THE UNIVERSITY
of ADELAIDE

Classifying and Clustering the Web of Things

Thesis submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

by

Sujith Samuel Mathew

Supervisors

A/Prof. Michael Sheng, A/Prof. Yacine Atif, and Prof. Zakaria Maamar

Faculty of Engineering, Computer, and Mathematical Sciences

The University of Adelaide, Australia

2013

ORIGINALITY STATEMENT

"I certify that this work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968. The author acknowledges that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the Web, via the University's digital research repository, the Library catalogue and also through web search engines, unless permission has been granted by the University to restrict access for a period of time."

Sujith Samuel Mathew
PhD Candidate
School of Computer Science
University of Adelaide
April, 2013

To Jesus, my Lord and Saviour, who walked this walk with me.

ACKNOWLEDGMENTS

This work has been challenging and would not have been possible if it wasn't for several people. Their support and encouragement cannot be expressed in words, but I acknowledge them here.

I would like to thank my supervisors, who relentlessly inspired me to pursue excellence. Dr. Michael Sheng taught me to always aim high and to not give up. Dr. Yacine Atif believed in me, encouraged me to run this race, and critically reviewed my work. Dr. Zakaria Maamar keenly refined my efforts and promptly corrected my work. Their experience, guidance, and positive attitude were pivotal for my study. I am deeply indebted to them.

I would like to thank my colleagues at the College of IT, UAE University for their support. A special thanks to Dr. Mohamed Adel Serhani, who faithfully and patiently reviewed my work.

Finally, I am indebted to my parents, my wife, and my two boys, for their constant support and prayers. I express my heartfelt gratitude to them for all their sacrifices, their love, and for standing by me every single day. This would have been impossible without them.

ABSTRACT

The *Web of Things* is emerging as a promising solution for realizing ubiquitous applications, where real-world things and people seamlessly connect and communicate. The challenges of integrating real-world things into the virtual environment of the Web have been the subject of much research recently. Such environments present a major challenge i.e. to retrieve combined environmental services and relevant information through prevalent means. Research in this direction is significantly challenging because of the growing number of heterogeneous things connected to the Web and the potential ubiquitous applications that would positively influence society, business, and industry. The substantial numbers of proprietary applications that integrate real-world things reveal the requirement for an open and scalable framework for including the plethora of things in a systematic and structured manner. Adequate solutions to manage the ever increasing number of things are paramount to induce flexibility, robustness, and usability in future ubiquitous applications.

In this dissertation, we propose a novel semantic structure to represent things on the Web. We classify things into an ontological structure based on required capabilities to participate in the Web. We introduce the *Ambient Space framework* to manage things in a scalable manner and use it to model the creation of *communities of things*. We describe the process of analyzing thing's semantic structure to cluster them into communities. We also discuss how things join these communities inherently to establish relationships with people on contemporary social networks. Finally, we present case-studies centered on environment sustainability which show applications where a number of things are managed using our framework in autonomous, heterogeneous, and dynamic environments. Our evaluations reveal the benefits of such applications, compared to conventional solutions, in support of sustainable environments.

Contents

List of Figures

List of Tables

List of Publications

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "The Web of Things - Challenges and Enabling Technologies", In Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence (N. Bessis, F. Xhafa, D. Varvarigou, R. Hill, and M. Li, eds.), Vol. 460 of Studies in Computational Intelligence, pp. 1-23. Springer Berlin Heidelberg. 2013.

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "Building Sustainable Parking Lots with the Web of Things", Personal and Ubiquitous Computing (PUC) journal, pp. 1-13, Springer-Verlag. June 2013.

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "A New Parking System based on the Web of Things", In Proceedings of the workshop on Computing and Networking for Internet of Things (ComNet-IoT), in conjunction with International Conference on Distributed Computing and Networking (ICDCN), Mumbai, India, January , 2013.

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "Towards an efficient sales pitch with the Web of Things", In the Tenth IEEE International Conference on eBusiness Engineering (ICEBE). Coventry, United Kingdom, 2013.

Mathew S.S. , "Managing Things in an Ambient Space", In the Ninth International Confer-

ence on Service Oriented Computing (ICSOC), pp. 226-232. Cyprus, 2012.

Maamar Z., Sheng Q.Z., Atif Y., **Mathew S.S.** and Boukadi K, "Towards an Approach for Weaving Preferences into Web Services Operation", Journal of Software. Vol. 7(7), pp. 1429-1439. 2012.

Trabelsi Z., Hayawi K., Al Braiki A. and **Mathew S.S.**, "Network Attacks and Defenses: A Hands-On Approach", Auerbach Publications (Taylor and Francis). 2012.

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "Web of Things: Description, Discovery and Integration", In International Conference on Internet of Things and Cyber, Physical and Social Computing (iThings/CPSCom), pp. 9-15. Dalian, China, 2011.
Citations: *6*

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "Ambient things on the Web", Journal of Ubiquitous Systems and Pervasive Networks (JUSPN). Vol. 1(1), pp. 1-8. 2010.
Citations: *2*

Atif Y., **Mathew S.S.**, "Ambient Learning Companion", in Proceedings of 2nd International Conference on Education and New Learning Technologies (EDULEARN10), pp. 4787-4791. Barcelona, Spain. 2010.

Atif Y., Serhani M.A., Campbell P. and **Mathew S.S.**, "Trusted Translation Services", In Collaborative Computing: Networking, Applications and Worksharing. Vol 10, pp. 778-791. Springer Berlin Heidelberg. 2009.

Chapter 1

Introduction

Nearly two decades ago, Mark Weiser envisioned *Ubiquitous Computing*, where computing power becomes invisibly integrated into the world around us and accessed through intelligent interfaces. He foresaw: "*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*" [1]. Today, we have moved leaps and bounds closer to this vision due to tremendous advances in information and computing technologies. These technologies are enabling a paradigm shift where there are increasingly more real-world things crossing the boundary from the physical world into the virtual world. In this virtual world, *things* and people meet, communicate, and cooperate with each other to enable businesses, industry, and society. The research community is focusing on some interesting innovations to accelerate the progress towards realizing Weiser's vision and some of these directions are the focus of this thesis. We begin by presenting the motivations of our work towards this vision.

1.1 Motivations

In the last two decades, the World Wide Web (*the Web*) has been a major catalyst to developments in society and technology. The Web has become the de-facto *platform* for exposing applications that manage businesses, connect people, and process information. Today, businesses thrive on Web applications that are distributed over heterogeneous platforms and across geographical boundaries. These heterogeneous platforms are integrated over the Web with interoperable components like Web services. With advances in information technologies and communication, would *the Web* be an ideal platform to realize ubiquitous applications [1], where people and real-world things seamlessly communicate? Indeed, the Web is evolving into the so-called *Web of Things* (WoT). The WoT is envisioned as the ubiquitous application environment where everyday *things* such as buildings, sidewalks, traffic lights and commodities are reachable, recognizable, readable, and even controllable [2, 3, 4].

Recent developments in the information and computing technologies world are enabling the realization of WoT vision. These advances include the falling cost of technology, the widespread integration of technologies like Radio Frequency Identification (RFID) and Wireless Sensor Networks (WSN), and the increasing availability of communication infrastructure like Wi-Fi and 3G. Moreover, advances in research are providing processing power and communication capabilities in increasingly smaller sizes allowing computational power to be embedded in the environment around us. The driving force behind these technology revolutions is the growth of the *Internet* as an economical, ubiquitous and efficient means of communication.

The ubiquity and usability of the Internet as a basic infrastructure for connecting people and applications has led whole cities to be accessible over wired and wireless Internet Protocol (IP) networks [5]. Extending the Internet to low powered and resource constrained things was

considered impractical. However, 6LoWPAN radically altered this status enabling efficient IPv6 communication for resource constrained devices. For the past few years, there has been a growing trend in using IPv6 addresses to uniquely identify physical things and research in this direction is termed as the *Internet of Things* (IoT) [6]. The IoT promises a communication infrastructure for real-world things to exchange states and functions with others on the Internet. Institutions such as the IP for Smart Objects alliance (IPSO) and the European Future Internet Initiative (EFII) are also accelerating this development to connect a variety of physical things into the Internet. The main intention is to propagate and manage the wide use of Internet as the common medium for communication and unleash hidden information in the physical world [7]. This hidden information is inherent in almost all environments, embedded within physical things like surveillance cameras, patient monitors, classroom lights, and parking sensors. However, an IoT infrastructure alone does not guarantee the success or usability of connected things. An application platform that integrates real-world things into virtual applications to enable business and industry is deemed necessary. This need for an application platform for the IoT has induced research activities towards WoT.

The Internet provides the necessary communication infrastructure for the Web, but it is the Web-based applications that are transforming society as a whole. Tim Berners-Lee commented on WoT; "*It isn't the documents which are actually interesting, it is the things they are about!*" in an indication to actually reach and operate on things directly via the Web [8]. The benefits of using the Web to build ubiquitous applications are many, three of the main benefits are:

- The Web is an open platform for application development and hence provides a high level of interoperability among resources.

- Web technologies and standards have matured over the years and are widely used. Hence

it is easy to reuse existing knowledge to build ubiquitous applications and services.

- The Web provides a scalable architecture for distributed systems, which is ideal for reaching real-world things.

The primary choice of a universal platform to build applications that use things would be, the Web. The contextual scope of WoT exceeds the boundary of today's Web as it is poised to enable real-world things to be accessed via a Web browser. This transformation of the Web promises to enhance people's personal lives and enable enterprises to tap into new business opportunities through efficient supply chains, novel marketing, and improved service monitoring. The emergence of the WoT represents a paradigm shift in how Web users behold and interact with the information hidden in real-world things [4]. With cars that notify their owners about tires that need to be changed and sports companies connecting their training shoes to the Web to compare performances [9], the opportunity to transform the value of products with augmented digital services boosts product experiences and hence consumer's delight. With things communicating with other things on the Web and participating in the decision-making process, novel applications are envisioned to enhance business and society.

Current research trends to realize the WoT vision has provided promising results [10]. Though the WoT vision promises attractive opportunities, there are some compelling issues that are yet to be addressed. While real-world things are expected to become essential components of Web-based applications there is a lack of standards on how to typify things. With the heterogeneity of potential things that are to be accessed on the Web, classifying things into common types is a huge challenge. Moreover, identifying the capabilities that allow a thing to participate in ubiquitous applications on the Web is also important as it is required to provide standard integration interfaces. Another issue is the enormous number of candidate things for

the WoT. The manual adaptation of things, one at a time into the WoT is impractical. A scalable framework that automates the process of subsuming things into Web application platforms is a necessity and a significant challenge to accomplish. In retrospect, our research is focused towards three directions that address issues listed in the next section. Firstly, we provide a semantic classification of candidate things. Next, we suggest a framework to manage these things within scaled down spaces where we identify repeating patterns of things and suggest the virtual representation of things within their spatial confinement. Finally, we model a community of things and discuss how things are subsumed into these communities. In the next sections, we provide an overview of the research issues that we focused on and our respective contributions.

1.2 Research Issues

Real-world things are heterogeneous and have proprietary applications. The benefits of using things as disparate entities are limited when compared to things functioning together in a community, within a common semantic context. For example, consider a scenario where a Web user is searching online for shops that sell custom made watches, located within the proximity of his home, and where there are available parking spots for the handicapped. This scenario requires a clear knowledge representation of the physical world and the knowledge should enable appropriate reasoning capabilities. Moreover, the representation should be both machine-readable and intelligible to people. Realizing such scenarios with the WoT necessitates the creation of a framework that seamlessly manages, controls, and communicates with real-world things from Web applications. Any approach that is adopted to realize WoT vision should retain the openness of the Web rather than provide proprietary solutions. The main challenge is to create a model that helps architects easily abstract and incorporate things into the Web so

as to design applications around them. Of the many issues that this new and exciting direction of WoT brings, some of the research issues that are addressed in this thesis are:

(a) Lack of a classification of candidate things for the WoT

(b) Lack of a scalable framework to subsume the numerous things into the WoT

(c) Lack of a model to propagate things to people on the WoT

(d) Dearth of case-studies that evaluate the benefits of applying the new paradigm of WoT

These issues are foundational and important to the success of the WoT's vision. We present our work as a comprehensive study that is directed towards solving these issues. We have focused our contributions, listed in the next section, to address these issues. These issues are described here while highlighting their research relevance.

(a) Lack of a classification of candidate things for the WoT

With a plethora of things becoming ubiquitous on the Web, the lack of integration and open communication between things on the Web, isolates important information and magnifies the existing problem of too much data and not enough information. There is a need to provide a relevant semantic specification of potential things that would share information on the Web. There is no relevant classification of real-world things to provide a semantic specification. Such a classification harnesses the complexity of modeling context-aware scenarios with a common approach to interface with a wide range of things. However, such a classification that encompasses all things with potential information to share on the Web is impossible, unless the classification is based on the minimalistic capabilities of candidate things required to participate in the Web. The *minimalistic capabilities* indicate the minimum requirements that need to be

satisfied for things to have an online presence on the Web. There is an urgent requirement for research contributions in this direction to reduce the random and ad-hoc solutions that are being adopted.

(b) Lack of a scalable framework to subsume the numerous things into the WoT

Earlier initiatives to realize WoT expose a thing's functionality either directly from the thing or from a proxy gateway [11]. Adapting these solutions on a large scale is a difficult proposition merely because of the enormity and heterogeneity of real-world things. Manually incorporating the vast number of things one at a time into the Web is an impossible task. Moreover, any approach to overcome this issue of integrating things into the Web on a large scale should encompass things and their operations for Web users in a scalable pattern. This scalable pattern is possible as *things* are confined into specific hierarchical spatial limits in the real-world, for e.g., a fixed projector (*thing*) in a classroom, is also located in a building that houses many such classrooms, and is also part of the campus where many such buildings are located. Hence, a representation of things using a pattern of spatial limits is possible and also reduces the efforts of integrating heterogeneous things. There is no in-depth study of an approach which partitions things considering spatial patterns which would help to scope and scale the representation of things for the WoT.

(c) Lack of a model to propagate things to people on the WoT

As things become available in the virtual world, they are expected to exhibit social attitudes and co-exist with people and other things on the Web. If all things start initiating connections with people and other things on the Web, a number of issues like breach of privacy, security, and increased network traffic are anticipated. Properties that govern the dynamics of things on the Web need to be elicited as well as the roles of people who access and control these things.

There is no significant work done to model the propagation of real-world things on the Web or to describe things as a potential contributing community on the Web.

(d) Dearth of case-studies that evaluate the benefits of applying the new paradigm of WoT

The benefits of adopting real-world things into the Web are being studied. Yet there is a dearth of case-studies where this new paradigm is applied to different real-world scenarios to study the possible benefits and challenges. Such case-studies are important to realize the effects that WoT brings to society.

1.3 Contributions

Focusing on the above research issues, the main research contribution in this thesis are:

(a) A classification model of candidate things for the WoT

The classification of candidate things increases and eases the deployment of things into the WoT as Web resources which are accessed through dedicated proxies, for instance Web services. Such a classification contributes to integrate things into the Web and help build semantics to adapt potential things into an application context. A classification also supports WoT architects to have a common referential taxonomy of candidate things on the Web. We propose a classification approach in which things are annotated with semantic metadata to increase interoperability as well as provide contextual information essential for situational knowledge.

Our initial attempt to model things and their properties on the Web [12] developed into an ontological structure used to classify and capture the semantics of real-world things [13]. The approach is capable of abstracting the required capabilities of a thing that has relevant

information to be shared on the Web. The ontology is based on the required minimal capabilities of candidate things to participate in the Web and thereby helps identify the missing capabilities that need to be augmented so that a particular thing is adapted into the Web. Research contributions towards the classification of things for the WoT are still in their preliminary stages. Our contribution is important as it plays a necessary and significant role in defining semantic specification of things while creating applications that subsume real-world things.

(b) Ambient Space: a scalable framework to automatically subsume heterogeneous things into the WoT

Millions of things are confined into hierarchical spatial limits in the real-world and are inherently proprietary i.e., they are owned by individuals or institutions. Coupled with a classification model, a framework to automatically integrate things into the Web is proposed. The proliferation of things into the Web is channeled hierarchically via an approach that addresses the Web enablement, and interoperability of things within scaled down spaces. While some solutions are being studied and deployed [14, 15], our framework takes advantage of the fact that things are duplicated within spatial patterns. We target the problem of realizing WoT by defining Ambient Spaces (AS) to be homogeneous virtual spaces that encompass Web-enabled things within an application context [16, 17]. The integration of these spaces enable the realization of the WoT in a scalable manner.

(c) A model for the creating a community of things to be represented on the WoT

The adoption of a social networking platform to integrate real-world things was proposed earlier [18]. However, the approach is restricted to privacy preservation of things and their users over existing social networking platforms. Moreover, the approach does not look at things as autonomous entities of the social network and substantial human intervention is expected. We

suggest a model that extends AS to realize the *Social Web of Things* (SWoT), where things are participating entities with people. The SWoT platform enables things to be identified, grouped, and transacted with. Based on relationships that things have on the SWoT, new ubiquitous applications are enabled through which things inherently market themselves, share and compare themselves over the Web, and things reveal their social learning value on a smart campus [19, 20, 21]. We model the process of automatically adopting things into the SWoT so as to create communities where people and things co-exist.

(d) Case-studies that evaluate the benefits of adapting things into the WoT

We present case-studies that explain the benefits of adapting things into the Web. We consider real-world scenarios where the WoT help in build sustainable environments like, classrooms, and parking lots. We evaluate the benefits and discuss how our framework is used for implementing these scenarios.

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows. In Chapter 2, we introduce fundamental concepts related to WoT, enabling technologies and present the state-of-the-art in research efforts. We discuss many salient technologies that are driving the success of the WoT, such as embedded systems, Web services, and the networking of things. In this chapter, we present the state-of-the-art in WoT research and discuss existing technologies and platforms that support the WoT. We present approaches to create Web-based things and discuss some proposed WoT architectures. We present some research directions, trends and challenges that are presently driving the WoT into new horizons. In contrast to our survey of current research trends, we reveal the focus of our work.

In Chapter 3, we describe our proposed classification of things. We highlight our classification approach and discuss important properties of candidate things for the WoT. The capability based classification that we propose is discussed in context of a technology-enabled learning environment to motivate the use of the classification. Our capability based classification of things lays a foundation to integrate different types of things into the WoT. The classification identifies things to be *Web Smart* when augmented with the necessary capabilities to participate in the WoT. We introduce the *Web Object Metadata* (WOM) that describes the semantic information related to the capabilities of Web Smart things. We also introduce the concept of *Ambient Spaces* (AS) which provides the virtual representation of a spatial environment composed of Web Smart things.

In Chapter 4, we reveal the *Ambient Space* framework which expounds on the virtual depiction of *Web Smart* things within scaled down spatial limits. In particular, we discuss how we augment a thing with the necessary capabilities so that it can participate on the Web. We also discuss the software components that enable the representation of real-world things on the Web. We present the AS framework's function as a gateway for Web applications to connect to Web Smart things. We also extend the WOM representation of things to provide more thematic representation for Web Smart things. In this chapter we also introduce the need for developing a community of things for studying the interactions between things and with people. We introduce our approach of using AS to cluster a large number of everyday things and create the *Social Web of Things* (SWoT), a platform to bridge the physical world with people on the Social Web.

In Chapter 5, we model the composition of things into AS and show how this is used to connect to people. This model lays the foundation for creating *communities of things* and describes how Web Smart things are grouped. The communities enable thing to thing as well

as things to people interactions. We describe the process of using our proposed WOM semantic structure to understand the social attributes of Web Smart things. We discuss the creation of two types of communities, the first is to automatically subsume Web Smart things into the SWoT and the second is to suggest a suitable associations with people on the Social Web. Thus, by using our AS framework, we introduce the SWoT as a platform where communities of things are modeled.

In Chapter 6, we describe two case studies promoting the WoT as a means to curtail energy sustainability issues. We describe the implementation of the Ambient Classroom, and the Ambient Parking Lot. We discuss the respective implementation and evaluate the benefits of the approach. Finally, in Chapter 7, we provide concluding remarks of this dissertation and discuss directions for future research.

Chapter 2

Background and Related Work

The research directions of the Internet of Things (IoT) have recently triggered the research community to adopt the interoperability of the Web as an application platform for integrating *things* on the Internet. Harnessing physical things into the virtual world using Web standards is also enriching the arena of conventional Web services to unleash data and functions of real-world things as service providers and consumers on the Internet. This evolution of the Web as a highly interoperable application platform for connecting real-world things is fueling the research area of the Web of Things (WoT). Current research on WoT is a catalyst for the realization of IoT, opening up novel and innovative possibilities.

In this chapter, we discuss the state-of-the-art in WoT research. This research field is quite new and the contributions from academia are still in their early stages. The research community is focusing on defining concepts, and working principles. In the first three sections, we focus on the fundamental principles enabling technologies that are driving this research, and technologies that help build applications for future Ambient Spaces (AS) with the WoT. Next, we present related work and active research directions where we discuss two proposed architectures before summarizing our review of the state-of-the-art.

2.1 The Web – An Application Platform for Real-World Things

The recent technology advances in the area of communication and embedded systems are enabling the realization of the WoT vision. Internet-enabled mobile phones, televisions, refrigerators, bathroom scales, and pacemakers are paving the way for everyday objects around us such as sidewalks, traffic lights, and other commodities to be identifiable, readable, recognizable, addressable, and even controllable via the Internet (TCP/IP) [3]. These trends are motivating researchers to study, model, design, and implement novel applications, which promote interoperability of heterogeneous physical things on the Internet.

However, networking alone does not enable the success or usability of IoT. Today, business and industry depend on applications that are built on Web architecture and Web services interoperability. While the IoT promises to provide the necessary networking and communication protocols to access real-world things, a parallel and recent research trend evaluates the Web as a platform for building applications that integrate real-world things on the Internet. The spread of the Internet provides the networking infrastructure for the use of real-world things, while research in the WoT provides the application and service layer for real-world things to interoperate over HTTP [8, 22].

The open platform of the Web has engaged application providers and users to generate and exchange information securely in various formats. Existing network infrastructures like Wi-Fi, Ethernet, and 3G leverage the use of Web technologies which facilitate the availability of tools, frameworks and information for developing Web based applications. These factors reduce the learning curve when using Web technology for extending the scope of Web applications to real-world things.

A high-level illustration to realize a Web-enabled thing is illustrated in Figure 2.1 [13].

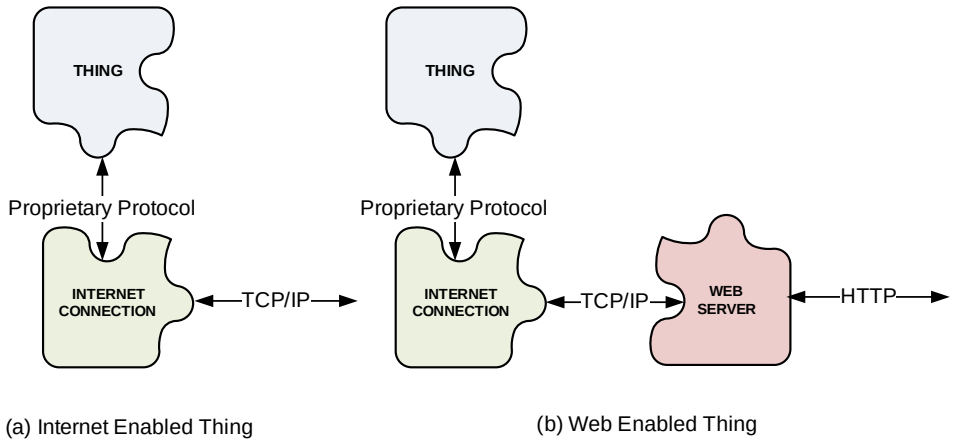


Figure 2.1: Realizing Web-Enabled Things

Augmenting a thing with Internet connection (i.e., IP address) ensures its accessibility over the Internet and results in an Internet-enabled thing, as shown in Figure 2.1 (a). When a thing is Internet-enabled and is, also, connected to a Web server it becomes Web-enabled, as shown in Figure 2.1 (b). The Web server may be either embedded or on a separate system. Advances in embedded technology have made it possible for realizing tiny Web servers to be embedded in everyday objects [15].

The wide proliferation of the Web has paved the path for Web services to be an indispensable technology, for interoperable applications. Extending service computing to Web-enabled things makes it possible for real-world things and their operations to be abstracted as Web services. In doing so, real-world things can offer their functionalities via SOAP-based Web services (WS-*) or RESTful APIs [11]. These services directly expose operations of real-world things to enterprise applications or even to other things and possibly involve a dynamic composition process at runtime.

The idea of integrating and networking real-world things to communicate with each other

has been prevalent for many years. Various technologies and standards have been proposed and some are already in use today. Many of these technologies, both software and hardware, have paved the path for the birth of WoT. The primary challenge for the WoT is to address the interoperability of the tirade of things that surround us today using Web standards and protocols. We present some of the early attempts that enabled communication of things on the Web and also look at technologies that are driving the research on the WoT.

Several industry alliances and standards have been defined like UPnP, DLNA, SLP, and Zeroconf to facilitate interoperability of real-world things. Each of these standards has individually been successful in enabling devices to communicate with each other. However, the availability of many standards creates an even more complicated challenge, which is interoperability between standards. With the diversity of things, their vendor specific properties, and the variety of services they provide, the use of standard-based interoperability approach is a never-ending process [23]. We briefly present few of these that have made considerable impact.

An early and noteworthy example is Zeroconf [24], which is a technique to provide reliable networking without any manual configuration and support services. Zeroconf is offered using several implementations. Probably the most known is Apple's Bonjour which is used to discover shared printers and for streaming media to Apple TV devices.

Universal Plug and Play (UPnP[1]) is a collection of networking protocols promoted by the UPnP forum and mainly used for devices to discover each other and establish connections over a common network. UPnP is based on protocols and standards like HTTP, HTTPU (i.e., HTTP over UDP), UDP, TCP, and IP. UPnP supports various services like discovery, description, control, event notification, and presentation. However, this cannot be widely used because in some embedded devices that are resource constrained it is impossible to host a large number of

[1] http://www.upnp.org/

these protocols [25, 26].

Conventional middleware technologies such as CORBA and Java-RMI have also been used to realize device interoperability by standardizing common protocols. The Jini project created a middleware where each device is loosely coupled as a Java object. Allard et al. introduced a framework to combine Jini with UPnP [27].

2.2 Technologies Driving the Web of Things

Alliances such as the IP for Smart Objects (IPSO[2]) and the European Future Internet Initiative (EFII[3]) have accelerated the trend to connect a variety of physical things to the Internet, with the intention of propagating the wide use of Internet as the common medium for communication. Also, with the ever-decreasing size of embedded systems and related software footprint, it has become possible to directly integrate Web servers into many appliances [15]. The challenges involved in using the Web as a platform to create applications that integrate real-world things into the Web has attracted attention both in academia and businesses [11, 22, 28]. Service Oriented Architecture (SOA) has proved to be a promising for integrating devices into a business IT network [29]. This has in turn led to the propagation of using Web services for the interoperability of real-world things.

2.2.1 Web Services

The use of Web services is paramount in establishing interoperable distributed systems on the Web [30]. This indispensable technology is extended to WoT to allow data or functionalities of real-world things to be abstracted and exposed as services on the Web. There are two

[2]http://www.ipso-alliance.org/
[3]http://www.future-internet.eu/

major classes that regulate the proliferation of Web services, RESTful Web services, and WS-* protocols stack [31, 32].

(a) RESTful Web Services

RESTful Web services are based on Representational State Transfer (REST) [32], an architectural style that considers every participating component to be an addressable resource. The state of each resource is essentially captured in a document referred to as a representation. The Web was built upon this architecture where each resource is a Web page identified by a URI (Uniform Resource Identifier). REST is lightweight, simple, loosely coupled, flexible and seamlessly integrates into the Web using HTTP as an application protocol. RESTful Web services are based on the following four concepts [33].

- Resource identification: Resources are identified through URIs which provide a unique and global presence for service discovery.
- Uniform interface: Resources are accessed using the standard HTTP verbs, GET, PUT, POST, and DELETE.
- Stateful interactions through hyperlinks: Interactions with resources are stateless and self-contained. To circumvent this, state can be embedded into response messages to point to valid future states of the interaction.
- Self-descriptive messages: Resources are decoupled from their representation so that their content can be accessed in a variety of formats like Extensible Markup Language (XML), Java Script Object Notation (JSON), plain text or YAML Ain't Markup Language (YAML).

These concepts make REST an ideal platform to expose services of real-world thing on the Web [14, 34]. IETF has constituted a working group called Constrained RESTful Environ-

ments (CoRE[4]), to develop a framework for resource oriented applications intended to run on constrained IP networks like WSN (Wireless Sensor Networks). This includes sensor based applications like temperature sensors, light switches, and power meters, applications to manage actuators like light switches, heating controllers, and door locks, and applications that manage devices. As part of the framework for building these applications, CoRE plans to define Constrained Application Protocol (CoAP) for the manipulation of resources on devices. CoAP is to be designed for interaction between devices on traditional IP networks, on constrained IP networks, and between constrained networks on the Internet. CoAP is a work in progress but some implementations are already emerging such as the Firefox extension Copper[5], Tiny OS[6], and libcoap[7]. The CoRE group has proposed the following features for the CoAP:

- RESTful design minimizing the complexity of mapping with HTTP,
- Low header overhead and parsing complexity,
- URI and content-type support,
- Discovery of resources provided by CoAP services,
- Subscription for resources and resulting push notifications, and
- Caching mechanism.

A REST-based approach was also used to define the TinyREST architecture [35], which introduces a new HTTP method, SUBSCRIBE, that enables clients to register their interests to device level services. The pREST or pico-REST [36] proposed by Drytkiewicz et al. is an

[4]http://datatracker.ietf.org/wg/core/charter/
[5]https://addons.mozilla.org/en-US/firefox/addon/copper-270430/
[6]http://docs.tinyos.net/tinywiki/index.php/CoAP
[7]http://libcoap.sourceforge.net/

access protocol, which brings the REST concept to the device level. The emphasis is on the abstraction of data and services as resources in sensor networks.

Though the work on REST-based services continues to encompass many WoT applications, in enterprise applications like banking or stock markets, where Quality of Service (QoS) levels are stricter, a more tightly coupled service paradigm like WS-* would be more ideal [33].

(b) WS-* Services

The key enabling technologies of WS-* are Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description Discovery and Integration (UDDI). Unlike RESTful services that uses HTTP as an application layer protocol WS-* services use HTTP as a transport protocol over which SOAP messages are sent. SOAP is an XML-based protocol defining a message based architecture and format [37]. SOAP messages are contained in an XML element called envelop, which contains two sub-elements called header and body. The header features application specific information and also QoS parameters while the body contains the content of the message intended for the endpoint. WSDL is also an XML-based language describing the syntactic communication for message exchange between endpoints [38]. SOAP messages and their structures are described by WSDL. Service endpoints are addressed either on the transport level, i.e., with URIs for SOAP/HTTP-bound endpoints, or on the messaging level via WS-Addressing [39]. UDDI is also an XML-based registry framework for discovering Web services [40]. WS-* is stateless but WS-Resource Framework describes the interaction of Web services where the state needs to be maintained [41]. The WS-* technology stack also covers many QoS features required to ensure the interoperability of distributed systems [42]. While this technology has been mature and stable for several years now, it has not achieved the expected widespread acceptance in the industry.

Research on integrating WS-* into real-world things has been active for many years. It is argued that WS-* is more suitable for business scenarios rather than resource-constrained things, since WS-* is heavy [33]. Some early research has focused on the use of WS-* to adapt to resource-constrained embedded systems. A Service-Oriented Device Architecture (SODA) [43] is proposed to integrate real-world things into today's businesses. The architecture exposes device functionalities as abstract business services. Pintus et al. [44] describe an SOA framework where real-world things are described using WSDL. The communications between these things are modeled as service orchestrations using BPEL (Business Process Execution Language). Projects such as SIRENA (Service Infrastructure for Real-time Embedded Networked Applications) [45] and SOCRADES (Service Oriented Cross-layer Infrastructure for Distributed Smart Embedded devices) [46] adapt an approach for networks with smart things. Many of these projects aim at making embedded systems directly accessible via Web services, i.e. by using the Devices Profile for Web Services (DPWS) stack [47, 48]. However, while this approach is suitable for some devices, many real-world things are also small and lightweight, and cannot manage the overhead introduced by the WS-* technologies, and consequently, require more efficient SOA implementations.

(c) Merging WS-* and RESTful Web Services

Since the goal of both paradigms is the same, there have been attempts of merging the two service architectures for WoT to make up for the disadvantages of each [14, 34]. The first direction is to have the RESTful architecture merged into the existing WS-* service architecture, which requires the implementation of alternate data model. Having two data models for the same service results in a very complex architecture and hence is not desirable. On the contrary the other direction is to have a software module that translates RESTful requests into WS-* requests [34]. This provides a structured approach ensuring a REST centric architecture while

maintaining the robustness of WS-* approach, but this affects the performance of RESTful APIs.

