University of Piraeus
Department of Digital Systems
Postgraduate Program “Digital Communications and Networks”

Analysis and extension of IoT ontologies towards autonomous reasoning of things

Supervisor: Kyriazis Dimosthenis

Koniari-Brili Maria-Eleni

Contents

Abstract 4
Περίληψη 5
1. Introduction 6
Internet of Things 6
1.1 Thesis Structure 7
1.2 IoT-related Technologies 8
1.2.1 Architecture of IoT 8
1.2.2 IoT Elements 8
1.2.3 Radio Frequency Identification (RFID) 8
1.2.4 Wireless Sensor Networks (WSN) 9
1.2.5 Addressing schemes 9
1.2.6 Data storage and analytics 10
1.2.7 Visualization 11
1.3 Application Domains 11
1.3.1 Personal and Home 11
1.3.2 Enterprise 12
1.3.3 Utilities 12
1.3.4 Mobile 13
2. Problem Statement 15
3. State of the Art 18
3.1. Ontologies 18
3.1.1. Sensor Web Autonomy (SWAMO Ontology) 18
3.1.2. CESN: An Ontology of Sensors and their Measurements 20
3.1.3. Extensible Observation Ontology (OBOE) 21
3.1.4 Device Ontology 23
3.1.5 Physics Domain Ontology 25
3.1.6 Estimation Ontology 27
3.1.7. SensorML 28
3.2. Distributed Coordination 30
3.2.1. Peer-To-Peer Protocols 31
Overview of P2P system structures 31

3.2.2 Chubby and ZooKeeper 34
Design 35
Background 36
4. Approach 37
4.1. Ontologies 37
4.2. Distributed Coordination 38
5. Social-aware Ontologies 40
Case Study 46
Future work 47
6. Conclusion 48
References 49

Abstract

This thesis has the objective to analyse and propose the development of an environment where devices (e.g. mobile phones, digital cameras etc.) can communicate between them, share experiences and adjust depending on information that receive from different “things”. Based on the idea of Internet of Things (IoT), where all objects will be equipped with diminutive tracking devices or identifiers machines which will convert our daily life by making her easier, we achieved to design an environment where different objects-things can share experiences. According to this theory (IoT theory) and the utilization of existing ontologies the question about how we can develop an application based on specific ontology which will enable experiences sharing and with the prospect of future development in the context of a Smart City, became more understandable and help us to develop a representative paradigm of a Smart City.

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως αντικείμενο την μελέτη και την ανάπτυξη ενός περιβάλλοντος όπου οι συσκευές (π.χ. κινητά τηλέφωνα, ψηφιακές φωτογραφικές μηχανές, κλπ.) μπορούν να επικοινωνούν μεταξύ τους, να μοιραστούν εμπειρίες και να προσαρμοστούν ανάλογα με τις πληροφορίες που λαμβάνουν από διαφορετικά "πράγματα". Βασιζόμενοι στην ιδέα του Ίντερνετ των πραγμάτων (Internet of Things-IoT), όπου όλα τα αντικείμενα θα πρέπει να είναι εξοπλισμένα με συσκευές εντοπισμού ή αναγνωριστικές μηχανές οι οποίες θα διευκολύνουν την καθημερινή μας ζωή, καταφέραμε να σχεδιάσουμε ένα περιβάλλον όπου τα διαφορετικά αντικείμενα θα μπορούν να μοιραστούν τις εμπειρίες τους . Σύμφωνα με αυτή τη θεωρία (θεωρία IoT) και την αξιοποίηση των υφιστάμενων οντολογιών το ερώτημα σχετικά με το πώς μπορούμε να αναπτύξουμε μια εφαρμογή, που βασίζεται σε συγκεκριμένη οντολογία και θα μπορεί να μοιράσει τις εμπειρίες των αντικείμενων και με την προοπτική της μελλοντικής ανάπτυξης στο πλαίσιο μίας Έξυπνης Πόλης (Smart City), έγινε πιο κατανοητή και μας βοήθησε να αναπτύξουμε ένα αντιπροσωπευτικό παράδειγμα μίας Έξυπνης Πόλης (Smart City).

1. Introduction

In this chapter we describe the meaning of Internet of Things and some related technologies which were the base for this development.

Internet of Things

The Internet of Things (IoT) use the logic of an automatic data transfer where people, objects and even animals by the usage of unique identifiers as a result do not have relationships such as human-to-human or human-to-computer. The IoT developed by combing different technologies such as wireless technologies, micro-electromechanical systems (MEMS) [1] and of course the Internet.

In the Internet of Things (IoT) different entities can act like "things". Some examples which will helps in the future present below:

The usage of Internet of Things in your body:

The company Vitality found the GlowCaps [2] which help people to place caps in the prescription bottles which through a wireless chip and with the appropriate message reminds in patients to take their prescription.

The usage of Internet of Things in your home:

The application Nest [3] which use sensors, a real time weather forecast and the daily activity in your home. The result is the reduction of monthly energy consumption up to 30%.

The usage of Internet of Things in your city:

The Smart Belly [4] trash products collect real time data and inform municipal services when they must empty the bin.

An important role in the development of the Internet of Things constitutes the IPv6. Steve Leibson says, that we could "assign an IPV6 address to every atom on the surface of the earth, and still have enough addresses left to do another 100+ earths." In other words, humans could easily assign an IP address to every "thing" on the planet. An increase in the number of smart nodes, as well as the amount of upstream data the nodes generate, is expected to raise new concerns about data privacy, data sovereignty and security.

