty member is teaching a Postgraduate class if he/she is at certain place according to the time specified.

2. $hasTimetable(?fm, ?ocal) \wedge hasCalendar(?fm, ?pcal) \rightarrow isTeachingClass(?fm, ?pc)$

where fm is a faculty member, $ocal$ is the organizational calendar (including a timetable), $pcal$ is the personal calendar and pc is a postgraduate class.

The rule states that if a Faculty member has an entry in his/her office calendar as well as the same entry in his/her personal calendar that he/she is teaching in a Postgraduate classs infers that he/she is teaching at the Postgraduate class.

3. $hasTimetable(?fm, ?ocal) \wedge Weather(?loc) \rightarrow isTeachingClass(?afm, ?pc)$

where fm is a faculty member, $ocal$ is the organizational calendar (timetable), loc is location, af is another/alternative faculty member (distinct from fm) and pc is a postgraduate class.

This rule deduces the activity of the person in a way that it mixes two facts and infers accordingly, where a Faculty member has however updated his/her activity as teaching but because of the worst weather conditions at that place infers that the alternate Faculty member is conducting his/her Postgraduate class.

Taking an example of Rule 3, here we have set a rule that if according to Organization Calendar the Faculty is liable to teach the postgraduate class and the exception will only be accepted if the alternative item is of the same or more importance than the agreed item. In such a case, duties of teaching a postgraduate class can only be transferred to an alternative faculty if the weather is bad and the actual faculty will not be able to turn up and conduct that class.

First two rules state as the examples of scheduled activities and the last one is the example of deduced activities achieved using inferred procedures.

As we have discussed in Chapter 7 that these rules are specific to our proposed system, however the knowledge base filled in with the help of proposed mapping methodology are generic for other systems to reuse/extend. To rely on the results achieved we have added degree of confidence on the system also discussed in the Evaluation chapter.

3.5 Information Retrieval of the Knowledge Base

To make a query for the information retrieval out of the knowledge base, an example to use the knowledge base would be to generate some queries to fetch the data as per requirement. Considering the scenario where a *Faculty Y* is stuck in the traffic conditions and searching for *Faculty X* for the required task a typical question might be "*What a Faculty X is doing at the particular time?*". However, the security is not in our scope but the problem of the authorization may be solved here because they (*Faculty X* and *Y*) are working in the same department and hence can be agreed to share some information.

One of the three main technologies of the Semantic Web are SPARQL [20] which is an acronym, for SPARQL Protocol and RDF Query Language. Through its design query part is used to query RDF data, the same way as SQL for relational data and XQuery for XML data. And protocol part transmits queries and output in between the client and a query engine via HTTP. A typical query is an RDF graph with variables.

An example of SPARQL query to ask about a faculty member's activity at a particular time is presented below:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ctx: <http://www.cs.le.ac.uk/ontology/contextmodel#>
     SELECT ?Faculty ?Activity
     WHERE
     {
     #all scheduled activities falls within the time span
     ?Faculty ctx:hasActivity ?Activity.
     ?Facuty ctx:hasProfile ?Profile
     ?Profile ctx:hasLastName ?ln
     ?Profile ctx:hasFirstName ?fn
     ?Activity ctx:hasTime ?Time.
     ?Time ctx:hasStartTime ?st.
     ?Time ctx:hasEndTime ?et.
           FILTER ((?ln>= "FacultyXLastName" &&
                   ?fn<= "FacultyXFirstName"))
           FILTER ((?st>= "2011-12-12 09:00:00" &&
                   ?et<= "2011-12-12 10:00:00"))}
```

The query written above asks about the *Faculty X*'s activity according to the time in the system developed.

3.6 Summary

This chapter introduced the context model that will be used to store and retrieve context information obtained from exchanges that occur between a user and the Services. It has also been discussed that how to retrieve and reason about the stored data.

In the next chapter, we discuss about how context can be acquired placing software sensors in the middle of the user and the Services along with the overall architecture of the proposed system.

Chapter 4 Sensors and Context Acquisition

This chapter presents the Activity Context Architecture in the context-aware environment using software sensors for the proposed system based on a layered framework. The architecture is the representation of the system that helps to understand how the system will behave; the structural solution meets all the technical and operational requirements built on the solid foundation. While there is a strong need to develop generic context models, it is also paramount to understand the context in which they are intended to be used. This becomes even more crucial when considering new input sources and different levels of abstraction of the context data. In general we should have an architecture that makes the development of context-aware systems easier. The proposed architecture supports user's flexibility and familiarity focused on the user experience and interaction already in use. It backs up market maturity by taking advantage of the existing platform along with the flexible design for reuse and extension. The combination of service oriented architecture with semantic web technologies may help build better context-aware systems.

We propose a context-aware service-based architecture, which presents the developer with a context platform that can be used inside other applications which are related with software sensors. The implementation of such systems usually consists of a client (a user or service invoking an interaction) and a service (providing a response). Exchanges from both ends are being monitored. However, WSDL [88] is used for providing service API information but as we are sensing activity therefore the data related to API would be not of much help in the system required. Although, WSDL shows how the service can be called, returning data structures and what parameters it expects, but despite of all that high relevancy regarding the web services, we seek more about the data that is reflected in the communication messages (i.e. SOAP messages) to know the user's activity.

Semantic Annotations for WSDL [36] enables semantic annotations for Web services not only for discovering Web services but also for invoking them. The emphasis of our proposed system exploits the activity of the user which is stated by the data provided in the SOAP messages not to describe the abstract functionalities of a service along with how and where to invoke it.

As the proposed infrastructure deals with the activity of the user; all the available data is considered helpful to gain insight into performed activities. Furthermore, our mapping methodology (explained in Chapter 5) is significant in providing meaning to the extracted data. The architecture is depicted in Figure 5 and we will discuss the components in more detail next.

Parts of this chapter are already published in [46], [48] and [49].

In this architecture context information is provided by various context sources, which include web Services. The data from the sources is either provided directly (the usual approach for hardware sensors) or through observation of message exchanges (the proposed approach for software sensors). This data forms input to reasoning which allows to determine a user's activity.

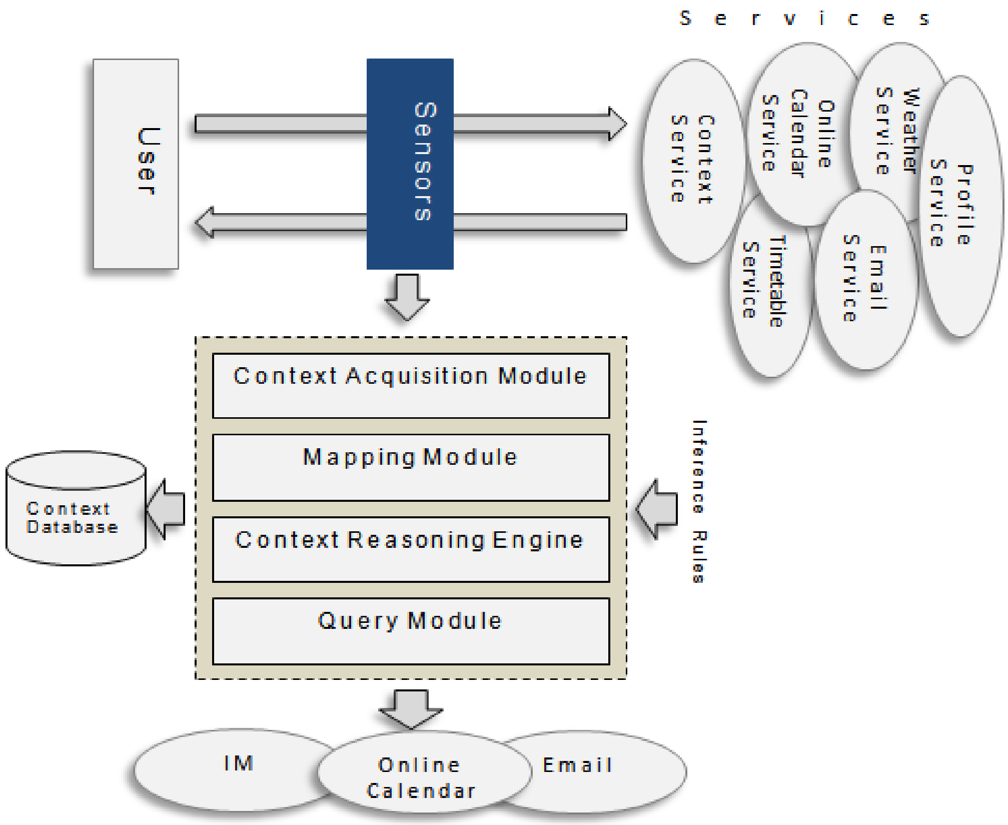


Figure 5: Activity Context Architecture in the Context-aware Environments using Software Sensors

We have divided this architecture in three parts which further explains each module of the architecture depicted in the Figure 5.

4.1 Sensing Context Information

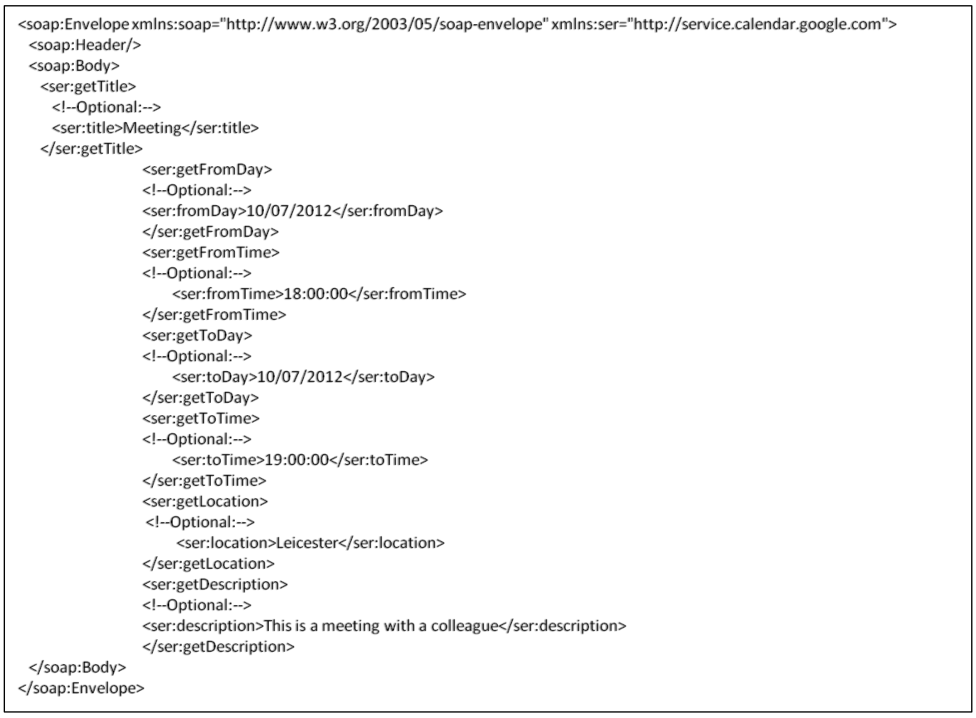
During the exchanges that occur between user and services data moves to and from the services in a SOAP envelope and remains unused for further processing. To make use of that data and transform that into a meaningful form we present the service-based activity context architecture in the context-aware systems scenario specifically for software sensors in Figure 5. which combines traditional web services with semantic web technologies to not only make use of that data but also out of that data we can actually sense the activity of the user without a physical sensor attaching to it, which is a trend so far.

The user's context information is provided by the exchanges occurred between a user and the Services, whenever a user is using any kind of service, she is actually providing data in the form so that a service can respond. This data is processed to achieve fruitful results. Here Sensors are capturing those exchanges and context acquisition module is taking that data to the system as in Figure 7. This raw data is further processed by mapping module which is the key to this research and has been described in the next sections. By putting that data to the context model is the beauty of this architecture which combines web services with the semantic web technologies to sense the activity of the user, as we know that raw data makes no sense but when that data is mapped into the structured knowledge with all the reasoning rules makes sense and hence we can query and reason the knowledge base by applying inference rules the already available data along with the data which is coming from sensors in the form of user's context, logically.

4.1.1 Context Providers

In this work Context Providers are all the services which provide the user's information (while sending their data towards the web server to complete their task on services) such as profile, calendar, timetable etc.

A Profile Service inputs the profile of the user and provides the entity information which can help to collect and aggregate information according to