2.2.2 Embedded Web Servers

Embedded Web servers are an indispensable component for the long term adaptation and spread of WoT. These tiny Web servers enable communication between real-world things over the simple and widely used HTTP protocol. Researchers have successfully embedded tiny Web servers into resource-constrained things like sensors and smart cards, making Web-enabled things a reality [49, 50]. Filibeli et al. describe an embedded Web server based home network system where Ethernut[8] based Web servers are embedded into home appliances and are controlled via the Web [51]. Priyantha et al. propose the interoperability of sensor nodes by implementing Web servers on the nodes [52]. Today, open source tiny Web server modules like FlyPort from OpenPicus[9] are readily available off the shelf. These modules are 35X48 mm in dimension, comes with an integrated 802.11 b/g/n Wi-Fi interface module, and a 16-bit processor. The internal flash of 256KB has been demonstrated to be sufficient for different Web applications that access and control real-world things like actuators and sensors. With the reducing physical size of hardware components and its software footprint many of the new appliances in the market, like Smart TVs and refrigerators are expected to come with embedded servers. Mathew et. al. suggest the creation of a new parking system based on the WoT using embedded Web servers to abstract parking services in a parking spot on the Web [21]. With computing power disappearing into the surroundings, embedded Web servers will be a common feature in many of the physical things we interact with, enabling intelligent WoT applications.

[8] http://www.ethernut.de/
[9] http://www.openpicus.com/

2.2.3 Networking Real-world Things

While there are several protocols that enable the physical connection to real-world things, like ZigBee, and Bluetooth, we discuss RFID and 6LoWPAN which have created sufficient impact to the possibility of realizing WoT.

(a) Radio Frequency Identification (RFID)

Research and development on the IoT gained its recent momentum from the wide spread deployment and use of RFID (Radio Frequency IDentification) technology. As illustrated in Figure 2.2 [16], information of a tagged thing like a can of beans, become available on the Internet when it is tagged with an RFID tag. An RFID reader reads the information on a tag and an Application server uses the reader's access protocol to access the information of the tags and expose the information on the Internet.

It is important to clarify the term 'things' while considering WoT or IoT applications and systems. While the term devices or appliances are often used to refer to things under consideration, some tend to believe that almost every "thing" can be included. The rationale for the latter is that information of any tagged thing is accessible on the Internet. The former argues that only connected devices on the Internet fall into the category of things, for example, in Figure 2.2 the application server is a connected device. Today, RFID readers are Internet enabled, and directly access the Internet without the use of an intermediate server and has been elevated into the category of connected devices or connected things. The Perci Framework enables mobile interaction with real-world objects where, the architecture uses Web services for Physical Mobile Interactions (PMI) [53]. Tagged physical objects are read by mobile devices in different interactive modes to gather specific information. The framework maps tagged objects onto different Web service parameters.

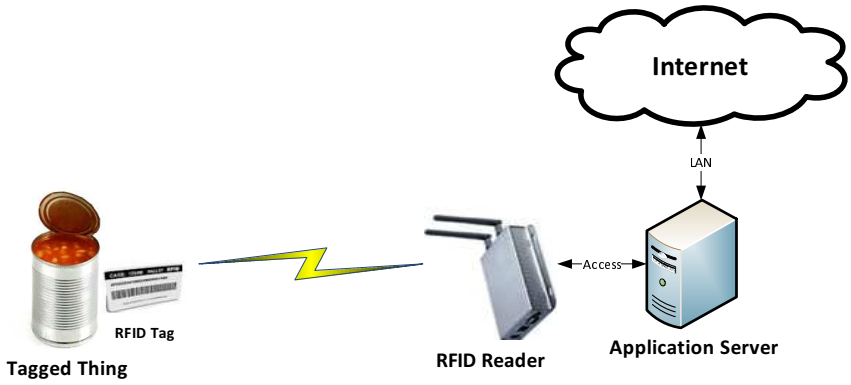


Figure 2.2: Internet Enabling Things with RFID

With more things tagged and accessible on the Internet, a major challenge is the management of large scale deployment of tagged things. These things are to be tracked and their state (e.g. location) is to be continuously updated. Discovering information like, current state, past state, as well as estimating the future states are important for the success of businesses that use them. The *traceability of products* is essential to a wide range of important business applications such as manufacturing control, logistics, distribution, product recalls, and anti-counterfeiting. Research in this direction has inspired the *PeerTrack* platform which has several traceability applications built-in for mobile asset management and supply chain management for tagged things [54]. A system for mobile asset management is deployed at the International Linen Service Pty Ltd. (ILS), a company that provides a suite of linen services for over two hundred customers in South Australia. In this system, trolleys are reusable containers for linens and are attached with RFID tags. They are transported among locations and are detected by RFID readers on arrival at a delivery location. The system developed on the PeerTrack platform offers an automated tracking and tracing service with the capability to monitor and control logistical operations in real-time for over three hundred different locations. Also, a visual monitoring

tool is deployed at each customer's site, together with the P2P services. The system traces and tracks the mobility of the trolleys, and ensures real-time inventory monitoring.

Web technology has matured over the years to successfully manage the interoperability of systems distributed on a worldwide scale. The adaptation of tagged things and their respective information to the Web poses many challenges and research questions. The advances in WoT technology intends to provide a scalable, simple and foolproof platform for the management of the ever-increasing presence of tagged things.

(b) 6LoWPAN

Interoperability of things would only be possible on a scalable, accessible, and end-to-end communication infrastructure. To enable embedded Web server on devices, they must first be connected to the Internet (Figure 2.1). Things with inherent information, that has to participate in a Web based application, are of various types and capabilities. The IP protocol stack should be adapted for devices with limited resources, like storage capacity and battery power. 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) [6] was launched by IETF which defines mechanisms that allow IPv6 packets to be sent to and received between resource constrained devices usually by adopting low-power radio communication protocols like IEEE 802.15.4.

A layer-wise comparison between the TCP/IP and the 6LoWPAN stacks is shown in Figure 2.3. Zeng et al. note that the necessity of an adaptation layer in the 6LoWPAN stack is mainly to fit one IP packet within one 802.15.4 MAC (Layer 2) frame [55]. The adaptation layer also manages header compression, packet fragmentation, reassembling and edge routing. 6LoWPAN is ideally implemented using User Datagram Protocol (UDP) as transport protocol which ensures efficient traffic because UDP has no overhead for opening, maintaining, and closing connections. Moreover, there is no retransmission of lost packets in UDP. For things

Figure 2.3: Comparison between TCP/IP and 6LoWPAN protocol stacks

that are resource constrained, the use of HTTP over UDP enables the creation of robust Web applications. 6LoWPAN also makes things on the Internet addressable at the IP layer.

2.3 Building Web of Things Applications

Existing tools, techniques, languages and models for building Web applications can be applied to the WoT. However, for applications on resource constrained things where ad-hoc and event driven interactions are necessary, the challenges need to be understood and some techniques need to be adopted. Some of these key techniques are presented here.

2.3.1 AJAX Approach

The AJAX (Asynchronous Java Script and XML) technique for Web application development creates highly interactive and efficient applications with efficient workload distribution between the client and the server modules [56]. The behavior of an AJAX Web application can be separated into two phases: loading phase and running phase [49]. In the loading phase the client collects files that contain style, content, and application code, while in the running phase, the

client executes the application code from the first phase, and interacts with the server by sending asynchronous requests, allowing the Web page to update itself dynamically. Experimental results show that during the first phase, numerous large-sized content are served and during the second phase smaller size content is sent to the client [49]. AJAX uses the browser to handle the workload and relieves the Web server. This is an interesting technique for the WoT applications, where the embedded Web servers have lesser resources when compared to the clients.

2.3.2 Mashup Approach

Mashups have eased the process of developing WoT applications. This new technique introduces a paradigm shift in the creation of Web applications by using Web 2.0 techniques and composing various Web services to achieve the application goals. Unlike portal-based solutions, the content aggregation for mashup applications occurs either on the client or on the server. This is beneficial for the WoT applications, since it is possible that the clients are more powerful computational systems than servers. In WoT, all the real-world things are abstracted as Web resources, which are addressable, searchable, and accessible on the Web. Guinard et al. [11], present two representative mashup styles, physical-virtual mashup and physical-physical mashup. For the former, the Energy Visualizer application was developed which has a Web interface to monitor power consumption and to control different home appliances. For the latter, to present the composition of services offered by things, the Ambient Meter was implemented on Sun SPOTs, which polls predefined URLs to get the energy consumption of devices in a room using HTTP based requests and responses.

2.3.3 Event Driven Approach

The limitation of resources on embedded Web servers is a challenge that needs to be considered when creating applications for the WoT. Hence, a feature that is important to WoT application is the need to push data from server to client, based on events instead of the client continuously requesting to pull data from server. Event driven approaches are efficient to implement stateless behaviors and hence software designed for embedded systems ideally use event-driven approaches to manage resource constrained hardware.

Comet allows a Web application to push data from the Web server to the client [57]. Comet is an umbrella term for techniques which are used for this. Interaction and implementation methods are of two types: Streaming and Long Polling. In Long Polling, the client polls the server for an event by sending requests and stops only when the server is ready to send some data. After it gets a response the client will restart a Long Polling request for the next event. In Streaming, the client opens a persistent connection from the browser to the server for all events. Neither side closes the connection and in the absence of events servers periodically send dummy data to keep the connection active. While these workaround methods are in practice they pose two drawbacks, they generate unnecessary traffic and they are extremely resource demanding on the Web servers. While Comet is better in data consistency and managing traffic workload, it has scalability issues. More recently, Web Sockets were proposed which use full duplex communication with a single TCP/IP socket connection. It is accessible from any compliant browser using simple Java Script API. The increasing support for HTML5 in Web and Mobile Web browsers makes it a very good candidate for pushing data for the WoT. Since WebSockets protocol has an initial handshake followed by message framing, layered over TCP, they can be implemented in a straightforward manner on any platform supporting TCP/IP [58].

2.4 Research Directions

Here, we consider the ongoing efforts to realize an architecture for the WoT and discuss related work. The proposed WoT architectures take into consideration the methods for integrating things on the Web while maintaining the interoperability and distributed nature of the Web. The use of Web services (WS-* and RESTful) is fundamental to any WoT architecture. We look at two such layered architectures used to expose things on the Web as services.

2.4.1 Integrating Things on the Web

As illustrated in Figure 2.1 (b), the requirement of Web-enabling a thing is to expose its operations as HTTP URLs. This can be achieved through, (1) Direct Integration: augmenting the Web server to the thing, or (2) Indirect Integration: expose operations of a thing through a proxy Web server [11].

(a) Direct Integration: Augmenting a Web Server

A thing is Internet-enabled and Web-enabled by augmenting it with the capability to communicate over TCP/IP and HTTP respectively. An embedded Web server exposes the thing's operations via URLs. When using RESTful APIs, a thing's operations are exposed through standard HTTP operations like GET or POST. Ostermaier et al. present a prototype using Wi-Fi modules for connecting things directly to the Web [15]. They enabled association of sensors, actuators, and things with each other through the Web. Guinard and Trifa present the direct integration of Sun SPOT with a Web server [11]. They implement the embedded server directly on the Sun SPOTs nodes. The server supports the four main operations of the HTTP protocol i.e., GET, POST, PUT, and DELETE, making the Sun SPOT Web-enabled. Akribopoulos et al. introduce a Web service-oriented architecture for integrating small pro-

grammable objects in the WoT [59]. Using Sun SPOT applications and sensor data exposed through Web services, they present use-cases for building automation and remote monitoring.

(b) Indirect Integration: Using a Proxy Web Server

The process of indirect integration involves a gateway or proxy server between the thing and the Web. A thing could have potential information that needs to be made available on the Web but does not have the necessary capabilities to communicate over TCP/IP or HTTP. The minimum requirement for such things to be an eligible candidate for the WoT is that they must be uniquely identifiable within a particular context. Mathew et al. refer to such a thing as a *Core* thing and examples of such things would be pallets, medicine bottles, or shoes, which can be identified uniquely on the Web using an identification system like RFID or Barcode (Figure 2.2) [17]. The information of Core things are made available to the Web through the proxy or gateway server, abstracting the proprietary protocol with uniform Web-based APIs.

2.4.2 SOCRADES Integration Architecture (SIA)

The SOCRADES Integration Architecture (SIA) describes the use of both WS-* and RESTful services to access devices from enterprise applications [46, 60]. The architecture includes a local/on-premise subsystem and a central or remote or cloud subsystem. A high-level illustration of the main layers of SIA with the modules of the local subsystem which interfaces with the devices is shown in Figure 2.4 [46]. SIA enables the integration of services of real-world things running on embedded devices with enterprise services. WS-* Web service standards constitute the main communication method used by the components of enterprise-level applications. This enables business applications to access real-time data from a range of devices through abstract interface based on Web services. SIA, also, supports RESTful services to be able to communicate

with emerging Web 2.0 services. This enables any networked device that is connected to the SIA to directly participate in business processes.

Figure 2.4: Main layers of SOCRADES Integration Architecture (SIA)

The Local/On Premise subsystem features a Local Discovery Unit (LDU) connected to devices seen on a LAN, and the Central subsystem (anywhere on the network) hosts enterprise-level applications. The LDU module scans the local network for devices and reports their connecting and disconnecting to the central system. It acts as an intermediary that provides uniform access to different classes of devices through a set of platform-dependent plugins. In the local subsystem at the Devices Layer there are several embedded devices that run various services. The legacy devices would require a Gateway to expose their proprietary functionalities.

SIA is able to interact with devices using several communication protocols, such as DPWS (Device Profile for Web Services), OPC-UA, and REST. The details of each layer and their functionalities are provided in [46].

2.4.3 The Web of Things Application Architecture

The WoT application architecture presented by Guinard enables the integration of things with services on the Web and facilitates the creation of Web based applications that operate on real-world things [34]. The primary goals of the architecture is to enable rapid prototyping of applications that integrate real-world things for developers, to offer direct Web based access to things to Web users, and to offer lightweight access to thing's data which would enable the data to be consumed by other things or software.

In the proposed layered architecture, each layer does not hide the lower layers but rather provides a hierarchy of separate abstractions to connect things as first class citizens of the Web. Based on requirements, an application can be built on top of each layer or a combination of layers. The various layers and their roles are illustrated in Figure 2.5 [34], we present a summary of these layers here; a detailed representation is given in [34].

Figure 2.5: Application Architecture for the Web of Things

The Accessibility layer focuses on providing a standard access to all kinds of connected things. This layer exposes things as RESTful Things using resource oriented architecture. REST [32] is used as a universal technique to interact with real-world things and four steps are suggested to Web-enable things:

1. Design the resources by identifying the functionality or services of a thing and organizing their interactions,
2. Design the representations and decide on how these will be provided,
3. Design the interfaces which indicate the actions of each service, and
4. Decide on direct or indirect integration into the Web.

Guinard indicates that a client-initiated interaction model, where a Web client continuously polls a Web-enabled thing, is costly when dealing with resource constrained things [34]. Instead a server-initiated model is proposed. Here, a Web client first registers with a Web server and is notified when an event occurs using WebSockets[10]. This creates a real-time approach when a Web client interacts with a Web-enabled thing hosting a Web server. Moreover, the use of tPusher, which adds the use of WebSockets to the WoT architecture, is suggested. *tPusher* moves the protocol to a gateway rather than a real-world thing [58]. This layer ensures that real-world things are exposed as first-class citizens on the Web. The Findability layer focuses on making services of real-world things searchable and available for service consumption or composition. Integration of things to the existing search engines with the use of semantic annotations is studied and the shortcomings of this approach were presented. The shortcomings are because of the inherent nature of real-world things, i.e., real-world things do not have

[10] http://www.websocket.org/

properties that can be indexed like documents on the Web and things are tightly coupled with their contextual information like the owner of a thing, or its location. A lookup and registration infrastructure adapted to WoT is proposed. The Sharing layer focuses on how Web-enabled things are shared ensuring privacy and security. The requirements stated for a sharing platform for WoT are security, ease of use, reflection of trust models, interoperability and integrated advertisement. The architecture proposes the use of existing social networking platforms like Facebook, Twitter, and LinkedIn, as sharing hubs for things. Since these platforms already manage access control and trust measures, these can be leveraged to manage the access and privacy of things privately. The Composition layer focuses on enabling end-users to create simple composite applications with Web-enabled things. Web 2.0 mashup techniques are reused to create Physical mashup editors considering the requirements of the real-world things. The requirements are, support for event-based mashup, support for dynamic building blocks, support for non-desktop platform and support for application specific editors [60].

2.5 Related Work

Though WoT architectures and enabling technologies are in its early stages of development the results are promising. The research contributions are maturing to create new platforms for designing and realizing innovative applications. There are yet many challenges and hurdles that need to be tackled. Some of the challenges that ongoing research is focused on are, providing discovery services for things on the Web, optimizing communication protocols, ensuring security and privacy of things, and the composition of things. Here, we first provide an overview of these research trends and then summarize our review of these works. We also present our research direction in the light of what is not yet done.

The spread of WoT is expected to flood the Web with real-time and real-world information and services that need to be discovered [61]. The use of sensors and other probing devices result in the dissemination of a large amount of real-time information. In contrast to the documents on the traditional Web, real-world things on the Web are expected to rapidly generate dynamic content because of their changing state. WoT applications inherently depend on the context of things and hence, search engines for WoT should focus on efficiently probing real-time data and discovering dynamic services. The efficiency and performance issues of search engines and related algorithms form some of the current areas of WoT research. Recently, Mayer et al. suggested a discovery service for Web-enabled things, where users employ RESTful interfaces for discovering Web resources [62]. A key challenge is to have an efficient search engine, which is able to reach and retrieve properties of real-world things on the Web. Very limited work has been done to define the properties of things in light of the dynamic nature of real-world things. Towards this shortcoming we suggest some of the states and properties that candidate things are expected to demonstrate on the Web [12], which we present in the next chapter.

Research has been focusing on adapting traditional client-server approaches to expose the functionalities of physical things. This approach is feasible when clients initiate interactions with servers to pull data and services from embedded applications that are expected to control things on the WoT. However, this approach is not efficient to monitor things in real-time. Since most monitoring applications are event-based, Web-enabled things should also be able to push data to clients whenever an event occurs, rather than having the client poll a thing continuously. Hence, standards such as HTML5 are being explored which is maturing towards efficient asynchronous bi-directional communication to enable interactions where the server initiates events [34].

Current research is also directing focus towards optimizing communication protocols for resource constrained devices. Efforts to adapt application layer protocols like Constrained

Application Protocol (CoAP) [63] based on REST and HTTP are active research areas, to enable flexibility and ease of use for a Web user and not just technical people. Refining protocols like 6LoWPAN [63, 64]to adapt the IPv6 protocols for low power radio networks is, also, highly relevant.

Issues of security, privacy, and trust in WoT promise many opportunities for research innovations [65]. To clearly contrast these three issues with respect to WoT, consider the example of a *car* on the Web. Privacy would involve dealing with issues that arise when the current owner decides to share the car with others on the Web. Security would deal with issues that pertain to who or what will have access to the car when it is in use. Trust would deal with issues of interactions between things on the Web, like if the garage door would open when the car arrives. The use of REST-based interfaces makes it possible to have secure interactions using HTTP authentication or HTTPS [66]. As things on the Web will be accessible and shared among many users, research in this area is crucial to the success and widespread use of the WoT. Research has been rampant with innovative ideas dealing with these issues in the IoT [67, 68], but the focus on WoT is yet to mature. The use of the social Web as a platform to ensure the trust and privacy of things has been advocated [18], to control Web-enabled things among trusted members on social Web sites.

Towards integrating and composing things into the Web, Trifa et al. implemented a gateway for Web based interaction and management of sensor networks through RESTful Web services [69]. Guinard et al. built an EPC Network prototype by using virtualization, cloud computing, and Web technologies [58]. In their prototype, an RFID reader behaves like a gateway between the cloud server and RFID tags. Welbourne et al. develop a suite of Web-based tools and applications that facilitate the understanding, management, and control of personal RFID data and privacy settings [70]. They deployed these applications in an RFID ecosystem

and conducted a study to measure the trends in the adoption and utilization of the tools and applications as well as users' reactions. A key research challenge is the study of things composition to create context-aware environments [71]. With the plethora of real-world things that can be participants on the Web, this challenge is sophisticated and many innovative solutions are yet to be proposed. Research is focusing on the study of consumer reaction and trends in applications for eHealth, smart homes [71, 72], and sustainable environments [73], which are being suggested using ambient intelligence.

Our survey presented above of the ongoing research towards realizing the WoT vision, provides a comprehensive overview of recent research directions. The research focus on WoT is still in its initial stages but the results provide an optimistic view of how real-world things are joining people on the Web. Many of these approaches addressed here tackle some of the many challenges towards realizing this vision. They are posed towards leveraging and adapting existing technologies for pushing and pulling information from real-world things on the Web. There are yet many challenges that are not addressed by the above approaches and we list some of these here.

- The above approaches do not consider a structured semantic classification of things. Such a classification would enable interoperability among things and the possibility of developing standard methods to adapt thing's states and functions into the Web.

- The above approaches do not consider the large scale of things that would eventually be available on the Web. It would be impossible to manually integrate things one at a time into the Web.

- Things interact with other things and people to form communities on the Web. With the promising future of the WoT, the above research approaches have not considered the

relationships between things nor have they suggested ways to create such communities.

Our study is focused towards three areas and these three complement each other. Firstly, we provide a semantic classification of candidate things. We suggest a framework to manage these things within a scaled down space where we identify repeating patterns of things and we model the virtual representation of things based on their spatial confinement. Finally we model the community of things and the possibilities of these communities interacting with people. Towards these approaches we present our work in the next few chapters.

2.6 Summary

In summary, the WoT is rapidly expanding as an important area of research. Developments in WoT research have the potential to realize Weiser's vision [1]. Many salient technologies are driving the success of the WoT, such as embedded systems, Web services, IPv6 and tagged things. In this chapter, we have presented the state-of-the-art in WoT research and discussed existing technologies and platforms that support the WoT. We presented different approaches to facilitate the realization of WoT based applications. The approaches are motivated by the level of comfort that people have when dealing with the Web, to portray the WoT as a simple and do-it-yourself technology. Advances in communication and real-time systems indicate that real-world things will soon become IP-enabled with embedded Web servers, making them addressable, controllable, and composable on the Web. In this context, ongoing research addresses some of the challenges posed in leveraging and adapting existing technologies for pushing and pulling information from real-world things. Many innovative ideas and applications are being designed and deployed on WoT where people and real-world things seamlessly interact over the Web. Research directions are presently driving the WoT into new horizons.

Chapter 3

A Classification of Things for the Web of Things

Current developments in Web of Things (WoT) applications are largely ad-hoc, proprietary, time-consuming, and require a considerable effort of low-level programming. However, the environment in which users interact with things (*the Web*) is dynamic in nature because of various reasons like, changing technology, changes in services, and evolving user requirements. Within this contrasting context, the specification of new composite services involving real-world things and reflecting user requirements could be frustrating and overwhelming. Moreover, this context is hardly applicable because of the volatility and size of the Web.

Managing the large number of things (i.e., to facilitate the description, deployment, and interactions of things) that is expected to flood the Web is exponentially complex. WoT developers require a new categorization and structure for addressing the context of things. The need for a scalable framework[1] for things to be easily and transparently adopted into the Web calls for the definition of a semantic structure for candidate things that are to be represented

[1]The Ambient Space framework is discussed in the next chapter

on the Web. Failing to do so may result in redundant efforts for integrating things in future ubiquitous applications, because of the absence of a common classification of things.

The first step towards realizing such a framework is to have clarity on the semantics of the potential things that would participate in the WoT. As a contribution towards our framework for managing things on the Web, this chapter presents our proposed classification of things based on their common capabilities. With the flux of things joining the Web, the relationships between things and people are yet to be studied. In this chapter, we also discuss some attributes related to the social relationships of things on the Web [12, 17].

The remainder of this chapter is organized as follows: Section 1 provides an overview of the background of our work and also motivates the need for classification. Section 2 describes the classification of things in terms of standard dimensions, and discusses the integration of related things into the Web. This section describes our proposed ontology of things, related reasoning modules, and presents the *Ambient Learning Space* to introduce an example of Ambient Spaces (AS). In Section 3, we discuss some important attributes to illustrate necessary and important aspects for consuming applications with regards to the relationships that exist between things and people. Finally, Section 4 provides concluding remarks for this chapter.

3.1 Overview

Targeting the lack of a structured classification for things, we propose a hierarchical classification and some schematic attributes, to produce an ontology for things. For illustration, we designed an ambient learning space which is discussed in the context of our proposed classification. We reason over the ontology to infer use-cases that subsume things within this learning space. We propose an OWL-based implementation of the ontology, and use a *SPARQL* rule-based system

to reason over this ontology. We also propose standard types in the multidimensional space of things used to represent things on the Web. As things connect to the Web, they exhibit certain properties related to their dynamic states as well as other social descriptors which we describe in this chapter.

Primarily, our contribution assists architects and designers describe things and their contextual behaviors on the Web. Moreover our proposed classification helps abstract a thing and quickly model it for integrating it into the Web for building context-aware applications. Our classification promotes standard interfaces and operations on things via the Web, to facilitate interoperability and reuse of things. Our capability based classification supplies an awareness of the capabilities that need to be augmented if a thing is adapted into the WoT. These Web-enabled things lead to models of composition to meet desired deployment scenarios of ubiquitous applications. We illustrate one such scenario, an *Ambient Learning* Space, which is a model of a ubiquitous learning environment to explain the need for a classification of things within such environments.

3.1.1 Design of an Ambient Learning Space

Real-world things are replicated in many spatial contexts like, a projector in every classroom, a patient-monitor in every hospital room, or a parking sensor in every parking spot. We define *Ambient Spaces* (AS) to be the virtual representations of one or more *Web-enabled things* that is within such specific spatial contexts, and has information or operations to be represented on the Web [17]. For example, an AS would be the virtual representation of a *thing* like a computer in a classroom, or the lights in a classroom, or all things in a classroom, or all classrooms on a floor. The granularity level of the things within AS is decided by the application developers based on the application context. Many novel applications are foreseen like, energy

sustainability, logistics, and supply chain management. The aggregation of things within AS provides a hierarchical and scalable approach to realize the WoT vision. In consecutive chapters we expound on how things are Web-enabled, the related software representations, the features of AS framework, and how things join AS. Here, we illustrate an *ambient learning space* as an example of an AS and describe our proposed approach to classify things within the context of a learning environment.

Our education system is pressured to align itself to contemporary profiles of learners through the deployment of technology-enhanced instruction. Learning technology has lately been driving this move where an increasing number of institutions have invested into technology-enhanced learning environments [74]. Prompted by technological changes and the availabilities of funding initiatives, research programs, and standardization initiatives (such as SCORM [75]), these technology-committed institutions are increasingly populating the educational landscape. During the last decades few has changed though, and universities remain conventional despite the rapid and wide proliferation of technology within our societies. The soaring enthusiasm of learners for gadgets is increasingly being used to disseminate knowledge, like learning something new while on-the-go. However, these learning instances occur informally in uncontrolled environments that are remote from context and learner profile, which makes it harder to find and advocate typical pedagogically-sound learning tasks [75]. On the other hand, ubiquitous computing environments bring context awareness to users to enable ubiquitous learning [76]. We designed the Immersive Learning classroom which is an instance of such an environment. The Immersive Learning classroom is an ambient learning space, caters to the ever-increasing use of ubiquitous systems in today's social and instructional setups. Consider this AS to be confined within an Immersive Learning classroom of a campus in a university context. Equipped with a touch-screen signage, the workspace includes a sound system, a projector, a video-conferencing

unit, and a digital screen (interactive board) as illustrated in Figure 3.1.

Figure 3.1: Design of Immersive Learning Room

Typically within the ambient learning space, a learner on campus checks the room's availability on behalf of a group of interested students via the Web or checks the schedule on the touch-screen signage outside the room. In either case, the room may be reserved by browsing to the classroom website and entering relevant reservation details. Once the room is reserved, all members of the group get notified on-line via a calendar-meeting request to join the scheduled session. As a booking is confirmed, a notification is also sent to the student's supervisor who may join the group using a Web based interface via the classroom's mounted video-conference unit. The student IDs are matched with the seats they occupy in the room. As soon as the students join their seats, the workspace is declared busy and attendance is automatically logged.

At the same time a video conferencing invitation is delivered to the supervisor who may be remotely located. A learning session consists of successive or simultaneous annotations on the interactive board from the tablet provided to each participant. The workspace can trace each member's contribution on the shared interactive board. The supervisor may also project some case studies on the interactive board through the Web, to throw personal notes or illustrations into the topic of discussion. The goal of this scenario is to identify the steps towards building such an environment to support learning processes.

With a plethora of things becoming ubiquitous on the Web, there is a need to model physical environments as depicted in Figure 3.1 [12], and plan to handle large number of things in AS. There are quite a few challenges in building a completely integrated system as the above Immersive Learning classroom.

- Accessibility to things via common interfaces is currently lacking, which is essential to build applications that exploit their capabilities in a given context.
- In addition, the heterogeneity of ubiquitous computing systems poses a major problem for system architects with respect to many protocols, component architecture, and data formats.
- There is no clarity on the common characteristics of things or processes for controlling them or querying them for tracking needs. Considering the fact that things are dynamic in space and time, managing the presence of things is quite a challenge.
- Also, defining the relationship between things in AS like the Immersive Learning room presents a significant challenge.

With the increasing number of things on the Web, it is obvious that there must be a hierarchy of things, to enable things to eventually manage other things. Towards engineering the virtual

access and control of large number of physical things on a common platform (i.e., the Web) it is important to identify the types of things and realize the various properties of these things. To represent the commonalities of things, a formal classification or ontology is a necessity [22]. We propose a classification based on the capabilities of things and provide an ontological structure for the same. Classification harnesses the complexity of modeling context-aware scenarios through a common approach and standardizes the interface for a wide range of physical things. A metaphoric illustration is the capability of database experts to quickly model a wide variety of data because of standardized data classification and structures. Next, we look at some related work in the area to corroborate our approach to classify things.

3.1.2 Supporting Work Towards Classifying Things

Kortuem et al. [77] address issues on modeling and representing smart objects in order to strike a balance between the objects and the infrastructure. This effort focuses on the design of industrial hardware. Instruments and tools in industrial scenarios are augmented with sensors, wireless communication capabilities, and display devices, to render them as smart. These tools are classified as activity, policy, or process-aware objects, based on their awareness, representation, and interaction. These types represent combinations of three dimensions with the aim to highlight the interdependence between design decisions and explore how these objects can cooperate to form IoT. However, this work is constrained to particular industrial devices and does not consider the vast majority of objects that could potentially be used to provide useful information. For example, objects that do not have sensing capabilities would not be classified as smart objects. In contrast, we propose a more comprehensive classification model where objects are abstracted into the Web based on factors like their capabilities and location. Beigl et al. [78] define smart physical things as things augmented with computing and communica-

tion capabilities, which can be accessed by computer applications. Similarly, Friedemann [79] envisions smart things to be able to wirelessly communicate with people and other smart things, with the ability to perceive the presence of surrounding objects. Today, these definitions do not formally encompass all things that could be on the Web, for example, an RFID tagged chair or a Personal Digital Assistant (PDA), and both are accessible on the Internet. There is no significant work done so far to classify things based on their capabilities or a specification of the various characteristics that would contribute to rendering a thing capable of contributing on the Web. Such a classification would facilitate the realization of a framework for integrating things into the Web and also enable the systematic deployment of things into WoT on a large-scale either as Web resources, providing information or as Web services providing autonomous services.

3.2 A Capability Based Classification of Things

With the advances in technology, the capabilities of real-world things have been increasing and this enhancement has introduced the term *Smart* to be prefixed to these things. Thus, we have Smart phones, Smart TVs, and Smart Homes. The proliferation of smart things with Web capabilities is driving the possibilities of realizing the WoT, but what makes a thing *smart*? With billions of things finding their way into the Web, people find themselves surrounded by things they interact with and share information. Can we define the parameters that describe such environments or AS?

Classification is described as a process of creating sets, into which things are mapped, in order to standardize processes for creating related infrastructures [80]. Classification ensures consistency and allows future ubiquitous application to interact with things through implicit

common interfaces hiding behind the inherent intricacies of things. To overcome the universality of things and their intractable complexity, the classification process must be addressed at a high level of abstraction to ensure simplicity and completeness of the proposed representation of things. Without such a classification, ad-hoc and redundant methods of dealing with things will prevail and interoperability will fail.

We abstract the capabilities of real-world things, and their interfaces on the Web to drive application functionalities when dealing with those things. These things are then usable for various scientific, environmental, business, and social needs via common Web interfaces. Hence, a classification based on capability to characterize things is deemed necessary as a foundational step before defining a framework to enable the management of the vast span of things to be deployed on the Web. This enables application designers to quickly model a thing, and define the interoperability of applications that use these things.