The idea of Internet of Things existed since 1980 in the Carnegie Melon University where the programmers could connect though the internet in a Coke machine and check if a cold drink wait them.

Kevin Ashton, cofounder and executive director of the Auto-ID Center at MIT, first mentioned the Internet of Things in a presentation he made to Procter & Gamble. The below is how Ashton explains the potential of the Internet of Things:

"Today the internet and the computers are totally depends on humans. The most of data which is available in the Internet were first created by humans. The problem to all of these is that people don't have time to take data and refresh them in the real world. The ideal solutions will be to have computers which will know everything about things and they will not want any help from us. This will help us to reduce the cost and we will have information when things need repair or replace or something else."

The image below describes the Internet of Things showing the end users and applications areas based on data.

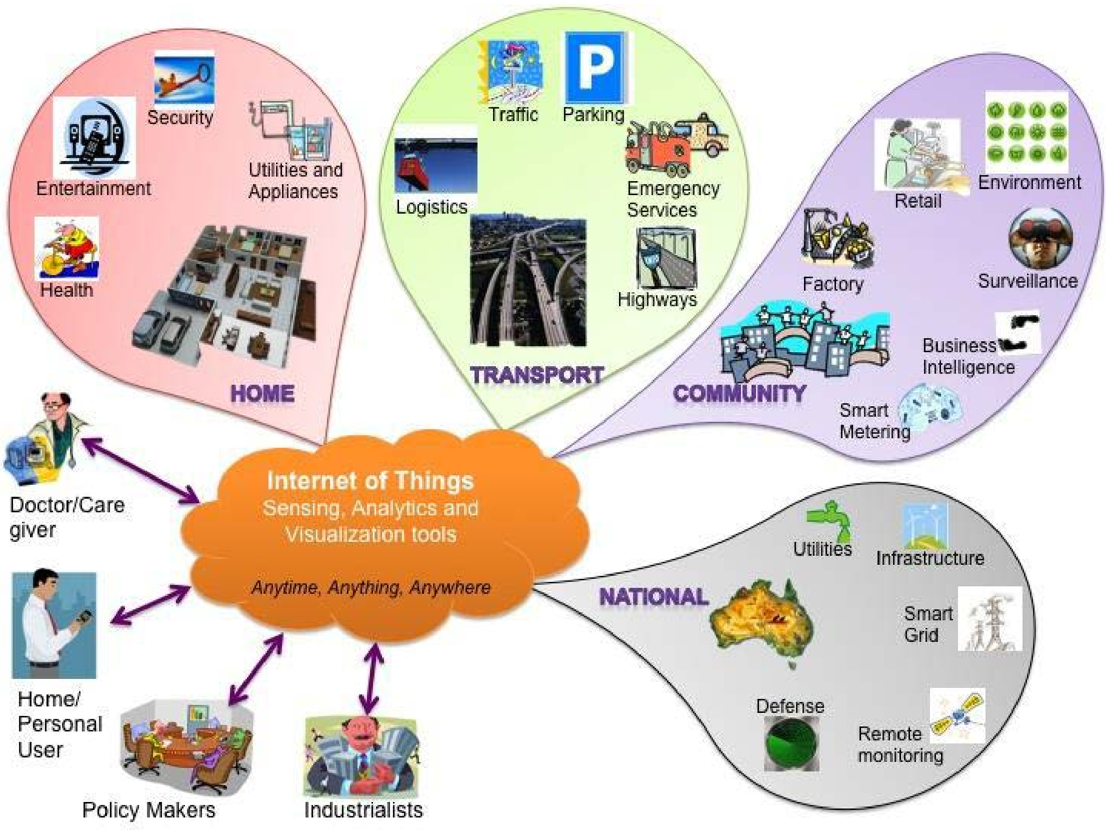


Figure 1- Representative example of IoT usage [5]

1.1 Thesis Structure

Chapter 1 introduces aspects related to IoT technologies, what is the architecture and what are the prospects. In the second chapter we will analyze the problem statement and some solutions, in the third chapter we will refer to the state of the art which already exists. In the fourth chapter will make an approach about the ontology which will be used to solve this problem, in the fifth chapter there will be analysis about the ontology which used and what exactly do this code, and in the last chapter will have the conclusion and some future work about this application.

1.2 IoT-related Technologies

1.2.1 Architecture of IoT

Architecture followed at the initial stages of IoT research will have a severe bearing on the field itself and needs to be investigated. Most of the works relating to IoT architecture have been from the wireless sensor networks perspective .European Union projects of SENSEI [6] and Internet of Things-Architecture (IoT-A) have been addressing the challenges particularly from the WSN perspective [7] and have been very successful in defining the architecture for different applications. This means that an architecture to overall IoT where the user is at the center and will enable the use of data and infrastructure to develop new applications. An architecture based on cloud computing at the center has been proposed. But it is not the best option for every application domain particularly for defense where human intelligence is relied upon but cloud centric architecture is the best where based services cost are required. However other architectures should be investigated for different application domains.

1.2.2 IoT Elements

In this section we will present some components that required for Internet of Things from a high level perspective. There are three IoT components: a) Hardware - made up of sensors, actuators and embedded communication hardware b) Middleware - on demand storage and computing tools for data analytics and c) Presentation - novel easy to understand visualization and interpretation tools which can be widely accessed on different platforms and which can be designed for different applications. Subsequently we describe a few enabling technologies in these categories which will make up the three components stated above.