```xml
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:ser="http://service.calendar.google.com">
  <soap:Header/>
  <soap:Body>
    <ser:getTitle>
      <!--Optional:-->
      <ser:title>Meeting</ser:title>
    </ser:getTitle>
                    <ser:getFromDay>
                    <!--Optional:-->
                    <ser:fromDay>10/07/2012</ser:fromDay>
                    </ser:getFromDay>
                    <ser:getFromTime>
                    <!--Optional:-->
                         <ser:fromTime>18:00:00</ser:fromTime>
                    </ser:getFromTime>
                    <ser:getToDay>
                    <!--Optional:-->
                         <ser:toDay>10/07/2012</ser:toDay>
                    </ser:getToDay>
                    <ser:getToTime>
                    <!--Optional:-->
                         <ser:toTime>19:00:00</ser:toTime>
                    </ser:getToTime>
                    <ser:getLocation>
                     <!--Optional:-->
                          <ser:location>Leicester</ser:location>
                    </ser:getLocation>
                    <ser:getDescription>
                    <!--Optional:-->
                    <ser:description>This is a meeting with a colleague</ser:description>
                    </ser:getDescription>
  </soap:Body>
</soap:Envelope>
```

Figure 6: An Example of SOAP Message Request (RAW) of a Calendar Service

user's own record. This can be the name, affiliations, education, etc. of the user wrapped in an SOAP envelope (request) written in the XML format and ready to be travelled to the web server (Figure 6). Calendar Service shows the scheduled activity of the user and is further explained in the coming sections. Email Service can sometimes be used to infer scheduled or deduced activities of the sender with the help of looking at sender, receiver and subject of the message only. A Timetable Service shows the scheduled activity of the user from the organization's side where s/he works. Weather Service can strengthen the deduced activity should the reasoning rules are applied on the collected context. The Context Service provides the actual context defined by users. If there is any conflict with users' provided context then heuristics can be used to resolve it.

4.1.2 Sensors

In this work a sensor means software sensor which is used to extract only context information from different services and provide this to the Context Acquisition Module. The sensed data is actually the interaction of a user with the Services. We have already presented background information regarding sensors in Chapter 2 that are placed in between the user and the services during interaction. These sensors play an important role in extracting the data from the exchanges occurring between a user and the services. While using the services we actually send user data to a service to be responded to and this data is carried through the SOAP protocol. SOAP (Simple Object Access Protocol) is an XML-based communication protocol to make applications exchange structured information to the web services over HTTP. SOAP allows getting around the firewalls and is a W3C recommendation.

While running services the SOAP exchange information is forwarded to the sensors. The SOAP monitor module of the services is not enabled by default for security reasons because it exploits the data regarding that service and the user. There are two kinds of handlers to set. One is in client side and the other one in the server side. Handlers allow you to intercept SOAP messages and can reside on both the client and the server side during a service invocation. The illustration shows the message flows involving the SOAP monitor:

$$Client \rightarrow SOAP\ Envelope \rightarrow TCP\ Monitor$$

$$TCP\ Monitor \rightarrow SOAP\ Envelope \rightarrow Server$$

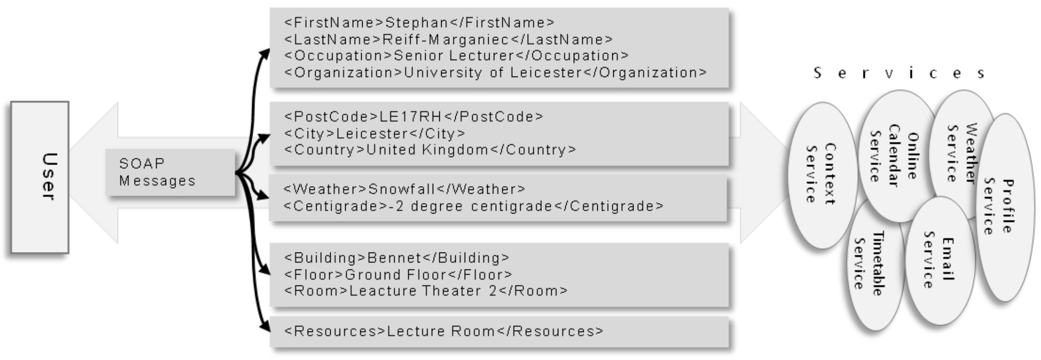

User
SOAP Messages
<FirstName>Stephan</FirstName>
<LastName>Reiff-Marganiec</LastName>
<Occupation>Senior Lecturer</Occupation>
<Organization>University of Leicester</Organization>
<PostCode>LE17RH</PostCode>
<City>Leicester</City>
<Country>United Kingdom</Country>
<Weather>Snowfall</Weather>
<Centigrade>-2 degree centigrade</Centigrade>
<Building>Bennet</Building>
<Floor>Ground Floor</Floor>
<Room>Leacture Theater 2</Room>
<Resources>Lecture Room</Resources>
Services
Context Service
Online Calendar Service
Weather Service
Profile Service
Timetable Service
Email Service

Figure 7: An Example of Formatted SOAP Message Request of A Calendar Service Showing An Example of the Data that can be Obtained Through Exchanges

We can have a handler to capture a SOAP request/response message exactly before it goes from or returns to the client. The SOAP messages are redirected to or from the TCP Monitor. The client connects to the TCP Monitor and TCP Monitor is configured to connect to the server.

The same applies when the SOAP Envelope is sent to the Client. With reference to Figure 7, we show the interaction between a user and services highlighting the data that has been extracted from the soap request and response messages. In Figure 7 we can see that first block of messages represents the data (derived to the activity) which has been extracted from using Profile Service, third one denotes to Weather Service and the second one and the final one depicts the data sensed while interacting with Calendar Service.

4.2 Linking Context and Services

There is still research going on to sense the activity of the user. Some of the systems only tell the online status (as an Activity) which is either updated by the user or if forgotten then the only understood context no matter between that period of time the user has done number of other activities. Some of the Advertising services (e.g. Gmail [26] (one of the applications of Google)) look at the description and subject of the email and advertise; no matter if that advertisement is necessary for the users at their situation or not and hence failed to respond according to context.

In traditional web services the SOAP messages carry data which is unprocessed for further use but carry a useful information regarding a user's interaction with that service and if processed can be useful in further ways. For example if a user is using a Calendar Service and she is putting data into the required form, for example she wants to add an event which requires details like: Title of the Event, Time (From to To), Location, Description etc. While inputting these details user is actually passing her crucial data through SOAP messages and this raw data in SOAP messages carry information of the User's Activity but in a form which is not further processable by machine and hence become a source to update the Calendar Service only. This part of the architecture (discussed in the next sections) explains how we make use of that data.

4.2.1 Context Acquisition Module

The Context Acquisition Module acquires user's context from different services with the help of sensors (discussed in previous section). These sensors capture the raw SOAP messages that are being used to invoke the web services.

For example while creating a new entry in the Calendar Service; a user inputs the data that travels from the client to the web service using the SOAP protocol. These SOAP messages carry the arguments a user passes to create

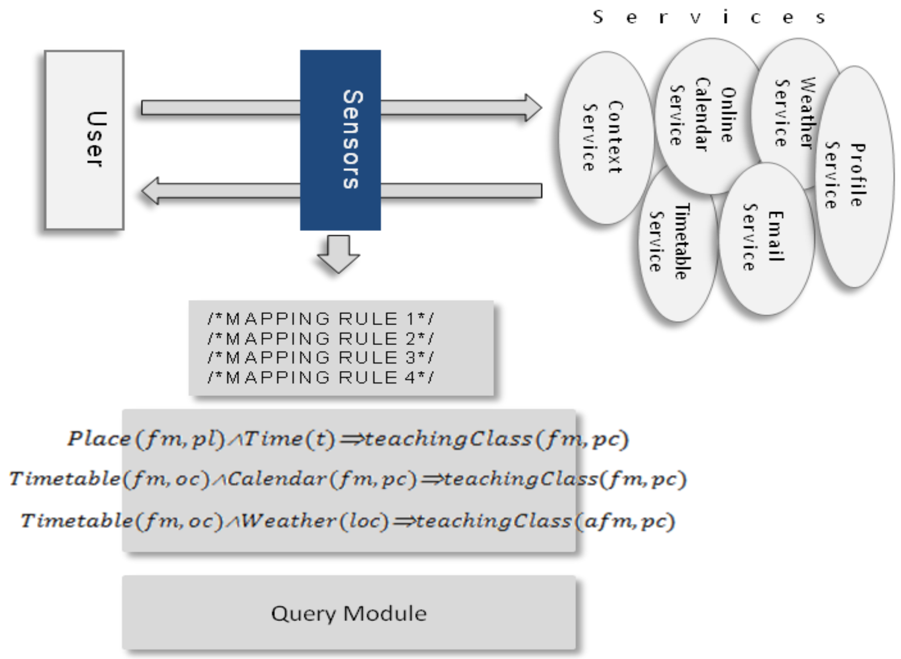


Figure 8: An Example of The Architecture Applying Mapping Rules and Semantic Rules

an event in the calendar, such as Title, FromDay, FromTime, ToDay, ToTime, Location and Description in the request.

In the system we place sensors to sense this data. In this particular example the request is more important than the response because a response might only be the acknowledgement (SOAP response message) stating that the entry is added into the calendar.

Taking another example of Weather Service, the request and response both are important to process because the request reveals the city of which the weather is asked and the response shows the Fahrenheit/Celsius, hence affects the activity of the user if the weather is pleasant, normal or bad.

4.2.2 Mapping Module

The Mapping Module maps the data captured from context providers into a semantic representation so that context can be further processed, shared and reused by other components. This approach addresses both the interoperability, extendibility and the reuse issues. As this approach is based on semantic technologies so that users can adapt the services according to their own needs. The mapping methodology is discussed in detail in Chapter No. 5.

4.2.3 Context Knowledge Base

The Context Knowledge Base stores this data into a triple, making a statement i.e. subject, predicate and object and instance statements specific to the individuals. It can provide various services with the help of the Query Module for querying, notifying, adding, deleting or modifying context knowledge stored in the context database.

4.3 Retrieving Context Information

To sense the activity of the user using hardware sensors there might be the need of proper image processing etc. but for software sensors there is a need of structured knowledge stored in a triple form so that automatic reasoning can be performed. An example is shown in Figure 8. From different exchanges between user and services provide us with SOAP messages and therefore the raw data. The challenge is to make it structured and transform it into a knowledge which is required for the inference mechanism. Reasoning cannot be done logically, if data is unstructured, hence the utmost need arises to structure the data and apply the reasoning onto it. Once knowledge is there then the need is to infer some more facts by applying rules onto the existing facts and this is the important feature. For example if you have SOAP messages containing the event title, time and location mentions the complete context, if the rules are applied properly. And if further added weather conditions of that particular location can deduce with the reasoning rules. Office calendar and work calendar may further strengthen the rules and hence sense the activity context with the combination of web services and semantic web technologies.

To instantiate an ontology that is new or already built; we are not only considering the method that emphasizes to create the new one that is proposed in the available literature [27] and [67], and not only inserting it without any logical way like [79] and [53] but we are also instantiating the existing knowledge and providing methodology which can further extend the ontology maintaining the structure with the help of lexical database.

4.3.1 Context Reasoning Engine

Once the data regarding user's context is instantiated in the web ontology language with the help of mapping methodology then the Context Reasoning Engine infers the scheduled and deduced context of the user by avoiding context conflicts into the knowledge base with the help of inference procedures. The semantic rules (some of them described in section 3.4) help to infer the data using Semantic Web Rule Language (SWRL) [32].

4.3.2 Context-aware Services

After obtaining the user’s context the Context-aware Services can be utilized so that these can adapt according to user’s situation in a context-aware environment.

Based on this architecture, we have implemented an activity-aware prototype (chapter 7) that senses the activity of the user in the context-aware environment using software sensors.

The query module aids the system providing the activity of the user with the available data, it takes input in the form of name and time of the person whose context you want to ask about and returns the activity of that person by following the infrastructure.

Some of the modules are specific to the system (proposed), however some of the modules like Sensors (which senses the interactions of the user in the form of request and response messages), Mapping Module (breaks the messages into the data that is helpful to forward to the knowledge base according to the tags they associate) and Knowledge base (that is filled with the data which leads to the activity of the user) are general in nature and can be used in other systems.

4.4 Summary

The building blocks of Activity Context Architecture in the context-aware environment using software sensors for the proposed system are presented. To elaborate the architecture it has been further divided into three parts; from where the context information is sensed, how this sensed data is linked and mapped with the semantic web technologies and how this data is retrieved applying the reasoning rules to infer the activity. The next chapter clarifies how the mapping methodology is done after acquiring data from sensors.

Chapter 5 Mapping

We use web services for our day to day tasks on the systems related to the situation. Web services require users' data relevant to run Services and users require their work happening. The SOAP protocol helps to make this exchange happen to or from the Services. SOAP request message to the service and response message to the user/service contain data which is valuable if interpreted right. The same data if processed cleverly makes a sense with the semantic web technologies which is the basis to make that data structured, a statement, inferable and queryable with extensibility and reusability and widely accepted features on top. To lift that XML written data to the web ontology language (OWL), we need to understand the tags and relevant values, which is a challenge. This chapter along with the model and architecture considers the core contributions of the research: the mapping methodology of XML data extracted from SOAP messages to the existing structure of the knowledge is explained, so to achieve the goal to sense activity context using software sensors in the context-aware environments. This approach states that these SOAP messages in general can also be processed to sense the activity because once we understand the nature of the tags/values and the relevancy of that tags with the structured model in the knowledge base then the same can be applied to the no. of different scenarios. We have tried to make it more general by mapping these tags and their values to a broader extent by looking at their tags itself along with the data dictionary so that terms might define its own self rather than a model;

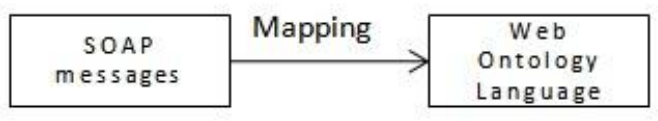


Figure 9: Mapping from SOAP messages to OWL

however inference rules are specific to the context model. This is further explained in the methodology section.

Parts of this chapter are already published in [47].

5.1 Available Approaches and Gaps

The related work in this field suggests only mapping from XML to OWL or from XML to RDF [27] and [67], which converts tags into the OWL looking at the existing structure unconditionally [79]. These works mainly suggest making an OWL file from an XML file or schema [53] without taking care of the existing structure (see Figure 9 and Figure 10). Some of the similar approaches from available work are presented in the form of elements, values and their relationships between the different systems of encoding are presented in Table 2.

If we look at the existing solutions then we will find out that those maintain the class hierarchy with the tree structure of the XML file without maintaining the actual structure of the knowledge. When we think about the knowledge for the activity sensing using software sensors these become preliminary and therefore are beneficial at the starting level of mapping. Because we are dealing here with the data to sense the activity so the only requirement is not the OWL file generated but the instances to their actual places and for that we need lexical database (i.e. WordNet) so that we can check and acquire the results accordingly. The real challenge starts when we are thinking about more generic forms of mapping, for example what we do with an instance which does not exist in the structure or an instance which needs to be inserted into the existing structure while maintaining the taxonomy as well. In the following sections we will discuss the methodology we propose to solve this. This approach helps find the tag and their value befitted in the structure of the model along the line for what that model is meant to be. For example when the first method becomes false the next method finds synonym and failure to which a tree like structure compiles hence supersedes the available approaches with an aid.

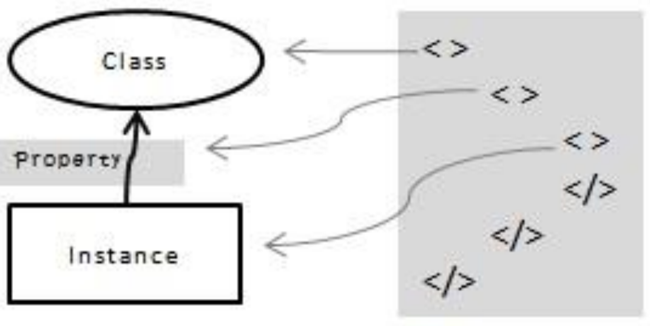


Figure 10: Representations of the Related Work for the Mapping from XML to OWL

XML	OWL
Represented by Trees with Labelled Nodes	Represented by triples (i.e. Subject-Predicate-Object)
Tree Structure	Concept Hierarchy
Schema	Model
Instances	Instances
<tag>	Resource/Literal (depends if the tag has a value)
Nested Tags	Resource with Object Properties (If no value or otherwise)

Table 2: Available Approaches to Convert XML to OWL

5.2 Tag to Instance Challenges & Solutions

Software Sensors in the context-aware systems environment require more intelligent kind of mapping when converting from SOAP (XML) messages to the semantic knowledge in OWL ontologies.

The tag to instance match is the most closest match to existing work. *(1)* It considers the datatype property attached to it, without considering the proper place of it in the structure, *(2)* what if the instance which has same like meaning exists in the structure or *(3)* the term which does not exists in the knowledge structure but its predecessor does in the hierarchy. An overview of all the steps of the mapping methodology to solve these problems is presented below:

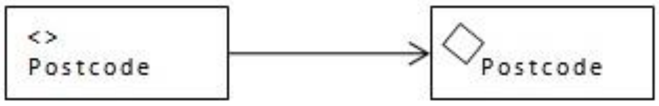


Figure 11: Mapping from Tag to Instance

1) Match the tag with the instance, if found then put the value of the tag as a literal.

2) Find the *synonym* for the tag into the lexical database (i.e. WordNet) and if any of the fetched synonyms is found in the structure then instantiate it.

3) Find the *tag* into the lexical database (i.e. WordNet) then make that term a hierarchy (e.g. *hypernym*) out of it and then match with every hierarchical term found, with the other terms in the knowledge structure, if any of the term is matched at any level of the hierarchy then follow the sequence or else make the tree out of the terms matched into the existing structure and traverse the instance the same way as *step 1* does.

5.3 Existing Structure of the Knowledge.

There is a dire need of a methodology which can help with the existing knowledge creating because available ontologies have consumed time and research intensively. Therefore, to maintain the existing knowledge structures and benefit from the effort in creating them, new knowledge or data needs to be inserted in a smart way. The appropriate solution is one that inserts new

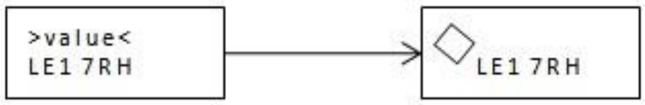


Figure 12: XML value to literal in OWL

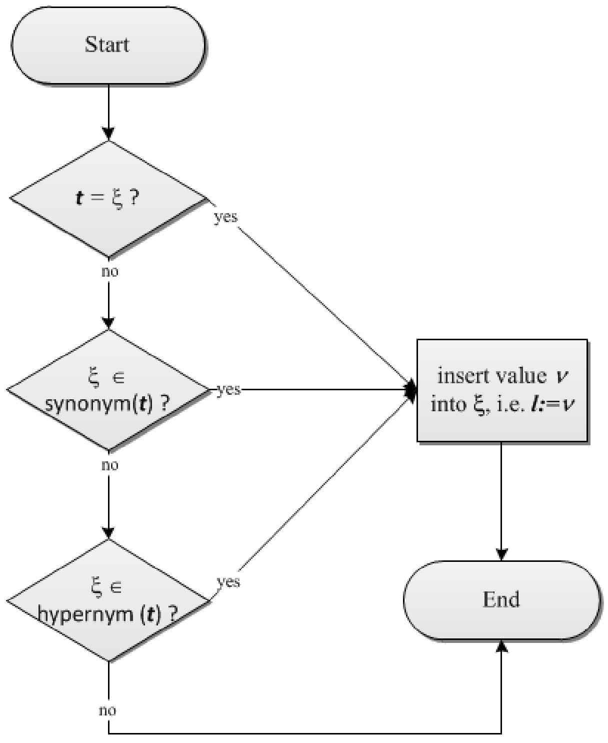


Figure 13: Mapping Methodology

instance data into the existing ontology structure without changing or recreating the ontology itself -- and the method needs to this automatically.

The next sections will introduce details on how data can be inserted into ontologies. It will present the details of how to achieve this for the different kinds of steps we identified in the methodology earlier.

5.4 Methodology

All the steps of Figure 13 including equations and algorithms are explained in this section.

5.4.1 Direct Match

The easiest kind of match is one where the tag matches the instance of the ontology see Figure 11 and Figure 12, and in this case we wish to insert the value from the tag directly as a literal associated with the instance.

Definition: Direct Match
Let
I be an instance of an ontology O,
l be a literal of instance I,
t be a tag in an XML schema Σ , and
v be a value for t.
then

$$\boldsymbol{l = v} \textbf{ iff } \mathbf{I} = \boldsymbol{t}$$

For example from Figure 6 the tag *<location>Leicester</location>*, where *tag=location* and *tag(value)=Leicester* can be instantiated in the knowledge base using our algorithm as a triple: *location hasValue Leicester* provided the tag matches.

5.4.1.1 Algorithm 1 (Direct Match)

Looking at the extract of the SOAP message we can clearly see that this is the combination of tags and values. To turn these values as valuables the first algorithm suggests to look at the concept of the first tag say for example *Location* which carries a value *Leicester*. The technique is applied afterwards if the match is found; as we know that this XML data could be transformed to a triple (Subject-Predicate-Object) by certain rules and that might be by adding a property which relates with both subject and object. This property is apparently not present in the XML data so the technique is to check the concept corresponding to the Instance it directs to and if found then create a datatype/object property if there is value next to it or otherwise respectively along with a prefix *has* with the *tagName* obtained. Then put the text node into the datatype property created to make an instance.