3.2.1 An Ontology of Things on the Web

A thing is defined as a physical entity that is operated from or has information to share on the Web. Considering their variations, things are abstracted into common representations to be integrated into the Web. We propose an ontology which models the characteristics that describe the dimensions of a thing on the Web to facilitate its virtual integration. A thing on the Web has dimensions like, its capabilities, location, and friends. We describe the capability dimension here and the rest are described in the next chapter. The capability dimension describes the essential details that elevate a real-world thing into a *thing on the Web*.

The virtualization of real-world things is achieved through a *Web Object Metadata* (WOM) shown in Figure 3.2, which defines the required capabilities of a thing to become a candidate WoT member. The WOM is the virtual representation of a real-world thing through the speci-

fication of relevant ontology like the WOM-Capability, which we explain here.

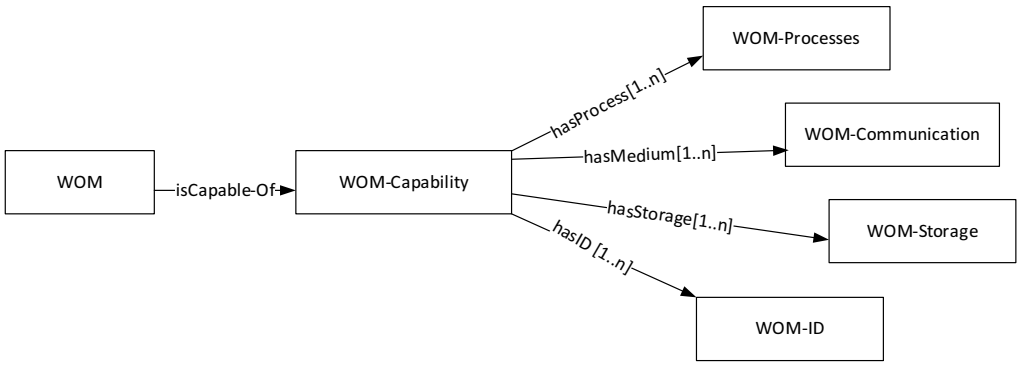


Figure 3.2: Model of Web Object Metadata (WOM)

The WOM-Capability ontology recognizes the four capability dimensions of real-world things for their participation in an Ambient Space (AS) to be Identity (ID), Processing (P), Communication (C), and Storage (S), referred to as the *IPCS* capability set. WOM-Capability ontology mandates the minimum requirement for a thing to participate in an AS to be a unique ID within the application context.

- *Identity (I):* A thing must be uniquely identifiable with the use of an appropriate identification system. Identification systems like Barcode, RFID, or IP address are used to locate a thing and access it as a unique resource within an AS. A thing could be identified using multiple identification systems, for example, a thing could have a Bluetooth address and an IP address. The requirement for a unique identity within the context of an application is a mandatory and minimal requirement.

- *Processing (P):* The processing capability of a thing defines the functions that allow a thing to be controlled or managed. At the chip level this defines a microprocessor, or an operating system that provides functions to control and manage a device, or a Web server

that processes HTTP requests.

- *Communication (C):* The communication interface of a thing is a system enabling interaction with other things. It describes how to read from or write to things. For example, a car stereo with a USB port and Bluetooth connectivity has two communication interfaces. A thing's functions exposed as a Web service provides Web-based Application Programming Interfaces (APIs) as communication interfaces for other things to interact with it. Properties of each interface (e.g., medium, protocol and privacy) describe how the communication is to be established.

- *Storage (S):* Storage describes the type and the amount of information that a thing retains. This capability enables the thing to record states and values. A thing could have multiple storage types and corresponding properties. Each storage type has properties such as name, storage type and capacity that describe its use.

Based on the IPCS capabilities, we classify things into different types. A taxonomy of things based on the different combination of IPCS capability dimensions provide an understanding of what capabilities a thing has or does not have. This allows developers of ubiquitous application to augment the necessary capabilities if they are required for a particular application context. Our taxonomy for things, define a *Core* thing to have the bare capability of being uniquely identified within a given context. A *Primitive* thing has a unique identity and includes one additional capability of the other three dimensions. These are further categorized as the *Fat*, *Plug,* and *Fuzzy.* In addition to the capability of being uniquely identified, *Fat* things has storage capability, *Plugs* have communication capabilities, *Fuzzy* things have processing capabilities. A *Complex* thing has a unique identity and combines two of the other three dimensions, as indicated in Table 3.1. A *Smart* thing combines all four capabilities of the IPCS set. The

combinations of Primitive and Complex things are shown in Table 3.1 where, *Identity* is not included as it is mandatory for all categories.

Primitive Things				
Names	**P**	**C**	**S**	**Examples**
Fuzzy	X			Washing machines, Microwave ovens, etc.
Plug		X		Headphones, Speakers, etc.
Fat			X	CD, DVD, etc.

Complex Things				
Names	**P**	**C**	**S**	**Examples**
Social	X	X		Remote controls, land-line phones, etc.
Sticky		X	X	USB stick, RFID tag, etc.
Gizmo	X		X	Calculator, Game Boy, etc.

Table 3.1: Classification of Primitive Things and Complex Things

The context in which a thing is used may vary based upon the application domain a thing participates in. Specifying the characteristics of things makes it easy to abstract things for various application contexts on the Web. For example, in an asset management application, a Personal Computer (PC) would only need to have a unique identity (e.g., an RFID tag) to indicate its presence. In a network management application, the same PC would be required to be classified using other capabilities such as communication interface (network ports, IP address), and storage (RAM, HDD). Thus, things are classified into one of these categories and analyzed to decide if they satisfy the required capabilities needed for a particular application.

We present our ontology using a UML based definition which has a number of benefits over traditional approaches to represent knowledge [81]. Moreover, UML is a widely accepted standard, expressive for both human-understanding and machine processing. Using UML descriptors the proposed ontological specification is illustrated in Figure 3.3.

An example, is illustrated in Figure 3.4 where an instance of the relationship between the

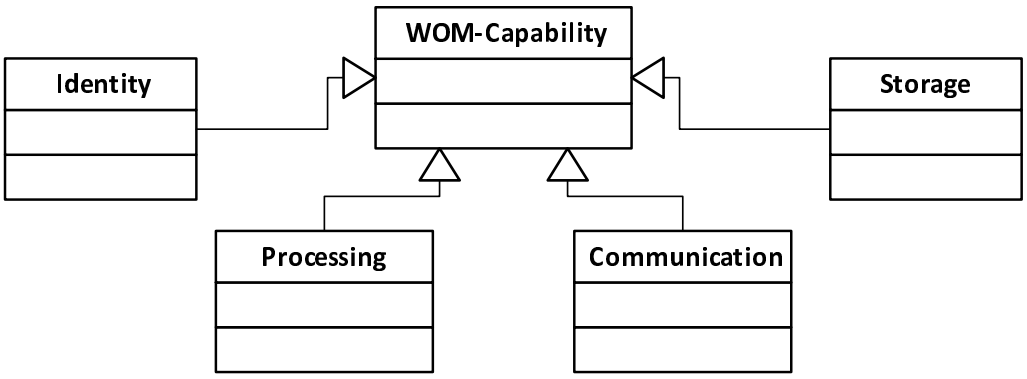


Figure 3.3: WOM-Capability class and IPCS sub-classes

classes are shown. A candidate thing (Web -Object) is first identified as a *Primitive* thing, Figure 3.4 (a), since it has an *Identity* and one of the other three capability dimensions. Next, the *Primitive* thing is identified as *Fuzzy* (Figure 3.4 (a) and (b)), since it has the *Processing* capability. This allows the architects of WoT applications to have knowledge of interfaces that need to be created for the thing's available Processing capabilities and also augment other capabilities if required for the required application context.

A thing needs to be essentially *Smart* to participate as an entity on the Web and we extended our ontology to define *Web Smart* things. A *Web Smart* thing inherits the dimensions of Smart thing i.e., using an object oriented connotation '*a Web Smart thing is-a Smart thing*'. Our ontology facilitates the process by which the IPCS capabilities of Web Smart things are satisfied by:

- *URL*: A thing is uniquely identified with a URL. RESTful adaptations of the URL provide necessary semantics to access the state and functions of things. Hence, a thing on the Web has a number of URLs $\{I_i, i = 1...n\}$, where n is an arbitrary number of IDs.

- *Web Server*: A thing processes HTTP requests through a Web server either augmented

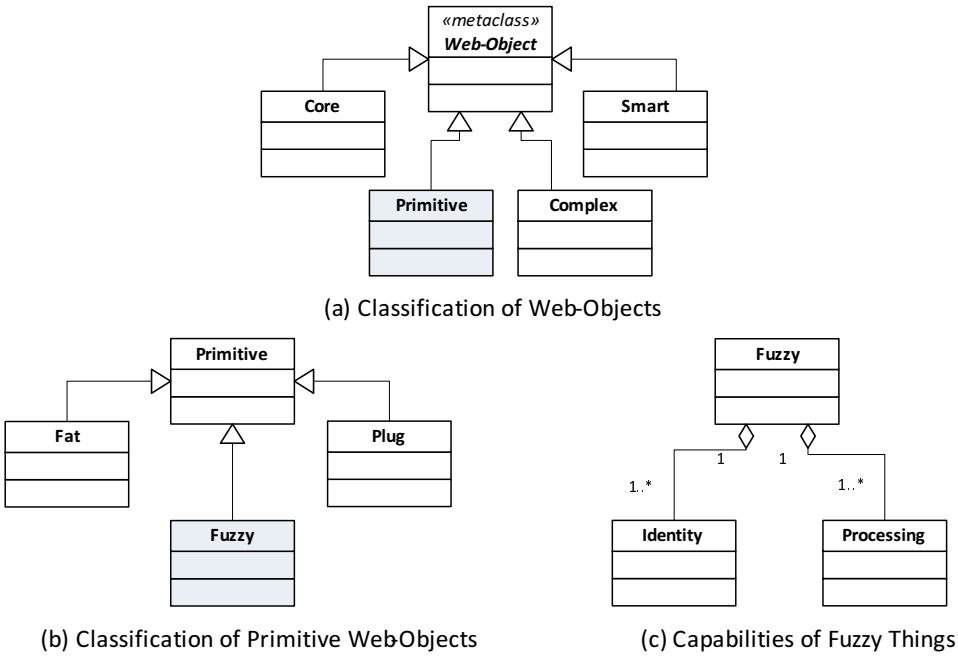


Figure 3.4: UML Descriptors for the Class Relationships

directly onboard or connected externally. These requests are processed and a respective representation is returned or converted to operations on the thing. Hence, a thing has a number of processes $\{P_i, i = 1...n\}$, where n is an arbitrary number of processes.

- *Web Services*: Communication with a thing is defined by RESTful APIs with methods like GET and POST. This enables the state and functions of things to be communicated using standard Web interfaces. Hence, a thing has a number of communications channels $\{C_i, i = 1...n\}$, where n is an arbitrary number of communication channels.

- *Storage*: A thing must have capabilities to cache Web resources and status information. The storage could be onboard or remote, for example a data center, or a private cloud. Hence, a thing has a number of storage options $\{S_i, i = 1...n\}$, where n is an arbitrary

number of storage options.

The role of the proposed ontology is to provide a unique vocabulary and description logic based on modeling things for rudimentary reasoning. The ontology consists of modules for the shared architectural knowledge layers, and services of things.

Querying the ontology is done using W3C recommended SPARQL language for RDF. The search potential of SPARQL is appropriate for querying our knowledge base [82]. The logic for dynamically querying the knowledge base is implemented using Jena APIs [83] and Protege[2] classes, which are open source frameworks for building Semantic Web applications. Jena is Java based and provides an environment to programmatically access the ontology. The knowledge base is queried with Jena APIs, like *executeSPARQLQuery()*, that takes a SPARQL query as a parameter and runs it against the knowledge base (example shown below).

```
JenaOWLModel owlModel =
      ProtegeOWL.createJenaOWLModelFromReader(input);
String queryStr =
      "SELECT ?primary WHERE ?primary rdfs:subClassOf WOT:Primary";
QueryResults results = owlModel.executeSPARQLQuery(queryStr);
```

A composite thing is the aggregation of different things. For example, in our ambient learning space the Immersive room is a composite thing made up of different individual things like camera, interactive board, and projector. If the individual things that form the composition cohesively contribute to all four capability characteristics (IPCS), then the composite thing is a *Smart* thing. Such mashups of physical things are abstracted on the Web by dynamically composing and assembling things for a particular application, where the capabilities of the participating things are utilized to create synaptic *Web Smart* things. Using the basic capabilities

[2]http://protege.stanford.edu/

that our ontology defines, further inferences can be generated to reflect composite capabilites using First Order Logic based representation. SPARQL is used to reason over the proposed ontology to infer composite things. In doing so, new types of things extend the existing ones to best satisfy specific application requirements. For example, composite things are identified using the following rules:

isPartOf (?y, ?x1) AND isPartOf (?y, ?x2)

isComplex (?x1) AND isComplex(?x2) $\rightarrow$ *isComposite (?y)*

In our proposed Ambient Learning Space, described in Section 3.1.1, the Immersive Learning classroom is created by mashing up things like projectors, sound systems, and video conferencing units. The capabilities of the various things in the room is composed to create the Immersive Learning room as a Smart room. The mashup process subsequently leads to descriptions and capability representations of discovered things: a user or software agent thus autonomously receive content or operate on things that are in AS. The ontology provides a common description format that is interpreted by Web services that may be typically involved in a delivery chain of ubiquitous services. A further specialization of the ontology facilitates adaptation to particular contexts that may be more relevant to the context at hand. However, the top-level specification addressed in this section enables a general semantic for future Web applications to develop ubiquitous experiences that require minimum user intervention. The combination of an ontological and rule based framework leads to a knowledge based structure of things within an AS. The main benefit of this structure is its semantic power in conceptualizing knowledge about things on the Web. The ontology of the related knowledge base acts also as a directory services for ubiquitous context-aware applications. These applications acquire and process information about surrounding environments based on implicitly derived information about Web Smart things. According to information fusion approaches, which may combine data sources

from multiple things, more complex contextual states are derived as a basis for triggering or offering new services.

Our classification facilitates the development, interaction, and integration of future Web-based ubiquitous applications among software architects using a common specification. It also enables reuse and extends the possibilities of standardizing the architectural framework for developing applications for WoT. The ontology specification aims at subsuming thing's diversity to hide inherent disparities into the surroundings, leaving only ontology-specified interfaces as perceivable access points to services and content of things. Although this domain may be further specified for some application contexts such as learning technology, multimedia, and health-care environments, we provide a top-level specification of the ontology to map thing's descriptions.

3.2.2 Prototype of Ambient Learning Space

Various applications for network management, asset management, and ambient learning space management are planned. Our proposed classification provides different levels of abstraction to model and reason over things in this environment. Ambient learning in an educational institute aims at matching the profiles of learners with instructional things, to reach a personalized learning experience [84].

Typically, learners are immersed in AS in our campus which communicates seamlessly with its inhabitants in a persuasive way that drives learners through a continuous learning cycle such as the one shown in Figure 3.5. Depending upon the context of an AS, learners may enter the learning cycle at any point. Next, we illustrate how we abstract classrooms in the building and abstract the things in a classroom.

The Immersive Learning Room (*Smart*) has the following features.

- Identity: It is uniquely identified with a room number and it is displayed on the classroom

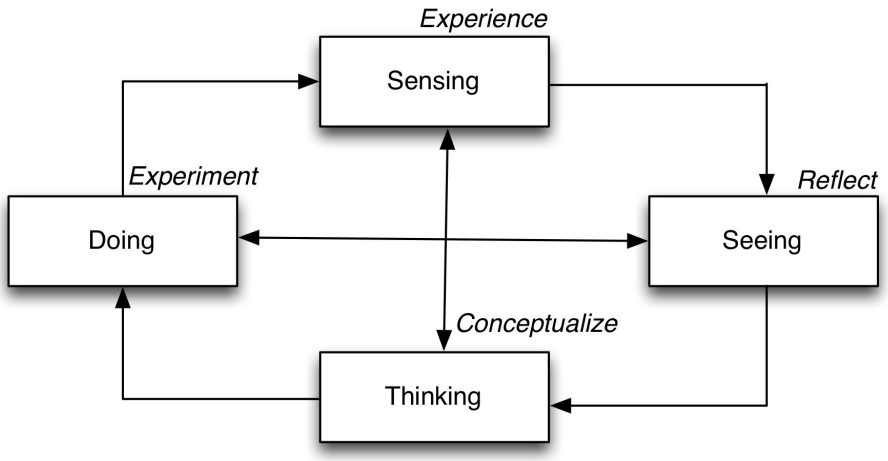


Figure 3.5: Experiential Learning Cycle

signage as shown in Figure 3.6(b). This signage is integrated into the campus Intranet and is uniquely identified on the network.

- Processing: The classroom signage has a computing system with touch screen interface. It provides various operations like assigning faculty to rooms, scheduling courses in rooms, and special message display.

- Communication: Classrooms provide interaction via video conferencing, IP phones and audio equipment as shown in Figure 3.6(a). This enables remote participation in classroom activities.

- Storage: Session details are stored via video recording capability, which is used to record the sessions that are scheduled in these rooms. The history of captured video recordings is made available to learners.

Classrooms without storage capability are classified as social venues since they satisfy only three (IPC) of the four characteristics. The provision of this classification enables appropriate

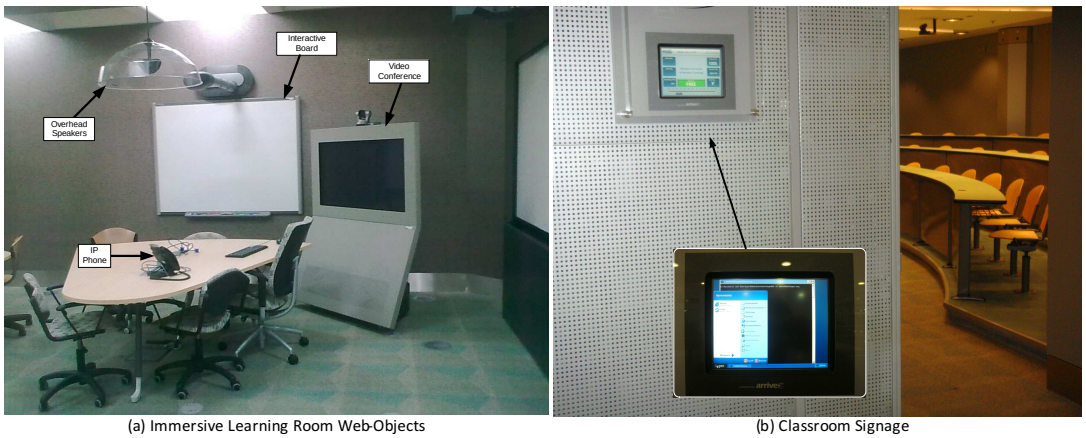


Figure 3.6: Immersive Learning Room

selection of learning objects (classrooms) for Immersive Learning room applications. For example, when learners have to choose to attend parallel learning sessions happening in Smart and Social venues, the application would give priority to Social venues since the Smart room session recording is available in a conflict-free time slot. This common classification enables the process of defining various things in the building which allows us to abstract and adapt new things for applications in the building.

3.3 Social Attributes of Things on the Web

Virtual interactions and the social implications have been the focus of research earlier, for example the Collaborative Virtual Workspace (CVW) from MITRE[3] corporation (1999). From a technical point of view, CVW is a framework for seamlessly integrating diverse tools that enable collaborative capabilities. From a social point of view, CVW is an environment for collaboration. It is a virtual space divided into rooms that integrates people, information, and

[3]http://www.mitre.org/news/digest/archives/1999/cvw.html

tools appropriate to a task, operation, or service and provides a context for communication and document sharing. CVW enabled time saving and increased the quality of distributed training, analysis, and decision making. CVW significantly advanced the ways in which MITRE's customers and other organizations conducted business. Social issues like security, ownership, rewards, and reporting structure was addressed to make it a successful endeavor. While CVW virtualized social interactions and collaborations between people, the WoT focuses on including real-world things into the virtual environment. Since real-world things are inherently proprietary in nature it is important to describe their social relationships for successfully modeling things within an AS and thereby for the WoT. There are various attributes that would define a thing on the Web in relation to their social interactions. We propose a few of these attributes to be considered with reference to thing's participation in an AS.

- **Lifespan:** The lifespan of things indicates the state of a thing and its transitions during its existence. As illustrated in Figure 3.7 [12] a thing takes on *Active* state after it is created or when it is enabled from a *Renewed* state. It can expire and those expired things must be handled to ensure a manageable and scalable environment (i.e., expired things can be removed in time and new things can be easily added in the environment). Determining the lifespan of things helps recycle or dispose those things that have expired. For example, an RFID tagged student ID card (primitive thing) could be recycled or destroyed after a student graduates. This approach applies as well to other contexts, such as a train ticket, or an airline boarding pass. The lifespan of the student ID starts with the journey a student commences as she/he joins the institution and it moves into an *Expired* state when the journey finishes. Based on policy, it can either move into a *Dead* state or be formatted and move into a *Renewed* state ready to be used again.

Figure 3.7: Lifespan of Things

- **Friends:** Foreseeing a world where everything is connected on the Web, a thing is bound to interact with other things and people. It is necessary to maintain the history of interactions. Hence, a thing has friends and the relationship is defined by properties of earlier interactions. The stored details of interactions with friends would ease decisions when interacting in the future (for example, to choose a friend for a particular service). This raises the requirement for creating a trusted circle of friends and the need to maintain relevant information of these relationships. Conversely this also allows the definition of competitors or relationships that are not with friends.

- **Ownership:** Real-world things are inherently proprietary i.e., things are owned by individuals or institutions. *Ownership* defines the possession rights of a thing. This property dictates the rights to make changes and provide or revoke different types of access to things on the Web. It is important to define this property for things represented on the Web to enhance user confidence that they have possession of their things on the Web.

- **Accessibility:** This property allows the owner of a thing to provide exclusive access to others for its use. Exposing the services using the existing social networks allows the users to be in their comfort zone and there would be no need to build a new database of trusted users. This raises questions about privacy, damage, and misuse when dealing with real-world things. Even if things are shared to known members of a community, misuse or damage could occur because access to physical objects is given through a virtual environment (the Web). To mitigate any risk, it is important to have a protocol in place where things accessed by a party (person or system) must agree to legal terms for its use. Access to a thing is either made via public, private, or protected modes. A thing is public when it is available for open access across the Web (e.g. sensors in parking slots). Private things have access restricted to predefined users restricting the access of things within a community of trusted users. For example, in the ambient learning space, adjusting the camera of the video conferencing equipment is restricted to occupants of the Immersive room and their supervisor (who may join the room via the Web). Protected things have private access but with access filtered or limited within a trusted group. For instance, turning on/off the video conferencing equipment is restricted to the supervisor while student in the room are allowed to only adjust the camera position.

- **Searchability:** Things available on the Web are accessed as URLs. Billions of things on the Web require the use of an easy and quick way of finding them. The chances of seeing Google adding "Things" as one of their search functionality, like Scholar or Images, is not far-off. As things become Web resources, Web crawlers have more to do in filtering and normalizing URLs.

3.4 Summary

In this chapter we proposed a capability based classification for candidate things on the WoT. Our capability based classification of things lays a foundation to integrate different types of things into the WoT. The approach identifies things to be Web Smart with necessary IPCS capabilities to participate in the WoT. The IPCS capabilities of Web Smart things are represented as Web Object Metadata (WOM). We also introduced the concept of Ambient Spaces (AS) which is the composition of one or more Web Smart things.

The use of Web-enabled things would allow their access, control and remote management on a common, widely accepted and existing platform of the Web, across the boundaries of vendors and manufacturers. Our framework for AS (described in the next chapter) exposes things as Web services to enable their access and control within different application contexts.

Chapter 4

Managing Things in Ambient Spaces

In the previous chapter we discussed our approach for classifying things based on their capabilities and also introduced some important attributes. Our proposed classification approach facilitates the identification of things with capabilities to participate in Ambient Spaces (AS) and term them as *Web Smart* things. The semantic representation provides knowledge of thing's capabilities or the lack thereof, in a structured manner. This enables WoT application architects to design AS, and equip candidate things with necessary capabilities so that they are seamlessly integrated into AS. However, a semantic representation that represents only the capabilities of things does not provide sufficient details to represent and manage the plethora of candidate things for the WoT. Semantics of various other factors like thing's location, social connections, and user reactions need to be captured. Moreover, a framework that is able to easily adopt things and describe their relationships is paramount in realizing the WoT.

In this chapter, we extend the Web Object Metadata (WOM) of things with other ontologies and elaborate further on the software representations of Web Smart things in Section 1. In Section 2 we discuss the main stakeholders that contribute to a thing's semantic structure. We then, discuss a framework that manages Web Smart things in the light of the Ambient Learning

Space (ALS) described in the previous chapter in Section 3. In Section 4 we discuss thing's augmentation with the required capabilities to be Web Smart. In Section 5, we introduce the need to use AS to create a community of things, where the relationship between things is exploited to establish possible connections with people. Finally in Section 6 we provide concluding remarks.

4.1 Web Object Metadata (WOM)

The perception of a real-world thing in virtual spaces is different from its actual representation in physical spaces. Also, a thing's virtual representation varies for different applications contexts. For example, a *car* is perceived differently for an automotive industry application and a business application. We focus on some of the necessary information that promotes a real-world thing to contribute within the context of a Web application. We extend the WOM structure here to enrich the representation of Web Smart things in an AS and to provide some thematic components to describe the thing's social attributes.

The WOM shown in Figure 4.1, is a collection of different ontologies defining concepts like capability, location, and friends of a thing to describe candidate WoT members. The representation is not exhaustive but is indicative of the representation that is required for the application contexts that we consider. The capability dimension of Web Smart things, the WOM-Capability ontology, recognizes the four dimensions of candidate elements to be Identity (ID), Processing, Communication, and Storage, referred to as the IPCS capability set.

The WOM is extended with the specification of other relevant ontologies.

- *Friend Of A Friend*[1] (FOAF) ontology is used to connect to other Web Smart things

[1] http://xmlns.com/foaf/spec/

(WOM) and people (for example, the URL of a person on a social networking site) as shown in Figure 4.1.

- WOM-Annotations ontology provides rich semantic content to capture a thing's history, user experiences and feedback. For the annotations we use *Meaning of a Tag* (MOAT) to represent tag details [85].

- WOM-Location ontology provides a record of how a thing is traced from the virtual space to its physical whereabouts. Geospatial ontologies[2] provide sufficient location and geographical information for locating WoT resources.

- The WOM-Profile hosts a summary description of a Web Smart thing's semantic information. The WOM components like that of capability and annotations contribute to the information in the WOM-Profile.

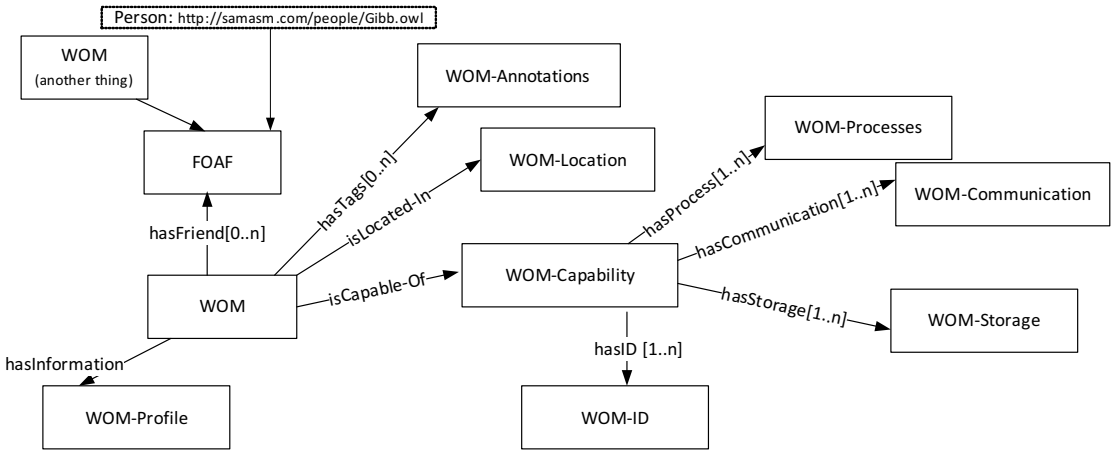


Figure 4.1: Extended Model of Web Object Metadata (WOM)

[2]http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/

Real-world things are inherently dynamic and proprietary in nature i.e., during the lifespan of a thing [12] it adorns various context values and also adapts to various ownership. Moreover, things also have various characteristics like manufacturer's details, price, date of manufacturing, model number, user experiences, and ownership history. The ontologies listed above capture some of the properties of Web Smart things and the properties are accessed from the WOM-Profile.

The semantic representation of things in WOM-Profile has two sets of elements. The first set of elements is tagged with *<wom:preset>* which is a representation of all properties of things that do not change like a thing's capabilities, details of production and manufacturer. The second set of elements is tagged with *<wom:dynamic>* which is a representation of properties that may change (owner, price, discounts and user experiences). A hypothetical and partial example of a WOM-Profile is illustrated through an example representing a *dishwasher* in Table 4.1.

In the example shown in Table 4.1, the preset part of WOM-Profile declares item code, country, manufacturer, and production date while the dynamic part reveals the owner, the price, a friend, and two user-defined tags. The WOM-Profile also includes their IPCS capabilities which are not shown in the XML representation in Table 4.1. It is also important to identify the major stakeholders that contribute towards creating relevant semantics of a Web Smart thing. These stakeholders contribute towards the WOM content and create the context for the creation, use, reuse and disposal of things on the Web.

4.2 Actors Building the WOM

In any fundamental computing setup, the main stakeholders are the providers and consumers of the services or infrastructure. The consumers use and update the system, while the providers

```
<?xml version="1.0"?>
<!-- Define all namespaces used in the profile -->
<rdf:RDF xmlns:rdf= ... xmlns:moat= ...
         xmlns:foaf= ... xmlns:wom= ... xmlns:dish= ... >
<rdf:Description rdf:about= ... >
<!-- WOM-Profile has two parts the Preset and the Dynamic-->
<wom:profile>
 <!-- Preset part defines the fixed parameters of a dishwasher-->
 <wom:preset>

     <dish:itemcode>Dish5AWS12</dish:itemcode>
     <dish:country>USA</dish:country>
     <dish:manufacturer>Omida</dish:manufacturer>
     <dish:prod>2012-11-05T08:15:30-05:00</dish:prod>

 </wom:preset>
 <!-- Dynamic part defines the changing parameters of a dishwasher-->
 <wom:dynamic>

     <wom:owner> ... </wom:owner>
     <wom:price> ... </wom:price>
     <wom:location> ... </wom:location>
     <!-- The dishwasher has a person as a friend -->
     <foaf:Person>
          <foaf:name>Gibbs</foaf:name>
          <foaf:mbox rdf:resource=.../>
     </foaf:Person>
     <!-- Annotations as tags are introduced by users -->
     <wom:tags>
          <!-- Tag name, the maker and meaning is included-->
          <moat:Tag>
              <moat:name>automatic</moat:name>
              <moat:hasMeaning>
                <moat:Meaning>
                  <moat:meaningURI rdf:resource=... />
                  <foaf:maker rdf:resource= ... />
                </moat:Meaning>
              </moat:hasMeaning>
          </moat:Tag>
     </wom:tags>

 </wom:dynamic>
</wom:profile>
</rdf:Description>
</rdf:RDF>
```

Table 4.1: Example of WOM-Profile (Partial)

deal with the manufacture, deployment and maintenance functions. The domain of WoT requires the addition of new stakeholders and redefinition of the traditional ones. The stakeholders within the WoT domain not only require providers and consumers but also needs to consider the role of owners and regulators who control the thing's inherent dynamic and proprietary state. Here, we briefly list the stakeholders, focusing on their contribution to the content of a thing's WOM-Profile.

- **Providers:** The providers are essentially the manufacturers that create the WoT elements. The providers will also hold the responsibility of recycling or discarding a thing at the end of its lifespan [12]. The maintenance and upgrades to a thing are the responsibility of providers while a thing is used by other stakeholders. The providers hold the right to change the content of a thing while maintaining history of changes. The providers contribute to the preset content <wom:preset> of a thing's profile and are responsible for ensuring the presentation of thing's composition, use, and disposal. The preset content of a thing's WOM-Profile is fixed and not changeable by other actors. Contact information of the providers needs to be provided, for the use of thing itself or any of the other stakeholders. The links to the user manual and the conditions of thing's usage are provided by the providers. The providers may also contribute to the dynamic content <wom:dynamic> of a thing's profile. Annotations for branding, price composition and marketing are initially added by the providers. The providers initiate the history of a thing's existence.

- **Consumers:** Consumers of a Web Smart thing are its users. These users could be other things or people. Unlike other domains, consumers are not owners here and are bound to access restrictions that are controlled by the present owner of a thing. The contribution

of consumers populates the dynamic content <wom:dynamic> of thing's profile. The consumers provide rich semantics to thing's use and add to the history of a thing. The content that the consumers provide to a thing essentially creates links with other things or people that are connected to the consumer. Thus the consumers play an important role in promoting things social connectivity.