1.2.3 Radio Frequency Identification (RFID)

RFID technology [8] is a very important discovery in the embedded communication prototype which enables design of microchips for wireless data communication. They help in automatic identification of anything they are attached to acting as an electronic barcode. The passive RFID tags are not battery powered and they use the power of the reader‘s interrogation signal to communicate the ID to the RFID reader. This has resulted in many applications particularly in retail and supply chain management. The applications can be found in transportation (replacement of tickets, registration stickers) and access control applications as well. The passive tags are currently being used in many bank cards and road toll tags which are among the first global deployments. Active RFID readers have their own battery supply and can instantiate the communication. Of the several applications, the main application of active RFID tags is in port containers for monitoring cargo.

1.2.4 Wireless Sensor Networks (WSN)

Recent technological advances in low power integrated circuits and wireless communications have made available efficient, low cost, low power miniature devices for use in remote sensing applications. The combination of these factors has improved the viability of utilizing a sensor network consisting of a large number of intelligent sensors, enabling the collection, processing, analysis and dissemination of valuable information, gathered in a variety of environments. Active RFID is nearly the same as the lower end WSN nodes with limited processing capability and storage. The scientific challenges that must be overcome in order to realize the enormous potential of WSNs are substantial and multidisciplinary in nature. Sensor data are shared among sensor nodes and sent to a distributed or centralized system for analytics. The components that make up the WSN monitoring network include:

a) WSN hardware - Typically a node (WSN core hardware) contains sensor interfaces, processing units, transceiver units and power supply. Almost always, they comprise of multiple A/D converters for sensor interfacing and more modern sensor nodes have the ability to communicate using one frequency band making them more versatile.

b) WSN communication stack - The nodes are expected to be deployed in an adhoc manner for most applications. Designing an appropriate topology, routing and MAC layer is critical for scalability and longevity of the deployed network. Nodes in a WSN need to communicate among themselves to transmit data in single or multi-hop to a base station. Node drop outs, and consequent degraded network lifetimes, are frequent. The communication stack at the sink node should be able to interact with the outside world through the Internet to act as a gateway to the WSN subnet and the Internet.

c) Middleware - A mechanism to combine cyber infrastructure with a Service Oriented Architecture (SOA) [9] and sensor networks to provide access to heterogeneous sensor resources in a deployment independent manner. This is based on the idea to isolate resources that can be used by several applications. A platform independent middleware for developing sensor applications is required, such as an Open Sensor Web Architecture (OSWA) [10]. OSWA is built upon a uniform set of operations and standard data representations as defined in the Sensor Web Enablement Method (SWE) [11] by the Open Geospatial Consortium (OGC) [12].

d) Secure Data aggregation - An efficient and secure data aggregation method is required for extending the lifetime of the network as well as ensuring reliable data collected from sensors. Node failure being a common characteristic of WSNs, the network topology should have the capability to heal itself. Ensuring security is critical as the system is automatically linked to actuators and protecting the systems from intruders becomes very important.

1.2.5 Addressing schemes

The ability to uniquely identify Things is critical for the success of IoT. This will not only allow us to uniquely identify billions of devices but also to control remote devices through the Internet. The few most critical features of creating a unique address are: uniqueness, reliability, persistence and scalability. Every element that is already

connected and those that are going to be connected, must be identified by their unique identification, location and functionalities. The current IPv4 may support to an extent where a group of cohabiting sensor devices can be identified geographically, but not individually. The Internet Mobility attributes in the IPV6 may alleviate some of the device identification problems; however, the heterogeneous nature of wireless nodes, variable data types, concurrent operations and confluence of data from devices exacerbates the problem further. Persistent network functioning to channel the data traffic ubiquitously and relentlessly is another aspect of IoT. Although, the TCP/IP takes care of this mechanism by routing in a more reliable and efficient way, from source to destination, the IoT faces a bottleneck at the interface between the gateway and wireless sensor devices. Furthermore, the scalability of the device address of the existing network must be sustainable. The addition of networks and devices must not hamper the performance of the network, the functioning of the devices, the reliability of the data over the network or the effective use of the devices from the user interface. To address these issues, the Uniform Resource Name (URN) [13] system is considered fundamental for the development of IoT. URN creates replicas of the resources that can be accessed through the URL. With large amounts of spatial data being gathered, it is often quite important to take advantage of the benefits of metadata for transferring the information from a database to the user via the Internet. IPv6 also gives a very good option to access the resources uniquely and remotely. Another critical development in addressing is the development of a light weight IPv6 that will enable addressing home appliances uniquely.
Wireless sensor networks (considering them as building blocks of IoT), which run on a different stack compared to the Internet, cannot possess IPv6 stack to address individually and hence a subnet with a gateway having a URN will be required. With this in mind, we then need a layer for addressing sensor devices by the relevant gateway. At the subnet level, the URN for the sensor devices could be the unique IDs rather than human-friendly names as in the www, and a lookup table at the gateway to address this device. Further, at the node level each sensor will have a URN (as numbers) for sensors to be addressed by the gateway. The entire network now forms a web of connectivity from users (high-level) to sensors (low-level) that is addressable (through URN), accessible (through URL) and controllable (through URC).

1.2.6 Data storage and analytics

One of the most important outcomes of this emerging field is the creation of an unprecedented amount of data. Storage, ownership and expiry of the data become critical issues. The internet consumes up to 5% of the total energy generated today and with these types of demands, it is sure to go up even further. Hence data centers which run on harvested energy and which are centralized will ensure energy efficiency as well as reliability. The data have to be stored and used intelligently for smart monitoring and actuation. It is important to develop artificial intelligence algorithms which could be centralized or distributed based on the need. Novel fusion algorithms need to be developed to make sense of the data collected. State-of-the-art non-linear, temporal machine learning methods based on evolutionary algorithms, genetic algorithms, neural networks, and other artificial intelligence techniques are necessary to achieve automated decision making. These systems show characteristics such as interoperability, integration and adaptive communications. They also have a modular architecture both in terms of hardware system design as well as software development and are usually very well-suited for IoT applications.