```
Algorithm 1
SOAP message is the xml document consists of element nodes and text nodes
let tagName = Element node presented in the SOAP Message
let tagNameValue = Text node of the corresponding tagName
let owlClass be any class of web ontology language
let owlSubClass be any subclass of an owlClass
let owlDataTypeProperty be the data type property of web ontology language
let owlObjectProperty be the object property of web ontology language
let owlInstance be an instance of web ontology language
let prefix = "has"
let childNodetagName be a child element node of tagName

if owlClass of ontology = tagName
 if tagName has childNode
   Add prefix+childNode as owlObjectProperty
   Set owlDomain = tagName
   Set owlRange = childNodetagName
 else
   if tagName has tagNameValue
   Add prefixHas as owlDataTypeProperty
   Set owlDomain tagName
   Set owlRange data type as String
   Set literal = tagNameValue
else
Goto Algorithm 2
```

This is a technique to convert the data into a triple form; next section adds the second match which deals with the concept if the term is not found using the first algorithm

5.4.2 Synonym Match

Slightly more complex is the situation where the tag and the ontology instance use terms that are synonyms. For example the SOAP message might use the term *location*, whereas the ontology uses the *placement* as presented in the Figure 14.

Definition: Synonym Match

Let

$\mathtt{I}$ be an instance of an ontology O,

l be a literal of instance $\mathtt{I}$,

t be a tag in an XML schema Σ, and

v be a value for t.

synonym(t) be a set of synonyms for t as obtained from a lexical database.

then

$$l = v \textbf{ iff } \mathtt{I} \in synonym(t).$$

According to the conceptual-semantic and lexical relations presented in the lexical database, placement is the synonym of location hence can be used alternatively in the taxonomy.

5.4.2.1 Algorithm 2 (Synonym Match)

This leads further to the concept with the lexical database (WordNet) which returns all the synonyms of the word (tagName) that are traditionally left or added without maintaining a structure. There is as such no restriction to

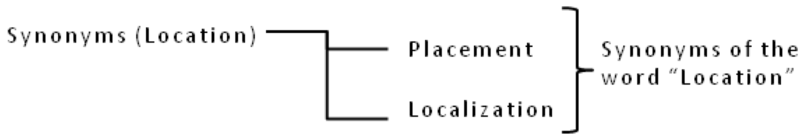


Figure 14: Synonym Match

choose the words to design an ontology as far as they reflect the domain for which they are being used. Taxonomy designer might choose a word *placement* instead of *location* (synonymous in nature according to WordNet) considering the synonym of the word which might lead to the redundancy if we create another word meant for same thing.. Here we have taken these things on board in *Algorithm 2*.

```
Algorithm 2
let Synonyms(tagName)_n be the set of all the synonyms of a word tagName
found on the WordNet lexical database

if Synonym(tagName) != null
  for Synonym(tagName)_n do
  Algorithm 1
   If Synonym(tagName)_n = true
   Set literal = tagNameValue
   break
else
Goto Algorithm 3
```

In the output of all the synonyms of the word location is presented in Figure 14. Now the *Algorithm 2* considers all available values i.e. *Placement*, *Localization*, etc. and checks each value against the concept mentioned in the model and if any of the word is found in the structure then the same methodology would be adapted as was in *Algorithm 1*, which we have discussed before.

5.4.3 Hypernym Match

Even more complex is the situation where no instance matches the tag directly or by being synonymous. This situation would call for a new instance to be created in the ontology identifying the proper place to it.

The previous approaches either merge by considering the super-sub-sub tags or create a new one. As we have stated previously that we have created our own structure (the context model) by identifying and to fulfil the need to context-aware systems making use of software sensors, we instantiate that model. We have also tried to make the method more generic considering the elements of SOAP body written in XML to an already available OWL file so that anyone can use it in their respective system.

Definition: Hypernym Match

Let

I be an instance of an ontology O,

l be a literal of instance I,

t be a tag in an XML schema Σ , and

v be a value for t.

hypernym(t) be a set of hypernym for *t* (that is the branch of the tree in which *t* is found) as obtained from a lexical database.

then

$$l = v \textbf{ iff } \mathrm{I} \in hypernym(t).$$

The proposed method uses the lexical database (e.g. WordNet) and its hierarchy of the words and then using a bottom up approach we identify the upper classes for the literals. In more detail, a given tag is checked against the Lexicon and the matching term in the hierarchy of that word is extracted. The hierarchy (super-super-class) of the term *location* is presented in Figure 15 in the reverse order. This shows ancestor classes of the term. The methodology suggests that check every corresponding node from top to bottom against the knowledge structure. For example if at any level of node that word is found then reverse the order to create respective class and instantiate the data according to the *first match*. To think about not to overload only top nodes are considered for instance if at level two (*object*) of node from top down the term matches then we will only create node 1

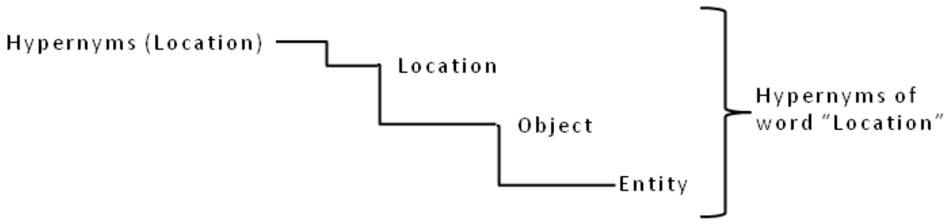


Figure 15: Hypernym Match

(*location*) as subclass of node 2 (*object*) rather going to node 3 (*entity*) and so on to avoid the redundancy.

5.4.3.1 Algorithm 3 (Hypernym Match)

This *Location* is a point or extent in the space with the *Object* as super-class and the *Entity* as super-super-class. Write all the respective values to tagNameHypernym(n, n-1, n-2,…) then check against all the classes and if found then create sub-class/super-class according to the structure and run the *Algorithm 1* for further processing.

The resultant Hypernyms of the word *Location* are presented in Figure 15.

Please note here that WordNet also offers the hyponyms of the word Location but the difference between hyponyms and Hypernyms is the reverse in hierarchy therefore we have avoided here.

```
Algorithm 3
let Hypernyms(tagName)$_n$ be the set of all the hypernyms of a word tagName
found on the WordNet lexical database

if Hypernym(tagName) != null
  for Hypernym(tagName)$_n$ do
  Algorithm 1
   if Hypernym(tagName)$_n$ = true
    do while Hypernym(tagName)$_n$!= null
    Set Hypernym(tagName)$_{n-1}$ = owlClass
    Set Hypernym(tagName)$_n$ = owlSubClass
    Set literal = tagNameValue
   break
else
Goto Algorithm 4
```

Hypernyms carries the super-class and super-super-class relationship of the word however the Hyponyms returns the sub-class and sub-sub-class relationship of the word it asked for and we are more interested about the structure of the knowledge not the knowledge overload. This is why we are fetching the Hypernyms rather than Hyponyms to avoid the information overload.

5.4.4 No Match

Finally, a situation might arrive where none of the previous three approaches allow us to identify a match. While it seems desirable to match as much information as possible, there might be data which simply does not add to the context and hence it is safe to ignore that. Furthermore, even if a data item could be adding to the context, one should keep in mind that any context data gathered is often not 100% accurate and could be contradictory to information simultaneously gathered, changes quickly and hence has a limited period of validity and is possibly made more or less reliable through reasoning anyhow. In the light of these factors missing some items can be seen as a negligible problem.

In these matches we have defined the essential properties of the algorithms by 1) specifying inputs in terms of SOAP messages, 2) specifying output with the intended results or even sometimes no output is also expected, 3) Definiteness by detailing the sequence of events and details of each step, 4) Effectiveness of the doable operations in the said matches and 5) Finiteness where the algorithms actually stops where no match occurs.

5.5 Degree of Confidence

Having gathered data from different sources and being able to derive new conclusions on them we should turn our focus to the issue of how certain we can be about the information derived. We term this concept degree of confidence as it should show how certain an enquiring user can be that the information she receives truly reflects the activity of the entity enquired about. This could for example be quite crucial if billing of consultants time or travel arrangements to meet someone will be based on such activity sensing.

Returning to our example, a faculty member might have private and work calendars. If they are inserting events in their work calendar which reflect working time activities, these are probably accurate for day time. However, social activities would usually take place outside working hours and hence the private calendar might be more accurate for such activities. In large organizations there might be further departmental and institutional calendars to be considered. The degree of confidence for the reliability of the data still needs to be tackled in such a way that another user (who is inquiring about actual user's activity) is confident on certain activities at certain times.

For example if a faculty is updating his work calendar for an appointment for work related issues i.e. from 08:00 to 5:30 pm then this directly affects the Work calendar but instead if he is updating the work calendar for after that time states is assumed as the user would have a meeting with the project with whom he is partnering and hence goes to less confidence with that calendar update.

For personal calendar, if a user is maintaining calendar entries for Friday afternoon to Sunday night might lead to the high in confidence than that with a work calendar. We can fall these ranges from range 0 to 1 with the 0 as least confidence and 1 as highest confidence. At this stage we propose a simple ranking of data sources, which is dynamic over time in that different preferences are given at different times.

Following our example, assume the user to have an Organization Calendar (*OC*), a Work Calendar (*WC*) and *a* Personal Calendar (*PC*) we get the following ranking:

```
If (time is 8:00am to 5:30pm) and
(day is Mon to Fri)
then
OC > WC > PC;
Else
PC > WC > OC
```

This states that during the normal working hours the organization and work calendars are more reliable than the private calendar, while out of working hours the calendars that are under the user's control gain in relevance.

The other way to strengthen the confidence on the activities is during mapping from XML to OWL, where the very first rule if proved true (i.e. tag matches with the presented instance in the existing ontology) has more confidence level than the one which is hierarchically driven by the lexical database. One could consider a more fine grained approach, and indeed this is one of our aspects of further work. For example we could strengthen the confidence on the activities is during mapping from XML to OWL, based on specific instances and also on how certain we are about the match that has been made when inserting data.

We have proposed the above explained techniques which map data of SOAP messages into the existing structure of knowledge in the web ontology language and further reasoning rules carry that data into a meaningful form once instances are created out of that data.

5.6 Summary

In this chapter we have identified that the availability of wide variety of data in SOAP messages need some methodology to map into the structured knowledge. It is further explained in the methodology that how we can achieve the desired results with the more specific or somehow generic form to it to fill-in the data model. The technique maps data of SOAP messages into an existing structure of web ontology language. Further reasoning rules carry that data into a meaningful form once instances are created out of that data. We have considered three cases in account, matching with existing instance, matching with same like instances, and inserting a new instance along with maintaining the hierarchy. To create an example for such systems we have populated the instances into the context model (designed for the proposed system) for context-aware environments.

Chapter 6
A Case Study for Supporting Context-aware Applications

We have presented an approach to sense activities through software sensors. We will now present a case study that will allow to evaluate the approach in more detail. The case study considers the setting of a University, with staff and students, calendar and timetable.

This chapter introduces the setting of the case study first; it then explores the goals and the specific scenarios that will be analysed.

Reference Document:

University of Leicester: Transparency Review Time Allocation Guidelines Definitions of Activity 2011/2012 is taken as an example to identify the no. of activities performed by a Faculty in the University environment.

Following section describes the use of services to communicate activities between staff and students in the University environment, where the proposed system might be helpful.

The Faculty and the Students in the University

In the University all the students and staff use services to inform any activities being held in the campus to one another.

To explain further, they use various web services to accomplish their day to day tasks relevant with the situations. For example Calendar Service (to maintain their scheduled activities), Email Service (to send and receive notification/request regarding any activity), Profile Service (to update the work portfolio), Weather Service (to check for the weather), and Timetable Service (to write down the timetable to perform no. of work related activities like classes, seminars, meetings etc.).

Even though, If we propose to add the Context Service (to let the users show what is their activity?) in the system to update the users' current activity context. It would be used in the favour of the users to avoid any disturbance at work or leisure level by announcing their activities. It is assumed that the disturbance level can be minimized if the users regularly update their context within an environment; this service might show the user's current activity like teaching, meeting, etc. These might be updated with a regular time interval or at the change of the activity. However, this tends to be a tedious task and uninspiring in a way that it requires a user's attention and interrupts the efficiency of the work being undertaken. While a user needs this attention to the task s/he is performing to. It is also to note that in such situations, we cannot rely completely on the services, which require regular update from the users.

Context-aware computing has a potential to solve a wide variety of day to day problems. This solution represents its benefit in a way that it does not disturb the actual service-oriented architecture, rather utilizes the services scenario and enhances it through utilization process. The system carries on the task of SOA in helping businesses respond more quickly and more cost-effectively to the market conditions that are changing. Though the case study discusses the university settings but also promotes at the service level rather classes level with the simplification of interconnection and usage of legacy system.

The system helps not only in University conditions in general but more specific to the user needs on the system, by improving the current features of the context-aware computing by intercepting and mapping the messages which are otherwise of no further use yet. It takes away menial tasks by taking care of the tags and their meaning to form a proper structure of the knowledge base, however the rules are not being generated automatically but even then it helps structure to be maintained for further use.

The solution's mapping part is quite general in nature and the inference rules are specific to the case study to examine the system for the basis of evaluation.

In the problem oriented approach to depict, analyze and identify the real life situation the major problems and to suggest solutions to the problems, a specific sequence of actions and interaction between user and system is presented in the next section.

6.1 Goal

Jim (a PhD student) wants to meet with his Supervisor Dr. Kim (also a co-author) regarding the final touches of the camera-ready version of a research article to be submitted within two days of time in one of the highest ranking journals.

6.1.1 Actors

- Mr. Jim: a PhD student.
- Dr. Kim: a Faculty Member (the Supervisor of Mr. Jim) of the University.

Traditionally, to achieve this goal the following steps are taken:

- Student types the email address of the Supervisor
- Student sends an email to the Supervisor asking him for the meeting
- Supervisor reads the email at some time
- Supervisor checks email and the calendar for the required time slot
- Supervisor fixes the meeting by replying the student

The above steps state the ideal situation according to the existing systems when everything runs in the favour of inviter and invitee of the event. However, the real world scenarios might be different than this and are described in the next section.

6.2 Scenarios

Consider the following scenarios:

6.2.1 Scenario 1: A Teacher Stuck in the Bad Conditions of Weather

A faculty member (A Teacher) is struggling to reach the campus before the start time of his lecture because of heavy snow fall in the previous night.

It is 8am and reported the heavy snow fall in the town. Dr. Kim is in his car driving to reach the campus of the University which is 7 miles away his residence. The traffic is moving car to car. While driving Dr. Kim is thinking about to conduct his class which is about to be started at 9am. He has Smartphone with him that provides all the university provided services including Calendar, Email, Weather, Timetable etc. From the traffic it is obvious that he would not be able to reach the campus before 9:30am. He pulls over the car and checks if any other faculty member is available by looking at the Weather and Timetable Service. He realizes that Dr. Sim (a faculty) whose lecture is after him (i.e.at 10am) with the same class would be very suitable as a substitution but he was uncertain about the road conditions of the other corner where Dr. Sim lives. Dr. Kim emailed Dr. Sim to act as a substitution for him stating that he will do the same for the 10am class of Dr. Sim.

Dr. Sim who is driving his car to reach the campus at 8am has forgotten his Smartphone at home.

6.2.2 Scenario 2: A Teacher in the Classroom

A faculty member (A Teacher) teaching a postgraduate class and trying to concentrate on a very core topic of the subject and trying to make it simple so every student in the class can easily understand it.

In this scenario Dr. Kim (a teacher) is teaching to the class of 70 students in lecture theatre. The topic is very new to most of the students and can be understood with the open discussions while explaining different situations. Dr. Kim is using the networked computer, the projector, white-board and any other visual aids to convey his message; on the other side students can have their notebooks or networked tablets with them for note taking. Dr. Kim can be approached by the outside entity using telephone (attached in the classroom), mobile (personal) or Email Service but usually mobile phones are kept off during the classes to avoid any disturbance.

While giving a lecture Dr. Kim saw a notification of a new email on the system and avoided to open it in front of the seventy students. After finishing the lecture (that lasted for almost two hours), Dr. Kim saw that Mr. Jim wants to meet with him regarding a final version of the research paper. Dr. Kim asked him to meet with him at around 1pm the same day by replying with email because at 4pm he has another meeting pertinent to the research project.

When Mr. Jim found no reply from Dr. Kim after sending him a request for the urgent meeting, he deduced that Dr. Kim as holding a lot of responsibilities must have any important work to do and he could not manage to see my request. Mr. Jim decided to attend one of the two hours PhD seminars to be started at 1:20pm.

6.2.3 Scenario 3: A Researcher in a Research Project Meeting

A faculty member (A Researcher) is attending a meeting regarding one of his current research projects. Students should not disturb the staff member, hence if they like to meet him they should be able to sense that the meeting is taking place and keeping the faculty member busy.

The clock shows 3pm, the meeting is held in the departmental conference room on the ongoing research work. Dr. Kim along with a whole team from his department is collaboratively working with other institutions on the research project. The attendees of the meeting have all the available resources with them e.g. connected Desktop and Laptop computer, Tablet, Smartphone, overhead projectors in the environment. All can use all services to carry on their tasks collaboratively. Dr. Kim is presenting in front of other members to devise the strategy for the on-going work and asking other members to follow the tasks accordingly. One of the attendees asks Dr. Kim to provide a tentative timeline for the tasks of next month to be undertaken. The meeting ends at 5pm following an open discussion and the next plans.

6.2.4 Scenario 4: A Supervisor Working Late in His Office

Dr. Kim is in his office working on one of his project work. He received a notification of the some new emails and decided to open it up after some time. The reason for this decision is that he is working on the system on one of his research projects with the office doors opened, and doesn’t want to be interrupted; but forgets to update his current context.

Dr. Kim at 6pm is preparing the timeline to be sent to other members for the tasks of research project.

In these kinds of real world situations one is not only concerned if the person is available, but if s/he can be disturbed. Obviously, one could ask people to update their context all the time to reflect their current status, but our work aims to remove that need by allowing to sense the context automatically.

The main success scenario is assumed after using the proposed system:

- Faculty logs in the system
- Faculty use context-based services to perform his day to day tasks for the organization
- Student logs in the system
- Student types the First Name and Last Name of the faculty
- Student types specific date and time, s/he needs to look at the faculty member's activity context
- Student receives the faculty's activity context as a response

6.3 Use Cases

Use cases help to find, record and describe functional requirements of the system needs and documentation of the specific functions.

6.3.1 Main Actors (Roles)

Actors might be a person, computer system or organization interacting with the system. The main interacting elements with the system here are the *Student* and the *Faculty Member* in the University. As we are interested to sense the activity of a faculty therefore we have to identify certain roles performed by a faculty.

According to the University of Leicester: Transparency Review (a document which is taken as an example to identify the activities) three main roles e.g. Teacher, Researcher and Administrator (Figure 16) are being performed by a Faculty. These roles point out the main activities performed by a faculty member. From these main roles derived activities (shown in Table 3) are like Teaching postgraduate students or undergraduate students, attending supervisory meeting or research project meeting, and sorting out masters related issues with students for two hours per day etc.

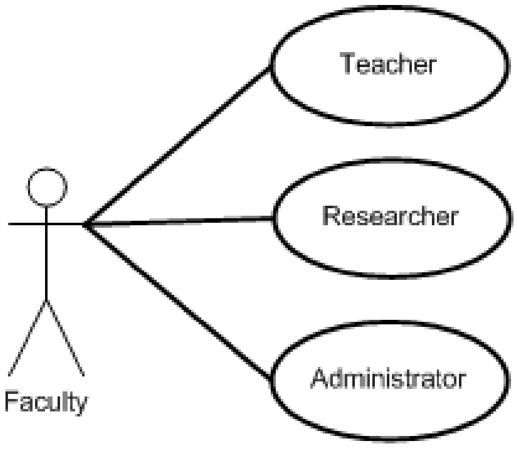


Figure 16: Roles of the Faculty Identified from the Transparency Review of University of Leicester

Some of the Activities of Dr. Kim (as a Faculty Member):
• Teaching Class, Lab, Surgeries at Postgraduate (PG) and Undergraduate (UG) Level for Self funding (SF) and Distance Learning (DL) Programs • Conducting Seminars, Tutorials, Workshops • Preparing material • Supervising the surgeries conducted with PG and UG level • Marking assignments and exam papers • Invigilating examinations • Assessing Dissertations • Conducting Viva-voce • Supervising students at PG and UG level • Undertaking Research Projects • Collaborating with the other members of the research project outside of the country using remote tools • Doing Field Research • Experimenting Project Work • Administrating Timetable • Interviewing Students • Counselling • Attending Boards and Meetings • Attending Seminars, Tutorials, Workshops • Working in the office for any other work relevant to the above activities

Table 3: Some of the Activities of a Faculty Member in the University

6.3.2 High Level View of the CANTEXT System

CANTEXT (Computer Activity aNd Time Entity Location Tracking) is a context-aware system proposed to sense the activity context of the user using software sensors as seen in Figure 17. Where the Faculty and the Organization use different services (e.g. Profile, Calendar, Timetable, Weather and Context), so the sensors sense the activity context of the faculty; and the Students or the Faculty query to see the activity context of the faculty member.

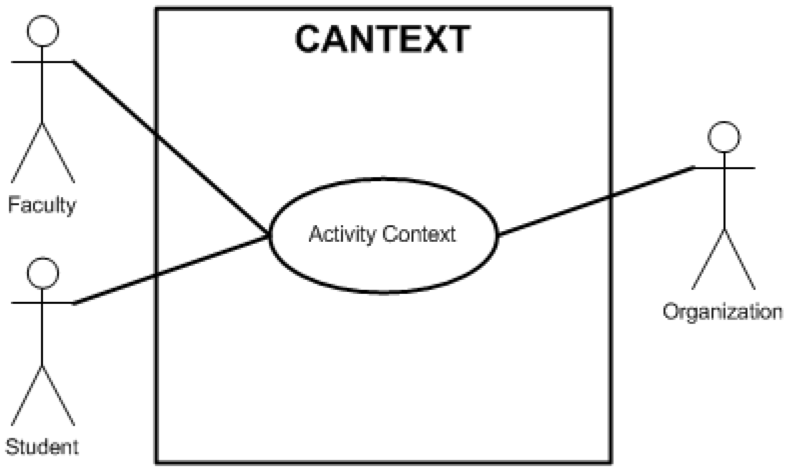


Figure 17: High Level View of the CANTEXT System

6.3.2.1 Calendar Service

Calendar Service (Figure 18) is one of the other services of *CANTEXT* that helps sensors to exploit activity context of the user. This service records the event details of the faculty member for example Event Title, Event Time, Event Location, Event Description etc. Here sensors capture those details to know the activity context of the faculty member.

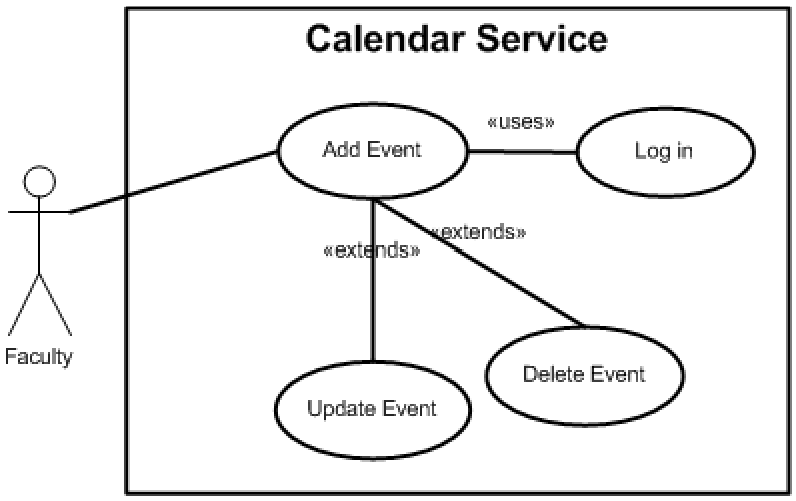


Figure 18: Calendar Service of the CANTEXT System

USE CASE TEMPLATE [2]	
Use Case Name	Add Entry into Calendar Service
Primary actors	Faculty Members
Supporting actors	
Description	In a university working environment it is a best practice to add the entries for the scheduled activities of an employee. These further can be shared amongst students or other staff to inform the class and laboratory timetable.
Triggers	Any working situation e.g. meetings or class rooms timetable etc.

Pre-conditions	Faculty members logged in into the system.
Normal flow	Course of Actions 1. Faculty member logs in the system. 2. Add a new entry into the service 3. Adds the details like: Title of the activity Time interval of the activity Single day or recurrence of the activity. Location of the activity to be held Description of the activity
System flow	The system carries the information from the calendar entries to map the entries into the knowledge base.
Flow exceptions	
Post conditions	The sensors received the data in terms of SOAP message requests and further carry the action.
Additional requirements	It is needed that the faculty is an employee of the particular university of which the systems is up and running.
Notes and issues	

6.3.2.2 Query Service

Query Service enquires from the knowledge base (that is filled with the user's context data) about the activity context of the faculty member of a particular time. This can be used by both Faculty and Student as seen in Figure 19.

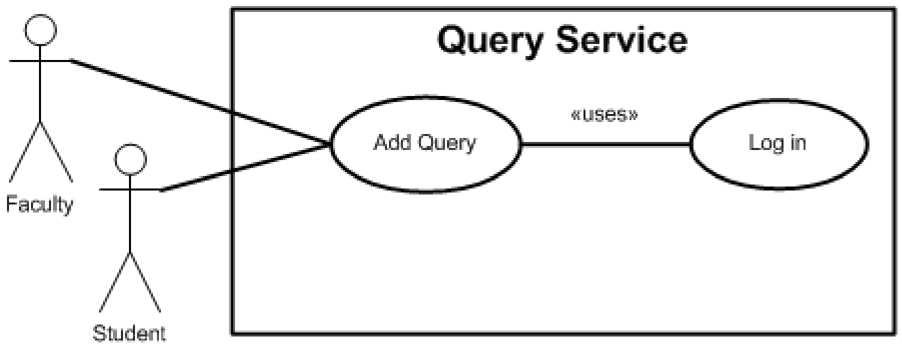


Figure 19: Query Service of the CANTEXT System

6.4 Evaluation Methodology

In this section we present evaluation methodology for the proposed system. The system need not only to sense and acquire the context related data but the investigation of mapping methodology strike on the overall context-aware system based on different scenarios. The need to place the data to the appropriate instances residing in the existing knowledge structure is the challenge in addition to the context knowledge base itself. To know the activity context of the user using services in the scenario, it is vital to place the weather, calendar, timetable, and profile related data with the actual instances. The evaluation would be carried out on the basis of the results achieved after mapping from messages to the knowledge base in the first, second and third match algorithms.

The same approach can transfer its origin to be generic in nature where the knowledge base might be reused by other services in the context-aware scenario. It is also of very important to note here that the inference rules that derive the outcome of the data are specific, in lieu of that we can utilize the data of the context model in other systems.

For example the data which has been collected without intervention of the user regarding the activity of the teacher (a person), where S/he is meeting his/her fellows outside to have a dinner might be utilized by the service-based advertisement system to send offers.

In the case study we proposed that how the goals can be achieved to accomplish the task of meeting a student with a faculty where number of other factors is also involved and the disturbance while carrying out a research task might be the huge. These use cases are particularly for using Calendar Service and some other services would be the future work.

The benefits of the proposed system lies with automated sensing: the activity of the users based on their context in the computerized environment. The same information might spread to number of different scenarios (for example health systems; to monitor the mature man's activeness related to his/her vulnerable health). We might calculate the healthiness of a person by analysing that how much times he leaves his/her house to perform day to day activities? Another extension may be in Disaster Management System; to cope up with any mishap (for example an earthquake), where the system may check the building, based on the activities that were part of that place, the no. of persons involved inside, respective to their sensed activities. This extends the scope of the thesis to not only University environment but to the other systems as well. This may further be extended to automatically make rules for the context-aware systems in general as a future work.

Chapter 7 Implementation and Evaluation

We explained the architecture, data processing model and mapping methodology and introduced our case study. We will now evaluate our work by considering the behaviour of the case study as well as considering our initial aims and objectives. Figure 20 shows the user interface for creating an entry in the Calendar Service running on the web server for the context-aware architecture (explained in Chapter 4). Whenever, a user tries to create or update an entry in the calendar service (i.e. interacts with the web service) the data travels in a SOAP Message in XML form carrying all the relevant data which is required to create or update an entry in the calendar.

Consider the sample SOAP message request extracted from the Calendar Service presented in Figure 21. In the message we can identify a few tags and values that are of interest, for example the *location* tag with its content *Leicester*.

After obtaining the request message, the data is extracted and sent as an input to the mapping algorithms in the form of arguments for the super tag name, sub tag name and sub tag value. As we know that XML are represented by trees with labelled nodes and semantic model is represented as subject-predicate-object. It is therefore needed the tag to be represented as Resources or Literal as per the needs and map this into the semantic form by

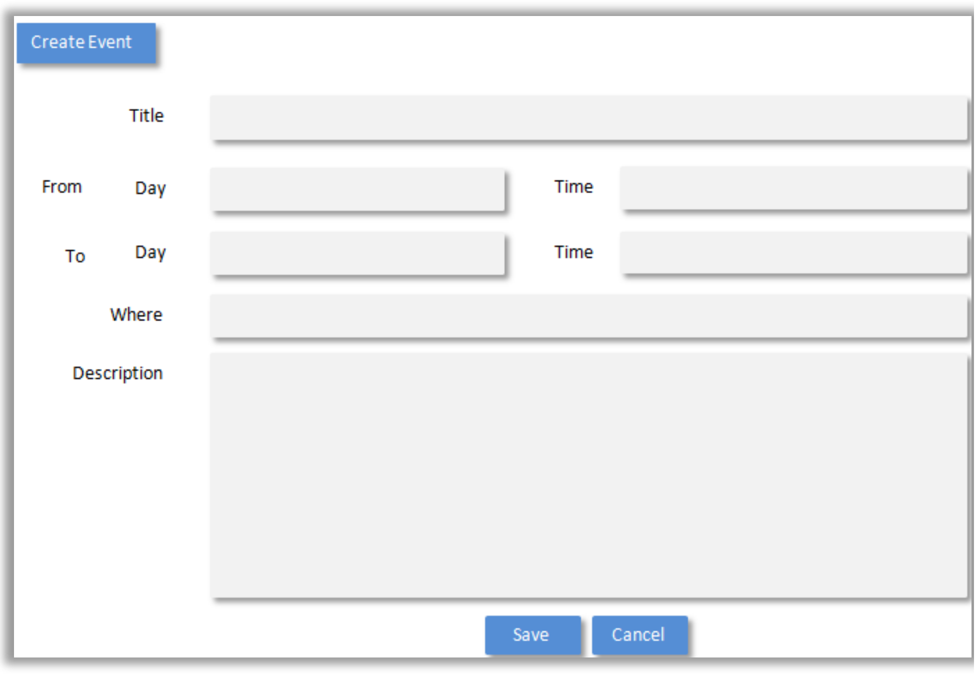


Figure 20: User interface to create an event in the Calendar Web Service

maintaining knowledge structure of the semantic model (i.e. the context model) for the wider use.

7.1 Algorithms

To effectively go through step by step procedure to implement mapping tasks four algorithms (exddplained in Chapter 5) are applied to transform the raw data, which has no impact on other services otherwise; but only used to input for the specific service. It might be helpful for users (either person or other service), whosoever may require the same by adding semantics to make it reusable and extendable in the future.

The first algorithm matches the tag with the instance of the context model and then instantiate the value of that tag as a literal in the ontology.