- **Owners:** Owners are consumers but have more rights to a thing's usage and content. The owners provide access restriction to a thing's operations and can loan or lease a thing. With proper authorization from regulators and providers, the owners can alter the dynamic content <wom:dynamic> of a thing and therefore change history. The options to re-brand or marketing a thing allows owners to change the value of a thing and promote its acceptance among other things or people.

- **Regulators:** While the other stakeholders provide content to value a thing, the role of the regulators prevails over other stakeholders. For example, government authorities or regulatory authorities that ensures the safe and judicious use of Web Smart things. The regulators provide details on rights and obligations of other stakeholders. They provide contractual details wherein other stakeholders and authorities are informed if there is a breach of contract. Because of the wide spread implication of the virtual use of physical things, liabilities and exceptions are to be clearly defined by regulators. The regulators provide content that are both preset and dynamic related to issues like privacy, trust, cyber-attack and legal implications. The role of regulators needs to be actively researched, investigated, and formulated with governments and international bodies so as to ensure the secure and sustainable use of things on the Web.

While it is important to understand the semantic structure of Web Smart things and the major stakeholders, it is also important to realize how the information is stored and retrieved from real-world things. Next, we discuss the AS framework for the integration of real-world things to be used in a WoT application context.

4.3 Ambient Space Framework

A framework provides a general structure intended for guiding the construction of something useful. We propose the AS framework to guide WoT architects to integrate real-world things for WoT applications. As introduced earlier an AS is the virtual encapsulation of one or more Web Smart things within a specific spatial context. The AS framework requires that we first identify and define the repeating spatial context. For example, consider a classroom as an AS. The classroom is located in a school building that houses many such classrooms, and is also part of the university campus where many such buildings are located. Next, we identify things that are replicated within this spatial context. For example, within each classroom are common things like, a projector, lights, and a camera. Then, we encapsulate these things as Web Smart things exposing their necessary operations. Thus, an AS provides a single portal to all Web Smart things in a classroom and is a common representation of all such classrooms in the campus. Figure 4.2 illustrates our example where the framework is used to design classrooms as AS representations within the context of a university campus that houses classrooms similar to the Immersive Learning room described in the previous chapter. This framework where things are grouped together, allow them to be represented and accessed in a standard way, thereby curbing the redundant and ad-hoc approaches of integrating real-world things into the Web.

Our AS framework make use of the fact that it is possible to represent things within hierarchi-

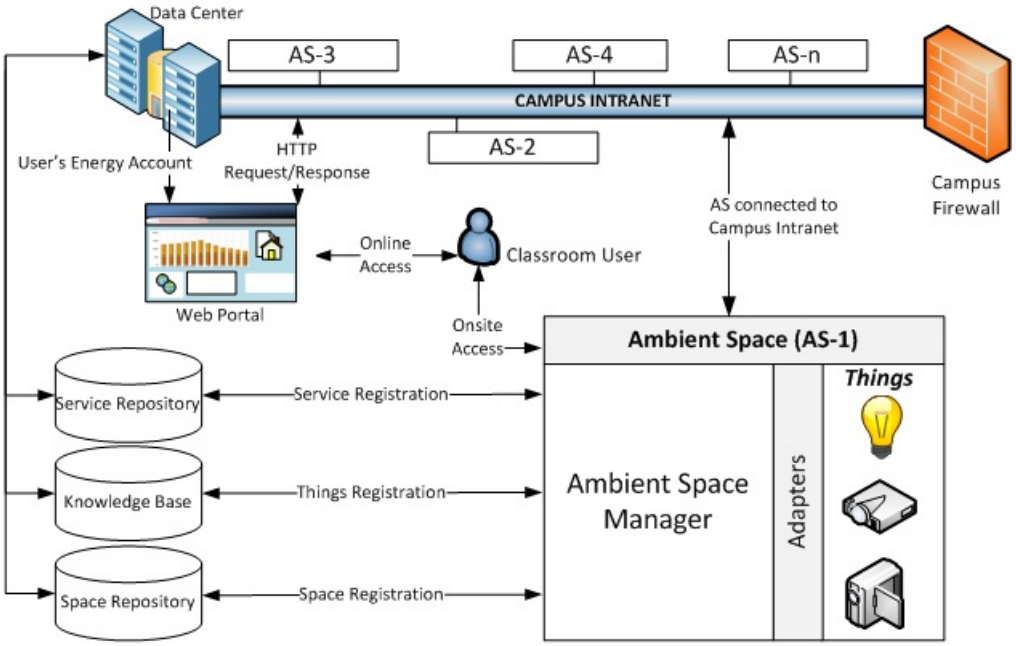


Figure 4.2: The Ambient Space framework for classrooms on a campus

cal spatial patterns to manage things. Also, considering that things are inherently proprietary i.e., they are owned by individuals or institutions, the proposed framework makes it possible for defining thing's ownership or accessibility in a smooth gradient within every AS. Within the context of AS, we define *scalability* as the efficient management of things when the number of Web Smart things increases. Using our framework, once an AS is defined there is a minimal effort in representing other similar spatial structures. Hence, our framework provides a scalable structure for realizing the WoT applications. We present case-studies of AS like classrooms and parking spots in Chapter 6 and evaluate benefits of such representations [16, 20, 21] within application contexts. Our framework can be adapted to many more scenarios, like hospital rooms, train compartments, bank branches, and company buses, all of which have *things* replicated within them.

As illustrated in Figure 4.2, the framework requires all things within an AS to be registered with a *Knowledge Base* using our proposed WOM. Following their registration, the knowledge base acts as a directory of things for applications that need to interact with them in an AS. The *Service Repository* provides a directory of services offered by various AS instances within the campus and the *Space Repository* describes the various parameters of an AS (for instance classroom location, seating capacity, and room availability). The *Ambient Space Manager* incorporates a software agent which queries and reasons on the capabilities of things within the AS; the details of the ontology is discussed in the previous chapter (Section 3.2.1). Web Smart representations of things in the classroom expose services that are to be integrated into the application context. The Web Smart things use *Adapters* to communicate with the candidate things that contribute to the application context. The Adapters incorporate software and hardware components to make-up for the capabilites that things do not have. A detailed explanation of how a real-world thing like a light bulb is adapted to be Web Smart is described in the next section.

Our proposed AS framework presented above uses the context of an educational learning environment. The framework is versatile to suit different environments. We provide another example of a parking lot, which is explained as a case-study in Chapter 6. Next, we explain how we elevate the capabilities of a real-world thing to make it Web Smart.

4.4 Creating Web Smart Things

A *Web Smart* thing is the virtual representation of a real-world thing within the virtual world of the Web. A thing would require the capabilities that we suggested earlier, i.e., the IPCS set, to be Web Smart so as to participate in an AS and thereby be part of the Web. When a

thing lacks any of these capabilities that are required to participate as a component of a Web application, it should be augmented with the missing capabilities.

To illustrate our framework, we built an application within a university campus that helps reduce electrical energy consumption. The details about this application and the related publication are found in chapter 6 [16]. For our case-study we selected inanimate objects like lights in classrooms and augmented them with required capabilities to be controlled and accessed via the Web. Within our classroom context, lights have two basic states, *on* and *off*. The first challenge is to transform light bulbs into a Web Smart thing in order to push and pull information. Their capabilities are to be enhanced so that they are controlled, sensed and their operations are accessible using Hyper Text Transfer Protocol (HTTP), in the context of an energy-preservation application. The second challenge is to specify the software components that would store the semantics of Web Smart things i.e., the WOM.

To tackle the first challenge, as described earlier, we have to augment a light bulb with Internet connection (i.e., IP address) to get an Internet-enabled light bulb, and when it is connected to a Web server, it becomes Web-enabled. Also, extending embedded Web servers and service computing to real-world things makes it possible for their states and functions to be abstracted as embedded Web services [63]. With the use of RESTful URIs, real-world things are able to offer their functionalities to other Web applications. These Web services directly expose operations of real-world things to enterprise applications, to other things, and even possibly involve a dynamic composition of things at runtime.

To address the second challenge, the software components are expected to reflect online scenarios, and must also be easy to store, retrieve and update. Hence, the WOM of Web Smart things should be stored in a format so as to ensure,

1. easy accessibility for applications and other Web Smart things,

2. minimum footprint i.e. it should be light-weight for efficient storage, and

3. easy comprehension for people and things, i.e., it should be intelligible to end-users.

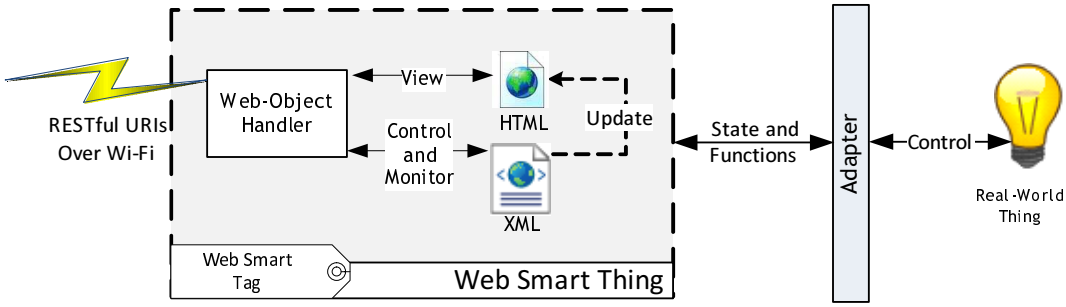


Figure 4.3: Transforming things into Web Smart things

As illustrated in Figure 4.3, the representation of states and functions of real-world things is stored in *XML* which ensures the exchange of structured data, and easy interoperability between Web Smart things. The XML contains the WOM of a Web Smart thing. The representation in *HTML* provides an enhanced perception of real-world things for users. The dynamic context of real-world thing is reflected in an XML and then the HTML is updated in real-time. The *Web-Object Handler* processes the requests for services of Web Smart things. The online access to these services is through unique RESTful URIs, directly referencing the thing. Since REST [32] is based on HTTP, the Web Smart thing must process HTTP requests and hence must host a Web server. For example, the URL[3] would trigger an actuator that turns a light *on* in a particular classroom. The *Adapter* transfers the state and function from a Web Smart thing to a real-world thing and from the real-world thing to a Web Smart thing. The *Web Smart Tag*, on the Web Smart thing provides a URL[4] to retrieve the XML file. The tag represents any technology like RFID, or NFC [86], used for wirelessly indicating the location of the XML file.

[3] *http://campus.com/floor1/classroom1/frontlights/On*
[4] *http://campus.com/floor1/classroom1/frontlights/WOMProfile*.xml

Research in RFID and NFC are advanced enough to realize this simple application [54, 86, 87]. For the purpose of our implementation we assume that the URL providing the location of the XML file describing the WOM of Web Smart things is available and accessible within an AS through these tags.

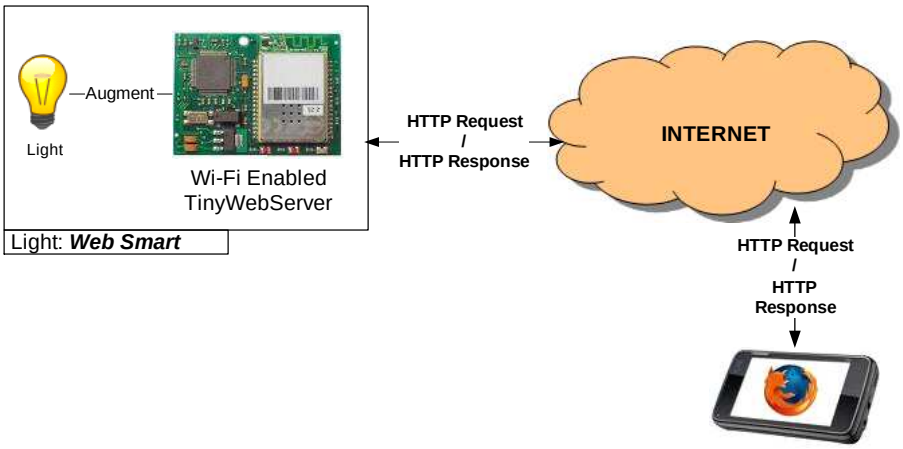


Figure 4.4: Light transformed into Web Smart Light

As illustrated in Figure 4.4 we realized Web Smart lights by augmenting a light bulb with a Wi-Fi enabled Tiny Web Server [88] module, which is 35X48 mm in dimension, has an integrated 802.11 Wi-Fi interface module, analog and digital I/O ports, and a 16-bit processor. The internal memory of 256KB is sufficient for the intended Web application. The module is configured to automatically connect to an available open Wi-Fi signal. The Web server parses the incoming URLs to determine the action and perform the corresponding operations. In our proposed framework the Ambient Space Manager module 4.2 exposes Web services to manage lights in a classroom as a mashup of Web-Object Handlers of light fixtures in a classroom. Dynamic Web pages access the analog ports which are augmented to the light to Web-enable it and raise its eligibility to *Web Smart*.

4.5 A Collection of Things

Our proposed AS framework integrates real-world things into the Web with Web service patterns like REST [32], and use the WOM to represent thing's semantics but there is need for a platform to enhance the large-scale integration, interoperability and sharing of things. The social networking platform has been found to be ideal for the composition and use of Web services [89]. Also, the adoption of a social networking platform to integrate real-world things was proposed, but the approach is restricted to privacy preservation of things and their users over a social platform [18]. There are yet many challenges to overcome to integrate things into the social networking platform. For instance, manually updating the increasing number of real-world things onto the social Web is impractical.

We target this issue by defining the *Social Web of Things* (SWoT), a community of Web Smart things where things interact with each other and also to other communities like those made up of people. The SWoT platform enables things to be identified, accessed, controlled and transacted with as a member of a community. Based on relationships that things have on the SWoT, new applications are enabled such as things inherently marketing, sharing and comparing themselves over the Web.

Several platforms, such as SenseWeb [90] and Cosm[5] have attempted to provide a social platform to share and integrate sensor feeds. However, these approaches are not designed to support decentralization, inherent integration and direct interaction with things, all of which are essential to the development of WoT. These existing ad-hoc social platforms require manual integration of things into the Web, whereas we advocate an integration solution with a one-time manual intervention to adopt things into the SWoT. As illustrated in Figure 4.5 Web Smart things represented by their WOM are grouped within AS. The use of AS allows the possibility to

[5] www.cosm.com

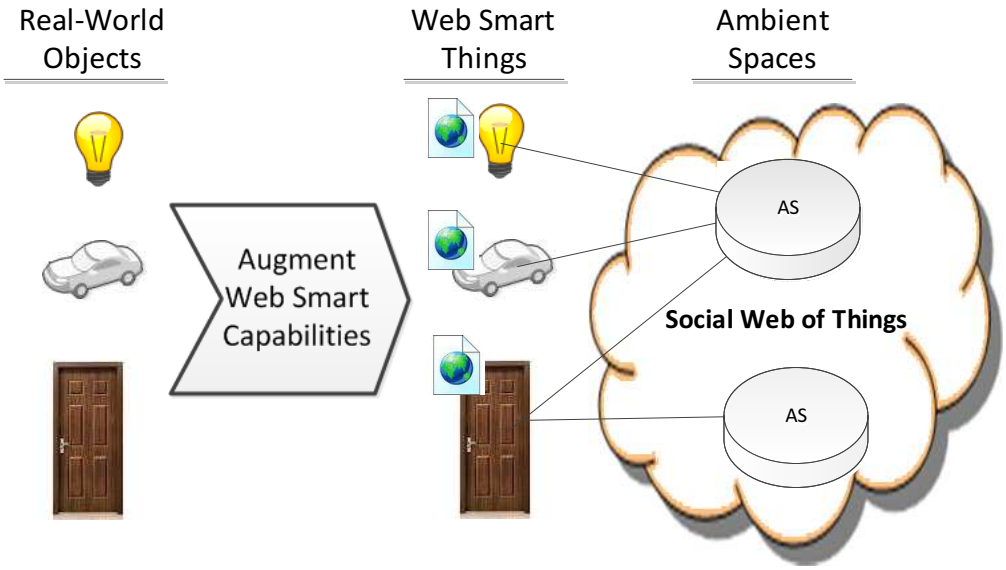


Figure 4.5: Social Web of Things created by mashup of Ambient Spaces

view things collectively, as a *community of things*, instead of viewing them as disparate things. The various AS together form the SWoT.

Our approach is to employ the use of Ambient Spaces AS to automatically cluster things together and then use the cluster members' social connections to drive things into social communities. To illustrate the reason for adopting this approach, we describe a scenario within a technology-enhanced learning environment. Consider that the AS within an application context of SWoT is a *Chemistry Lab*. The lab has a number of weighing balances spread across several venues in a campus, which are associated with faculty members who are aware about the availability and operational features of these balances. Some of these faculty members have defined certain schedules and restrictions for the use of these balances. Seven new Web Smart digital balances are ordered for the lab. When they arrive at one of the lab venues, they have to be manually registered with the system one at a time. This approach is cumbersome and not

scalable. Instead, we propose that the new balances first identify and associate themselves with their *own kind* i.e., join a group of similar balances on campus, and then send friend requests to faculty members associated with the older balances in the group. This scenario depicts how automatic integration of things is beneficial compared to manual interventions every time things are introduced into AS.

To achieve the integration of things into SWoT, they need to be grouped or clustered based on common attributes. We extend our concept of AS to group things through designated similarity criteria. These groups are used to identify and include new things that enter an AS and also to reach people using existing social links of group members. Despite their heterogeneous properties, we take advantage of the fact that real-world things may have induced similarities, which are revealed through their WOM-Profiles. We expound on our approach in the next chapter.

4.6 Summary

In this chapter, an extended WOM representation was presented. The software representations of Web Smart things within an XML structure which constitute the WOM-Profile was discussed. We presented the AS framework which functions as a gateway for WoT applications to connect to Web Smart things and we discussed the process of augmenting things to make them Web Smart. We discussed the need for developing a community of things and introduced our approach of using AS to cluster large number of things to create the *Social Web of Things* (SWoT).

In the next chapter, we discuss the process of categorizing things into the SWoT. We develop a model for thing's communities and propose a matchmaking algorithm, to incrementally populate the SWoT.

Chapter 5

Towards a Community of Things on the Web

In the previous chapters, we have described the *Ambient Space* (AS) framework that defines and manages the properties and functions of Web Smart things respectively. However, since things have inherent relationships with other things and with people, there is a concern over the unrestricted sharing and open access of things on the Web. The open access to real-world things results in security issues and privacy violation [91, 92]. Hence, a Web based platform that connects things and ensures the security and privacy of these things would be required.

Today, the social networking platforms on the Web or the *Social Web*, like Facebook, Google+, and LinkedIn have grown to bridge geographical and political boundaries to form virtual communities where people pursue goals and share mutual interests. These existing Social Web platforms have been suggested as a suitable environment to bridge the gap between the physical world and the Web [18]. The Social Web platform curtails security and privacy issues because members of these platforms are able to easily define access restrictions to their personal information or things and also have their trusted circle of associations. The processes

and methods of connecting real-world things on the Social Web have been well studied and described in the recent past [18, 93]. Though this approach of hosting real-world things on the Social Web provides, the possibility of reusing social associations that already exist in these platforms, and hence also ensures a level of privacy, this approach still has an issue. The enormity of candidate things reduces the efficiency of managing things on the Social Web; because things need to be registered one at a time into the Social Web or else they remain isolated and disconnected. It is impractical to have *owners* of things register things manually, one at a time.

To manage the plethora of isolated things, we propose the creation of *communities of things* based on inherent similarities among things and then adopt things into these communities [94]. These communities of things or the *Social Web of Things* (SWoT) function as a gateway to automatically launch the growing number of Web Smart things into existing Social Web platforms. We use SWoT to make it possible to initiate new and meaningful relationships between things and see how to use these relationships to automatically propagate things to people. We must note here that the discussion is far from exhaustive but highlights our continuing work towards creating communities of things.

The relationships that a community member (*thing*) has with people on the Social Web are used to infer new relationships for other things within the community. Thus the creation of *communities of things* provide a platform to extend and manage the social associations of things. SWoT automatically subsumes things, establish relationships, and provides the necessary platform to populate the WoT. To create communities of things, we exploit our concept of *Ambient Space* (AS) to instantiate group structures where related things are clustered together as communities [94] and thereby expanding the possibility of populating things into the WoT.

In this chapter, we model the adoption of things into SWoT using our proposed AS framework to realize the ubiquitous interconnection of things on the Web. SWoT promises the

creation of coherent communities of Web Smart things bridging them with people on the Web. SWoT is proposed as a platform to propel things to thrive in the future WoT. In Section 5.1 we describe the concept of the SWoT and the different layers of this platform. In Section 5.2 we describe how AS is used to create a community of things and suggest two types of communities. The process of modeling communities of things is described in Section 5.3. We discuss issues of creating communities when sufficient information is not available in Section 5.4 and finally we provide concluding remarks in Section 5.5.

5.1 The Social Web of Things

The success of existing Social Web platforms is dependent on the possibility of creating and populating communities based on social associations between people. The Social Web presently connects people together into communities by comparing similar attributes between people. Our aim is to extend this approach to Web Smart things in AS. A possible application of such a community of things would be that commercial things would autonomously promote themselves among potential buyers, i.e., market themselves to people through *social adoption*. We define *social adoption* as a process wherein things initiate associations with people.

The SWoT layers are illustrated in Figure 5.1, where the lines represent associations and *Comm 1* to *Comm n* are communities of people in existing Social Web platforms. Each AS represents a community of things. The *Ambient Space Manager* (Figure 4.2) module of our proposed AS framework, handles the process of group creation and social adoption. We propose two stages for realizing community of things,

1. Grouping[1] similar type of things into AS, and

[1]The terms *Grouping* and *Clustering* are used synonymously.

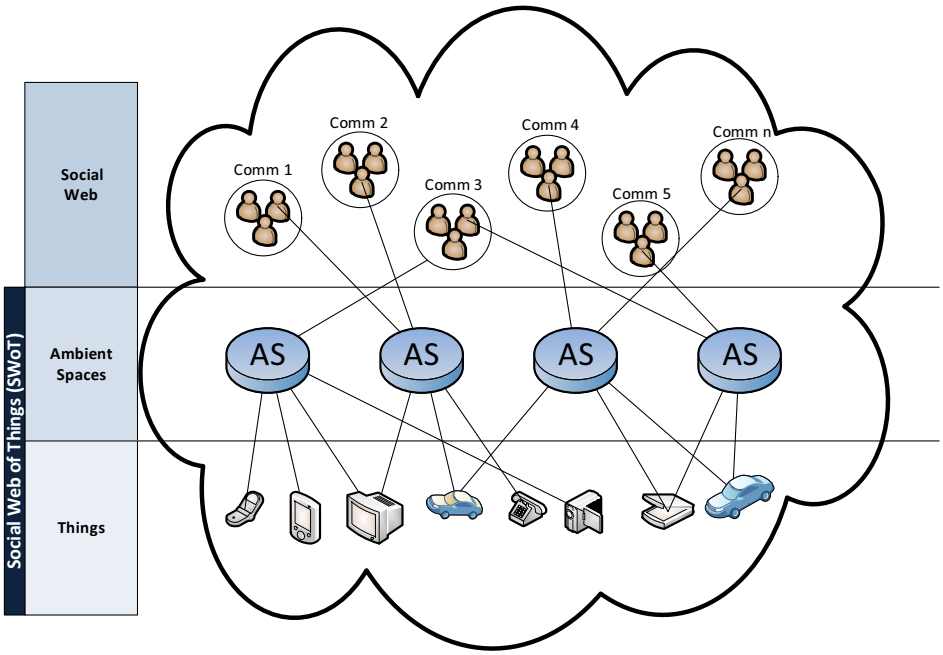


Figure 5.1: Connecting People and Things through Ambient Spaces

2. Linking things in AS to people in the Social Web

The details of the two stages are described in the next section. For the first stage we need to define *similarity* between things. In the second stage, we use opportunistic relationships that some things in the group have with people to dynamically suggest peers to people in existing Social Web communities. The two stages are described in Section 5.2. The rationale of the AS layer is to maximize the chances of a thing to match contextually similar peers, which have already been adopted by members of the Social Web, and use this membership to adopt new relationships between people and things [94, 95]. The associations to the Social Web communities are enabled through the AS layer which represents communities of things.

5.2 Building a Community of Things

The challenge of building AS to represent a community of things is twofold. Firstly, *Web Smart* things need to be compared to determine similarities so that they are assigned to appropriate AS. Deciding the parameters for comparing thing's similarities is a complex task as they need to be relevant within an application context to ensure that AS provides the necessary semantic coherence between grouped things. Secondly, it would be impractical to assume that all the required parameters of candidate things are available or accurate. To address these challenges, we make the following assumptions:

- We confer similarities between candidate things, by comparing respective WOM-Profiles. We assume that two things are similar when their WOM-Profiles are similar. As described in the previous chapter, the WOM-Profile of a thing provides the semantic description of a thing. WOM-Profiles are represented as XML documents. Two methods are adopted to compare WOM-Profiles. First, the *structure* of the two XML documents i.e., the tags themselves and the hierarchical arrangement of the XML tags are compared. Secondly, the *content* of these tags are compared to determine if things are similar. For example, consider the XML tag `<dish:manufacturer> Omida </dish:manufacturer>` in a WOM-Profile. If the WOM-Profiles of two things have the same tag `"<dish:manufacturer>"` and the same content `"Omida"` we reason that there is a high probability that the two things are similar. In our approach, we take into consideration the hierarchy of tags as a path structure, while comparing WOM-Profiles . The comparison is made with the complete WOM-Profile of two things, including the tags and the content.

- We employ *semi-supervised clustering* approach to group Web Smart things into clusters. In our approach, for semi-supervised clustering, we start by selecting one *predefined seed*

(thing) for each cluster. This is a one-time process initiated by an administrator of the SWoT who has knowledge of candidate things that need to be clustered. The process is initiated on a static repository of things that are candidates for the clustering. We then run a clustering process from the Ambient Space Manager module, where all candidate things are compared with all predefined seeds and the best fit among the available clusters is determined. We discuss the approach in Section 5.4.

- For the clustering process, we assume that WOM-Profiles are complete and have accurate information that represents a thing's state, functions, and social connections. We discuss the implications of incomplete or inaccurate WOM-Profile later in Section 5.4, where we propose an approach for *unsupervised clustering* processes.

- We assume that the Ambient Space Manager locates the WOM-Profiles of candidate Web Smart things in an AS by navigating to the URL that is available on Web Smart Tags as explained in the previous chapter (Section 4.4).

As discussed in the previous chapter, the WOM-Profile is divided into the `<wom:preset>` part and the `<wom:dynamic>` part (Table 4.1). The former part describes fixed or static information of *Web Smart* things introduced by its producers or manufacturers. The latter part describes the changing information of a *Web Smart* thing introduced by consumers. The two parts represent two genre of information i.e., the `<wom:preset>` part describes a thing with properties like capabilities, dimensions, identifications, and handling instructions, whereas the `<wom:dynamic>` part describes other aspects of a thing like, state, location, social connections, and user comments. Our approach to subsume things into communities, creates AS of two types, one for each stage of adoption into the Social Web. Figure 5.2 illustrates the two types of AS where things T_1 to T_m are integrated within AS communities represented by AS_1 to AS_l. The first stage

creates AS based on `<wom:preset>` and the second stage creates AS based on `<wom:dynamic>` part of the WOM-Profile. Next, we explain these two stages.

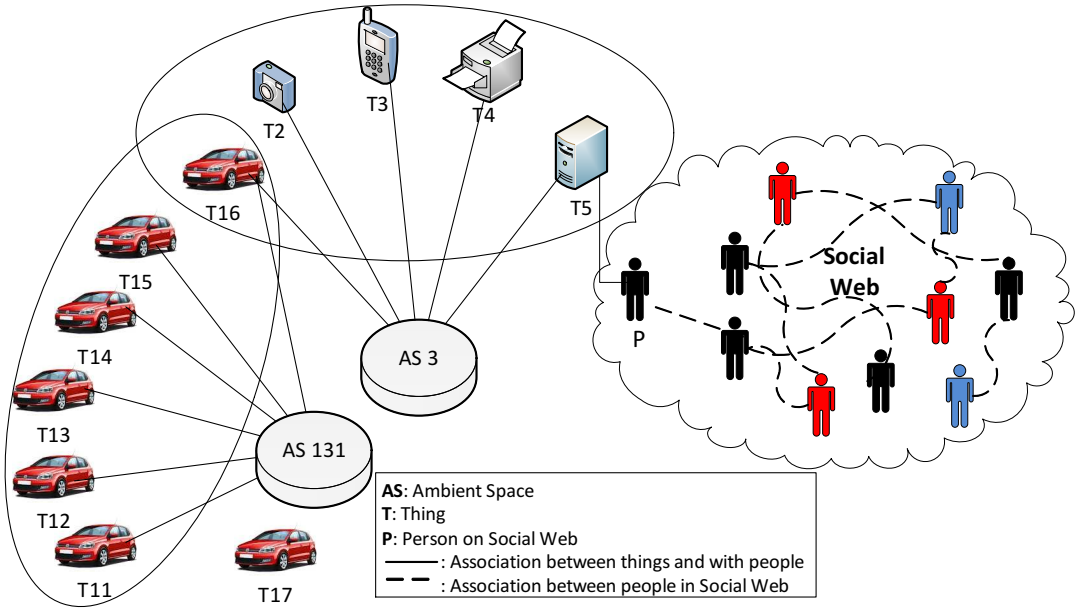


Figure 5.2: Two types of Ambient Spaces

5.2.1 Grouping Similar Type of Web Smart Things

In this first stage, we create AS like AS_{131} in Figure 5.2. AS_{131} community represents a cluster of things that are similar i.e., T_{11} to T_{16}. For example, the similarity attributed to create AS_{131} could be the same *manufacturer* which is an attribute found in the `<wom-preset>` part of the WOM-Profile. Clusters like AS_{131}, created in the first stage, allows the automatic categorization of a new thing like T_{17} into a community like AS_{131} that has things similar to T_{17}. These groups are initiated with at-least one member (*manually inserted*) which acts as a seed or centroid to adopt future members. Similarities that exist between the preset parts (`<wom:preset>`) of

the things WOM-Profiles are used to create clusters around the predefined seed. During a clustering process, every new thing that is Web-enabled but not in an AS is adopted into an AS by comparing similarities of its WOM-Profile (`<wom:preset>`) with the available cluster seeds. If a thing is not matched into any cluster, then the thing would be adopted during another clustering cycle or manually administered as a new seed to form a new AS. The seed in each AS provides a common representation for the similar things that are grouped within them. The clustering process ensures the re-election of the seed i.e., the clustering process may change the centroid of the cluster, we explain this in Section 5.3.3. Hence, over a period of time the seed becomes a purified representation of a cluster. This stage is executed first, to ensure that all possible candidates are part of existing communities. This ensures that a new community is formed only after it is initiated by authorized personnel, thus providing a level of control whereby things are not allowed social associations with people if they do not match into an existing community of things. For example, considering the group of weighing-balances in the Chemistry Lab (Chapter 4), within the context of that application the newly procured weighing-balance would first be adopted into an existing group of weighing-balances on campus, before connecting with the faculty members on campus.

5.2.2 Grouping Web Smart Things with Similar Social Relationships

In the second stage, we focus on similarities between things like T_{16}, T_2, T_3, T_4, and T_5, to create a community like AS_3. Our approach suggests relationships for all things in $AS_3$with person P because of existing relationship between a thing (T_5) in the community and P. The associations that P has within the Social Web is exploited to determine possible adoption of things in AS_3. Such associations are bound to introduce privacy and security issues and the various possibilities are discussed in [18]. AS_3 community integrates things based on the

similarity in the `<wom-dynamic>` part of the WOM-Profile, for instance, the same *owner*. The clustering process is also used to project relationships with people in the Social Web. In this stage, only things that are part of existing communities created in stage 1 (see Section 5.2.1) are considered.

These AS are created with at least one socially-connected member, for example T_5 (*manually inserted*), which acts as a seed or centroid for nurturing social connections of future members. The clustering process matches the WOM-Profiles (`<wom:dynamic>`) of candidate things and groups them into AS. If a new thing is not matched into any cluster i.e., none of the WOM-Profiles of cluster seeds match with the new thing's WOM-Profile, then the thing is manually administered as a new seed to form a new AS. Social connections to things are suggested to members (people) on the Social Web based on the social connections of the seed and in subsequent iterations the social connections of other things in the AS are also used. Hence, the clustering process here creates AS of things which are related to social parameters like `<wom:owner>` or `<foaf:Person>` found in the `<wom:dynamic>` part of the WOM-Profiles. These social similarities are used to suggest *friend* relationships to people on the Social Web.

Hence, SWoT furnishes a platform for people and real-world things to coexist. Here, we do not explain the process of connecting to a specific social networking platform as this has been studied and demonstrated previously [18, 93]. Instead, we model the creation of communities that automate the process of identifying associations in the Social Web and also reduce the overhead of manually registering all things with social networking platforms.