1.2.7 Visualization

Visualization is critical for an IoT application as this allows interaction of the user with the environment. With recent advances in touch screen technologies, use of smart tablets and phones has become very intuitive. For a lay person to fully benefit from the IoT revolution, attractive and easy to understand visualization have to be created. As we move from 2D to 3D screens, more information can be provided to the user in meaningful ways for consumers. This will also enable policy makers to convert data into knowledge which is critical in fast decision making. Extraction of meaningful information from raw data is non-trivial. This encompasses both event detection and visualization of the associated raw and modelled data, with information represented according to the needs of the end-user.

1.3 Application Domains

There are several application domains which will be impacted by the emerging Internet of Things. The applications can be classified based on the type of network availability, coverage, scale, heterogeneity, repeatability, user involvement and impact. We categorize the applications into four application domains: (1) Personal and Home (2) Enterprise (3) Utilities and (4) Mobile. This is depicted in Figure 1 which represents Personal and Home IoT at the scale of an individual or home, Enterprise IoT at the scale of a community, Utility IoT at a national or regional scale and Mobile IoT which is usually spread across other domains mainly due to the nature of connectivity and scale. There is a huge crossover in applications and the use of data between the domains. For instance, the Personal and Home IoT produces electricity usage data in the house and makes it available to the electricity (utility) company which can in turn optimizes the supply and demand in the Utility IoT. Internet enables sharing of data between different service providers in a seamless manner creating multiple business opportunities. A few typical applications in each domain are given.

1.3.1 Personal and Home

The sensor information collected is used only by the individuals who directly own the network. Usually Wi-Fi is used as the backbone enabling higher bandwidth data (video) transfer as well as higher sampling rates (Sound). Ubiquitous healthcare has been envisioned for the past two decades. IoT gives a perfect platform to realize this vision using body area sensors and IoT backend to upload the data to servers. For instance, a Smartphone can be used for communication along with several interfaces like Bluetooth for interfacing sensors measuring physiological parameters. So far, there are several applications available for Apple iOS, Google Android and Windows

Phone operating system that measure various parameters. However, it is yet to be centralized in the cloud for general physicians to access the same. An extension of the personal body area network is creating a home monitoring system for aged-care, which allows the doctor to monitor patients and elderly in their homes thereby reducing hospitalization costs through early intervention and treatment. Control of home equipment such as air conditioners, refrigerators, washing machines etc., will allow better home and energy management. This will see consumers become involved in IoT revolution in the same manner as the Internet revolution itself. Social networking is set to undergo another transformation with billions of interconnected objects. An interesting development will be using a Twitter like concept where individual Things in the house can periodically tweet the readings which can be easily followed from anywhere creating a *TweetOT*. Although this provides a common framework using cloud for information access, a new security paradigm will be required for this to be fully realized.

1.3.2 Enterprise

We refer to the Network of Things within a work environment as an enterprise based application. Information collected from such networks are used only by the owners and the data may be released selectively. Environmental monitoring is the first common application which is implemented to keep a track of the number of occupants and manage the utilities within the building (e.g., HVAC, lighting). Sensors have always been an integral part of factory setup for security, automation, climate control, etc. This will eventually be replaced by wireless system giving the flexibility to make changes to the setup whenever required. This is nothing but an IoT subnet dedicated to factory maintenance. One of the major IoT application areas which is already drawing attention is Smart Environment IoT. There are several testbeds being implemented and many more planned in the coming years. Smart environment includes subsystems as shown in Table 1 and the characteristics from a technological perspective are listed briefly. It should be noted that each of the sub domains cover many focus groups and the data will be shared. These applications are grouped according to their impact areas. This includes the effect on citizens considering health and wellbeing issues; transport in light of its impact on mobility, productivity, pollution; and services in terms of critical community services managed and provided by local government to city inhabitants.

1.3.3 Utilities

The information from the networks in this application domain are usually for service optimization rather than consumer consumption. It is already being used by utility companies (smart meter by electricity supply companies) for resource management in order to optimize cost *vs.* profit. These are made up of very extensive networks (usually laid out by large organization on regional and national scale) for monitoring critical utilities and efficient resource management. The backbone network used can vary between cellular, Wi-Fi and satellite communication. Smart grid and smart metering is another potential IoT application which is being implemented around the world. Efficient energy consumption can be achieved by continuously monitoring every electricity point within a house and using this information to modify the way electricity is consumed. This information at the city scale is used for maintaining the load balance within the grid ensuring high quality of service. Video based IoT which integrates image processing, computer vision and networking frameworks will help develop a new challenging scientific research area at the intersection of video, infrared, microphone and network technologies. Surveillance, the most widely used camera network applications, helps track targets, identify suspicious activities, detect left luggage and monitor unauthorized access. Automatic behavior analysis and event detection (as part of sophisticated video analytics) is in its infancy and breakthroughs are expected in the next decade as pointed out in the 2011 Gartner Chart Water network monitoring and quality assurance of drinking water is another critical application that is being addressed using IoT. Sensors measuring critical water parameters are installed at important locations in order to ensure high supply quality. This avoids accidental contamination among storm water drains, drinking water and sewage disposal. The same network can be extended to monitor irrigation in agricultural land. The network is also extended for monitoring soil parameters which allows informed decision making about agriculture.