```xml
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:ser="http://service.calendar.google.com">
  <soap:Header/>
  <soap:Body>
    <ser:getTitle>
      <!--Optional:-->
      <ser:title>Meeting</ser:title>
    </ser:getTitle>
                  <ser:getFromDay>
                  <!--Optional:-->
                  <ser:fromDay>10/07/2012</ser:fromDay>
                  </ser:getFromDay>
                  <ser:getFromTime>
                  <!--Optional:-->
                       <ser:fromTime>18:00:00</ser:fromTime>
                  </ser:getFromTime>
                  <ser:getToDay>
                  <!--Optional:-->
                       <ser:toDay>10/07/2012</ser:toDay>
                  </ser:getToDay>
                  <ser:getToTime>
                  <!--Optional:-->
                       <ser:toTime>19:00:00</ser:toTime>
                  </ser:getToTime>
                  <ser:getLocation>
                   <!--Optional:-->
                        <ser:location>Leicester</ser:location>
                  </ser:getLocation>
                  <ser:getDescription>
                  <!--Optional:-->
                  <ser:description>This is a meeting with a colleague</ser:description>
                  </ser:getDescription>
  </soap:Body>
</soap:Envelope>
```

Figure 21: An Example of SOAP Message Request

The second algorithm finds the synonym of the tag into the lexical database provided if the first algorithm can’t find any match and iterates the process until any of the word from the set of the synonyms of tag name is found in the structure. After finding the tag name, it calls *Algorithm 1* to carry the process of instantiation.

The third algorithm fetches the hierarchy in the super-class, super-superclass… manner and not only instantiates the data but also creates the class if need there be at the required hierarchy level suggested by lexical database (i.e. WordNet).

Note that in all the cases that instantiation of the literal is conditional only if tag name is found or the process is ended with the run of *Algorithm 4*.

The next section describes the inference procedures applied to infer the facts out of the available knowledge base after instantiation.

7.2 Inference Mechanism

Inference mechanism is to try to derive answers from the knowledge base by the available triples after getting instances mapped from the SOAP messages. Because simple queries asking about the data stored in the model are easy to formulate and ask, but do not allow to harness the full power of the semantic technologies at hand. We can enhance the richness of questions to be asked by allowing the use of additional reasoning rules that can combine existing facts to derive new facts. Many of these rules can be quite specific to certain domains, reflecting interpretation of data in those domains. For example, in University there might be sufficient information on rooms and schedules available to derive that a Professor being in a teaching room in the postgraduate block is probably teaching a Postgrad Class. Here is one of the examples of rules that are explained in Chapter 3:

$$hasPlace(?fm, ?pl) \wedge Time(?t) \rightarrow isTeachingClass(?fm, ?pc)$$

where *fm* is a faculty member, *pl* is the name of a place or a building, *t* is the time zone and *pc* is a postgraduate class.

As we know that from the calendar entry system if a person is creating an entry then the name of the calendar associates with him would be that person, then comes the place which has been captured by the location of the calendar, then Time is captured through FromDay, FromTime, ToTime etc., afterwards in the title of the entry faculty writes that he is teaching which shows his activity and then in the final we can capture from his timetable that he is a professional and is teaching certain class.

As all the available data makes sense now because it is stored in the statement form which is inferable. Therefore according to this rule, if a Faculty X is in certain place with the specific time implies that s/he is teaching.

Please note that these rules are not separately compiled but will run automatically whenever any query is generated to fetch the data about that particular individual. Other rules are explained in Chapter 3. Next section explains the queries generated to achieve the task.

7.3 Queries

We have discussed the context model and the insertion of context earlier, so here we are looking at making use of it. A typical question to ask in the context of the scenario might be “What is my supervisor doing on 12 Dec 2011 from 10:00 to 11:00?”. We assume here that the person asking, that is the research student is authorized to ask about the teacher.

Formally, as we are using the RDF based ontology; we can ask questions such as this through the SPARQL query language:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ctx: <http://www.cs.le.ac.uk/ontology/contextmodel#>
      SELECT ?Faculty ?Activity
      WHERE
      {
      #all scheduled activities falls within the time span
      ?Faculty ctx:hasActivity ?Activity.
      ?Facuty ctx:hasProfile ?Profile
      ?Profile ctx:hasLastName ?ln
      ?Profile ctx:hasFirstName ?fn
      ?Activity ctx:hasTime ?Time.
      ?Time ctx:hasStartTime ?st.
      ?Time ctx:hasEndTime ?et.
            FILTER ((?ln>= "FacultyXLastName" &&
                    ?fn<= "FacultyXFirstName"))
            FILTER ((?st>= "2011-12-12 09:00:00" &&
                    ?et<= "2011-12-12 10:00:00"))}
```

The above SPARQL query (explained in Chapter 3) looks very much like an SQL query, and asks about a faculty member's activity at a particular time – some of the arguments (values after the ?) could be instantiated with specific values, directly filtering the results to those matching.

7.4 Evaluation

To check whether the provided results are matching with the expected results we have applied adequacy evaluation. It is very hard though that when evaluation becomes adequate to fully satisfy certain requirements and can be very broad in a sense.