5.3 Modeling a Community of Things

Here, we explain our proposed clustering process and describe how similarity functions are used to create communities of things. The set of things considered are confined to an AS and a corresponding set of WOM-Profiles $W = \{w_1, w_2, \ldots, w_m\}$ represent the things within the AS. We structure our description as follows:

1. We explain the different stages of creating clusters using a flowchart.
2. Next, we explain our similarity model with examples to describe how similarities between things are measured.
3. We then explain the clustering algorithm, where things are compared and grouped into AS based on their similarity.
4. Finally, we provide an illustrative application with an evaluation that highlights the benefits of such communities.

5.3.1 Grouping Things into Ambient Spaces

Our proposed clustering approach is illustrated in Figure 5.3. The preset and dynamic parts of the WOM-Profiles of Web Smart things are initially separated as shown in Figure 5.3 (a). This is followed by the clustering process shown in Figure 5.3 (b), which is executed separately for the preset profile and the dynamic profile. The clustering process starts with the Profile Pre-Processing stage where the *structure* and *content* are separated. The *structure* includes all the XML tags in the WOM-Profiles. These are extracted from the WOM-Profile to create the structural model. The structural model is represented as paths which contains the tags in a hierarchical order. This process is explained with examples in the following section. Similarly

the *content* includes all the text or values that are within the XML tags. The content model is represented as unique terms with stop-word removal and stemming [96], where stop words like, "is", "it", or "a" are removed and unique terms are extracted.

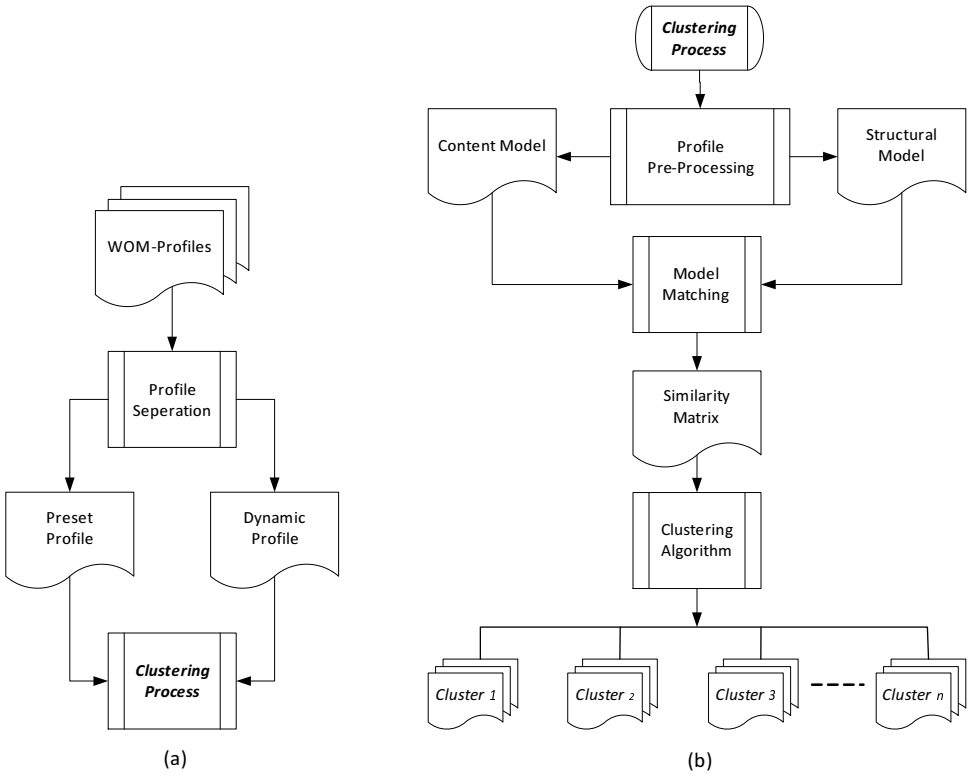


Figure 5.3: Clustering Web Smart things into Ambient Spaces

The content model and the structure model are then separately measured, and respectively matched in the model matching process, as shown in Figure 5.3 (b). The results of the matching process are combined to form a pair-wise similarity matrix which depicts the similarity between each pair of candidate WOM-Profiles. The clustering algorithm uses the similarity matrix to group things into separate clusters. We adjust the participation of structural comparison and

content comparison by providing configuration parameters. The use of structural similarity with content similarities produces more relevant clusters in heterogeneous environments such as those represented by WOM-Profiles [97]. Next, we explain similarity measures used to create the Similarity Matrix and then expound on the Clustering Algorithm.

5.3.2 Modeling the Similarity of Web Smart Things

As described previously, the similarity between things is determined by separately considering the preset and dynamic parts respectively for the two stages of thing's adoption into the SWoT. To determine the similarities between WOM-Profiles, we compare both the structure and relevant content of the XML descriptors in the profile.

Accurate clustering requires a precise definition of the closeness between a pair of things, in terms of either the similarity or the distance. A variety of similarity and distance measures have been proposed and widely used, like cosine similarity and the Jaccard correlation coefficient. Similarity is also conceived in terms of distance [98]; the Euclidean distance and relative entropy has been applied in clustering to calculate pair-wise distances. Next, we describe the measures that we use to compare the similarities between the WOM-Profiles of things, so as to cluster things.

a) WOM-Profile Similarity Measure

Here we describe the measure that our approach adopts to determine the similarity between WOM-Profiles. Our approach measures the *content* and the *structural* similarities separately and then combines the results with different configuration parameters. This gives *relative importance* to the structure and content, depending on the type of things that are considered. Moreover, the structural and content similarity is measured separately for the preset and dynamic parts of the WOM-Profile.

For the preset part (`<wom:preset>`) of a WOM-Profile, the content similarity between two WOM-Profiles w_x and w_y is defined as $contPreSim(w_x\,,\,w_y) \times \delta$. The structural similarity between w_x and w_y is defined as $structPreSim(w_x\,,\,w_y) \times (1-\delta)$. δ is an AS configuration parameter and the value ranges between 0 and 1 to determine the relative importance of *contPreSim* (similarity of preset content) and *structPreSim* (similarity of preset structure) in the similarity measure.

Likewise, in the dynamic part (`<wom:dynamic>`) of two WOM-Profiles w_x and w_y , the similarity of content is defined by $contDynSim(w_x\,,\,w_y) \times \varepsilon$, and similarity between structures is defined by $structDynSim(w_x\,,\,w_y) \times (1-\varepsilon)$. ε is also an AS configuration parameter, the value of which ranges between 0 and 1, and determines the relative importance of *contDynSim* (similarity of dynamic content) and *structDynSim* (similarity of dynamic structure) in the similarity measure between WOM-Profiles.

Hence, equations 5.1 and 5.2, define the similarity between the preset parts and dynamic parts of the WOM-Profiles respectively.

$$profPreSim(w_x, w_y) = (contPreSim(w_x, w_y) \times \delta + structPreSim(w_x, w_y) \times (1-\delta)) \quad (5.1)$$

$$profDynSim(w_x, w_y) = (contDynSim(w_x, w_y) \times \varepsilon + structDynSim(w_x, w_y) \times (1-\varepsilon)) \quad (5.2)$$

The use of δ and ε helps to create communities that have closely associated semantic space which reflects major associative patterns and reduces the effect of less important patterns. For example, in equation 5.1, if δ is assigned 1, then the similarity is measured only based on content and not on structure as $structPreSim(w_x, w_y) \times (1-\delta)$ evaluates to 0. If equal participation

of both content and structure is required then, δ is assigned 0.5. Using the similarity measure in equation 5.1, a pair-wise WOM-Profile comparison is done to generate a similarity matrix for clustering things into AS. Equation 5.2 is used for pair-wise comparison of WOM-Profiles to generate a similarity matrix for possible inclusion into a social community. A clustering algorithm such as k-means is applied to determine clusters from the similarity matrix. Next, we describe how the similarity is determined for the structure and content of WOM-Profile.

b) WOM-Profile Structural Similarity

The structural similarity of the WOM-Profile depends on how the profile is organized, which is represented by the hierarchy of XML tags. As mentioned earlier, we assume two things to be similar if their WOM-Profiles have similar XML tags and structure. Information such as element names, data types, and hierarchy of elements is used to determine the structural similarity between WOM-Profiles. The elements in the profile are organized in a tree structure. We match the structure of WOM-Profiles by dividing the profile into distinct paths. These paths are used to measure structural distances between the profiles. Given a set of WOM-Profiles $W = \{w_1, w_2, \ldots, w_m\}$, a set of distinct paths $P = \{p_1, p_2, \ldots, p_f\}$ are extracted from W as described next.

A path p_i contains element name from the root element $\{R\}$ to the leaf element which hosts the content. The structural model of a profile w_i is a vector $\{p_{i,1}, p_{i,2}, \ldots, p_{i,f}\}$, where each element of the vector represents the frequency of a path in P that occurs in the WOM. Finally, given two profiles w_x and w_y, and their corresponding vectors $\{p_{x,1}, p_{x,2} \ldots, p_{x,f}\}$ and $\{p_{y,1}, p_{y,2}, \ldots, p_{y,3}\}$ respectively, the distance between the two profiles is computed using the Euclidean distance. Equation 5.3 evaluates the structural similarity considering the preset part of WOM-Profiles.

$$structPreSim(w_x\,,\,w_y) = \left(\sum_{i=1}^{f}(p_{x,i} - p_{y,i})^2\right)^{\frac{1}{2}} \tag{5.3}$$

The Euclidean measure is a standard metric and is widely used for clustering problems. It is also the default distance measure used for the k-means algorithm [99], which we use for the clustering process.

Example: Let us assume a set, W containing four WOM-Profiles $\{w1, w2, w3, w4\}$ as shown in Figure 5.4, where w_1 and w_4 a have similar structure. Each profile has the root element $< R >$ and the leaf elements $< L_i >$ hierarchically represented. The content of the profiles are represented as $T = \{t_1, t_2, \ldots, t_q\}$ which are q distinct terms that make up the content.

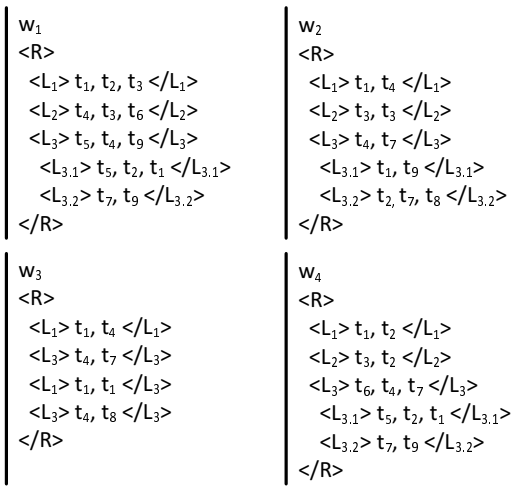


Figure 5.4: Structure and Content Representation of four WOM-Profiles

The structure is processed and represented as a vector. The structure of all the profiles is put together as a path matrix $M_{f\times n}$, where f is the number of distinct paths in P and n is the number of profiles in W, as shown in Table 5.1. Each element of M represents the frequency of a distinct path appearing in a profile.

Paths	w_1	w_2	w_3	w_4
$R-L1$	1	1	2	1
$R-L2$	1	1	0	1
$R-L3-L3.1$	1	2	0	1
$R-L3-L3.2$	1	0	0	1
$R-L3$	1	1	2	1

Table 5.1: Path Matrix for Structural Model

c) WOM-Profile Content Similarity

The content similarity considers the value (content) of the various XML tag elements in the WOM-Profiles that are being compared. From our example in Figure 5.4 a set of distinct terms from $T = \{t_1, t_2, \ldots, t_q\}$, is extracted from the profiles $W = \{w_1, w_2, w_3, w_4\}$. A term matrix, $S_{q \times n}$ where q is the number of distinct terms and n is the number of profiles in W is constructed as shown in Table 5.2. The first column in Table 5.2 lists $T = \{t_1, t_2, \ldots, t_q\}$, which forms the corpus of distinct terms from the WOM-Profiles, which are used to determine content similarity. Each of the subsequent columns in Table 5.2 is a vector representing the frequency of term occurrence within respective candidate WOM-Profile.

Terms	w_1	w_2	w_3	w_4
t_1	2	2	3	2
t_2	2	1	0	3
t_3	2	2	0	1
t_4	2	2	3	1
t_5	2	0	0	1
t_6	1	0	0	1
t_7	1	2	1	2
t_8	0	1	1	0
t_9	2	1	0	1

Table 5.2: Term Matrix for Content Model

A term t_i that appears in the content of the WOM-Profile is a word that is retained after the stemming process and removal of stop-words [96]. The content model of a profile w_i, is modeled as a vector $\{t_{i,1}, t_{i,2}, ..., t_{i,q}\}$, where each element in the vector represents the frequency of a term that appears in the document. Once the profiles are represented as term vectors the similarity of the two profiles corresponds to the correlation between the vectors. This is measured as the cosine similarity [100]. An important property of the cosine similarity is that it is independent from the length of the profile. As the lifespan of a Web Smart thing progresses, its history also increases which is included in the dynamic part of the WOM-Profile. Hence, the dynamic part of the WOM-Profile could be very long, and therefore the cosine similarity would be appropriate for measuring the similarity of the dynamic profile.

Given two WOM-Profiles w_x and w_y, the cosine similarity is:

$$contPreSim(w_x, w_y) = \frac{w'_x.w'_y}{|w_x| \times |w_y|} = \frac{\sum_{i=1}^{n} w_{x,i} \times w_{y,i}}{\sqrt{\sum_{i=1}^{n}(w_{x,i})^2} \times \sqrt{\sum_{i=1}^{n}(w_{y,i})^2}} \tag{5.4}$$

In equation 5.4, w'_x and w'_y are the term vectors for the respective profiles. Equation 5.3 is used for both the preset and the dynamic profiles when determining content similarities.

In the clustering algorithm, things are grouped based on a notion of "distance" or "similarity". While equation 5.4 (measures the content similarity) uses the cosine *similarity* function, equation 5.3 (measures the structural similarity) uses the Euclidean *distance* to measure the distance between things' WOM-Profiles. These measures have to be combined to be used in the k-means clustering algorithm. The Euclidean distance is the default measure of closeness that the k-means algorithm uses to cluster things into an AS. To enable this feature for our use of k-means algorithm, we convert the cosine similarity measure into a distance value using $\{Dist = 1 - Sim\}$ where, Sim is the similarity measure calculated using the cosine similarity function (equation 5.4). Thus, the clustering algorithm is fed the pairwise similarity matrix

which is created using equations 5.1 and 5.2 for preset or dynamic cluster creation respectively.

5.3.3 Clustering Algorithm

In our approach to cluster real-world things, we propose the use of k-means algorithm [101, 102] to compare and cluster the WOM-Profiles of Web Smart things. The k-means algorithm is iterative and commonly used for clustering in applications with large data sets. There have been various attempts to improve the efficiency of k-means algorithm [103]. Here, we are not focusing on the efficiency of the algorithm but rather on the possibility of creating clusters from large data sets that the WoT would generate. We propose a more efficient approach in Section 5.4, where we discuss the issue of incomplete WOM-Profiles. The k-means algorithm takes the similarity matrix as input and generates clusters of things represented by their WOM-Profiles.

k-means algorithm:

The algorithm is used to partition a given set of WOM-Profiles into k clusters. Here, the algorithm is applied to the grouping of things that are represented by an n-dimensional vector space. The algorithm clusters a set of vectors that represent WOM-Profiles, $W = \{w_j | \, j = 1, ..., m\}$ where $w_j \in T_m$ which is the m things in an AS.

In k-means, each projected cluster (AS) is represented by a cluster centroid or *seed*. We denote the set of cluster representatives as $C = \{c_i | i = 1, ..., l\}$. The k-means algorithm attempts to minimize the total *Euclidean distance* between each thing w_j and its closest seed c_i. The objective *(Obj)* of the algorithm is represented in equation 5.5, which is determined by minimizing the sum of squared errors.

Algorithm 5.1 k-means algorithm to create clusters of things.

```
01: Input:
    02: - Set of WOM-Profiles W
    03: - Pair-wise similarity matrix
    04: - the number of clusters k
05: Output:
    06: - Cluster representatives C
    07: - Cluster membership vector M
---------------------------------------------------------------
    08:  - Initialize k seeds from W
    09:  - Use these k seeds as initial set of cluster representatives C
    10: Repeat
        //Step1: Things Assignment
    11:  - Calculate cluster mean c_j.
    12:  - Reassign things in W to the closest cluster mean.
    13:  - Update M such that m_i ∈ M is the cluster ID of w_i.
        //Step 2: Re-election of seeds
    14:  - Update C such that c_j is mean of points in j^th cluster.
    15: Until convergence of objective function equation 5.5
---------------------------------------------------------------
```

$$Obj = arg_c min \sum_{i=1}^{l} \sum_{w_j \in C_i} ||w_j - c_i||^2 \tag{5.5}$$

Algorithm 5.1 describes the steps used by k-means to clusters W (line 02) iteratively alternating between two steps, i.e., reassigning the cluster-id of all points in W, and updating the cluster seed based on things in each cluster. The algorithm takes the pair-wise similarity matrix as input (line 03) to create clusters. This matrix is created by the pair-wise comparison of WOM-Profiles of things. The value of k (line 04) is an input to the algorithm, where $k \leq m$.

In general, the value of k is suggested based on prior knowledge of number of clusters that would appear in W. Here, initially we adjust k manually to the number of seeds we are considering for cluster creation in a given application. Subsequent runs of the algorithm would take the number of available clusters as the value of k and adapt new things into the available clusters. In our approach, the need for an additional cluster occurs when new things cannot be grouped into any of the existing clusters. In this case, the value of k is incremented with the number of new seeds. The algorithm works as follows.

First the k seeds are initialized (line 08). The algorithm then iterates between the following two steps until convergence equation 5.5 (lines 10 - 15).

Step 1- Things assignment (lines 11 - 13): Each thing is compared and assigned to its closest seed. This results in partitioning available things into clusters. The cluster membership vector M is also generated (line 13). The closeness to the cluster mean is decided based on a threshold value that is manually assigned for each AS.

Step 2- Re-election of seed: The single best representative for a cluster is then elected as the seed (line 14). The algorithm minimizes distortion, which is the sum of the squared Euclidean distances between each thing $w_j$and its closest seed c_i. Each step of the k-means algorithm refines the choices of seeds to reduce distortion.

The algorithm converges when the assignment values no longer change (line 15). The objective function described in equation 5.5 reduces as there is a change in the assignment and re-election steps. The convergence is guaranteed in a finite number of iterations.

The algorithm is strongly dependent on the selection of the seeds (k) and also convergence of the algorithm is to a local minimum. Repeating the algorithm for a number of times could solve the problem of local minimum, but still the seeds need to be selected carefully by one that has knowledge of the WOM-Profiles or things represented. If expertise to select k is not

available, then a simple and naive approach to get around this limitation is to run the algorithm repeatedly with different seeds till suitable clusters are formed. In this approach once the initial seeds are selected by an expert and suitable clusters are created, then grouping new members into existing clusters becomes easier. With more similar members in a cluster, the purity of the cluster increases. Next, we discuss an example where pure clusters are beneficial for a business scenario.

5.3.4 Benefits of Clustering Things into Communities

The behavior of a thing within the proximity of other things alters their group dynamics and facilitates business functions like decision-making, selection, tracking and logistics. With the propagation of Web Smart things, a study of thing's behavior in groups mandates the effectiveness of future businesses. We present an illustrative scenario to depict the benefit of creating communities of things as explained above. The scenario has two parts where community of things directly affects the future of supply chain management and product marketing. We also evaluate one of the benefits of creating things communities when considering future marketing trends where things promote themselves to potential customers inherently.

a) Marketing Scenario

Sam's Household Ltd; (*Sam's*) is a chain of retail stores that sells home appliances around the country. The business decided to invest in a new Web based application that manages the functions of the store which is built on a social networking platform called *Sam's Ambient Space Manager (SASM)*.

Things group themselves: In the morning a container with 300 boxes arrived at Sam's. The store manager receives a *friend request* from a social community in SASM called *Dish5A3_ Series*. The store manager inspects the request to see that there are 49 members in the community

where each member is a new dishwasher and realizes that they are part of a shipment of 60 dishwashers that were ordered last week. The store manager accepts the friend's request and authorizes the sales team to start the sales while the container gets unloaded, sorted, and arranged in the store within the next few days.

Things market themselves: The *Gibbs* family is a regular visitor to Sam's. They bought a dishwasher from Sam's a few years back and are planning to buy a new dishwasher. Mr. Gibb receives a *friend request* to join a new community in SASM called *Dish5A3_Series*. Mr. Gibb inspects the friend request and is immediately taken to a Web page to view the details of the new dishwasher at *Sam's*. He is presented with the specification to study the product. He also compares with other models that are available in SASM. Mr. Gibbs accepts the request and tags one of the dishwashers with "I like the black one!". Social networking dynamics in SASM automatically sends an invitation to Mrs. Gibbs to check out her selection.

The above scenario represents an ideal situation where our approach and framework is implemented and used. In the first case it would have ideally taken three to four days to unload, scan, sort and arrange 300 boxes before the store can start the sale of the merchandises in the container. Moreover, the order was placed for 60 dishwashers and only 49 were received. We see here that the lead time to initiate sale is reduced drastically as the dishwashers automatically grouped themselves to an existing community of dishwashers in SASM and declared their presence to the store manager. While in the second case we see that the dishwashers have started marketing themselves among potential customers while they are being unloaded. The information for contacting Mr. Gibb was retrieved from existing members (dishwashers) of a group where Mr. Gibb's present dishwasher is a member. Such scenarios boost the marketing efforts of businesses as things autonomously market themselves. With WoT domain maturing, we foresee a complete makeover of business functions like supply chain management and

marketing.

b) Evaluation

Things marketing themselves on the social platform create a viral phenomenon with the plethora of things autonomously contacting potential clients. The ability of things to cluster themselves into communities would reduce the number of messages that are sent out as only one member of each cluster needs to send out promotional messages. This is ideal if all members of a cluster are strongly related to each other. Our similarity functions and clustering algorithm enable the creation of *pure* clusters.

Purity measure shown in equation 5.6 is commonly used in clustering to evaluate the coherence of a cluster. It measures the degree to which a cluster contains similar Web Smart things from a single category. For an ideal cluster, which only contains things from a single category, its purity value is 1. In general, higher the purity value better is the quality of a cluster. Every time a new candidate is added to a community of things, we calculate the purity of the community. The purity measure is calculated as follows.

$$Purity(C_i) = \frac{1}{n_i} max_h(n_i^h) \tag{5.6}$$

In equation 5.6, C_i is a particular cluster with size n_i and $max_h(n_i^h)$ is the number of things that are from the dominant category in the cluster C_i. The term n_i^h represents the number of things from cluster C_i assigned to category h.

Since our research domain is still in very initial stages, it is difficult to get a real-time sample data of Web Smart things and therefore we decided to simulate the generation of the WOM-Profiles for the purpose of evaluation. For our simulation, we first created a corpus of terms for the content of the WOM-Profiles. Next, we generated the `<wom-dynamic>` part of the

WOM-Profiles by randomly picking values from the dictionary where each profile represented a Web Smart thing. Next, using our model we generated clusters by matching profiles that had similar `<wom-dynamic>` content (see Section 5.2.2).

We run an experiment for the scenario where things market themselves in a Social Web context, and we observe a considerable reduction in the number of messages that would be sent out when clustering is employed, as the number of clusters created increases in a graceful gradient. For our simulation we varied the number of things between 100 and 1000. The data size is reasonable when considering things within an Ambient Space (AS), for example things in a warehouse or a showroom. Using a number of iterations the average number of clusters created was calculated. Our simulation suggests the benefit of creating communities of things to enable business functions like marketing. Figure 5.5 illustrates the comparison of things sending out messages versus messages going from clusters to potential customers.

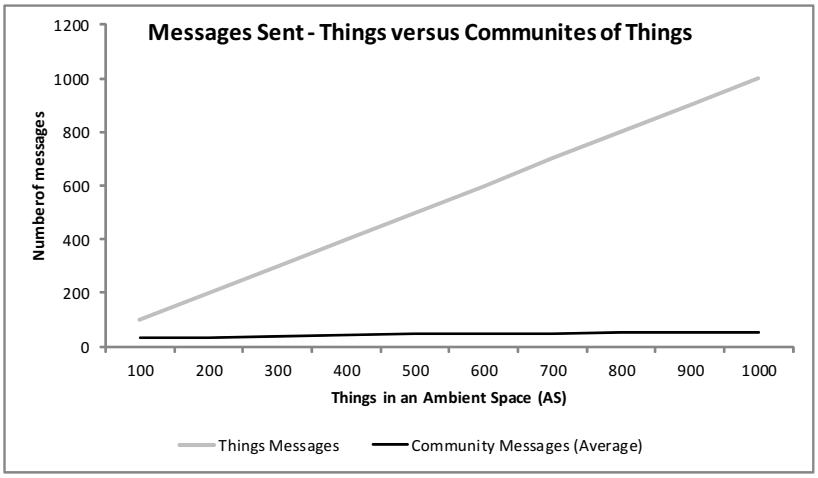


Figure 5.5: Comparing cost of communication: Community versus Individual Things

5.4 Handling Incomplete WOM-Profiles

WOM-Profiles provide semantic and virtual representations of real-world things on the Web. In our approach described above, we used the k-means algorithm to cluster things represented by their WOM-Profiles. Research on k-means algorithm has been extensive and is still active. We discuss the algorithm to provide a more efficient approach to clustering things. We must note here that the discussion is far from exhaustive but highlights our continuing work towards creating communities of things.

Clustering is usually referred to as unsupervised learning. Unsupervised learning assumes that there is no dependent random variable, output, or response i.e., a set of input observations is gathered and treated as a set of random variables and analyzed as is [104]. Since there is no use for tagged information, unsupervised learning is typically used for applications where the tagged information is sparse or missing. Supervised learning methods require things to be well tagged with information. In our algorithm we have utilized the semi-supervised approach where there are a few things tagged with information and these function as seeds to initialize clustering. Semi-supervised learning algorithms are a good compromise between purely unsupervised and purely supervised methods. The k-means algorithm is an iterative clustering algorithm that partitions given set of things into a set of clusters, k. The k-means algorithm which is typically an unsupervised learning method, does not provide control over the final cluster creations. In our approach we introduced initial seeds and thus altered it to function as a semi-supervised algorithm. This allows more likelihood of creating clusters that correspond to the introduced seeds. There is a large dependency on the initial selection of seeds. Also, the algorithm only groups things into a single cluster, but to increase the chances of things belonging to separate clusters each thing in a cluster can be associated with a weight which indicates the level of its

participation in a given cluster.

In our approach above, we use AS to group and adopt things to SWoT where we considered the WOM-Profiles to be complete and accurate. In reality this might not be the case always as there is a possibility of missing content and hence the WOM-Profiles would be sparse.

5.4.1 Assumptions to Address Sparse WOM-Profiles

We assume that the `<wom:preset>` part is accurate since it is fixed and appended to a thing's profile by the producers of a thing and is ideally tamper proof. Using our approach this assumption enables the creation of AS with similar things as mentioned in Section 5.2.2. However, the process of creating AS to suggest social connections with people would be vulnerable to missing or incomplete information in the `<wom:dynamic>` part of the WOM-Profile since, it is not fixed and subject to end-user interactions. The possibility of creating clusters to connect things to social communities as mentioned in Section 5.2.2 diminishes if the information in `<wom:dynamic>` part of the profile is missing. Here we propose to use unsupervised clustering when there is sparse information or missing information in the `<wom:dynamic>` part of the profiles. We propose the use of Random Indexing (RI) K-trees [105], which is adapted for our particular use. Next, we explain the process and also our proposed adaptation.

5.4.2 Adapting the K-Tree and Random Indexing Approach

The k-tree approach is suited for clustering large collections candidate things with low time complexity O(n log n) for building k-tree. Moreover, k-tree adopts new things as they arrive with a low time complexity of O(log n) for insertion into the tree. It also creates a hierarchy of clusters which the k-means do not. The k-tree approach is designed to operate with dense vectors and not with sparse. When sparse vectors are used, the performance of the k-tree algo-

rithm diminishes even though the amount of information is lesser. Since the vectors generated by WOM-Profiles have very high dimensionality this becomes a very expensive process. The approach adopted is to increase the density of the vectors that are used by k-trees.

RI is an approach to increase the density of sparse vectors [106]. RI is a scalable and incremental approach for the implementation of profile models to increase the density of sparse WOM-Profiles. We use the distribution of terms in WOM-Profiles to create high dimensional vectors and the directions of these vectors represent various semantic meanings and contexts of real-world things. Random Indexing first creates random context vectors of lower dimensionality and then combines them to create a term occurrence matrix in the dimensionally reduced space. Each term in the collection is assigned a random vector and the document term occurrence vector is then a superposition of all the term random vectors. There is no matrix decomposition and hence the process is efficient because of reduced time consumption. The vectors created by RI are of unit length in the k-tree and therefore the seeds or centroids need to be normalized to unit lengths at all times.

As explained in previous sections the WOM-Profiles are divided into structure model and content model. The content model is created by measuring the frequency of terms occurrences. The assumption is that terms in the profile indicate the type of thing in the real-world and hence things with the same content would be closer together in the Ambient Space representation. When comparing the structure model, it is proposed in [105] to use tag presence or tag frequencies to create structure model. In contrast we propose the use of path vectors (see Table 5.1), which is a more appropriate definition of profile structure. A comparison of the quality of our approach will be undertaken in our continued research.

5.5 Summary

In this chapter we discussed our approach to create communities of Web Smart things. To this end we proposed a WOM semantic structure to understand the social attributes of Web Smart things. Based on the similarities between the semantic attributes of things, clusters are created which represented communities. Two types of communities were created, the first was to subsume things into the SWoT and the second was to suggest connectivity to people on the Social Web. Thus, by exploiting the AS framework, we were able to create a platform where communities of things are modeled and their interactions can be studied. The SWoT communities provide a bridge for real-world things to connect to people on the Web and displays coherent communities of Web Smart things.

In the next chapter we discuss the benefits of WoT in various environments. We provide case-studies that exploit the use of AS framework to realize sustainable environments.

Chapter 6

Case Studies and Implementation

While the research focus and direction towards *Web of Things* (WoT) is maturing there is a dearth of case studies that identifies application areas and related benefits. In this chapter we present some case studies that we conducted to realize WoT applications. We also evaluate the benefits of WoT applications that adapt real-world things into the Web. In the first case study we describe the use of *Ambient Space* (AS) to create sustainable parking lots [20]. We analyze the benefits of using our parking application when compared to traditional solutions based only *Wireless Sensor Networks* (WSN). In the second case study, we describe in detail the use of AS to save energy in an educational campus environment, which was introduced earlier to discuss the AS framework [16]. Here again, we analyze the benefits of retrieving real-world information and creating WoT application to save energy.

6.1 Case Study 1: Building Sustainable Parking Lots

Sustainability measures are surging with the increased awareness of the benefits and long-term implications these measures have on our planet. Sustainability efforts drive economic

growth, greater prosperity, and new business opportunities [107]. Peak-time traffic witnesses environmental pollution and considerable amount of delays which results in an overall economic loss to society. Research innovations in the WoT can curtail some of these issues by creating scalable and sustainable environments like parking lots, which provide motorists with access to convenient parking spots. We present a scalable parking lot application that exposes parking services by creating a mashup of real-world things in a parking lot. Our application uses service oriented architecture, allowing motorists to reserve parking spots online. The application uses our proposed AS framework to interact with things in a parking lot. The application leverages the use of HTTP and Wi-Fi for the Web-enablement and interoperability of things within a parking spot, and elevates the parking spot to a *Smart Parking Spot* (SPS) on the Web. With the use of SPSs, our evaluation shows a maximum of 40% time saved to find parking spots and also considerable reduction in air pollution.

In this case study, we tackle two recurrent problems occurring in parking lots. Firstly, motorists spend considerable amount of time looking for free parking spots, and in effect contribute to increasing environmental pollution. The financial impact of reducing the time to search for a parking spot was accounted to more than a billion euros per year in a study done in France [108]. The report estimates about 70 million hours spent in a year by motorists in France looking for parking. Secondly, the numbers of reported auto thefts indicate alarming statistics, for example, in Canada 93,000 stolen cars were reported in 2010 [109]. With the increasing number of vehicles on our roads, these are issues that most modern cities grapple with. We approach these problems with the technology advances in the WoT research. Road rage, accidents, abandoned trips, and fuel wastage are some of the other issues induced by the unproductive task of finding a secure parking spot.