1.3.4 Mobile

Smart transportation and smart logistics are placed in a separate domain due to the nature of data sharing and backbone implementation required. Urban traffic is the main contributor to traffic noise pollution and a major contributor to urban air quality degradation and greenhouse gas emissions. Traffic congestion directly imposes significant costs on economic and social activities in most cities. Supply chain efficiencies and productivity, including just-in-time operations, are severely impacted by this congestion causing freight delays and delivery schedule failures. Dynamic traffic information will affect freight movement, allow better planning and improved scheduling. The transport IoT will enable the use of large scale WSNs for online monitoring of travel times, origin-destination (O-D) route choice behavior, queue lengths and air pollutant and noise emissions. The IoT is likely to replace the traffic information provided by the existing sensor networks of inductive loop vehicle detectors employed at the intersections of existing traffic control systems. They will also underpin the development of scenario-based models for planning and design of mitigation and alleviation plans, as well as improved algorithms for urban traffic control, including multi-objective control systems. Combined with information gathered from the urban traffic control system, valid and relevant information on traffic conditions can be presented to travelers. The prevalence of Bluetooth technology (BT) devices reflects the current IoT penetration in a number of digital products such as mobile phones, car hands-free sets, navigation systems, etc. BT devices emit signals with a unique Media Access Identification (MAC-ID) number that can be read by BT sensors within the coverage area. Readers placed at different locations can be used to identify the movement of the devices. Complemented by other data sources such as traffic signals, or bus GPS, research problems that can be addressed include vehicle travel time on motorway and arterial streets, dynamic (time dependent) O-D matrices on the network, identification of critical intersections, and accurate and reliable real time transport network state information. There are many privacy concerns by such usages and digital forgetting is an emerging domain of research in IoT where privacy is a concern. Another important application in mobile IoT domain is efficient logistics management. This includes monitoring the items being transported as well as efficient transportation planning. The monitoring of items is carried out more locally, say, within a truck replicating enterprise domain but transport planning is carried out using a large scale IoT network.

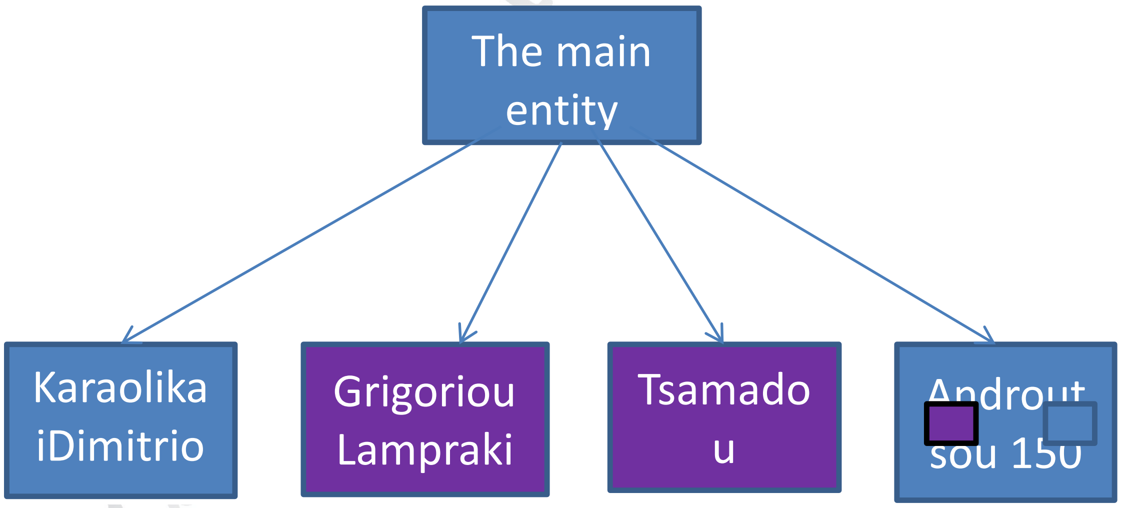
2. Problem Statement

The following chapter will approach the idea of what we want to accomplish overcoming already existing problems.

Until 2020 there will be nearly 26 billion devices on the IoT and also 30 billion devices will be wirelessly connected to the IoT. Today there are over 22 billion of mobile terminals which used and over 1 billion users of the Internet worldwide. We understand that the amount of things is huge.

This is the first problem we must face. The very big numbers of things make our life difficult because it isn't easy to manage all of them with effectiveness and with the best way. This happens because everything has different characteristics, software, data, services etc. So it is difficult to communicate and exchange experience between them especially when they don't have something in common. For example, think that we have two people from different countries and nobody knows the other language or a common language. So the result is that they can communicate and share experiences something that causes problems. The same happens with the objects because we don't have the results we want, the management isn't fast and causes a lot of problems as it isn't easy to decide with which technique you can succeed the best result and the better communication where a lot of objects can participate so we end up in a situation which covers our needs and it will be successful.

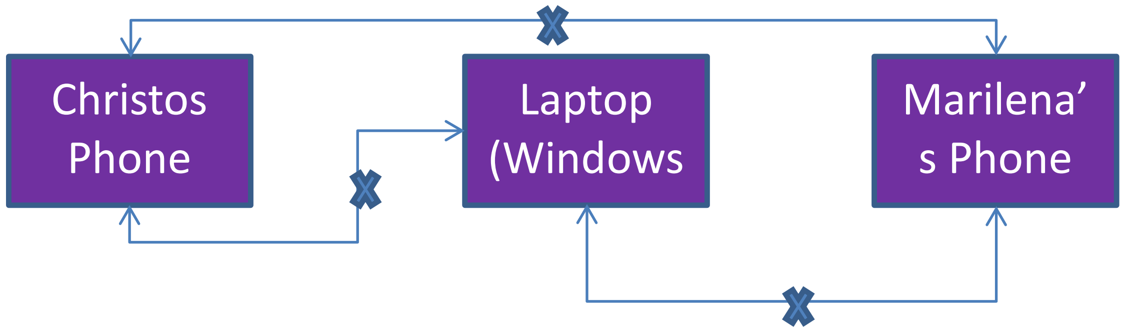
The second problem we have to deal with it, is the location bounded.