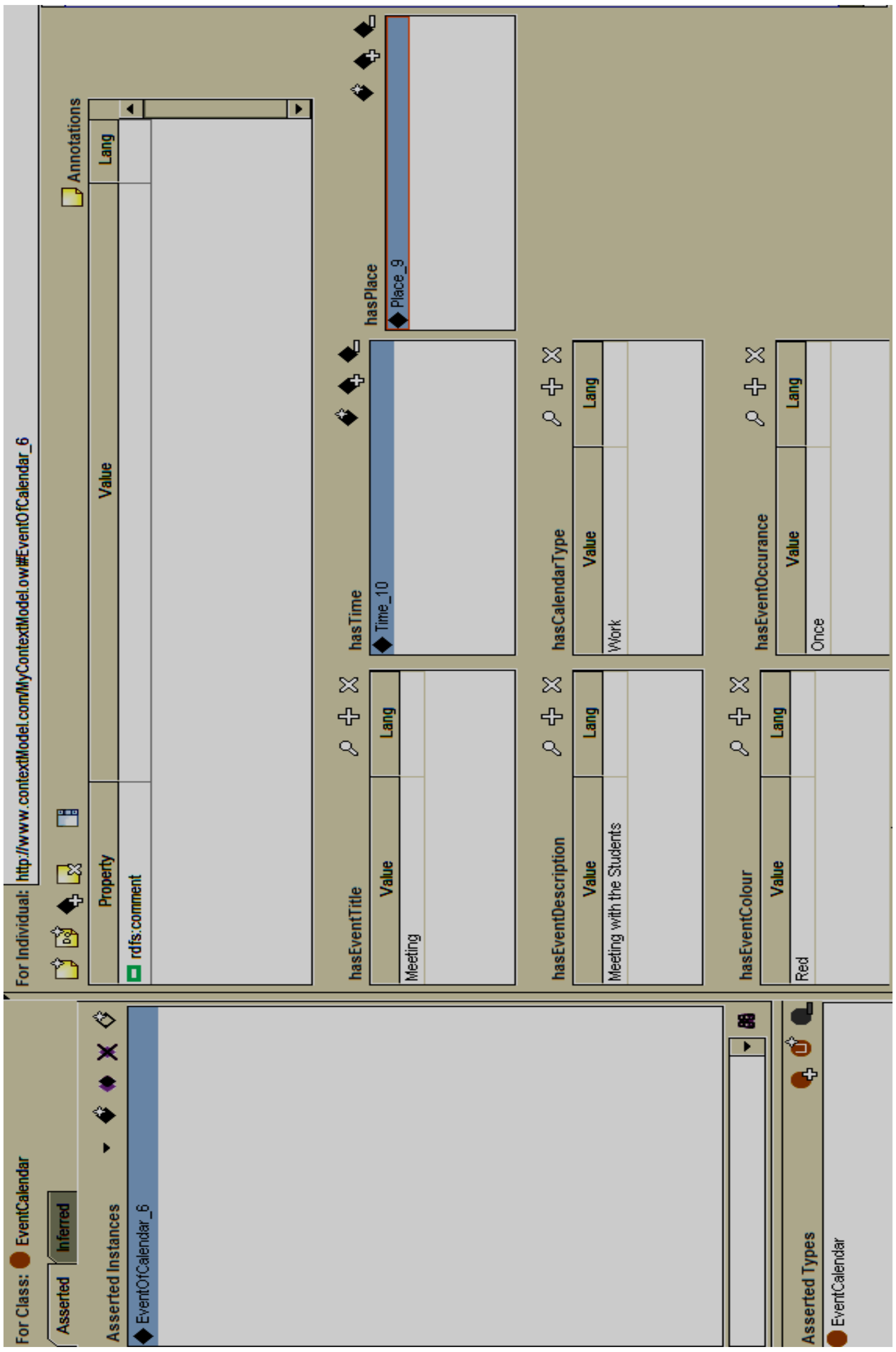
The evaluation is carried out on the basis of information stored after mapping rules; the data is collected from the knowledge base. The criterion to which the evaluation carries out is:

Fulfilment of the research objectives: The overall research objectives are that a user gets the activity of the user in the context-aware environment; that is considerable only when the data (sensed and processed afterwards) fits in the appropriate place after matching rules so that the further processes might be carried out.

Words	Synonyms	Context Model
EventTitle	Consequence (attached Word) title, deed, championship,	DoC_{-1}
StartDay	start, beginning (attached Word) Day	DoC_0
StartTime	start, beginning (attached Word) time	DoC_0
EndTime	end, ending, goal, close (attached Word) time	DoC_0
EndDay	end, ending, goal, close (attached Word) Day	DoC_0
Location	location, placement, localization	DoC_0
Description	description	DoC_0

Table 4: Test Case 2 Showing the Confidence Level on Synonym Match

Matching Rules: One of the main goals of evaluation is to check the elaborated test cases mentioned in this chapter. Where the words match with the vocabulary provided, or all the possible synonyms or the hierarchy of the words maintaining structure are traversed in the appropriate places in the context model.


For Class: EventCalendar

Asserted | Inferred

Asserted Instances

EventOfCalendar_6

Asserted Types

EventCalendar

For Individual: http://www.contextModel.com/MyContextModel.owl#EventOfCalendar_6

Annotations

Property	Value	Lang
rdfs:comment		

hasEventTitle

Value	Lang
Meeting	

hasTime

Time_10

hasPlace

Place_9

hasEventDescription

Value	Lang
Meeting with the Students	

hasCalendarType

Value	Lang
Work	

hasEventColour

Value	Lang
Red	

hasEventOccurance

Value	Lang
Once	

Figure 22: An Example of the Output for the Test Case 1

Degree of Confidence: A check is also felt mandatory to comfort the user with the feeling or belief to rely on the system with more to the less confidence, if the processed data fits at the first match to the last match respectively.

Circumstantial with the system, it can be said that we are looking for the user’s data to be fit in the appropriate place to further carry inference procedures and retrieval methods for the system to run smoothly.

To start with all this we have to look from the start of the system, where this valuable data coming from. It is assumed that the SOAP file retains its conventions (by conventions here means the names of the methods and variables are user-friendly). For example if a method is requiring user to input the location, it is assumed that a method name is either *getLocation()*, *haveLocation()*, *inputLoction()*, *locationGet()*, *locationInput()* etc. where at least the root of the function name or a variable describes what it actually means with the actual words or synonyms of that regardless of what the pre-root and post-root of the word describes. The example showing sample of SOAP request message is presented in the former sections.

Moving forward with the assumption that developer has provided with proper words or same meaning words in the services. In general we might assume two cases in the first if the name does not follow the functionality for which it executes (however the user friendly notion in the programming is very good practice to follow) or ambiguity in the words used otherwise. In the first case it might mismatch the terms used in the context model but in another case where a name is used which does not exist in the lexical database as well tends to the No Match situation to occur. To further go at every step we have evaluated with the following test cases where data needs to be fit in the context model specifically and the user confidence as general.

7.4.1 Test Case No. 1

In test case 1, it is assumed that the words match with the vocabulary provided or match with the actual context model. We believe that almost everything the normal calendar can relate to populate the data is covered in the context model. This data is an input to the knowledge base hence the requirement for that would be very crucial.

In the first case the results are quite satisfactory (an example is shown in Figure 22) and provide a push to strengthen the level of confidence on the data. We apply a check to which level the required results are achieved to make us more confident; if the same meet with the run of very first algorithm rather the last one. Further cases are elaborated in the next sections.

7.4.2 Test Case No. 2

In the second case where a check is made against all possible synonyms generated from the WordNet of that particular tag that matches with the model as shown in Table 4. Here DoC_{-1} corresponds to the values that can negatively affect the instance because of their synonyms can be used as a different word in different concepts and DoC_0 is considered as less affecting synonyms or the words and have more clear meaning. Opting results at this case reflects less confidence than the first case and can also affect the structure of the knowledge if the proper word is not matched with the required one.

First column of Table 4 shows the terms we obtained from the SOAP messages, in the second column there are fetched values (synonyms) of the word, if one word or two are attached synonyms are available, and the last column shows if there might be confusion with the words that might disrupt

Words	Hypernyms	Context Model
EventTitle	event+heading->text->writing->entity	DoC_0
StartDay	start+day->time->measure->entity	DoC_0
StartTime	start+Instance->happening->event->entity	DoC_0
EndTime	end+Instance->happening->event->entity	DoC_0
EndDay	end+Day->time->measure->entity	DoC_0
Location	location->object->entity	DoC_{-1}
Description	description->statement->message->communication->entity	DoC_0

Table 5: Test Case 3 Showing the Confidence Level on Hypernym Match

the already available structure for example synonym of the word *title* might be deed and *championship, title (according to WordNet)* which are the kind of synonyms that can alter the meaning and will be a reason to cause ambiguity amongst the concept where a *Title* might also state the championship of the person, although this means the *event title* here. This term will work less efficiently in this situation or we may say that this will cause less confidence on the system compared to the first case.

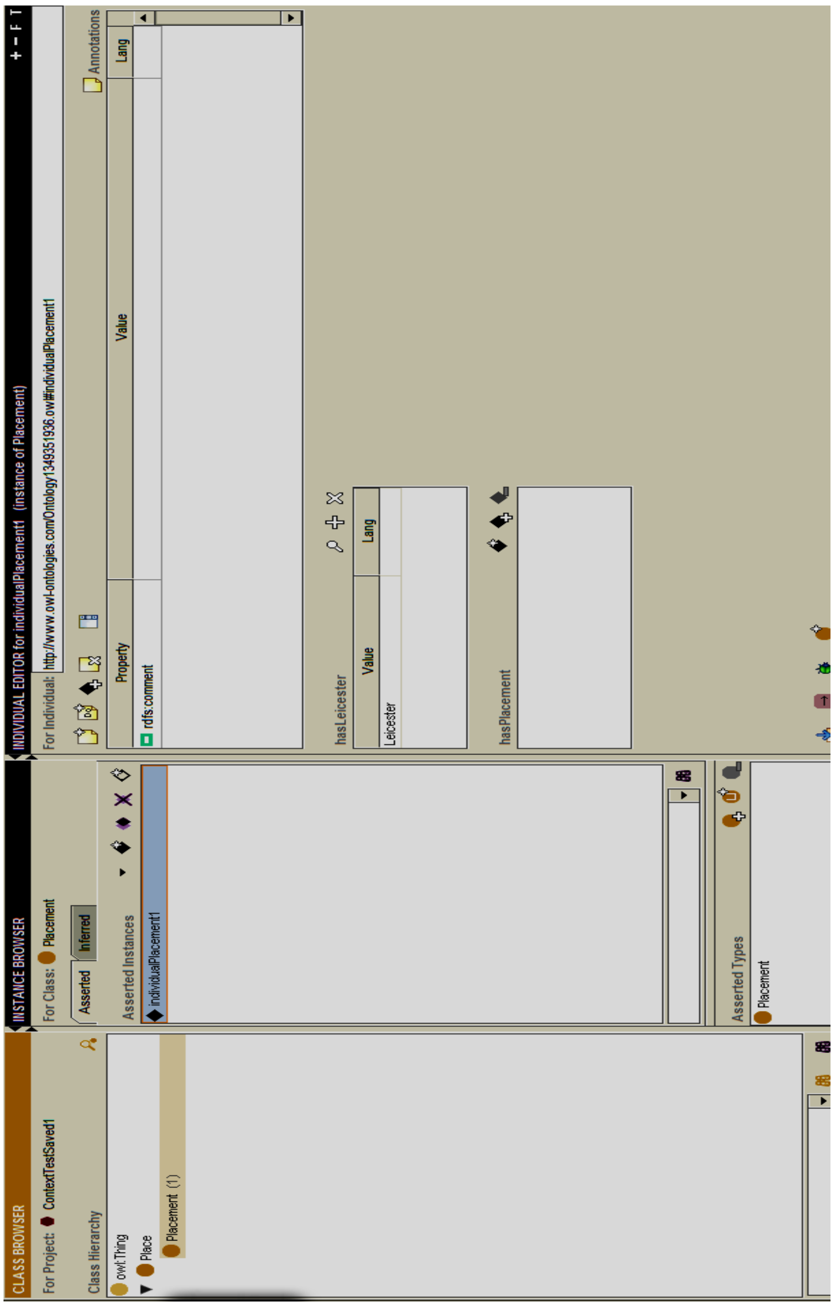


Figure 23: An Example of the Output for the Test Case 2

An example of output for the test case 2 is shown in Figure 23, where the tag *location* is replaced with the word *placement* in the ontology because *location* was not present in the model and the value is instantiated on the word *placement* because the *tagNameValue* is *Leicester*.

7.4.3 Test Case No. 3

In this case where the check is on the hierarchy of the word it matches to the tag name. This hierarchy represent the classes subclasses represented in the nodes and places the nodes in a way that all the words related to the upper level of the hierarchy are checked against the structure, and if at any of the level the node is found then program runs the reverse direction to place every super node of the term populating the value with the appropriate place with the actual word. The Table 5 shows the degree of the confidence on the hierarchy of the terms:

Table 5 shows the results achieved while fetching the tree of the word (tag name) out of the WordNet lexical database. The degree of confidence works the same way as test case 2 where DoC_0 has neutral level in the confidence and DoC_{-1} has less. As we can see that *Location* has a tree *location->object->entity* where *object* not only means tangible thing but also can be *purpose*, *aim* or *goal of a specific action* and can be used in variety of ways and a cause to create confusion in the concept. An example of output for the test case 3 is shown in Figure 24, where tag name *location* is not found and the algorithm created the class *object* and then created a subclass *location* (according to lexical database) and then instantiated *location* with the value *Leicester*. To match whether these provide the expected results for which the system was proposed the adequacy is checked with the scenarios mentioned in the chapter 6.

7.4.4 Scenario Evaluation

After results achieved at the scenario level, it is observed the data is instantiated at the required space in first, second and third match (shown in Table 6) with the more degree of confidence in the first place with less in the last, respectively. Where *(+)* means more confidence, *(-)* means less confidence and *(un)* means unchanged/neutral degree of confidence on the data. Degree of confidence with the first iteration strengthens the results with the achieved structure hence the knowledge base instantiates the data, where it needs to be; in relation with the classes and instances.