We have used our AS framework to build a context-aware parking lot (see Figure 6.1) that

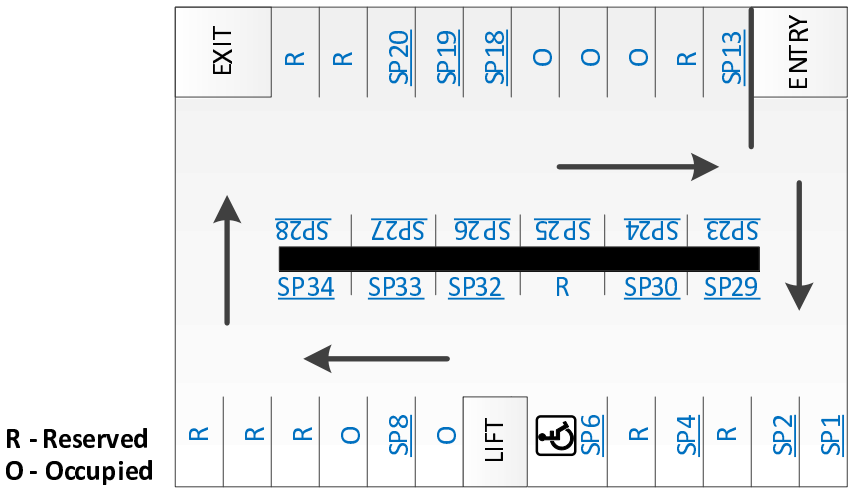


Figure 6.1: Smart Parking Spots in an Ambient Parking Lot

abstracts each parking spot as an AS. Within each parking spot is the same type of things, like lights, parking sensors, and status display units (LCD). When these things are made *Web Smart* a parking spot is transformed into a *Smart Parking Spot* (SPS).

Our application exposes the functions of things in a parking spot as Web services (RESTful services) to enable the interoperability of all SPSs and provide services to access and control SPS functions. RESTful services are employed as they are ideal for service-enabling resource-constrained things like sensors, displays, and actuators in a loosely coupled way. Instead of using relatively new protocols that are designed for resource-constrained device like 6LowPAN [6] or (Constrained Application Protocol (CoAP), our application leverages HTTP and Wi-Fi for the easy and universal Web-enablement as well as interoperability of things within a parking spot.

The use of small, low power Wi-Fi enabled Web server exposes a parking spot as an independent unit, and reduces the use of wiring and cabling for creating the parking lot's networking infrastructure. This allows our application to scale for any size of a parking lot, reducing the

cost and expanding the design possibilities of parking spaces. Each SPS is a Web resource and allows automatic allocation, remote reservation, and secure parking by binding a parking spot with a motorist's smart phone. The online reservation module of SPS reduces the time spent to locate and occupy a parking spot, and reduces the pollution within a parking lot. We have evaluated the use of SPS for reserving parking spots and measured the reduction in the time spent to find free parking space. We also measured the reduction in carbon emission from vehicles which was directly related to the amount of time saved to find a parking spot.

6.1.1 Motivational Scenario

We illustrate the user experience of our application through a motivational scenario that is an ideal condition where our application is implemented in a multistory paid parking lot.

Jon drives to his three-hour meeting on the twelfth floor of an office building. As he enters the parking lot of the building, his smart phone welcomes him to the parking lot and directs him to a pre-booked spot, which he reserved online before starting the trip. Jon quickly locates the spot and as he eases his car into it, his phone indicates the remaining parking time and announces the receipt of three messages. Jon reads the first one, which indicates that this is his fifth time at the spot and the next visit would be free (as a bonus). The next message indicates that his friend Ben, who is heading to the same meeting, is in the parking lot as well, and the third message indicates a book sale at his favorite bookshop on the fifth floor. As it has become a habit, Jon uses his smart phone to trigger a washing service in the parking to clean and polish his car while he is in the building. He notices Ben, cruising around for a free spot and waves out as he walks to the elevator.

This scenario illustrates the benefits of adopting parking spots as SPSs to plan daily tasks using WoT-based approaches. The seamless integration of physical parking spots into Web

applications and exposure of their services to mobile clients, reinforce the vision of AS, where people and physical things interact to deliver a new genre of services that encompass real-world entities.

6.1.2 Survey of Automated Parking Solutions

Here, we present the challenges for realizing our application through a review of related work driving this realization.

Existing guided parking frameworks [110] use Wireless Sensor Networks (WSN) where each parking spot has a lamp which indicates if it is occupied (red) or not (green). This is realized through a parking sensor that detects the presence of a vehicle in the parking spot. Digital displays in the lot indicate the number of available parking spots on each level of the building or for each bifurcated parking area.

However, during peak-time traffic situations, this does not guarantee the spot when a vehicle reaches it, because there are many takers for a spot. Vehicles continue to move around in the parking lot for various lengths of time and in the process pollute the air with vehicular emissions. Today, many parking lots employ an infrastructure of sensor networks to manage the available parking spots and provide safer parking. These networks are restricted by the lack of resources on the sensors, the heterogeneity of sensor nodes and the difficulty to redesign or extend the network [52, 111]. Parking solutions based on WSN relies on a centralized storage of information because of resource restrictions on sensors [111]. Therefore, these solutions have restrictions on scalability and are not flexible. They require cabling and related networking infrastructure, which makes it difficult for existing parking lots to adopt these solutions or extend the existing ones. Moreover, these solutions require various types of nodes for sensing, routing, and management.

In retrospect, our application is easy to deploy and scales well, as each node (parking spot) is equipped with a Wi-Fi module and a Web server connected to things like parking sensors, displays, and lights. This means encompassing each parking spot to be an independent hardware component and is also virtually represented on the Web. Each parking spot is an independent Web resource having a state and services displayed online (Figure 6.1). For example, SP33 in Figure 6.1, is a hyperlink which when clicked reveals the options for reserving the spot.

There are also many online applications that interface with parking lots like SFpark (sfpark.org) in San Francisco and Parkopedia (parkopedia.com). SFpark tracks the availability of parking spaces and garages in some areas of San Francisco using sensors that detect free spaces. Though it is for a small area and expects to manage the traffic by varying the cost of parking, the solution provides accurate data about free parking spots. In contrast, our approach provides two other benefits. Firstly, our system couples a parking spot and a user's smart phone ensuring that the vehicle is retrieved only by the owner of the vehicle (i.e. the person who possess the smart phone used for registering the spot) and secondly our system provides rich semantic data which is able to capture user centric information so as to provide user incentives as mentioned in our scenario. Another popular solution in Europe is Parkopedia that provides parking solutions and also an online interface for users to submit reviews which are manually monitored for validity and acceptance. The system provides limited services and the data is not openly available to people who would use the parking spots. In contrast, our system provides an open architecture where people access the parking spot directly through Web based URLs. The semantic structure provides easy access to third party service providers to directly plug in their services, like a washing service as indicated in our scenario.

The resources and respective representations are distributed across every parking spot instead of a centralized location and are accessed as Web services. The use of Web service for

enhancing the scalability of WSN has been successfully tested [52]. In comparison to this approach, which uses SOAP, the use of REST architecture for Web services has introduced a new paradigm for resource-constrained devices [32]. This new approach enables the realization of the WoT as an interoperable and open technology. Yet, the heterogeneous nature of things within a given space requires the use of gateways to bridge RESTful operations of things with the vendor-specific protocols [11, 66] of real-world things. Our application augments candidate things like sensors in a parking lot with Web capabilities, allowing parking services to be exposed as RESTful services over HTTP without the use of gateways.

6.1.3 Ambient Parking Lot Application

As described earlier, an AS is a virtual space represented as a mashup of one or more *Web Smart* things providing services that access or alters the state of the physical space. We use our proposed AS framework to design the *Ambient Parking Lot* application. Here we present the necessary representations for our application created by the mashup of many SPSs.

Application Architecture: Parking lots are organized spaces for retaining vehicles that are not moving for a period of time. An ambient parking lot application creates a mashup of SPS operations to create a virtual space on the Web where motorists and things like smart phones, parking sensors, and vehicles interoperate to provide secure and timely parking services in the real-world. The multiple layers of the ambient parking lot application is illustrated in Figure 6.2.

The *Smart Parking Client* (SPC) layer enables user interactions and feedback from the ambient parking lot applications. The SPC is essentially a mobile application that resides on any Web-enabled device like a smart phone that is used for reserving a parking spot. The SPC

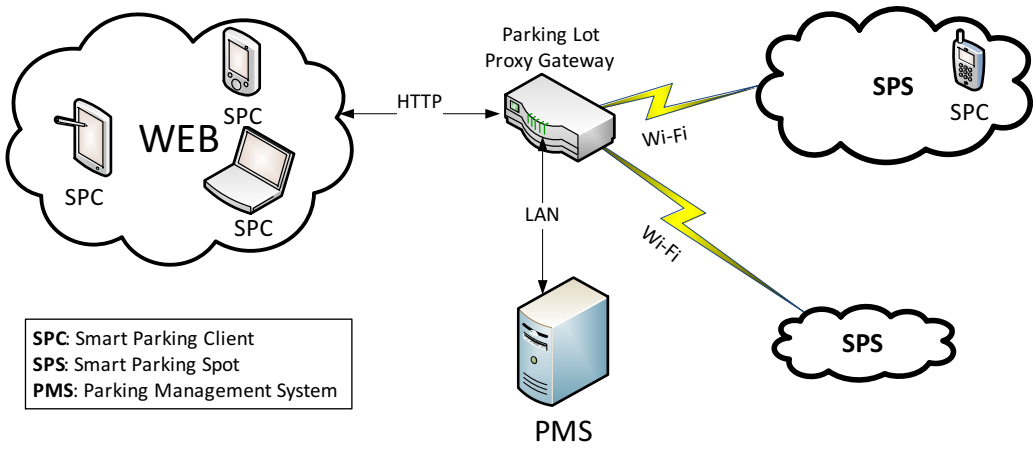


Figure 6.2: Layers of an Ambient Parking Lot Application

also verifies the user within the parking lot when it is coupled with a reserved spot. Once inside the parking lot, the SPC becomes part of the parking lot network infrastructure and seamlessly communicates with other entities in the parking lot.

The *Smart Parking Spot* (SPS) layer represents an Ambient Space (AS) where things like sensors, lights and LCD displays are things within a parking spot. Each SPS is wirelessly integrated to the network within the parking lot using Wi-Fi. This has two benefits, the amount of wiring and cabling is reduced and it also extends the parking to multiple floors or to a larger area easily, by scaling the number of access points. When an SPS is coupled with an SPC (e.g., smart phone of a motorist) it is in an *Occupied* state, otherwise it is either *Free* or *Reserved*. The state is reflected online and also displayed onsite. The coupling and decoupling of SPS and SPC go through a verification process which ensures the security of the vehicle in the parking lot.

The *Parking Management System* (PMS) hosts a Web based application that allows motorists to reserve a parking spot over the Web. The reservation facility is purchased through

online payment for a period of time selected by the motorist. A database connected to the PMS indexes the URLs of the RESTful services of all SPS and corresponding things within them enabling the easy access and search for SPSs.

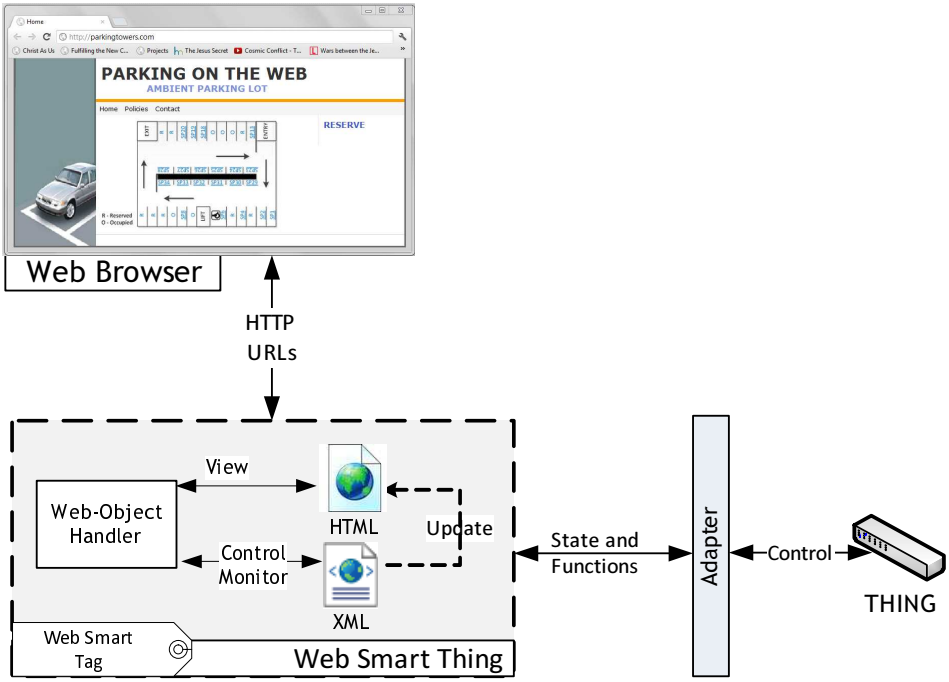


Figure 6.3: Exposing information on a thing as RESTful Web services

Things in an Ambient Parking Spot: To represent a thing like a parking sensor or a digital display on the Web, it must be made available as a Web Smart thing. The minimal requirement for representing a thing on the Web is shown in Figure 6.3 where a Web Smart thing exposes the information and control of a thing as RESTful services accessible over HTTP. The Web-Object Handler receives and responds to the requests for services. The representation of Web Smart things must reflect real-time scenarios and also be retrievable and updatable. The

representation of states and functions of real-world objects in XML ensures easy interoperability between Web Smart things and their representation in HTML enhances human perception of real-world objects. The XML contains the Web Object Metadata (WOM) of a thing as described earlier. The integration of many SPSs creates an ambient parking lot. We discuss in the next section the design and implementation of the various components to realize an ambient parking lot.

6.1.4 Design and Prototype Implementation

An ambient parking lot with the PMS, three SPS in different states and an SPC is illustrated in Figure 6.4. The Data Center hosts information on the parking reservations, vehicular services and the various points of interest within the parking lot.

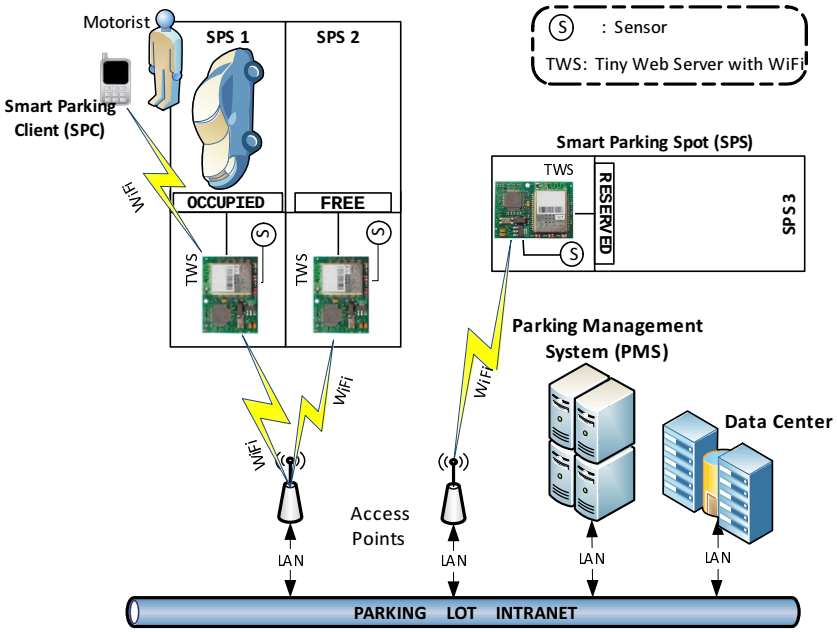


Figure 6.4: Design of Ambient Parking lot with three Smart Parking Spots

Parking Management System (PMS): The process of parking is initiated by a motorist who requires a parking spot for a period of time. The motorist invokes the parking lot URL using a Web-enabled smart phone (SPC). The Web page hosted by PMS, provides the options for *automatic* or *manual* selection of a parking spot. The automatic option selects the first available spot (e.g., SPS1) for the motorist as shown in Figure 6.5. The manual option displays the parking lot which allows the motorist to select a spot or click on a point of interest like the *Lift* or the *Exit*. On selecting a particular point of interest a list of available parking spots closest to the selected one are displayed to the motorist.

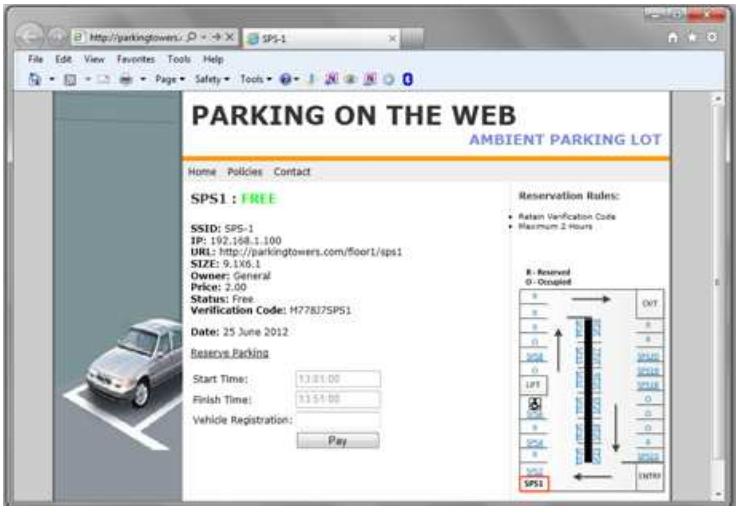


Figure 6.5: Reservation page and online display of the parking spots

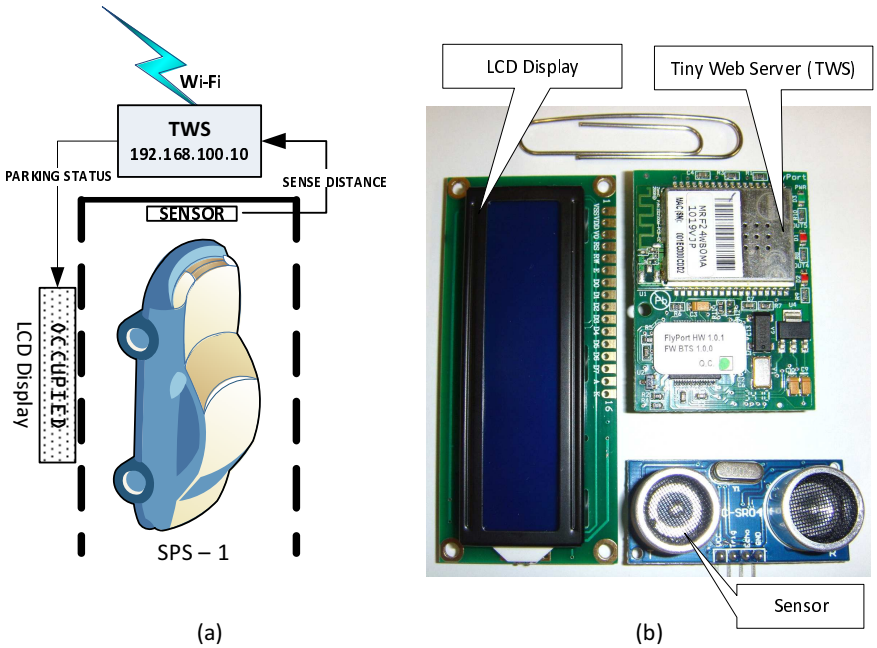
The ambient parking lot is represented by a URL (e.g., `http://parkingtowers.com`), which represents the general namespace for accessing any SPS within the lot. For example, the URL `http://parkingtowers.com/floor1/SPS1?start=1301&stop=1351`, reserves SPS1 for 50 min from 13:01. HTTP responses like '*200 OK*' or '*303 See Other*' indicate the results of accessing the URLs. Once the online payment is completed the state of the selected SPS is set to

Reserved (R). The spots that are *Occupied* (O) cannot be selected at the given point in time since there is a possibility that the time may be extended by the motorist that has occupied the spot. Reserving an SPS in advance reduces the time to find a parking spot like Jon did in our scenario.

A database stores the URLs of RESTful services and their relations to each SPS. An SPS hosts its WOM-Profile with representation of the various parameters like location, dimension, ownership, cost, state, driving direction of the aisle, or if it is a disabled spot. Hence each SPS is a Web resource and also a data store. This creates an efficient and scalable model for distributed storage of information accessed using RESTful services.

Smart Parking Spot (SPS): It is an AS that integrates things like lights, sensors, displays, and a motorist's phone into the ambient parking lot [21]. An SPS is realized by augmenting it with a Wi-Fi enabled Tiny Web Server (TWS) which communicates over TCP/IP, and has storage space. Figure 6.6 illustrates the SPS design and the essential hardware components. The TWS is connected to an ultrasound sensor and a digital display, to sense the presence of a vehicle in the spot and display the state information respectively. The SPS is Wi-Fi enabled and functions as an access point providing wireless access to the parking lot Intranet. Each SPS has a unique Service Set Identification (SSID), which enables a motorist's smart phone (SPC) to uniquely identify a spot. This encapsulates each SPS into an independent hardware unit, which enhances the scalability of our application and enables flexible design of parking lots. This also makes it possible for existing parking lots to easily adapt our application.

Identifying each resource uniquely within a given namespace is necessary for building RESTful services and enables the direct access to an SPS. An SPS is uniquely identified by appending



(a) Information flow in a Smart Parking Spot (SPS)

(b) Hardware components used for SPS prototype

Figure 6.6: Things in a Smart Parking Spot (SPS)

the parking lot's URL with the SSID[1] of the Wi-Fi module which makes it a unique resource within the namespace and the Web. The parameters of URLs are parsed to decide on corresponding operations like, reserving a spot, verifying motorists, or extending time. The TWS provides services to query and update the WOM parameters.

Once a user has reserved an SPS, the state parameter of the SPS is updated to *Reserved*. When a vehicle occupies an SPS which is in a *Free* or *Reserved* state the sensor senses the presence of the vehicle and a timer (e.g., 60 sec) is started. The SPC (e.g., a smart phone) verifies itself and is paired with the SPS before the timer expires, failing which an alarm is raised. Once an SPC and an SPS are paired, the state of the SPS is updated to *Occupied*, as

[1]Service Set Identifier for Wireless networks

shown in Figure 6.6 (a). On departure from the SPS, the sensor again triggers a timer (e.g., 60 sec), before which the SPC must identify itself. If the SPC identification fails an alarm is raised as it could indicate that the vehicle is being retrieved by someone who did not reserve the spot. If the SPC successfully identifies itself, once the vehicle leaves the state of the SPS is updated to *Free* and it is available for reservation.

HTTP is generally used with pull technology for accessing information, which in this case would require the PMS to continuously poll all SPS to check their state to see if vehicles are occupying or leaving them. This is not efficient and hence we use HTTP callback to indicate SPS state when an event occurs. For example, when a vehicle occupies an SPS, an HTTP POST message is sent to the PMS using a predefined URL and with relevant information in the message body. This improves the efficiency of the PMS to handle requests only when specific event occur, like when a car occupies or leaves an SPS.

An SPS provides services, which are composed together to enable efficient parking management. An instance of service composition is illustrated in Figure 6.7, where events like *Booking* a parking spot triggers several services, including (i) the *SpotBookingWS* where the reservation parameters are verified, (ii) the *SpotSensorWS* that triggers the sensor to verify the availability of the spot, and (iii) if available, the *SpotDisplayWS* indicates on the digital display that the spot is '*Reserved*'. When a car parks at the spot, the sensor triggers a Parking event that invokes the *SpotVerifyWS* with a verification code from the occupant. On successful verification, the *SpotDisplayWS* is invoked to change spot state to *Free* on the digital display or if the verification fails the *SpotAlarmWS* is invoked.

Smart Parking Client (SPC): The SPC is a mobile application that retains the parameters of the parking reservation and also communicates with SPS. Once the reservation is complete,

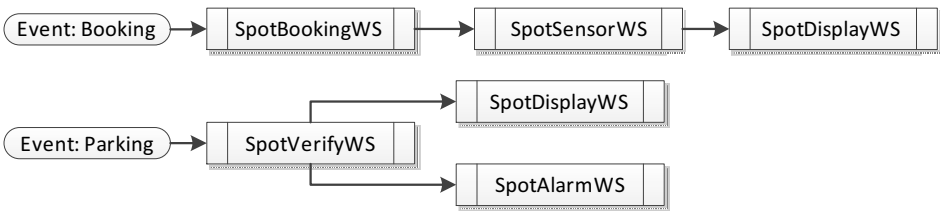


Figure 6.7: Composition of parking services

the reservation parameters like the SSID of the reserved spot, the relative location inside the parking lot and a verification code are stored by the SPC. The application also allows the user to manually enter the reservation parameters, in case the reservation was done from another device like a PC. On arrival at the SPS, the SPC identifies and connects to the SPS based on the reservation parameters that are stored, and then the verification code is exchanged. This identifies the user as the one who made the reservation. On departure, the SPC connects again to the SPS and exchanges the verification code. This ensures that only a person with the smart phone (SPC) that made the reservation can retrieve the vehicle.

Prototype Implementation: The hardware components used to design SPS is shown in Figure 6.6 (b). We used a TWS module, FlyPort [88], which is 35X48 mm in dimension with an integrated 802.11 Wi-Fi interface module and a 16-bit processor. The internal flash of 256KB is sufficient for the intended Web application. Dynamic Web pages access inputs and output ports which enable the manipulation of a thing from the Web. We configured the TWS with a unique SSID to serve as a Wi-Fi access point and as a part of the parking lot IP network infrastructure. The Web application parses the incoming RESTful URLs to perform corresponding operations. An LCD display and an ultrasound sensor are connected to the ports of the TWS to display the state and determine the presence or absence of a vehicle respectively. Ultrasonic sensors like the HC-SR04 are cheaper, simpler and stably detect echoes from barriers from 2cm to 500cm

as compared to vision sensors, laser sensors and image sensors [112]. The TWS hosts an HTML page and an XML page to represent the WOM of things in the parking spot.

The PMS implementation involves a standard Web application that is built using JSP, Servlets and MySQL database. The Web application allows users to reserve a parking spot for a selected time period on a day. All reservations expire by the end of the day. The SPC is developed on the Android 2.3.1 platform using `android.net.wifi.WifiManager` methods to identify an SPS with a particular SSID. The SPC searches for a particular SSID and connects to the SPS. The verification code is exchanged and the state of the SPS is updated using the reservation parameters. This couples the SPC and the SPS and also identifies the SPC as an integral part of the ambient parking lot.

6.1.5 Analysis and Evaluation

Here, we determine the extent to which the use of SPS in the proposed ambient parking lot application helps to minimize the time to find a free spot and preserve a sustainable environment.

For evaluation purposes, we consider *Ben* and *Jon*, going to attend a meeting during peak-time traffic conditions. While Ben chooses to drive directly to the meeting, Jon decides to use his smart phone to reserve an SPS from the Web using the *automatic* (i.e., the system allocates a free SPS) option and then drive to the venue. We assume that both, use the same route, and reach the parking lot at the same time.

This scenario assumes that Jon will find a free parking spot and that Ben may not find one. Also, the time spent for Jon to reserve a parking spot is typically less that the time spent by Ben to find a free parking spot. Inspite of the advantage that Jon has in this scenario, we wanted to measure if there was a significant amount of time saved when SPS is used. Moreover, we also wanted to use our findings to measure the amount of reduction in vehicular pollution,

to see if there was any major impact.

The performance metrics of our evaluation are cruising time, total time and emission rates. We define cruising time as the time spent in the parking lot to search for a free spot and occupy it. The total time includes the time to reserve an SPS, time to drive to the parking lot and cruising time. Within parking lots, vehicles assume various rates of acceleration to reach a desired spot i.e. they slow down or accelerate while looking out for a parking spot. Considering the various rates of acceleration for different vehicles the average emission rates for NO (nitrogen oxides), HC (hydrocarbons), CO (carbon monoxides), and CO_2 (carbon dioxide) were determined to be 1.44 mg/sec, 0.76 mg/sec, 10.51 mg/sec, and 3.22 g/sec respectively [113].

#	**Ben's Travel**	**Jon's Travel**
1		Online reservation of SPS
2	Drive to the parking lot for 30 min	Drive to the parking lot for 30 min
3	Cruise for a free parking spot	Cruise to the pre-booked SPS
4	Occupy the parking spot	Occupy the SPS

Table 6.1: Traveling events to the parking lot for Ben and Jon

Based on the traveling events of Ben and Jon as shown in Table 6.1, we are interested in the following questions for the duration of the peak-time: (1) How much cruising time is spent to reach a parking spot? (2) What is the total time taken for reaching the parking spot? And, (3) What is the amount of carbon emission?

The possibility of finding a free parking spot at a given point of time is dependent on two factors the number of cars arriving and occupying the free spots, and the time of arrival at the parking lot, i.e., the later they arrive into the peak-time lesser would be the chance of finding a free spot. Hence, the chance to get a free spot reduces as time advances into the peak-time. We also consider maximum cruising time, beyond which the motorist leaves the parking lot without

finding a free spot.

To complete an exhaustive evaluation we would require Ben and Jon to arrive at the parking lot at various time slots incrementally for the duration of peak-time traffic (120 minutes). For example, what would be the results if they arrive 35 min into the peak-time or 42 min or 56 min into the peak-time and so on? This is expensive and cumbersome to evaluate in real-life and hence we simulate the scenario, where Ben and Jon arrive at the parking lot at time t which is incremented for every minute of peak-time traffic duration.

For our simulation, we consider the total number of parking spots to be N (800 spots) and the simulation is run for the duration of peak time traffic (120 minutes). The time to reach the parking lot for both Ben and Jon is T minutes, a constant value of (30 minutes). A random number of cars C_t arrive at the parking spot at time t following a Poisson distribution, based on which the number of available parking spots reduce. Cruising time depends on the available spots at a given time and also the maximum cruising time X. The maximum cruising time is a constant value (30 minutes), beyond which the motorist leaves without finding a free spot.

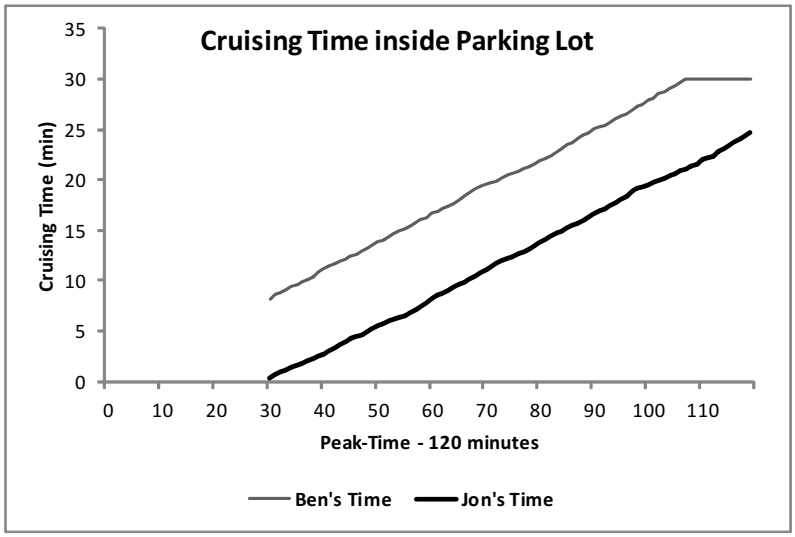
Let $A_t = A_{t-1} - C_t$, be the available parking spots at time t, then Ben's cruising time UC_t to an unreserved spot and Jon's cruising time RC_t to an SPS, are formulated as follows:

- UC_t at time t is $((N - A_t)/N) * X$
- RC_t at time t is $((N - A_{t-T})/N) * X$, where A_{t-T} is available spots at time $t - T$, because the spot was reserved T minutes earlier.

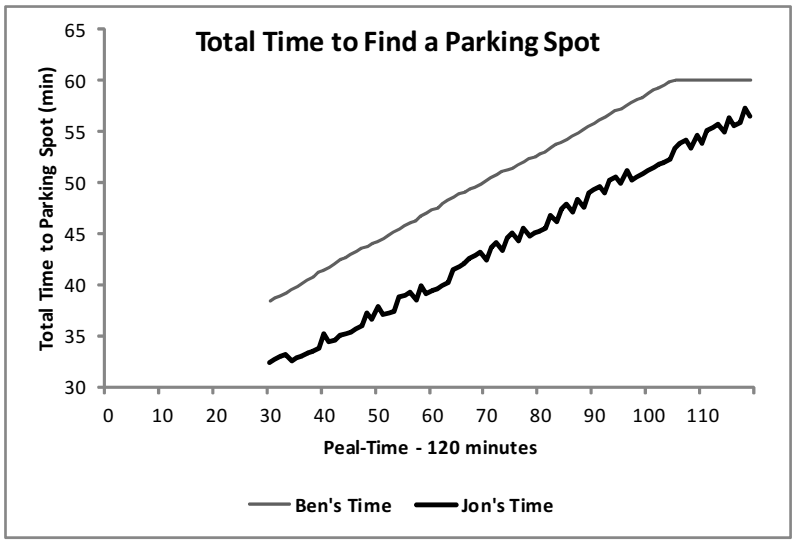
The simulation compares Ben's and Jon's time to find a parking considering they would reach the parking lot at varying degrees of peak-time traffic. Graphs of the simulation results are shown in Figure 6.8 [20], which are based on average results collected after 200 iterations. Figure 6.8 (a) and (b) for both Ben and Jon, the x-axis coordinates of the graphs indicate

their arrival time at the parking lot, i.e., the number of minutes into peak time after a 30 min drive. The y-axis coordinates indicate cruising time and total time in Figure 6.8 (a) and (b), respectively.