The example above can describe our problem. We have a basic entity which can communicate with others entities. Every entity of them is in a different place and has different characteristics. The question is how these entities can communicate between them and share experiences. The answer is that they can't communicate effectively for many reasons.

- ✓ Basic entity can't communicate and transmit information in the same time because this will create collision in the system and it will cause problems. The basic entity is the entity which all users trust and derive information with the purpose of a better operation. This entity give "instructions" to each smaller entity with information that need so that to get the experience which is necessary to operate better.
- ✓ The entities can't communicate between them and exchange their experiences. This happens because there is no trust between them. For example the laboratories which find in Tsamadou and the laboratories which find in Androutsou 150 can't communicate and exchange information about the students and the tasks that happen because the trust between them is limited and they depend on the basic entity. So this causes problems because we can't avoid the communication with the basic entity.
- ✓ The thing that only one entity share information to other entities is not fast and also we don't ensure security to our system. All entities wait to get information from only one entity as a result to have collisions. Furthermore there is a waiting queue because every entity waits until the basic entity unoccupied. This makes our system slow without effectiveness. The security of the system is poor because if the basic entity breaks down, the system can't work because we don't have self-management in the entities. The same happens when someone hacks our system. All entities depend on one entity so if we have "incursion" all the system will be influenced as a result the no proper functioning.
- ✓ To achieve all the above requires money. So if the system fails and causes a lot of problems of communication between the entities and not provides the results we want it is not a lucrative system. This means that we don't have any profit and all the system failed. So we must find clever solutions to face this problem and all the above.

The third problem that we called to face is that the things owned-operated by different entities (people-organizations). In other words every thing or a group of things belongs to a different entity or organizations or they have different operated system. All of these cause problems because there is no trust between different organizations and also the different software can't share experience between them as they don't have the same communication code and the same possibilities.



To solve all the above we have the ontologies and the distributed coordination.

Ontologies enrich the information and do a better management.

An ontology is defined as “a formal, explicit specification of a shared conceptualization" and is used to represent knowledge within a domain as a set of concepts related to each other. There are four main components that compose an ontology: Classes, relations, attributes and individuals. Classes are the main concepts to describe. Each class can have one or several children, known as subclasses, used to define more specific concepts. Classes and subclasses have attributes that represent their properties and characteristics. Individuals are instances of classes or their properties. Finally, relations are the edges that connect all the presented components.

If all entities use the same ontology we can have better results because it will be easier to share experiences between things and also the management of the millions of things will be easier. Whatever software and experience have the things we have the opportunity to share experiences and communicate as a result to create a dynamic system which will act autonomous and it will not influenced by external factors.

With distributed coordination we achieve a better management through the things. In other words as the communication systems growing in size and complexity created the need to develop systems which will be autonomous. This means that everything will receive the necessary information through different shared experiences and when it completes the necessary number of information that it needs then it will act autonomous. This will help to have better management and we will come through the collision in the communication systems. With this plan repealed the meaning of one master entity which provides all the information to the sub entities. Every entity acts all alone at the time where it will get all the necessary information. Also all entities can communicate and share their own experience and information that they have. With that way we get a system faster and cheaper because every entity act autonomous and it not wait the main when it will be available so as to draw all the information that it need. Also this cost nothing as we use the existing system without make changes and interferes in the software.

In addition, this idea offers us trust and security as it is not easily accessible of malware software and we don’t have one main entity which determines the operation of the system. So if an entity breaks down will not influence the whole system and all the rest works normal. Also it is not easy to interfere in that system with malware software because you have to face a lot of different software. This cost a lot and it takes a lot of time as you must find in what malware software every entity is “sensitive”.

So the conclusion is that we have a system autonomous, reliable and with high quality security which offers us the best results and better management through the million of things.

3. State of the Art

In this chapter we shortly present related technologies with respect to ontologies in the IoT domain.

3.1. Ontologies

3.1.1. Sensor Web Autonomy (SWAMO Ontology)

The SWAMO [14] ontology follows the structure of SensorML and its major decomposition of concepts. In SensorML, all concepts are considered to be processes. Processes may be viewed as physical or logical, and may be aggregated or atomic.

In SensorML, the view of a sensing system is the idea of a Process concept, which is a superclass with four major classes: System, Component, Process_Chain, and Process_Model. The System and Component classes represent composite and atomic physical devices, respectively. The Process_Chain and Process_Model classes represent composite and atomic logical processes, respectively.

A satellite platform is a characteristic example of a composite physical system. It may be composed of several subsystems, including several Earth sensors, a power subsystem, a communications bus, and a computing system. Each of these subsystems may be further decomposed into some set of atomic components. The granularity of decomposition is generally dependent upon the degree of control required for the control system on the platform.

The same, with the logical processes, process chains are composed of one or more subprocess chains or process models. These subprocesses represent logical procedures that must be performed on a physical device. Their degree of decomposition depends upon the required detail of sensing system planning and the sensing system service descriptions.