The user puts the confidence on the system with the reliability of the data she fetches. The notion of the confidence is measured here with the first second and third number of matches. For example a word which matches instantly with the context model and instantiates its value to the structure might be relied more upon than the synonym of that word or hierarchy in the last (discussed in the sections Test case 1, 2 and 3). Table 6 illustrates the same confidence level on the different level of matches, whatsoever comes first and becomes true, and the degree of confidence at the overall system level might be considered as a future work, where we might quantify all the positive, negative and unchanged values of all the services in the system to provide user with more confidence. This assures the user to rely on the results that whether Mr. Kim is *teaching* or *stuck in the worst weather situations*.

7.5 Discussion

From the evaluation, we comprehend that the placement of data is linear in a nature that it depends upon the words we choose to develop the methods and

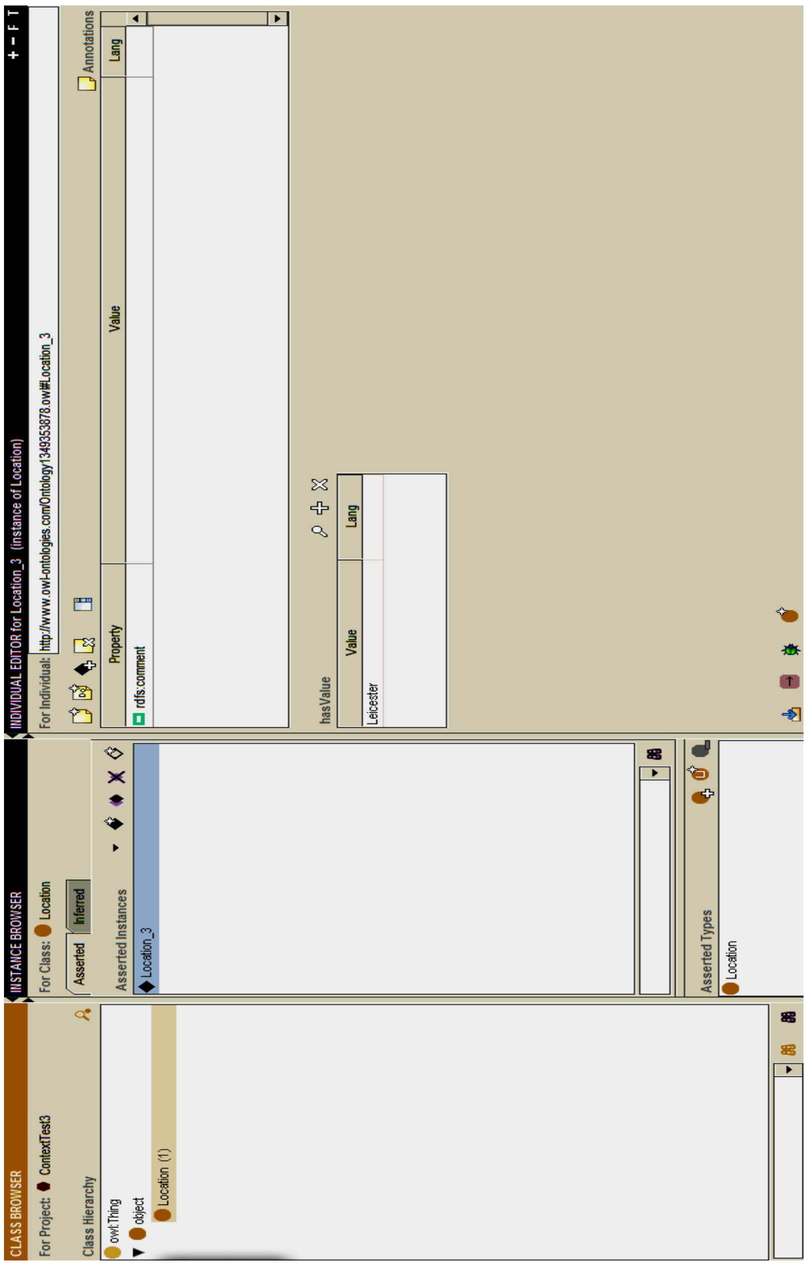


Figure 24: An Example of the Output for the Test Case 3

variables in the web services, which affect the SOAP messages that are generated and in the last might affect our proposed system by placing the data for that particular instance for which it is meant to be. The architecture

	$Direct\ Match$	$Synonym\ Match$	$Hypernym\ Match$
$EventTitle$	$DoC +$	$DoC -$	$DoC\ (un)$
$StartTime$	$DoC +$	$DoC\ (un)$	$DoC\ (un)$
$EndTime$	$DoC +$	$DoC\ (un)$	$DoC\ (un)$
$Location$	$DoC +$	$DoC\ (un)$	$DoC -$
$Description$	$DoC +$	$DoC\ (un)$	$DoC\ (un)$

Table 6: Adequacy Evaluation using Check and Match Methodology

facilitated to sense the context with the help of software sensors. The design of data model addressed the core terms related with the context in the context-aware environments. To come up with the need of context-awareness issues relevant with software sensors, the structure was modelled

In the number of test cases: at the run of the very first case, we assumed that the terminology of the knowledge management complies with the same words as are used with the web services' methods and variables and we saw the fruitful results. Initial results support the system, although we observe that for example if the service asks to instantiate the word *title* as an *eventTitle* for the Calendar, but what if the structure models it as *championship title* according to the synonyms brought by WordNet. For example if the user's *Profile* inside the knowledge management structure points out the word *title* (as the championship *title* won by that person). However, the system wanted to place an instance of Event Title to be placed inside the Calendar's entry (*Title*) rather than Profile's achievement. This would definitely affect the system because from *calendar title* we assume the user's activity; however from the *professional title* it means the status of holding an award or recognition. In the same way when we see the fetched hierarchy from the lexical database for the word *location* makes a tree like *entity->object->location*, however the structure of the context model presents *object* as the superclass of *computerized* or *non-computerized* entities rather *location*. It is therefore submitted that while evaluating the scenarios for the degree of confidence, if we give a *plus* to every right value for the individual, *minus* for every misleading or unidentified value and an *unchanged* for less affecting values; even then we are closer to good results producing the required output for the system. Some of the terms can create less confidence on the data and can be avoided with showing the asker (the one who asks about the user's activity) the degree of confidence along with the actual results. These results might be individually presented with every other service or combined as a whole to assure the user's confidence on the system.

However, there might be some cases where these all cases might not be of that use where the ambiguous terms are written to represent the data that might lower the fruitful outcome of the system and sometimes might also become aftermath. It has been observed that it is a good practice to follow the user friendly names while developing hence suggested in the system.

The populated data in the context model may further be used/extended in the different applications if need there be. Though, the semantic rules rely on the context model specifically because those meant to drive the outcome from the available data according to the user needs on the system. For example the context data related to activity of the person might be used in the advertisement field to know whether a person X has already had his lunch or about to go for one according to its own inference.

After all of this, still the results are quite satisfactory and the future needs would be the attachment of the degree of confidence at the various levels for the guidance as per the activity of the person.

7.6 Summary

In this chapter, the implementation and evaluation of the system with check and match methodology is explained. In the first it is started with fetching SOAP request messages from the service-based environment meant for the calendar web service. It is further shown that how this data maps into the knowledge base with the help of first, second and third algorithm. The queries and inference procedure carried further to process that data to achieve the desired results. In the last adequacy evaluation is elaborated to check the data that fits appropriately to the knowledge base after three matches for the further process and evaluation is also carried out with confidence level which is demonstrated for the reliability on that data that that fits at the required instances with the literals in it.

Chapter 8
Conclusion

To sense the user's activity context using software sensors in the context-aware environment, is the main objective of this research. To attain it the overall aims are 1) sensing activity context of the user with the help of software sensors 2) providing semantic model for the data for interoperability and reasoning and 3) mapping sensed data into the semantic model.

8.1 Research Contributions

To discuss the research contributions we have described the aims and objectives presented below:

1) Sensing activity context of the user is an important work in this research because the challenge was to identify the framework that supports and the way we can exploit the user's activity using software sensors.

Service oriented architecture helps to collect the user's context while user exchanges messages to services to perform her task. Sensors have been developed to acquire the exchanges to work as an extra layer in the context-aware architecture.

2) After acquiring this raw data the need of the research was how this data should be processed. While talking about the context-aware systems the requirement was not only to provide a simple data model but a model that supports future trends and can be accessible, scalable, reusable, and

extendable. Not only this, but the model should also support reasoning rules to infer the facts from the knowledge base for the context-aware systems. It should also fulfil all the concepts related to the context-aware systems to follow the user needs on systems. To accomplish that task the context model has been designed for the system to provide the raw data with a semantic model to process and store in the knowledge base.

3) Mapping sensed data into the semantic model was the challenge that describes how to transform the raw data into the semantic model. To instantiate the acquired data into the knowledge base while maintaining the structure is the challenge this research addressed. Providing a meaning to the XML tags and values into the context model's instances is a hard work that is solved by giving three techniques that explain the mapping methodology as a whole. Direct Match is done with matching the instances that exist in the model. However, Synonym and Hypernym matches are done with using a lexical database (i.e. WordNet). These matches help direct the context related data to the knowledge base so that the semantic rules be applied to infer the outcome.

8.2 Future Directions

The presented context sensing architecture and mapping methodology support the applications that sense the activity context based on software sensors. After getting the required output according to the set aims from the proposed system there are still directions to do more in our research.

For example the questions like, for how long the context data is needed? which kind of context related data is needed? might create some opportunities to find out in a systematic and scientific manner at the context acquiring level.

The other research issues and directions in this field might be to parse the description of the message. This might describe insight for the activities, for example what is the core of the activity? which entities would be accompanied?, what is the agenda? what are the extra resources of that place that have come along with the users?

Addition of other services that might help the system to strengthen the confidence and attachment of degree of confidence with respect to scheduled or deduced data, are some of the issues of the future at the various layers.

As Google has also started the knowledge graph based on the semantic technologies; we think that by enhancing the proposed system it can help to advertise more efficiently where the system relies on the activities rather than the keywords of the message only.

8.3 Concluding Remarks

As context-aware systems have evolved over time to facilitate users and service oriented architecture have provided a test bed. To expand their limitations we need to bridge the gap between the semantic web technologies and the former one. To lessen the burden on the hardware sensors in the pervasive computing, our system relies on software sensors to sense the activity of the user. It is therefore service-based architecture, context model and mapping methodology have been proposed to deal with the issues regarding the system sensing with software. The data is fetched using sensors from the traditional services, semantic processing model is provided with the context model, the mapping methodology helps transforming the data (by maintaining the semantic structure), which is otherwise not in use in the environment for this purpose.

The proposed system follows the future trends and standards and provides new ways of thinking to the existing systems for activity context sensing. We believe that this work will help in the field of context-aware systems where context sensing is done with the help of software sensors and activity context is of more interest.

Appendix A
An Example of the Mapping

Direct Match:

```
//check class
OWLNamedClass superTagNameClass=
loadedOntologyCore.getOWLNamedClass(superTagName);
System.out.println("SuperTagName: "+superTagNameClass);

String strSubTagName=subTagName;

      if(superTagNameClass != null){
      //create string for Object property
      String superTagPrefix="has"+strSubTagName;
      //check the superTagProperty or relationship
      try {
      OWLObjectProperty
      superTagProperty=loadedOntologyCore.getOWLObjectProperty(sub
      TagName);
      System.out.println("SuperTagProperty: "+superTagProperty);
      //search propety

            if(superTagProperty != null){
            RDFResource
            superTagRange=superTagProperty.getRange(true);
            RDFResource superTagDomain =
            superTagProperty.getDomain(true);
            System.out.println("SuperTagRange:-->
            "+superTagRange);
            }
            else{
            //create property
            superTagProperty =
            loadedOntologyCore.createOWLObjectProperty(superTagPre
            fix);

            OWLNamedClass tempSubTagClasss =
            loadedOntologyCore.getOWLNamedClass(subTagName);
            //check the subclass and create subclass

                  if(tempSubTagClasss != null){
                  superTagProperty.setDomain(superTagNameClass);

      superTagProperty.setRange(tempSubTagClasss);
```

```
                    }
                    else{
                    tempSubTagClasss =
                    loadedOntologyCore.createOWLNamedClass(subTagNam
                    e);

                    tempSubTagClasss.addSuperclass(superTagNameClass
                    );

                    tempSubTagClasss.removeSuperclass(loadedOntology
                    Core.getOWLThingClass());
                    superTagProperty.setDomain(superTagNameClass);
                    superTagProperty.setRange(tempSubTagClasss);


               }
          }
```

Synonym Match:

```
int[] ids = wordnet.getSenseIds(word, pos[0]);

for (int i = 0; i < ids.length; i++) {
  System.out.println("Sense: " + ids[i]);
  String description = wordnet.getDescription(ids[i]);
    if (words != null) {
  System.out.print("Synset: ");
  for (int j = 0; j < words.length; j++)
System.out.print(words[j] + " ");
  }
  System.out.println("\n-");
}
```

Hypernym Match:

```
String word = "location";
String pos = wordnet.getBestPos(word);
String[] result = wordnet.getHypernymTree(ids [0};
for (int i = 0; i < result.length; i++) {
  System.out.println(result[i]);
}
```

Appendix B
Publications

1. S. Reiff-Marganiec, K.T. Pathan, Y. Hong: User Activity Recognition through Software Sensors. Distributed Networks: Intelligence, Security, and Applications, CRC Press, 2013.
2. K.T. Pathan, S. Reiff-Marganiec, et al.: Reaching Activities by Places in the Context-Aware Environments using Software Sensors. Journal of Emerging Trends in Computing and Information Sciences, volume 2, 2011.
3. K.T. Pathan, S. Reiff-Marganiec, Y. Hong: Mapping for Activity Recognition in the Context-Aware Systems Using Software Sensors. In *Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC '11)*. IEEE Computer Society, pages 215-221, 2011.
4. K.T. Pathan and S. Reiff-Marganiec: Towards Activity Context using Software Sensors. M. ter Beek In *Proceedings of YR-SOC 2009*. EPTCS volume 2, pages 27-35, DOI. 2009.

Poster

1. K.T. Pathan, and S. Reiff-Marganiec**:** I Know! What You Do All Summer. *Fifth Festival of Postgraduate Research, University of Leicester*. 2010.

Bibliography

[1] A. Asthana, M. Cravatts, P. Krzyzanowski: An Indoor Wireless System for Personalized Shopping Assistance. pages 69-74, 1994.

[2] A. COCKBURN: Basic Use Case Template. *Humans and Technology, Technical Report*, volume 96, 1998.