With peak-time increasing we observe that there is an increasing gradient in the time taken to find a free spot with Ben taking longer time to find a spot. As the graph illustrates, towards the end of the peak-time Ben leaves without finding a parking spot while Jon continues to occupy SPS. The results in Figure 6.8 (a) show an approximate average of 40% reduction in cruising time when using an SPS, compared to an unreserved parking spot. Similarly, the results in Figure 6.8 (b) show, the total time to reach an SPS inclusive of travel time and the online reservation time is better than using an unreserved spot. After 200 iterations of the simulation it was noticed that there is an average of 32% reduction in total time to reach an SPS when compared to unreserved parking spot. The maximum recorded was 40% reduction which is shown in the figure.



(a)



(b)

Figure 6.8: (a) Cruising time of Ben and Jon, and (b) Total time of Ben and Jon

The contrasts between the rates of emission for NO, HC, CO, and CO_2 is shown in Figure 6.9 [20], for Ben and Jon. The graphs indicate considerable reduction in emission components when the time to find a parking is reduced. Collectively with the use of SPS there is an average of about 38% reduction in emission.

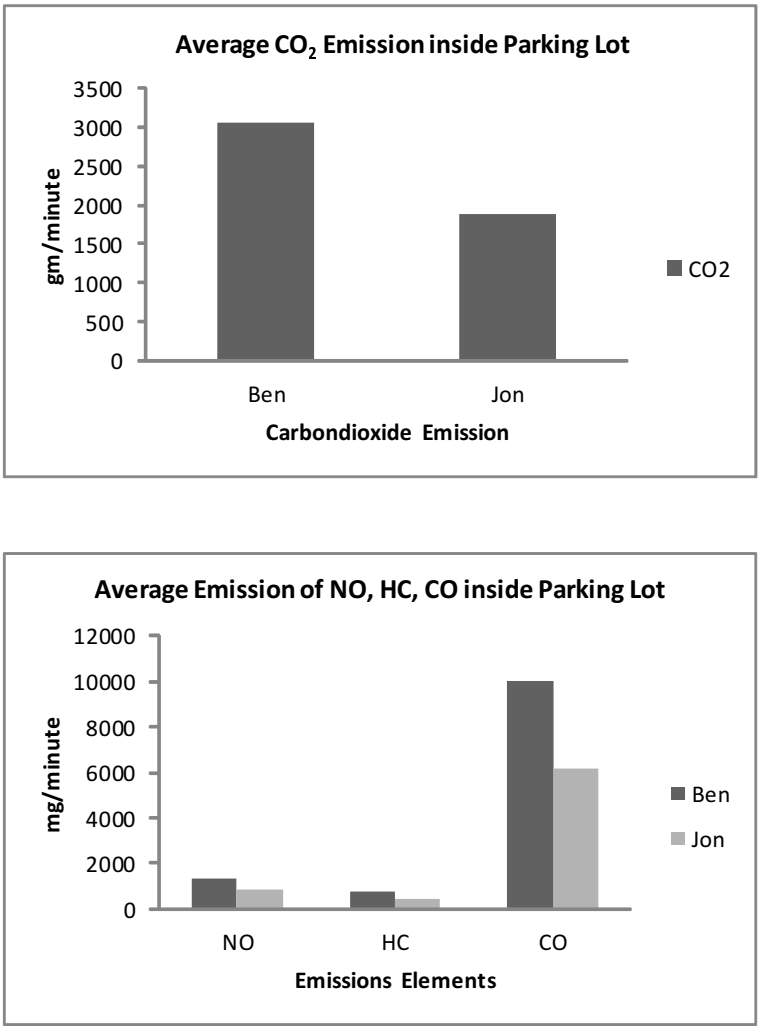


Figure 6.9: The emission rates for SPS and unreserved parking spots

6.1.6 Discussion and Conclusion

Abstracting parking spots into the WoT as ambient spaces creates a flexible model for parking lots. With the WoT technology which encompasses the use of RESTful services, distributed

data, Web enabled things, and Wi-Fi access our application enhances the scalability and flexibility of a parking lot network infrastructure and related information. This in effect reduces the total travel time, and cruising time for motorists and also reduces the rate of pollution, creating more sustainable parking environments. With the use of SPS, our evaluation shows a maximum of 40% time saved to find parking spots and also considerable reduction in air pollution. Using our proposed AS framework, we plan to design Web applications for searching, locating and reserving choice parking spots, to reduce overall traffic congestion and provide safe parking. We plan to study the impact when *two or more* parking lots within a city adopt our application. We also continue to explore the options of optimizing the application by working towards measuring the performance when one TWS is used for multiple parking spots.

6.2 Case Study 2: A Sustainable Campus with the WoT

Within the scope of enhancing social existence, sustainability is the responsible consumption of valuable resources to increase the longevity of their availability. Sustainability measures are surging with the increased awareness of the long term implications that these measures have on our planet and as a result many countries are contemplating the introduction of energy taxes. Electric power is an important energy resource that has paramount effect on the existence of the civilized world as we know it today. Studies done worldwide indicate that increased power consumption and its ecological consequence like light pollution is partially attributed to the fact that people spend most of their time in building structures, which are a major source of artificial light consumption, even when they are not occupied [114, 115].

A gas leak or an engine left running would create a sense of urgency for many people because of wasted resources or impending danger. Similarly, empty rooms with the lights left on indicate

the depletion of valuable electrical energy, but most people would care less. We utilize research innovations in WoT to curtail some of these issues by classifying and abstracting real-world objects as things on the Web. We present an application that composes *Web Smart* things to create *Ambient Spaces* (AS) where people and electrical supplies seamlessly communicate. A case-study within the scope of a university campus is presented which describes and evaluates the realization of AS to sustain electrical energy.

We exploit the advances in WoT research to curtail the problem of excessive power consumption in classrooms of a university campus. Two factors that contribute in solving the problem are the real-time indication of an individual's contribution to power consumption and accountability metric to tax or reward usability patterns. We envision sustainable environments where intelligence is built into electrical supplies and equipment to bring context awareness to their users who are accounted for the use. Towards the initial phase of realizing this vision within the scope of a university campus we identified classrooms and common things within them as real-world objects to be abstracted on the Web. We used our AS framework and recently introduced the modules for realizing the Ambient Classroom application [16] using service oriented architecture to efficiently manage the energy consumption in classrooms.

6.2.1 Survey of Ambient Infrastructure

Earlier approaches towards realizing ambient environments using Web technologies have been suggested [116, 117]. Our application factors these developments and proposes a design that exploits the simplicity of using Web technologies, sensors, and Wi-Fi for large scale deployment.

Many solutions to manage room ambiance use sensors to detect daylight or human presence to control artificial lights. These approaches result in disturbances caused by lights going on or off randomly because of room design, weather condition, or noise levels [118]. One of the

reasons for adopting sensors is because there is no clarity on how long the lights are required. Within the scope of a university campus we perceived the duration of classroom usage by using a room reservation service. This ensures enough light during a session and also saves energy as rules are set to automatically switch off lights after a session.

The use of Web service for realizing scalable sensor networks has been successfully tested using SOAP based Web service [52]. The use of Representational State Transfer (REST) architecture for Web services has introduced a new and better paradigm compared to SOAP for resource-constrained devices, enabling the realization of the WoT as an interoperable and an open technology. We used a REST based approach since it is lightweight and simple to service enable resource constrained things like 'lights' so as to realize an Ambient Classroom. We conduct a case study to evaluate the feasibility of our application to sustain energy by ensuring the proper use of lights in a campus environment.

6.2.2 Ambient Classroom Framework

Our framework illustrated in Figure 6.10 was introduced earlier in Chapter 4. Here we expound on the Ambient Space Manager (ASM) module and its components. Each classroom is a homogeneous physical space because they have the same type of things and is abstracted as an AS. Our framework leverages the use of HTTP and Wi-Fi for the Web-enablement and interoperability of things within a classroom. We employ REST based services to enable resource constrained things like sensors and actuators in a loosely coupled way. Moreover, a thing augmented with Wi-Fi capabilities reduces the need of wiring and cabling and therefore provides the flexibility to install it anywhere in the classroom. This reduces implementation cost, and expands design possibilities of AS. A Web based interface enhances the accessibility of electrical supplies in classrooms. The ASM hosts five modules, *Things Control and Sensing* (TCS),

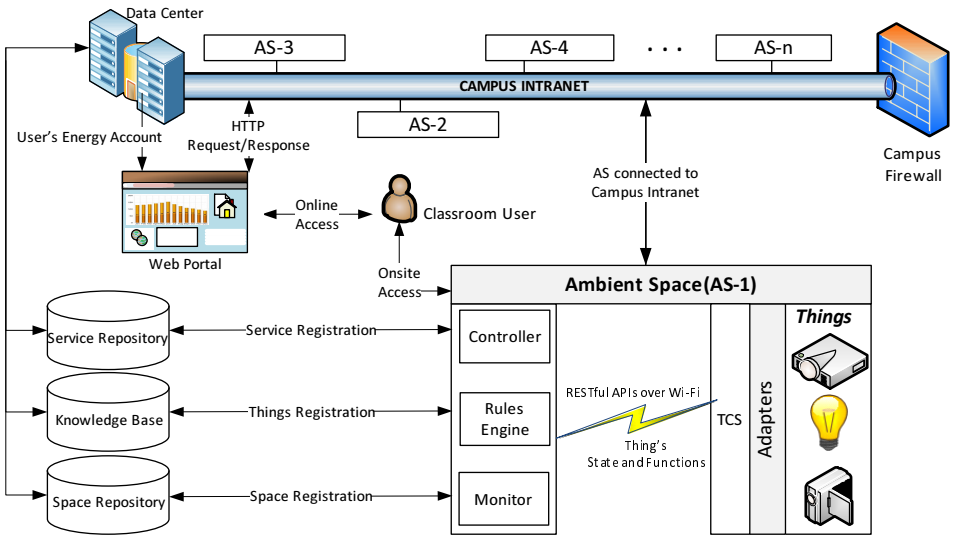


Figure 6.10: Classrooms abstracted using Ambient Space framework

Controller, *Rules Engine*, and *Monitor*.

The TCS module translates the states and functions of real-world objects into a format that is accessible over HTTP using RESTful services. TCS incorporates Wi-Fi enabled access to actuators and sensors to control and sense the state of things in a classroom. It is the last unit of information flow as it may have the least computing resources. The *Adapters* provide the necessary capabilities for things to participate in an AS as Web Smart things. The necessary capabilities like a unique identification, the ability to handle HTTP requests and responses, and necessary storage space are augmented to things in a classroom.

The other three modules provide an integrated interface to classroom services. These modules expose classroom services to users and Web applications that query the state and function of various things in the room. Moreover, these modules commission and manage the life-cycle of things within a classroom. The three modules have specific tasks. The *Controller* ensures secure access to things within the AS based on predefined access protocols. The *Rules Engine*

constrains a thing's usage based on the capabilities and operational intelligence which are specific to the physical things within them. The *Monitor* interfaces with sensors to capture the states of AS for various measurements.

Each AS has a designated set of repositories which is part of the campus *Data Center*. The *Space Repository* holds details like, location, purpose, and seating capacity, of classrooms. The *Knowledge Base* is a repository of things classified according to their capabilities and associated with each classroom and the *Service Repository* provides a directory of services exposed by each classroom. Moreover, the *Data Center* also functions as a private cloud for storing information of things that do not have sufficient storage capability.

Classroom Users are made aware of their energy usage based on the duration of time they use a classroom. Each user has a virtual account called the *Energy Account* which indicates a user's contribution to the energy consumption. The *Energy Account* of each user is made available through a Web portal, and incentives for optimal usage are planned.

6.2.3 Challenges of Building an Ambient Classroom

To realize the application a case study was conducted to understand the challenges and provide awareness and accountability of the utilization of lights in classroom. As an initial step we measured the duration that lights within classrooms of a university campus were left on. The classrooms are venues for scheduled courses and the lights are needed only for the scheduled duration. However, the perceived comparison illustrated in Figure 6.11, shows that lights in the classrooms are left on for a much longer duration than required.

Based on feedback from faculty that use these rooms, the wastage was attributed to forgetfulness, lack of awareness, and some users indicated that within a large campus it was time consuming and tedious to return back to switch the lights off at a later point of time. This

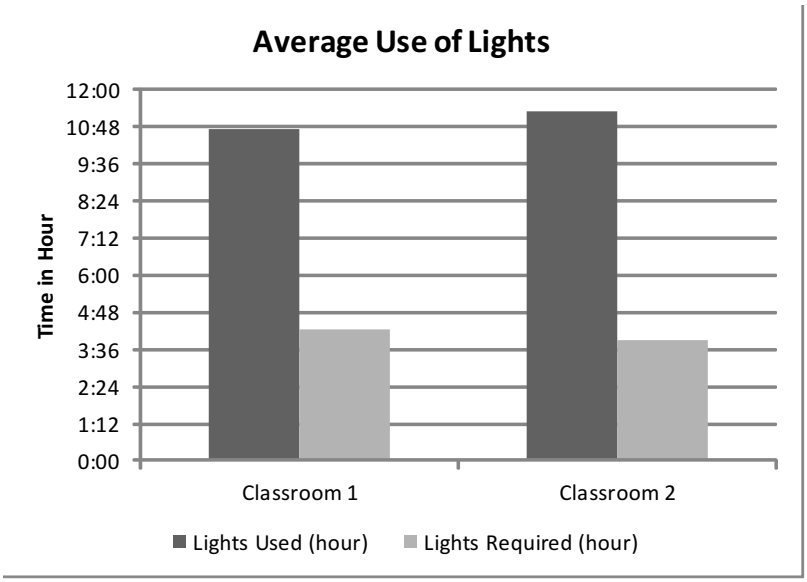


Figure 6.11: Comparison of the actual usage of lights *versus* the required usage in two classrooms

study was done for seven days and the average use of lights was compared with average requirement. The community within the campus is generally unaware of the impact of their possible negligence. They are not informed to adjust their attitude towards an environment-friendly ecosystem and help sustain energy. We expect that once energy loss is accounted to individuals, it would increase their awareness and allow them to be careful in reducing their energy footprint. Towards this vision we realized an Ambient Classroom. We were faced with a number of design challenges in the process of prototyping the main components of the framework to realize a classroom as an AS which we discuss in the following sections.

Encapsulating things as Web Smart things: We had to decide on the software components that would essentially represent Web Smart capabilities of real-world things. As illustrated in Figure 6.12, the access to these things is through unique RESTful URLs. The Web-object

Handler handles the requests for services. The AS Manager modules exposes classroom services created as a mashup of Web-object Handlers of various things in the classroom. Abstracting different things in the classroom as Web Smart things is the function of the TCS module (Figure 6.10 and Figure 6.12). The Web Smart tag provides the address (URL) of the XML file that hosts the WOM-Profile of a thing.

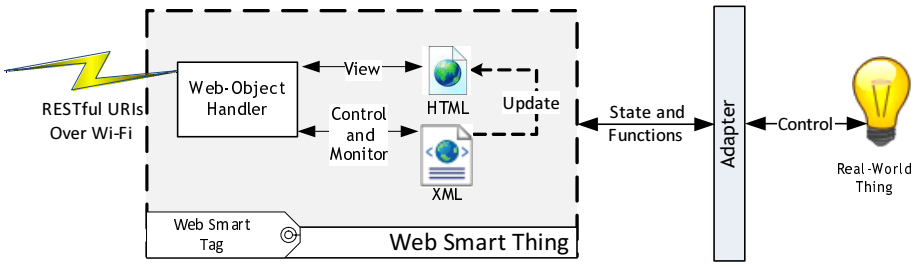


Figure 6.12: A Web Smart thing representing a real-world thing

Adapting things into the Web: Extending embedded Web servers and service computing to real-world objects makes it possible for their states and functions to be abstracted as embedded Web services [63]. In doing so, real-world objects could offer their functionality over HTTP using the semantic value of RESTful URLs. These services directly expose operations of real-world things to enterprise applications or even to other things and possibly involving a dynamic composition process at runtime.

Within our context of a classroom, lights have two basic states, on and off. The challenge is to Web-enable inanimate things like light bulbs in order to push and pull information. Their capabilities are to be enhanced so that they can be controlled, sensed and their operations are accessible using HTTP. Rules are to be set to turn lights on or off based on room usage and also to monitor the light so as to determine if any of the lights need to be replaced. Also, the state of the lights had to be stored and retrieved. Hence we decided to augment the lights with

a tiny Web server that has storage capability and also communicates wirelessly to enable the seamless access of things' functionalities and resources.

Monitoring Thing's State: Detection of light malfunctions in a remote location is impossible without physically observing them. As a norm, maintenance staff is informed of such issues either by performing periodic checks or reports from room occupants. It is time consuming and costly to physically locate such issues in a large campus with hundreds of venues. Hence it is necessary to detect real-time state of lights in classrooms and also to report issues ensuring minimum communication overhead to reduce administrative costs.

One possibility is to use light sensors which would enable the detection of the presence or absence of light. With the state of sensors stored on the Web Smart thing, it would enable the detection of light status online. However, HTTP is generally used with pull technology for accessing information, which in this case would require a remote client to continuously poll all sensors to see if the lights are on or off. This is not an efficient use of network bandwidth. We use HTTP callback to indicate light status when an event occurs, where using a predefined URL an HTTP POST message is sent to the *Monitor* module (see Figure 6.10) with relevant information in the message body. This improves the efficiency as the communication occurs only when specific events occur and not continuously. For example, an event is generated when a light is switched on and that triggers a POST message to be sent containing the expected state i.e., *On,* and the sensor reading which must be same. If it is not the same then the *Monitor* determines that there is a malfunction with the lights. Another possibility is to allow classroom occupants to navigate to the Web Smart thing from any Web-enabled device like a smart phone and tag the Web Smart thing (lights), with a predefined tag options like, *Fix*, which would generate an event to trigger the invocation of the HTTP callback URL.

Onsite Interactions with Things: Interactions with classroom services had to be provided both onsite i.e., within classroom environment and also online i.e., from the Web. Creating Web Smart lights enables their online access but the onsite access of a thing's state and functions is also necessary to enrich the experience of people interacting with Web-enabled things in the classroom.

Existing building management solutions look at managing energy at a macro level, i.e., electricity consumption is measured by taking meter readings for a building [119]. Using our application, a holistic view of power consumption is presented. Micromanaging usage of lights at room level instead of a whole building allows the usage to be attributed to individuals that occupy the room. This creates a requirement for classroom reservation services to be available online and onsite for individuals or systems with proper authorization. This would also allow authorized entities to have their requests routed to a proper light startup and shutdown service.

We have designed the interactions to classroom services to be provided through a digital signage with touch screen interface since touch screens have enriched the experience of interacting with devices and appliances. The signage displays the details of ongoing sessions and classroom numbers. A *Calendar Service* and a *Reservation Service* allow the scheduling of classrooms for courses. A genuine user with a valid username and password or a Web application like a time tabling application reserves classrooms based on available time slots.

Energy Accounts on the Web Portal: Richly formatted feedback system that reflects the use of energy helps to have a clear understanding of energy usage [11]. An increased level of awareness through feedback has been shown to motivate household occupants to reduce energy use significantly and in effect reduce energy bills [120]. However, in our case study the intention is to increase awareness and accountability of classroom occupants using a format that would

directly relate to their real-time usage of things in a classroom and not to bill them. Hence, instead of using standard units of power we decided to use time as the unit to indicate usage. Faculty on campus has their *Energy Account* updated based on the duration of respective classroom occupancy. In comparison to the average usage on campus each *Energy Account* keeps its owners aware of their actions, when using energy supplies. The Energy Accounts of each user is made available on a Web portal which is accessed using login credentials.

6.2.4 Design and Prototype Implementation

Here we describe the realization of a classroom as an AS. The proposed application is realized as an integration of different technologies and services. The planned tasks were, 1) to Web enable lights with a Tiny Web Server and to use sensors to detect their status, and 2) to use a booking system for allotting and reserving classrooms with online and onsite interface.

A classroom signage is used to provide room reservation and calendar services that was realized with a prepackaged product with the required features and integrated it into our application as illustrated in Figure 6.13. The signage is part of the campus Intranet and is a programmable unit which enables onsite and online interaction with calendar services as shown in Figure 6.13 (a). With this interface, a user can log into the portal and navigate to the particular classroom and then use the calendar service to reserve a room or cancel it. This unit hosts the *Controller*, the *Rules Engine* and the *Monitor* modules which allow operations on the classroom lights. For each scheduled session the signage displays who would use the room and the duration of the session.

Web Smart lights are realized by augmenting them with a Wi-Fi enabled tiny Web server [88], which is 35X48 mm in dimension, has an integrated 802.11 Wi-Fi interface module and a 16-bit processor, as illustrated in Figure 6.13 (b). The internal memory of 256KB is sufficient

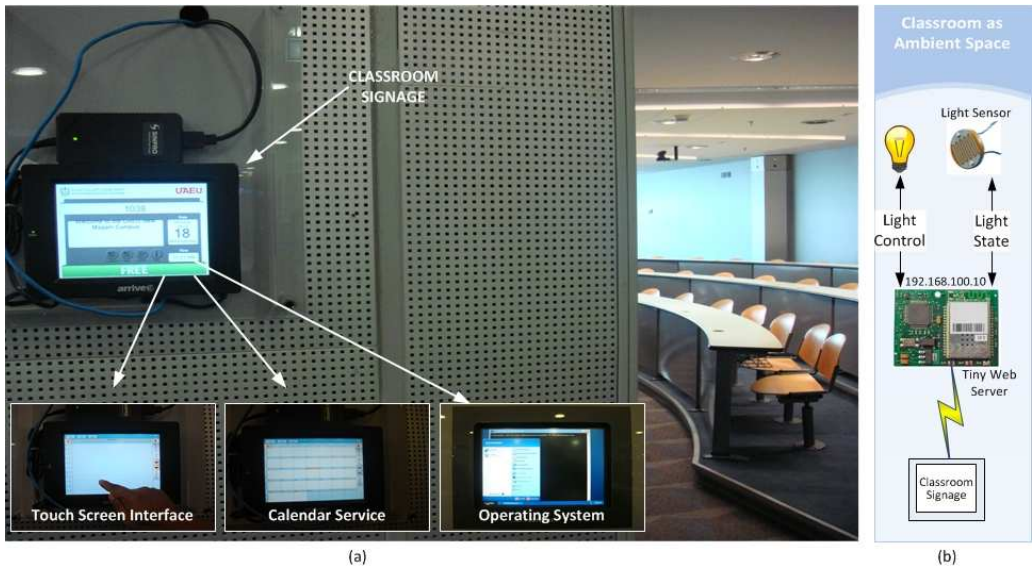


Figure 6.13: (a) Room signage fixed outside a classroom and connected to the campus Intranet. Insets show the operating system, the monthly view of the calendar service and the touch screen interface. (b) Things Control and Sensing (TCS) subsystem.

for the intended Web application. Dynamic Web pages access inputs and output ports which can augment any equipment to Web-enable them and raise their eligibility to Web Smart. We configured the module with a static IP address to be part of the Intranet. The lights and sensors are connected to the module to control the lights and measure their intensity. The Web server parses the incoming URLs to determine the action and perform the corresponding operations.

A tuple with four variables represent an instance of a user's Energy Account. The variables $\{R_i, U_j, T_s, T_e\}$ signify room ID, user ID, start time, and end time. For each scheduled session T_e (time to end), is configured to five minutes more than the scheduled duration. At time T_e the `Light-ControlWS()` service is triggered to switch the lights off unless they are switched off earlier from the signage. T_e is updated if the lights are switched off earlier than T_e and then the tuple is used to update the user's energy account. The tuple represents the user's contribution

Algorithm 6.1 Report generation for classroom ambiance

```
Proc Detect_Light_MalfunctionWS(R_i)
Begin
//Loop through light fixtures in classrooms
For each L_j in R_i do
    //Check light status
    If Check_LightWS(L_j) <> False then
        Pass(L_j,R_i)
    Else
        Fail(L_j,R_i)
    End If
End For
End
```

towards usage of lights for the particular session and this information is made available to the user. Presented below are the procedures to detect the possible light malfunctions, where `Detect_Light_MalfunctionWS` (Algorithm 6.1) and `Check_LightWS` (Algorithm 6.2) compose services provided by TCS. Using `Detect_Light_MalfunctionWS()` every light fixtures in a room (R_i) is checked and reports are generated. `Check_LightWS()` triggers the light services to switch the lights on in a classroom and then use the corresponding sensor to check if they actually are. If TCS records a malfunction, these are listed as issues and the maintenance division is informed of the same.

The notations used in the above procedures are as follow.

- **Light-SenseWS(L_j)**: This RESTful service senses the illumination of lights Lj and confirms the functionality or the lack thereof. An example of the RESTful expression of this service is, `http://univcampus.edu/building1/classroom1/FrontLights/check` where the parameter L_j is the FrontLights.

Algorithm 6.2 Crosscheck the triggered state and sensor result

```
Proc Check_LightWS (L_j)
Output: True, False
Auxiliary: i: Boolean
Begin
//Check if status of light is same as sensor reults
For i -> False,True
    If Light-ControlWS(L_j, i) XOR Light-SenseWS(L_j) <> False
        Output: False
        Exit
    End If
End For
    Output: True
End
```

- **Light-ControlWS(L_j, switch)**: This Web service is triggered for the light L_j by parsing HTTP URLs send to the tiny Web server. An example of a RESTful expression of this service is, `http://univcampus.edu/building1/classroom1/FrontLights/On` where the parameter switch, is true.
- **Ri**: Represents the room (classroom) for which the detection is done.
- **Lj**: Represent a set of light fixtures that is controlled with one Web server.
- **Pass(Lj, Ri)**: Records proper functioning of lights L_j in classroom R_i.
- **Fail(Lj, Ri)**: Records failure of lights L_j in classroom R_i.

6.2.5 Preliminary Evaluation

We conducted an evaluation to get insights into the impact of our application on saving energy and consumers' attitude of having energy accounts. To achieve this goal, we converted two classrooms into Ambient Classrooms to monitor the use of lights for seven days. The averaged results shown in Figure 6.14 positively contrast with the results perceived without our application (Figure 6.11 *versus* Figure 6.14). During the evaluation, it was observed that many faculty members switched the lights off promptly from the classroom signage. The major reason for this change in behavior is reported by them to be because of their increased awareness provided by their *Energy Accounts* which monitored their energy use.

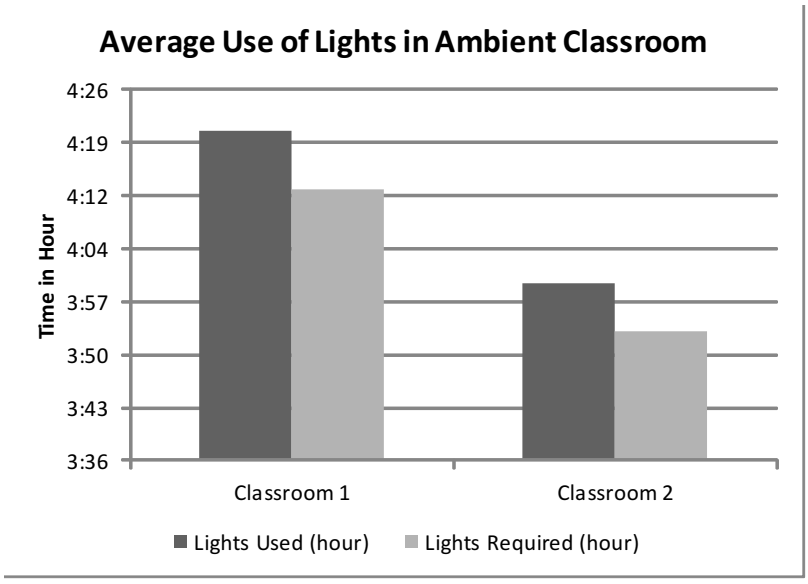


Figure 6.14: The Average use of lights in ambient classrooms

In our application, the TCS module is resource challenged because of limited storage size, and processing capabilities. It is a design proposition to have substantial amount of the processing

and storage in modules that are located at an earlier stage in the flow of information and control. We have demonstrated this by having the TCS module as an endpoint in our framework and used the other three modules to host the other major functionalities. The infrastructural cost of implementing our framework is a limitation. We believe this limitation is soon overcome since most modern campuses are Wi-Fi enabled and there is also a decline in the cost and size of communication devices. We continue to expose other classroom related services for projectors and cameras which would enable more efficient use of power.

6.2.6 Discussion and Conclusion

Abstracting physical units like classrooms as AS makes it possible to monitor and control environmental parameters, like lighting, over the Web. Energy Accounts are useful parameters for tracking an individual's energy footprint and introducing incentives for reduced energy consumption. It raises awareness towards sustaining energy and personalizes energy usage.

6.3 Summary

In this chapter, we demonstrated the implementation and benefits of using AS to represent things as Web resources. The two case studies helped us evaluate the various challenges for implementing our application. Both the case studies were focused on providing a more sustainable environment by allowing users to proactively make choices that effect the environment. The growing use of the Web and the interaction between people and things on the Web allows the realization of such scenarios. In the next chapter, we conclude with our final remarks and future work.

Chapter 7

Conclusions

In this chapter, we summarize the contributions of this dissertation and discuss some future research directions towards the realization of the Web of Things.

7.1 Summary

The increased proliferation of *things* with potential information that benefits society, business, and industry is driving research towards realizing *Ubiquitous* computing [1]. In the recent years, the *Web of Things* (WoT) is emerging as a promising technology for effectively bridging the digital gap with real-world things through the Web. With the increasing number of real-world things connected to the Web, the WoT has taken on a new significance and importance, attracting significant industry and research interest [8, 9, 15, 22, 28, 64]. However, existing ad-hoc and proprietary methods of interacting with things are not sufficient to meet the scale of candidate things for the WoT. Adequate solutions towards this problem are expected to enable flexible, robust, and usable ubiquitous Web applications.

In this dissertation, we have proposed an approach for classifying candidate things for the

Web, whereby a scalable framework to subsume real-world things into the Web is subsequently derived. We also provided a *Social Web of Things* (SWoT) structure as a gateway to populate the future WoT. Several case-studies and related prototype implementations illustrating our approach have been discussed showing WoT applications realization. A detailed survey of the state-of-the-art, and a critical review of the current developments are also presented. Next, we summarize our main research contributions as follows.

(a) A classification and semantic model of candidate things for the WoT

Some weaknesses seen in previous WoT related literature originate from the fact that things do not share a full understanding of their semantics, which has largely affected the integration of things into the Web. We proposed a capability-based classification for potential things to join the Web. The *Identity, Processing, Communication,* and *Storage* (IPCS) capabilities of things are structured as a *Web Object Metadata* (WOM). Our proposed WOM structure is also extended by integrating other ontologies that provide the necessary semantics to represent things on the Web. Our approach to classify things lays down a foundation for integrating different types of things into the WoT. This approach requires things to be *Web Smart* (i.e., things that include all IPCS capabilities for the Web) to participate in the WoT.

Our capability-based classification and related semantics of Web Smart things provide a structured approach towards the access, control, and management of things, on the common, widely accepted, open, and existing platform of the Web. Our classification also supports WoT architects to have a common referential taxonomy of candidate things on the Web. Our approach annotates things with semantic metadata to increase interoperability as well as provide contextual information, which are essential for contextual knowledge. The classification of candidate things eases thing's deployment into the WoT, contributes to the structured integration of things into the Web, and supports their semantics to adapt to application contexts.

Our initial attempt to model things and their properties on the Web [12] developed into an ontological structure used to classify and capture the semantics of real-world things [13]. This approach is capable of abstracting the required capabilities of a thing that has relevant information to be shared on the Web. Research contributions towards the classification of things for the WoT are still in their preliminary stages and hence our contribution is important as it plays a foundational role in defining semantic specification of things for creating applications that subsume real-world things.

(b) Ambient Space framework to automatically integrate heterogeneous things into the WoT

We introduced our concept of *Ambient Spaces* (AS) and presented the *AS framework* which functions as a gateway for Web applications to connect to Web Smart things. The process of augmenting things with Web Smart capabilities was described and an extended WOM representation of things was presented. Illustrative examples were also discussed.

While ad-hoc solutions are being studied and deployed, our framework takes advantage of the fact that things are duplicated within spatial patterns, for example, every classroom has a projector, or every parking spot has a parking sensor. By providing examples we revealed how our *AS framework* exploits the fact that things are replicated within such hierarchical spatial bifurcations. We targeted the problem of realizing the WoT by defining AS to be homogeneous virtual spaces that composes Web Smart things in an application context [17]. Thus an AS represents the virtual encapsulation of Web Smart things that resides within a physical space. The integration of many such AS instances enable the realization of WoT in a scalable manner. The proliferation of things into the Web is channeled hierarchically through our AS framework to harness the interoperability of things within scaled down physical spaces. On the other hand things are inherently proprietary, i.e., they are owned by individuals or institutions, and

have social associations. We also discussed important properties of things to identify social implications of their virtual exposure.