The SWAMO ontology starts with a description of the Process Class. Referring to the Process class description has four slots specified: inputs, outputs, parameters, and metadata. The inputs, outputs, and parameters slots are similar to each other. All are the type of Class; however, each is constrained to class instances of a different type. The value for the inputs slot must be an instance of type Input, the value for the outputs slot must be an instance of type Output, and the value for the parameters slot must be an instance of type Parameter. The cardinality is one-to-many since a valid process description must have at least one input and one output, but may have multiple inputs and outputs. The cardinality of parameters is zero-to-many since parameters are optional. There are no other constraints on these slots. The metadata slot is of type Class and is constrained to an instance of type Metadata. Since SensorML metadata is optional, its cardinality is zero-to-one.

The Input, Output, and Parameter classes represent three distinct concepts, but their information structures are almost the same. An object-oriented implementation of these concepts may subclass them from a common superclass. These three concepts all have an identifying name slot, a definition slot, and a unit_of_measure slot.

The name and definition slots are constrained to type String, are required, and are individually. These slots are not defined in further detail to allow for implementation-specific descriptions. The unit_of_measure slot is of type Class, is constrained to be an instance of type UCUM [15] (The UCUM describes a standard way to specify and covert units of measure in a wide variety of systems), and has cardinality of one to one.

The Metadata class [16] describes the SensorML concept of extra information made available for implementation-specific uses. Such uses may include semantic searching for processes that produce results with specific characteristics or specifying requirements of a process for matching actors with processes. The Metadata class has two slots, characteristics and capabilities, which are optional, but if provided it may only have a single value of type String. They avail as the placeholders for the SensorML specification of the information associated with processes and systems.

The UCUM class represents a Unified Code for Units of Measure description. In the UCUM there are included two slots. The dimensions slot identifies the number of fields in the base unit description. For example, the description of one Tesla is —Wb/m2‖, which has a dimension of three. The first base value is webers (Wb), the second is divided by (/), and the third is square meters (m2). It is also important to distinguish the upper and lower case versions of a unit of measure symbol; for example, —t‖ represents mass in metric tonnes while —T‖ is a metric unit of measure for the strength of a magnetic field in teslas. This information is necessary for parsing units of measure, so the dimensions slot value is required and only one may be provided.

Finally, the base slot captures the description of the unit of measure. Continuing with the example above, —Wb/m2‖ is the base value for the teslas position, commonly stated as —webers per square meter‖. The base slot is of type String and its cardinality if one to one. With the values provided in these slots, a unique unit of measure can be extracted and used for conversions, comparisons, and other sensor web decision making. The remaining sections will describe in detail the other parts of the SWAMO ontology. These are the Platform ontology, which represents physical systems; the Model ontology, which represents logical systems; the Agent ontology, which represents the reasoning and control aspects of SWAMO; and the SOS ontology [17], which represents the observations available from the sensor web. For SWAMO, there is actually only one ontology.

3.1.2. CESN: An Ontology of Sensors and their Measurements

The CESN [18] sensor ontology describes the relationships between sensors and their measurements. The main concepts that found in the CESN sensor ontology are same with the concepts that described in SensorML (2007) and in the Marine Metadata Interoperability device ontology project (MMI, 2009)[19], and CSIRO [20]Sensor Ontology (CSIRO, 2009). These and several others are under “control” by the W3C Semantic Sensor Incubation Group with the active participation of their developers, and the entire subject of semantically enabled sensors is in its infancy. In the figure below, the core concepts in the CESN sensor ontology are the physical sensor devices themselves, *Sensor;* the *PhysicalProperty* that a *Sensor* can measure; and the measurement that a sensor has taken, *PhysicalPropertyMeasurement*. In other words, a sensor object can measure only one physical property. A class named *Instrument*, model objects that can contain sensors and measure more than one physical property. In turn, an instrument is usually deployed on some kind of *Platform*, which typically constrains its relationship to the environment in which it is deployed. Also not shown is the class *Deployment,* which represents the deployment of an instrument at a particular time and place, and so can be used to relate instrument readings to expected or unexpected events signaled by the data modeled by the structure sketched in figure below.

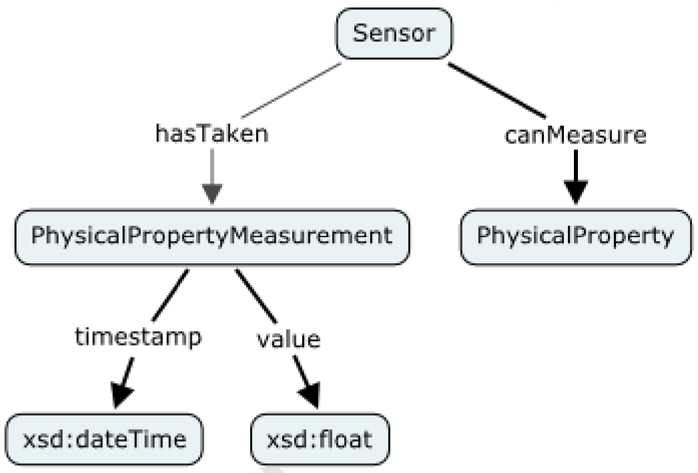


Figure 2-CESN Sensor Class [21]