[3] A. Dey, D. Salber, G. Abowd, et al.: The Conference Assistant: Combining Context-Awareness with Wearable computing. In *Proceedings of the Third International Symposium on Wearable Computers, 1999. Digest of Papers*, pages 21-28, 1999.

[4] A. Ferscha, S. Vogl, W. Beer: Ubiquitous Context Sensing in Wireless Environments. KLUWER INTERNATIONAL SERIES IN ENGINEERING AND COMPUTER SCIENCE, pages 98-108, 2002.

[5] A. Hopper: The Clifford Paterson Lecture, 1999. Sentient Computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, volume 358: pages 2349-2358, 2000.

[6] A. Ranganathan and R.H. Campbell: An Infrastructure for Context-Awareness Based on First Order Logic. Personal and Ubiquitous Computing, volume 7: pages 353-364, 2003.

[7] A. Schmidt, K.A. Aidoo, A. Takaluoma, et al.: Advanced Interaction in Context. LECTURE NOTES IN COMPUTER SCIENCE, pages 89-101, 1999.

[8] A. Schmidt, M. Beigl, H.W. Gellersen: There is More to Context than Location. Computers & Graphics, volume 23: pages 893-901, 1999.

[9] A. Schmidt: Ubiquitous Computing–Computing in Context. Lancaster University, UK, 2002.

[10] A.K. Dey and G.D. Abowd: CybreMinder: A Context-Aware System for Supporting Reminders. LECTURE NOTES IN COMPUTER SCIENCE, pages 172-186, 2000.

[11] A.K. Dey and G.D. Abowd: Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of CHI 2000 Workshop on the What, Who, Where, When and How of Context-Awareness*, 2000.

[12] A.K. Dey, G.D. Abowd, D. Salber: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer-Interact, volume 16: pages 97-166, 2001.

[13] B. Bouzy and T. Cazenave: Using the Object Oriented Paradigm to Model Context in Computer Go. In *Proceedings of the First International and Interdisciplinary Conference on Modelling and Using Context*, 1997.

[14] B. Schilit and M. Theimer: Disseminating Active Map Information to Mobile Hosts. Network, IEEE, volume 8: pages 22-32, 1994.

[15] B. Schilit, N. Adams, R. Want: Context-Aware Computing Applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85-90, 1994.

[16] B.E. Brewington, and G. Cybenko: Keeping Up with the Changing Web. COMPUTER, pages 52-58, 2000.

[17] B.P. Clarkson: Life Patterns: Structure from Wearable Sensors. Massachusetts Institute of Technology, 2003.

[18] C.J. Matheus, M.M. Kokar, K. Baclawski: A Core Ontology for Situation Awareness. In *Proceedings of the Sixth International Conference on Information Fusion*, pages 545-552, 2003.

[19] D.L. McGuinness, F. Van Harmelen: OWL Web Ontology Language Overview. http://www.w3.org/2004/OWL/, W3C Recommendation, volume 10, 2004.

[20] E. Prud'hommeaux, A. Seaborne: SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/, *W3C Working Draft*, volume 4, 2008.

[21] F. Bennett, T. Richardson, A. Harter: Teleporting-Making Applications Mobile. In *Proceedings of the Mobile Computing Systems and Applications*, pages 82-84, 1994.

[22] Facebook. https://www.facebook.com/, last accessed on 07 August 2012.

[23] G. Chen and D. Kotz: A Survey of Context-Aware Mobile Computing Research. In *Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College*, 2000.

[24] G.D. Abowd, C.G. Atkeson, J. Hong et al.: Cyberguide: A Mobile Context-aware Tour Guide. Wireless Networks, volume 3: pages 421-433, **1**997.

[25] G.M. Voelker and B.N. Bershad: Mobisaic: An Information System for a Mobile Wireless Computing Environment. KLUWER INTERNATIONAL SERIES IN ENGINEERING AND COMPUTER SCIENCE, pages 375-394, 1996.

[26] Gmail. https://mail.google.com/, last accessed on 05 July 2012.

[27] H. Bohring, and S. Auer: Mapping Xml to Owl Ontologies. Leipziger Informatik-Tage, volume 72: pages 147-156, 2005.

[28] H. Chen, T. Finin, A. Joshi, et al.: Using OWL in a Pervasive Computing Broker. Defense Technical Information Center, 2005.

[29] H. Chen, T. Finin, A. Joshi: An Ontology for Context-Aware Pervasive Computing Environments. The Knowledge Engineering Review, volume 18: pages 197-207, 2004.

[30] H. Yan and T. Selker: Context-Aware Office Assistant. In *Proceedings of the Fifth International Conference on Intelligent User Interfaces*, pages 276-279, 2000.

[31] H.L. Truong, S. Dustdar, D. Baggio, et al.: InContext: A Pervasive and Collaborative Working Environment for Emerging Team Forms. In *Proceedings of the 2008 International Symposium on Applications and the Internet (SAINT 2008)*, IEEE Computer Society. 2008.

[32] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, volume 21: page 79, 2004.

[33] J. Bacon, J. Bates, D. Halls: Location-Oriented Multimedia. Personal Communications, IEEE Wireless Communications, volume 4: pages 48-57, 1997.

[34] J. Bauer, R.D. Kutsche, R. Ehrmanntraut: Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic. Technische Universität Berlin, März, 2003.

[35] J. Budzik, and K.J. Hammond: User Interactions with Everyday Applications as Context for Just-in-Time Information Access. In

Proceedings of the Fifth International Conference on Intelligent User Interfaces, pages 44-51, 2000.

[36] J. Kopecky, T. Vitvar, C. Bournez and J. Farrell: Sawsdl: Semantic Annotations for WSDL and XML Schema. Journal of Internet Computing, IEEE, volume 11, pages 60-67, 2007.

[37] J. McCarthy: Notes on Formalizing Context. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 13: pages 555-555, 1993.

[38] J. Pascoe: Adding Generic Contextual Capabilities to Wearable Computers. In *Proceedings of the Second International Symposium on Wearable Computers*, pages 92-99, 1998.

[39] J.E. Bardram: Activity-Based Computing: Support for Mobility and Collaboration in Ubiquitous Computing. Personal and Ubiquitous Computing, volume 9: pages 312-322, 2005.

[40] J.I. Hong and J.A. Landay: An Architecture for Privacy-Sensitive Ubiquitous Computing. In *Proceedings of the Second International Conference on Mobile Systems, Applications and Services*, pages 177-189, 2004.

[41] J.S. Wilson: Sensor Technology Handbook. Newnes, 2005.

[42] K. Cheverst, K. Mitchell, N. Davies: Design of an Object Model for a Context Sensitive Tourist GUIDE. Computers & Graphics., volume 23: pages 883-891, 1999.

[43] K. Cheverst, N. Davies, K. Mitchell, et al.: Developing a Context-Aware Electronic Tourist Guide: Some Issues and Experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 17-24, 2000.

[44] K. Henricksen, J. Indulska, A. Rakotonirainy: Generating Context Management Infrastructure from Context Models. Industrial Track,

In *Proceeding of the Fourth International Conference on Mobile Data Management (MDM),* pages 1-6, 2003.

[45] K. Henricksen, J. Indulska, A. Rakotonirainy: Modeling Context Information in Pervasive Computing Systems. LECTURE NOTES IN COMPUTER SCIENCE, pages 167-180, 2002.

[46] K.T. Pathan, S. Reiff-Marganiec, A.A. Shaikh, et al.: Reaching Activities by Places in the Context-Aware Environments using Software Sensors. Journal of Emerging Trends in Computing and Information Sciences, volume 2, 2011.

[47] K.T. Pathan, S. Reiff-Marganiec, Y. Hong: Mapping for Activity Recognition in the Context-Aware Systems Using Software Sensors. In *Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC '11)*. IEEE Computer Society, pages 215-221, 2011.

[48] K.T. Pathan, N. Channa, N.H. Arijo, et al.: Architecture for Sensing Activity Context using Software Sensors in the Context-aware Service-based Environments. Journal of Sindh University Research Journal (Science Series), volume 45, pages 01-06, 2013.

[49] S. Reiff-Marganiec, K.T. Pathan, Y. Hong: User Activity Recognition through Software Sensors. Distributed Networks: Intelligence, Security, and Applications, CRC Press, page 243, 2013.

[50] L. Finkelstein, E. Gabrilovich, Y. Matias, et al.: Placing Search in Context: The Concept Revisited. ACM Transactions on Information Systems, volume 20: pages 116-131, 2002.

[51] L.A. Klein: Sensor and data fusion concepts and applications. Society of Photo-Optical Instrumentation Engineers (SPIE) Bellingham, WA, USA, 1999.

[52] M. Baldauf, S. Dustdar, F. Rosenberg: A Survey on Context-Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing, volume 2: pages 263-277, 2007.

[53] M. Ferdinand, C. Zirpins, D. Trastour: Lifting Xml Schema to Owl. Web Engineering, pages 776-777, 2004.

[54] M. Fowler and K. Scott: UML Distilled: A Brief Guide to Standard Object Modeling Language. Addison-Wesley Longman Publishing Co., Inc., 2000.

[55] M.C. Daconta, L.J. Obrst, K.T. Smith: The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. Wiley, 2003.

[56] Merriam-Webster Dictionary. http://www.merriam-webster.com/dictionary/context/, last accessed on 04 August 2012.

[57] N. Marmasse and C. Schmandt: Location-Aware Information Delivery with ComMotion. LECTURE NOTES IN COMPUTER SCIENCE, pages 157-171, 2000.

[58] N. Mitra, Y. Lafontitle: SOAP version 1.2 part 0: Primer. http://www.w3.org/TR/soap12-part0/. W3C. 2007.

[59] N. Shadbolt, W. Hall and T. Berners-Lee: The Semantic Web Revisited. Intelligent Systems, IEEE, volume 21: pages 96-101, 2006.

[60] O. Lassila and R.R. Swick: Resource Description Framework (RDF) Model and Syntax Specification. CiteSeer, 1998.

[61] P. Brown and J.D.X. Chen: Context-Aware Applications: From the Laboratory to the Marketplace. Personal Communications, IEEE, volume 4: pages 58-64, 1997.

[62] P. Gray and D. Salber: Modelling and using Sensed Context Information in the Design of Interactive Applications. LECTURE NOTES IN COMPUTER SCIENCE, pages 317-336, 2001.

[63] P. Otzturk and A. Aamodt: Towards a Model of Context for Case-Based Diagnostic Problem Solving. In *Proceedings of the Interdisciplinary conference on modelling and using context (Context-97)*, pages 198–208, 1997.

[64] P. Prekop and M. Burnett: Activities, Context and Ubiquitous Computing. Computer Communications, volume 26: pages 1168-1176, 2003.

[65] P.J. Brown: The Stick-e Document: A Framework for Creating Context-Aware Applications. ELECTRONIC PUBLISHING-CHICHESTER, volume 8: pages 259-272, 1995.

[66] P.P.S. Chen: The Entity-Relationship model—toward a Unified View of Data. ACM Transactions on Database Systems (TODS), volume 1: pages 9-36, 1976.

[67] R. Ghawi and N. Cullot: Building Ontologies from XML Data Sources. In *Proceedings of the Twentieth International Workshop on Database and Expert Systems Applications (DEXA'09).* IEEE. pages 480-484, 2009.

[68] R. Hull, P. Neaves, J. Bedford-Roberts, et al.: Towards Situated Computing. HP LABORATORIES TECHNICAL REPORT HPL, 1997.

[69] R. Want, A. Hopper, V.F.J. Gibbons: The Active Badge Location System. ACM Transactions on Information Systems, volume 10: pages 921-102, 1992.

[70] R.F.B. Neto and M.G.C. Pimentel: Toward a Domain-Independent Semantic Model for Context-Aware Computing. In *Proceedings of*

the Third Latin American Web Congress (LA-WEB 2005), IEEE. volume 10, 2005.

[71] R.M. Gustavsen: Condor–an Application Framework for Mobility-Based Context-Aware Applications. In *Proceedings of the Workshop on Concepts and Models for Ubiquitous Computing*, 2002.

[72] S. Loke: Context-aware Pervasive Systems: Architectures for a New Breed of Applications. Auerbach Pub, 2006.

[73] T. Berners-Lee, J. Hendler, O. Lassila: The Semantic Web. Scientific American, volume284: pages 28-37, 2001.

[74] T. Bray, J. Paoli, et al.: Extensible Markup Language (XML). *World Wide Web Journal*, volume 2(4): pages 27-66, 1997.

[75] T. Gu and H.K. Pung: Toward an OSGi-Based Infrastructure for Context-Aware Applications. IEEE Pervasive Computing, pages 66-74, 2004.

[76] T. Gu, X.H. Wang, H.K. Pung, et al.: An Ontology-Based Context Model in Intelligent Environments. In *Proceedings of Communication Networks and Distributed Systems Modelling and Simulation Conference*, 2004.

[77] T. Halpin: Information Modelling and Relational Databases: From Conceptual Analysis to Logical Design. Morgan Kaufmann, 2001.

[78] T. Hofer, W. Schwinger, M. Pichler, et al.: Context-Awareness on Mobile Devices-the Hydrogen Approach. In *Proceedings of the Thirty Sixth Annual Hawaii International Conference on System Sciences*, volume 10, 2003.

[79] T. Rodrigues, P. Rosa, J. Cardoso: Mapping XML to Existing OWL Ontologies. In *Proceedings of the International Conference WWW/Internet*, pages 72-77, 2006.

[80] T. Strang and C. Linnhoff-Popien: A Context Modeling Survey. In *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp*, 2004.

[81] T.R. Gruber: A Translation Approach to Portable Ontology Specifications. KNOWLEDGE ACQUISITION, volume 5: pages 199-199, 1993.

[82] Twitter. https://twitter.com/, last accessed on 07 August 2012.

[83] V. Akman, and M. Surav: The use of Situation Theory in Context Modeling. Computational Intelligence, volume 13: pages 427-438, 1997.

[84] W. Mark: The Computer for the 21st Century. Scientific American, volume 265: pages 94-104, 1991.

[85] W3C. http://www.w3.org/, last accessed on 03 August 2012.

[86] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana: Web Services Description Language (WSDL) 1.1, 2001.

[87] Organization for the Advancement of Structured Information Standards. https://www.oasis-open.org/, last accessed on 07 August 2012.