(c) A model for creating a community of things in WoT

We presented a model which uses AS to cluster a large number of things in order to enable the SWoT. We discussed our approach which creates communities of Web Smart things by using our proposed WOM semantic structure. Based on the similarities between the semantic attributes of things, clusters are created to represent communities. Two types of communities are proposed, the first was to include things into SWoT and the second was to drive things to people. Thus, by exploiting the AS framework, we were able to create a platform where communities of things are modeled and their interactions can be realized. SWoT communities provide a bridge for real-world things to connect with people on the Web and display coherent communities of Web Smart things.

The adoption of a social networking platform to integrate real-world things was proposed earlier [18]. However, that approach is restricted to privacy preservation of things and their users over existing social networking platforms. Moreover, the approach does not view things as autonomous entities of the social network. Besides, substantial human intervention is expected. We suggested a model that extends AS to realize the SWoT platform which enables things to be automatically adopted, identified, grouped, and interacted with. Thus, the SWoT functions as an intermediate layer for real-world things to connect with people on the Web. Based on relationships that things have on the SWoT, new ubiquitous applications are envisioned through which, things inherently market themselves, share and compare themselves over the Web, and things reveal their social learning value on a smart campus [19, 20, 21].

(d) Case-studies to evaluate the benefits of adapting things into the WoT

We implemented and demonstrated the benefits of using AS to represent things as Web resources. Two case studies were presented, which helped us address various challenges for implementing our framework. Both case-studies were focused on providing a more sustainable environment by allowing users to proactively make choices that positively affect the environment. The growing use of the Web and interaction between people and things on the Web allows the realization of such scenarios.

In the first case study we built an *Ambient Classroom* prototype which minimizes the excessive use of electrical energy in a campus. In the second case-study, we prototyped an *Ambient Parking* lot which was targeted towards reducing vehicle emission by providing faster access to parking spots. We evaluated the benefits and limitations, and presented a validation of our framework when implementing these scenarios.

7.2 Future Directions

Although there is a growing community of researchers investigating the challenges and potential of the WoT in the last few years, several research issues are yet to be addressed. Research investigations such as, the management of large amount of dynamic data generated by things, the security and privacy concerns about virtualizing real-world things, an evaluation framework for WoT, and the efficient discovery of things on the Web, are important and challenging. In particular, we identify the following directions for our future research which relate to efficient compositions of things on the Web, the support for security when virtualizing things and dynamic nature of of mobile things.

- **Efficient composition of things on the Web:** We presented a capability based classification of things, their related WOM structure, and a novel framework whereby things

that are replicated in physical spaces are represented as Web Smart things in AS. The framework provides a scalable approach of encapsulating things and their related services in AS. We are planning to focus our future work on improving thing's representation by reducing the size of semantic information as this information would have to be stored on relatively small things with lesser storage capacity and power. Also, our approach features the grouping of things into AS communities by manually introducing the initial seed using k-means clustering approach. The use of clustering approaches like fuzzy clustering approaches need to be investigated to study the efficiency of creating communities of things. Realizing a fully automatic composition of things in AS is still very difficult and presents several challenges. Indeed, a complete solution for delivering the composition of things is still ongoing and far from mature. Other issues like the verification and optimization of composite things are also critical to the success of the WoT platform.

- **Support for the security of things on the Web:** Beyond the functional aspects of adopting things into the Web, the non-functional properties are equally important for the WoT platform to be adopted. Non-functional properties like security, privacy, and trust in the WoT promise many opportunities for research innovations [65]. As things on the Web will be accessible and shared among many users, research in this area is crucial to the success and widespread use of the WoT. Research is growing with innovative ideas and some early approaches dealing with these issues are published such as [10, 67, 68]. Also, the use of the existing social Web as a platform to ensure the trust and privacy of things has been advocated [18], to control Web-enabled things among trusted members of social Web sites. Real-world things are proprietary and hence are personal to an individual or an institution. Hence, a wider scope of study focusing on guaranteeing the fairness of the values and QoS criteria when a community of things interacts among its members and

outside is yet to be presented.

- **Support for the mobility of things in AS:** In our future work, we also intend to extend our AS framework to support seamless access of composite services among mobile things. Indeed, during the invocation of a thing's services, especially the one having prolonged activities or with complex tasks, users are likely to be switching between things. Also, multiple things may enter or leave an AS within temporal limits. WoT applications can not be allowed to terminate and start again simply because users switch their interaction with things or things move in and out of AS. Instead, the access of thing's services should be continuously operational. In other words, users and applications should not experience a break during their interaction with mobile things. This is extremely important for time critical applications and environments (e.g., medical or defense). Another important direction is to support collaboration between multiple (mobile) things in the same application context.

Bibliography

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 94–104, 1991.

[2] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *Services Computing, IEEE Transactions on*, vol. 3, no. 3, pp. 223–235, 2010.

[3] G. Mulligan, "The internet of things: Here now and coming soon," *IEEE Internet Computing*, vol. 14, no. 1, pp. 35–36, 2010.

[4] B. Christophe, M. Boussard, M. Lu, A. Pastor, and V. Toubiana, "The web of things vision: Things as a service and interaction patterns," *Bell Labs Technical Journal*, vol. 16, no. 1, pp. 55–61, 2011.

[5] A. Danigelis, "Cities with widespread wireless internet." http://dsc.discovery.com/technology/tech-10/wireless-cities-top.html, 2010.

[6] J. Hui and D. Culler, "Extending ip to low-power, wireless personal area networks," *Internet Computing, IEEE*, vol. 12, no. 4, pp. 37–45, 2008.

[7] J. Vasseur and A. Dunkels, *Interconnecting smart objects with ip: The next internet*. Morgan Kaufmann, 2010.

[8] T. Berners-Lee, "The web of things." Special theme: The Future Web. The European Research Consortium for Informatics and Mathematics (ERCIM) News. http://ercim-news.ercim.eu/en72/keynote/the-web-of-things, January 2008.

[9] P. Olson, "Googling your lost keys, and the coming revolution in smart products." Forbes Tech, http://www.forbes.com/sites/parmyolson/2012/09/14/googling-your-lost-keys-and-the-coming-revolution-in-smart-products/.

[10] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the internet of things to the web of things: Resource-oriented architecture and best practices," in *Architecting the Internet*

of Things (D. Uckelmann, M. Harrison, and F. Michahelles, eds.), pp. 97–129, Springer Berlin Heidelberg, 2011.

[11] D. Guinard and V. Trifa, "Towards the web of things: Web mashups for embedded devices," in *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*, 2009.

[12] S. Mathew, Y. Atif, Q. Sheng, and Z. Maamar, "Ambient things on the web," *Journal of Ubiquitous Systems and Pervasive Networks (JUSPN)*, vol. 1(1), pp. 1–8, 2010.

[13] S. Mathew, Y. Atif, Q. Sheng, and Z. Maamar, "Web of things: Description, discovery and integration," in *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pp. 9–15, 2011.

[14] D. Guinard, M. Mueller, and V. Trifa, "Restifying real-world systems: A practical case study in rfid," in *REST: From Research to Practice* (E. Wilde and C. Pautasso, eds.), pp. 359–379, Springer New York, 2011.

[15] B. Ostermaier, M. Kovatsch, and S. Santini, "Connecting things to the web using programmable low-power wifi modules," in *Proceedings of the Second International Workshop on Web of Things*, WoT '11, (New York, NY, USA), pp. 2:1–2:6, ACM, 2011.

[16] S. S. Mathew, Y. Atif, Q. Z. Sheng, and Z. Maamar, "The web of things - challenges and enabling technologies," in *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence* (N. Bessis, F. Xhafa, D. Varvarigou, R. Hill, and M. Li, eds.), vol. 460 of *Studies in Computational Intelligence*, pp. 1–23, Springer Berlin Heidelberg, 2013.

[17] S. S. Mathew, "Managing things in an ambient space," in *Service-Oriented Computing - ICSOC 2011 Workshops* (G. Pallis, M. Jmaiel, A. Charfi, S. Graupner, Y. Karabulut, S. Guinea, F. Rosenberg, Q. Sheng, C. Pautasso, and S. Mokhtar, eds.), vol. 7221 of *Lecture Notes in Computer Science*, pp. 226–232, Springer Berlin Heidelberg, 2012.

[18] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable web of things," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pp. 702–707, 2010.

[19] Y. Atif, S. Mathew, and A. Lakas, "Ubiquitous learning specification in a pervasive smart campus model," *Pervasive and Mobile Computing*, 2012. Submitted.

[20] S. S. Mathew, Y. Atif, Q. Z. Sheng, and Z. Maamar, "Building sustainable parking lots with the web of things," *Personal and Ubiquitous Computing*, pp. 1–13, 2013.

[21] S. Mathew, Y. Atif, Q. Sheng, and Z. Maamar, "A new parking system based on the web of things," in *Proceedings of the workshop on Computing and Networking for Internet of Things (ComNet-IoT), in conjunction with International Conference on Distributed Computing and Networking (ICDCN)*, Mumbai, India, January , 2013.

[22] D. Raggett, "The web of things: Extending the web into the real world," in *SOFSEM 2010: Theory and Practice of Computer Science* (J. Leeuwen, A. Muscholl, D. Peleg, J. PokornÃœ, and B. Rumpe, eds.), vol. 5901 of *Lecture Notes in Computer Science*, pp. 96–107, Springer Berlin Heidelberg, 2010.

[23] Z. Song, A. Cardenas, and R. Masuoka, "Semantic middleware for the internet of things," in *Internet of Things (IOT), 2010*, pp. 1–8, 2010.

[24] D. H. Steinberg and S. Cheshire, *Zero configuration networking: The definitive guide*. O'reilly, 2010.

[25] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, "Smews: Smart and mobile embedded web server," in *Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on*, pp. 571–576, 2009.

[26] H. Song, D. Kim, K. Lee, and J. Sung, "Upnp-based sensor network management architecture," in *Proc. ICMU Conf*, 2005.

[27] J. Allard, V. Chinta, S. Gundala, and I. Richard, G.G., "Jini meets upnp: an architecture for jini/upnp interoperability," in *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*, pp. 268–275, 2003.

[28] "The web of things." White Paper, Tridium Inc. http://www.tridium.com/galleries/white_papers/WP-SedonaWeb.pdf, 2009.

[29] D. Lizcano, M. JimÃ©nez, J. Soriano, J. Cantera, M. Reyes, J. Hierro, F. Garijo, and N. Tsouroulas, "Leveraging the upcoming internet of services through an open user-service front-end framework," vol. 5377, pp. 147–158, 2008.

[30] Z. Maamar, Q. Sheng, Y. Atif, S. S. Mathew, and K. Boukadi, "Towards an approach for weaving preferences into web services operation," *Journal of Software*, vol. 7, no. 7, 2012.

[31] W. W. Group, "Web services architecture." http://www.w3.org/tr/ws-arch.

[32] R. Fielding, *Architectural styles and the design of network-based software architectures.* PhD thesis, University of California, 2000.

[33] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: making the right architectural decision," in *Proceedings of the 17th international conference on World Wide Web*, WWW '08, (New York, NY, USA), pp. 805–814, ACM, 2008.

[34] D. Guinard, *A Web of Things Application Architecture - Integrating the Real-World into the Web.* PhD thesis, Ph.D. dissertation, 2011.

[35] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, "Tinyrest - a protocol for integrating sensor networks into the internet," in *in Proc. of REALWSN*, 2005.

[36] W. Drytkiewicz, I. Radusch, S. Arbanowski, and R. Popescu-Zeletin, "prest: a rest-based protocol for pervasive systems," in *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, pp. 340–348, 2004.

[37] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte, and D. Winer, "Simple object access protocol (soap) 1.1." http://www.immagic.com/eLibrary/ARCHIVES/SUPRSDED/W3C/W000520N.pdf, 2000.

[38] D. Booth and C. K. Liu, "Web services description language (wsdl) version 2.0 part 0: Primer." http://www.w3pdf.com/W3cSpec/WDSL/1/wsdl20-primer.pdf, 2007.

[39] W3C, "Web services addressing." http://www.w3.org/2002/ws/addr/.

[40] L. Clement, A. Hately, C. Von Riegen, T. Rogers, T. Bellwood, S. Capell, J. Colgrave, M. Dovey, D. Feygin, R. Kochman, *et al.*, "Uddi version 3.0. 2, 2004," *URL http://uddi. org/pubs/uddi v3. htm.*

[41] OASIS, "Web services resources framework (wsrf 1.2), 2006." http://www.oasis-open.org/committees/wsrf/.

[42] F. Curbera, F. Leymann, T. Storey, D. Ferguson, and S. Weerawarana, *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more.* Prentice Hall PTR Englewood Cliffs, 2005.

[43] S. de Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker, "Soda: Service oriented device architecture," *Pervasive Computing, IEEE*, vol. 5, no. 3, pp. 94–96, 2006.

[44] A. Pintus, D. Carboni, A. Piras, and A. Giordano, "Connecting smart things through web services orchestrations," vol. 6385, pp. 431–441, 2010.

[45] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 1, pp. 62–70, 2005.

[46] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. Souza, and V. Trifa, "Soa-based integration of the internet of things in enterprise services," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pp. 968–975, 2009.

[47] I. Samaras, J. Gialelis, and G. Hassapis, "Integrating wireless sensor networks into enterprise information systems by using web services," in *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, pp. 580–587, 2009.

[48] A. Sleman and R. Moeller, "Integration of wireless sensor network services into other home and industrial networks; using device profile for web services (dpws)," in *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pp. 1–5, 2008.

[49] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, "The web of things: Interconnecting devices with high usability and performance," in *Embedded Software and Systems, 2009. ICESS '09. International Conference on*, pp. 323–330, 2009.

[50] I. Agranat, "Engineering web technologies for embedded applications," *Internet Computing, IEEE*, vol. 2, no. 3, pp. 40–45, 1998.

[51] M. C. Filibeli, O. Ozkasap, and M. R. Civanlar, "Embedded web server-based home appliance networks," *Journal of Network and Computer Applications*, vol. 30, no. 2, pp. 499 – 514, 2007.

[52] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: design and implementation of interoperable and evolvable sensor networks," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, (New York, NY, USA), pp. 253–266, ACM, 2008.

[53] G. Broll, E. Rukzio, M. Paolucci, M. Wagner, A. Schmidt, and H. Hussmann, "Perci: Pervasive service interaction with the internet of things," *Internet Computing, IEEE*, vol. 13, no. 6, pp. 74–81, 2009.

[54] Y. Wu, Q. Z. Sheng, D. Ranasinghe, and L. Yao, "Peertrack: a platform for tracking and tracing objects in large-scale traceability networks," in *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, (New York, NY, USA), pp. 586–589, ACM, 2012.

[55] D. Zeng, S. Guo, and Z. Cheng, "The web of things: A survey (invited paper)," *Journal of Communications*, vol. 6, no. 6, 2011.

[56] J. Garrett *et al.*, "Ajax: A new approach to web applications," *http://experiencezen.com/wp-content/uploads/2007/04/adaptive-path-ajax-a-new-approach-to-web-applications1.pdf*, 2007.

[57] A. Russell, "Comet: Low latency data for browsers," *The Dojo Toolkit, http://infrequently.org/wp-content/LowLatencyData.pdf*, 2006.

[58] D. Guinard, C. Floerkemeier, and S. Sarma, "Cloud computing, rest and mashups to simplify rfid application development and deployment," in *Proceedings of the Second International Workshop on Web of Things*, WoT '11, (New York, NY, USA), pp. 9:1–9:6, ACM, 2011.

[59] O. Akribopoulos, I. Chatzigiannakis, C. Koninis, and E. Theodoridis, "A web services-oriented architecture for integrating small programmable objects in the web of things," in *Developments in E-systems Engineering (DESE), 2010*, pp. 70–75, 2010.

[60] D. Guinard, "Mashing up your web-enabled home," vol. 6385, pp. 442–446, 2010.

[61] B. Ostermaier, K. Romer, F. Mattern, M. Fahrmair, and W. Kellerer, "A real-time search engine for the web of things," in *Internet of Things (IOT), 2010*, pp. 1–8, 2010.

[62] S. Mayer and D. Guinard, "An extensible discovery service for smart things," in *Proceedings of the Second International Workshop on Web of Things*, WoT '11, (New York, NY, USA), pp. 7:1–7:6, ACM, 2011.

[63] Z. Shelby, "Embedded web services," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 52–57, 2010.

[64] G. Mulligan, "The 6lowpan architecture," in *Proceedings of the 4th workshop on Embedded networked sensors*, EmNets '07, (New York, NY, USA), pp. 78–82, ACM, 2007.

[65] V. Gupta, P. Udupi, and A. Poursohi, "Early lessons from building sensor.network: an open data exchange for the web of things," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pp. 738–744, 2010.

[66] E. Wilde, "Putting things to rest. school of information," tech. rep., UC Berkeley, Tech. Rep. UCB iSchool Report 15, 2007.

[67] R. H. Weber, "Internet of things–new security and privacy challenges," *Computer Law & Security Review*, vol. 26, no. 1, pp. 23 – 30, 2010.

[68] C. M. Medaglia and A. Serbanati, "An overview of privacy and security issues in the internet of things," in *The Internet of Things* (D. Giusto, A. Iera, G. Morabito, and L. Atzori, eds.), pp. 389–395, Springer New York, 2010.

[69] V. Trifa, S. Wieland, D. Guinard, and T. Bohnert, "Design and implementation of a gateway for web-based interaction and management of embedded devices," *Submitted to DCOSS*, 2009.

[70] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, and G. Borriello, "Building the internet of things using rfid: The rfid ecosystem experience," *Internet Computing, IEEE*, vol. 13, no. 3, pp. 48–55, 2009.

[71] M. Boussard, B. Christophe, O. Le Berre, and V. Toubiana, "Providing user support in web-of-things enabled smart spaces," in *Proceedings of the Second International Workshop on Web of Things*, WoT '11, (New York, NY, USA), pp. 11:1–11:6, ACM, 2011.

[72] T. Mantoro, M. A. Ayu, and E. E. Elnour, "Web-enabled smart home using wireless node infrastructure," in *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '11, (New York, NY, USA), pp. 72–79, ACM, 2011.

[73] A.-G. Paetz, E. Dütschke, and W. Fichtner, "Smart homes as a means to sustainable energy consumption: A study of consumer perceptions," *Journal of Consumer Policy*, vol. 35, no. 1, pp. 23–41, 2012.

[74] Y. Atif, Y. Badr, and Z. Maamar, "Towards a new-digital learning ecosystem based on autonomic web services," in *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, pp. 180–185, 2010.

[75] W. Huang, Y.-X. Zhao, Y. Xiao, and X.-H. Sun, "A cloud service environment framework for scorm compatible content," in *Circuits, Communications and System (PACCS), 2011 Third Pacific-Asia Conference on*, pp. 1–4, 2011.

[76] G.-J. Hwang, C.-H. Wu, J. C. R. Tseng, and I. Huang, "Development of a ubiquitous learning platform based on a real-time help-seeking mechanism," *British Journal of Educational Technology*, vol. 42, no. 6, pp. 992–1002, 2011.

[77] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the internet of things," *Internet Computing, IEEE*, vol. 14, no. 1, pp. 44–51, 2010.

[78] M. Beigl, H.-W. Gellersen, and A. Schmidt, "Mediacups: experience with design and use of computer-augmented everyday artefacts," *Computer Networks*, vol. 35, no. 4, pp. 401 – 409, 2001. <ce:title>Pervasive Computing</ce:title>.

[79] F. Mattern, "From smart devices to smart everyday objects," in *Proceedings of Smart Objects Conference*, (Grenoble, France), 2003.

[80] G. C. Bowker and S. L. Star, *Sorting things out: Classification and its consequences*. The MIT Press, 28 August, 2000.

[81] S. Cranefield, S. Haustein, and M. Purvis, "Uml-based ontology modelling for software agents," *Information Science Discussion Papers Series No. 2001/07*, vol. http://hdl.handle.net/10523/1081, 2001.

[82] E. Sirin and B. Parsia, "Sparql-dl: Sparql query for owl-dl.," in *OWLED*, vol. 258, 2007.

[83] B. McBride, "Jena: a semantic web toolkit," *Internet Computing, IEEE*, vol. 6, no. 6, pp. 55–59, 2002.

[84] Y. Atif and S. S. Mathew, "Ambient learning companion," in *EDULEARN10 Proceedings*, 2nd International Conference on Education and New Learning Technologies, pp. 4787–4791, IATED, 5-7 July, 2010.

[85] A. Passant and P. Laublet, "Meaning of a tag: A collaborative approach to bridge the gap between tagging and linked data," in *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China*, (Beijing, China), p. 48, 2008.

[86] R. Want, "Near field communication," *Pervasive Computing, IEEE*, vol. 10, no. 3, pp. 4–7, 2011.

[87] K. Finkenzeller, *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. Wiley, 2010.

[88] F. W.-F. module from openPICUS. http://www.openpicus.com/site/technology/products, Accessed on 21/03/2013.

[89] Z. Maamar, H. Hacid, and M. Huhns, "Why web services need social networks," *Internet Computing, IEEE*, vol. 15, no. 2, pp. 90–94, 2011.

[90] W. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao, "Senseweb: An infrastructure for shared sensing," *MultiMedia, IEEE*, vol. 14, no. 4, pp. 8–13, 2007.

[91] M. Langheinrich, "A privacy awareness system for ubiquitous computing environments," in *UbiComp 2002: Ubiquitous Computing* (G. Borriello and L. Holmquist, eds.), vol. 2498 of *Lecture Notes in Computer Science*, pp. 237–245, Springer Berlin Heidelberg, 2002.

[92] M. L. Mazurek, J. P. Arsenault, J. Bresee, N. Gupta, I. Ion, C. Johns, D. Lee, Y. Liang, J. Olsen, B. Salmon, R. Shay, K. Vaniea, L. Bauer, L. F. Cranor, G. R. Ganger, and M. K. Reiter, "Access control for home data sharing: Attitudes, needs and practices," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, (New York, NY, USA), pp. 645–654, ACM, 2010.

[93] J. Mitchell-Wong, R. Kowalczyk, A. Roshelova, B. Joy, and H. Tsai, "Opensocial: From social networks to social ecosystem," in *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*, pp. 361–366, 2007.

[94] S. Mathew, Y. Atif, Q. Sheng, and Z. Maamar, "Towards an efficient sales pitch with the web of things," *IEEE International Conference on eBusiness Engineering (ICEBE)*, Coventry, United Kingdom, 2013.

[95] R. M. Morgan and S. D. Hunt, "The commitment-trust theory of relationship marketing," *The Journal of Marketing*, vol. 58, pp. 20–38, July, 1994.

[96] M. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 40, no. 3, pp. 211–218, 2006.

[97] T. Tran, R. Nayak, and P. Bruza, "Combining structure and content similarities for xml document clustering," in *Proceedings of the 7th Australasian Data Mining Conference - Volume 87*, AusDM '08, (Darlinghurst, Australia, Australia), pp. 219–225, Australian Computer Society, Inc., 2008.

[98] G. Salton, *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[99] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, p. 14, California, USA, 1967.

[100] B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, (New York, NY, USA), pp. 16–22, ACM, 1999.

[101] R. Xu and I. Wunsch, D., "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[102] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Mining and Knowledge Discovery*, vol. 2, no. 3, pp. 283–304, 1998.

[103] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 881–892, 2002.

[104] K. J. Cios, R. W. Swiniarski, W. Pedrycz, and L. A. Kurgan, "Unsupervised learning: Clustering," in *Data Mining*, pp. 257–288, Springer US, 2007.

[105] C. M. Vries, S. Geva, and L. Vine, "Clustering with random indexing k-tree and xml structure," vol. 6203, pp. 407–415, 2010.

[106] M. Sahlgren, "An introduction to random indexing," in *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE*, vol. 5, 2005.

[107] D. L. McCollum, V. Krey, and K. Riahi, "Beyond rio: Sustainable energy scenarios for the 21st century," vol. 36, pp. 215–230, 2012.

[108] E. Gantelet and A. Lefauconnier, "The time looking for a parking space: strategies, associated nuisances and stakes of parking management in france," tech. rep., SARECO research report, Association for European Transport and contributors, 2006.

[109] B. Shannon and M. Dauvergne, "Police-reported crime statistics in canada, 2010.." Statistics Canada Catalogue no. 85-002-X. http://www.statcan.gc.ca/pub/85-002-x/2011001/article/11523-eng.htm.

[110] R. Lu, X. Lin, H. Zhu, and X. Shen, "Spark: A new vanet-based smart parking scheme for large parking lots," in *INFOCOM 2009, IEEE*, pp. 1413–1421, 2009.

[111] M. Chen and T. Chang, "A parking guidance and information system based on wireless sensor network," in *Information and Automation (ICIA), 2011 IEEE International Conference on*, pp. 601–605, 2011.

[112] Y. Dong-he and L. Xi-ang, "Research of intelligent mobile robot's obstacle avoidance based on ultrasonic sensor," *Computer Engineering and Design*, vol. 28, no. 15, pp. 3659–3661, 2007.

[113] H. Frey, A. Unal, N. Rouphail, and J. Colyar, "On-road measurement of vehicle tailpipe emissions using a portable instrument," *Journal of the Air & Waste Management Association*, vol. 53, no. 8, pp. 992–1002, 2003.

[114] O. Masoso and L. Grobler, "The dark side of occupant's behaviour on building energy use," *Energy and Buildings*, vol. 42, no. 2, pp. 173 – 177, 2010.

[115] F. Hölker, C. Wolter, E. Perkin, K. Tockner, *et al.*, "Light pollution as a biodiversity threat.," *Trends in ecology & evolution*, vol. 25, no. 12, pp. 681–682, 2010.

[116] X. Wang, J. S. Dong, C. Chin, S. R. Hettiarachchi, and D. Zhang, "Semantic space: An infrastructure for smart spaces," *Computing*, vol. 1, no. 2, pp. 67–74, 2002.

[117] C. Prehofer, J. Van Gurp, V. Stirbu, S. Satish, S. Tarkoma, C. Di Flora, and P. Liimatainen, "Practical web-based smart spaces," *Pervasive Computing, IEEE*, vol. 9, no. 3, pp. 72–80, 2010.

[118] "The energy observer, energy efficiency information for the facility manager quarterly issue - december 2007." http://www.michigan.gov/documents/dleg/EO-12-07-218809-7.pdf.

[119] Y. Agarwal, T. Weng, and R. K. Gupta, "The energy dashboard: improving the visibility of energy consumption at a campus-wide scale," in *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, (New York, NY, USA), pp. 55–60, ACM, 2009.

[120] S. Darby, "Making it obvious: Designing feedback into energy consumption," in *Energy Efficiency in Household Appliances and Lighting* (P. Bertoldi, A. Ricci, and A. Almeida, eds.), pp. 685–696, Springer Berlin Heidelberg, 2001.

Glossary

A

AS - Ambient Space

ASM - Ambient Space Manager

I

IoT - Internet of Things

IPCS - Identity, Processing, Communication, and Storage

P

PMS - Parking Management System

R

RFID - Radio Frequency IDentification

S

SPC - Smart Parking Client

SPS - Smart Parking Spot

SWoT - Social Web of Things

T

TWS - Tiny Web Server

W

WOM - Web Object Metadata

WoT - Web of Things

Curriculum Vitae

Full Name: Sujith Samuel Mathew

Email: sujith@adelaide.edu.au

Home Page: http://cs.adelaide.edu.au/~sujith

Research Interests

Distributed Computing, Ubiquitous Computing, and Mobile Computing

Education

- **Ph.D. in Computer Science, 2013**
  - School of Computer Science, the University of Adelaide (UoA). Adelaide, Australia.
  - Dissertation: *Classifying and Clustering the Web of Things*
  - Supervised by A/Prof. Michael Sheng, A/Prof. Yacine Atif, and Prof. Zakaria Maamar

- **Master of Technology (M.Tech) in Software Engineering, 1998 - 2000**
  - Visveswaraiah Technological University, Karnataka, India.
  - First Class with Distinction
- **Masters of Computer Applications (MCA), 1994 - 1997**
  - University of Mysore, Karnataka, India
  - First Class with Distinction

Industry Experience

- From 2004 to 2006, Group Leader and Technical Evangelist, Infosys Technologies Ltd.
- From 2002 to 2004, Software Engineer, Huawei Technologies Ltd.
- From 2000 to 2002, Software Specialist, Digital Equipment Corporation (DEC).

Teaching Experience

- Instructor for Computer Science and Electronic Commerce departments , College of IT, UAE University, (since 2006)
  - *Courses Taught*: E-Business Processes, E-Business Models, Internet Computing, Data Structures using Java, Algorithms and Problem Solving, Object Oriented Programing, Database Systems, and Advance Database Systems.

Publications

Refereed Journal Publications:

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "Building Sustainable Parking Lots with the Web of Things", Personal and Ubiquitous Computing (PUC) journal, pp. 1-13, Springer-Verlag. June 2013.

Maamar Z., Sheng Q.Z., Atif Y., **Mathew S.S.** and Boukadi K, "Towards an Approach for Weaving Preferences into Web Services Operation", Journal of Software. Vol. 7(7), pp. 1429-1439. 2012.

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "Ambient things on the Web", Journal of Ubiquitous Systems and Pervasive Networks (JUSPN). Vol. 1(1), pp. 1-8. 2010.

Refereed Conference Publications:

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "Towards an efficient sales pitch with the Web of Things", In the Tenth IEEE International Conference on eBusiness Engineering (ICEBE). Coventry, United Kingdom, 2013.

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "A New Parking System based on the Web of Things", In Proceedings of the workshop on Computing and Networking for Internet of Things (ComNet-IoT), in conjunction with International Conference on Distributed Computing and Networking (ICDCN). 2013.

Mathew S.S. , "Managing Things in an Ambient Space", In the Ninth International Confer-

ence on Service Oriented Computing (ICSOC) , pp. 226-232. 2012.

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "Web of Things: Description, Discovery and Integration", In International Conference on Internet of Things and Cyber, Physical and Social Computing (iThings/CPSCom). , pp. 9-15. 2011.

Citations: *6*

Atif Y., Serhani M.A., Campbell P. and **Mathew S.S.**, "Trusted Translation Services", Collaborative Computing: Networking, Applications and Worksharing. , pp. 778-791. Springer. 2009.

Atif Y., **Mathew S.S.**, "Ambient Learning Companion", in Proceedings of 2nd International Conference on Education and New Learning Technologies (EDULEARN10), pp. 4787-4791. Barcelona, Spain. 2010.

Refereed Book Chapter:

Mathew S.S., Atif Y., Sheng Q. and Maamar Z., "The Web of Things - Challenges and Enabling Technologies", In Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence (N. Bessis, F. Xhafa, D. Varvarigou, R. Hill, and M. Li, eds.), Vol. 460 of Studies in Computational Intelligence, pp. 1-23. Springer Berlin Heidelberg. 2013.

Book:

Trabelsi Z., Hayawi K., Al Braiki A. and **Mathew S.S.**, "Network Attacks and Defenses: A Hands-On Approach" Auerbach Publishers, (Taylor and Francis). 2012.

Professional Services

- Program Chair, The 3rd International Workshop on Internet of Ubiquitous and Pervasive Things (IUPT), Halifax, Nova Scotia, Canada, June 25-28, 2013.
- Program Chair, The 2nd International Workshop on Internet of Ubiquitous and Pervasive Things (IUPT), Niagara Falls, Ontario, Canada, August 27-29, 2012.
- Technical Program Committee member, The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN), Niagara Falls, Ontario, Canada, October 21-24, 2013.
- Technical Program Committee member, The 4th International Conference on Advances in Future Internet (AFIN), Rome, Italy, August 19 - 24, 2012.
- Technical Program Committee member, The 3rd International Conference on Advances in Future Internet (AFIN), French Riviera, France, August 21-27, 2011.
- Web Administration Chair, The 7th International Symposium on Web and Mobile Information Services (WAMIS), Singapore, March 22-25, 2011.
- Web Administration Chair, The 6th International Symposium on Web and Mobile Information Services (WAMIS), Perth, Australia, April 20-23, 2010.
- Organizing committee member for UAE National Programming Contest (NPC) 2008 and 2009.

Technical Skills

- Worked in all phases of both project and product lifecycles of creating software. Exposure to analyzing, designing, implementing, and testing various layers of wireless communication protocol stacks like Bluetooth and 3G.
- Proficient in building JEE applications.
- Experience working on the IBM CELL Broadband Engine (BE), implementing parallelized algorithms in C.
- Databases worked on: Oracle, MySQL, and SQL Server.
- Languages proficient in: C, C++, and JAVA.
- Experience building mobile applications in Palm OS, Win CE, and Android platforms.