3.1.3. Extensible Observation Ontology (OBOE)

The aim of the Extensible Observation Ontology (OBOE) [22] is to operate as a formal and generic conceptual framework for describing the semantics of observational data sets (ex. datasets consisting of observations and measurements). Extensible Observation Ontology (OBOE) also describes a structured approach for organizing domain specific ontologies through the use of “extension points. ”OBOE extension points allow ontology classes, properties, and constraints to be easily defined for a particular domain specific terminology, and existing domain extensions to be interrelated. Therefore, OBOE can serve as a framework for defining new domain ontologies as well as interoperating and relating existing ones. Figure 1graphically describes the basic core structure of OBOE, which consists of five classes named Observation, Entity, Measurement, Characteristic, and Measurement Standard, and six properties named has Context, of Entity, has Measurement, has Value, has Precision, uses Standard, and of Characteristic. In additional properties may be added to the Measurement class to capture when and where measurements were recorded, who recorded each measurement, the protocols of measurements, and so on. Similar properties can also be added to the Observation class. Today, most details of observational data are not recorded. Instead, the physical representation of data is often optimized for data collection; for use within a specific tool, or for a particular analysis, e.g., to perform a calculation requiring a site-by-species matrix. As a consequence, contextual information concerning data is typically implicit, where context is (possibly) encoded by attribute labels, implied by the proximity of attributes (i.e., neighboring data), stored in metadata as natural-language descriptions, or altogether missing. Consider the first data table in Figure 1. The column of data labeled “Ht” can be assumed to represent height, giving information about the characteristic of some entity that was measured. However, no explicit information is given about the entity itself. The neighboring column labeled “Sp” suggests that each height measurement was for a species given in the column. Further, if all such species values in the column correspond to types of reef coral, one could surmise the kind of entities to which the height measurements pertain. The goal of OBOE is to provide a generic model for making such information explicit, which can then enable automated approaches for merging data (e.g., in this case with other coral data) and data discovery (e.g., for researchers looking for data on coral heights).The rest of this section details the core OBOE classes and properties shown in Figure 1, and demonstrates the use of OBOE for capturing observation data and for extending OBOE with domain ontologies. Although not discussed here, OBOE is encoded using the OWL-DL ontology language, which gives a description-logic formalization of the various parts of the OBOE ontology described below.

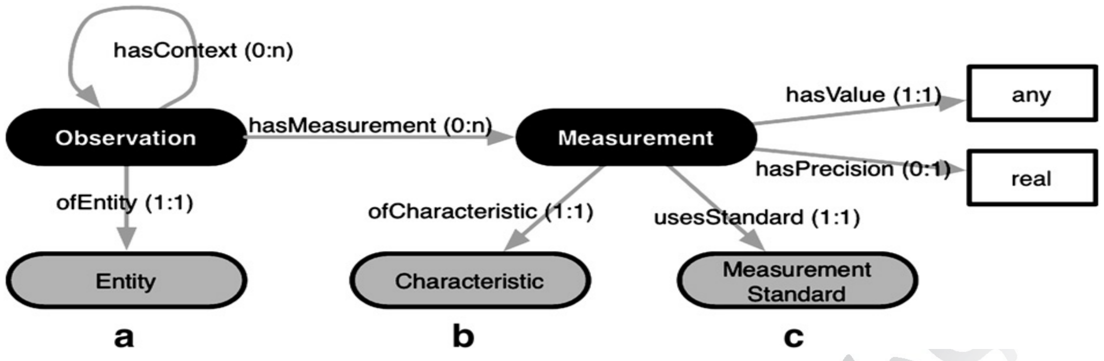


Figure 3-Properties of Extensible Observation Ontology (OBOE) [23]

Each Observation is of some Entity, and can provide context for the Observation of another Entity. A Characteristic of an Entity can be represented through a Measurement. Measurements relate Characteristics to a Measurement Standard via a Value and, if applicable, a Precision. Measurements are taken by a Recorder (human or non-human) using a Protocol at a particular Time and Place (shaded properties, see text for details). Observations may have multiple Measurements. Entity (a), Characteristic (b), and Measurement Standard (c) (shaded classes) provide extension points for domain-specific ontologies. Numbers in parentheses denoted min: max cardinality for properties.

3.1.4 Device Ontology

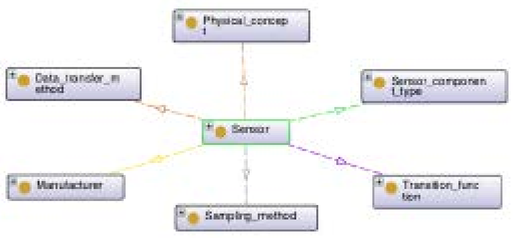


Figure 4-Sensor and Device Ontology Classes [24]

The Device Ontology [25] is accessed to identify what things should be looked up to satisfy an application's requirements. We consider that applications built on top of an IoT middleware should be network and device agnostic. Therefore, it becomes the task of the middleware to identify what devices to seek in order to provide needed services. For this purpose, the ontology should clearly describe and yet not over-specify device metadata. Our main contribution is the high-level abstraction for device metadata, especially regarding the internal components of devices. Internal components are the electronic chips and hardware parts, built inside the device, that together define its technical functionalities. Hence, looking at each of them separately as independent entities is not informative, as their functionalities are tightly related to one another's. That being said, understanding the characteristics of a singular chip requires an understanding of the whole device's internal schema, which grows to be too complex to include. We can further argue that they can just be considered as a black box, especially that those components are not directly accessible by applications. However, we chose to allow users to describe the internal components of devices by their name and type only. Another main contribution is that our ontology holds knowledge that is independent of device deployments, e.g., information related to the device's actual location. Instead, deployment information is presented in the metadata, reported by devices, during the discovery registration process. The ontology becomes thus easily pluggable with any middleware or application. To elaborate on the ontology, we consider that IoT devices can be divided in four main classes:

- ✓ Sensor: A device that has the capability to measure a physical property of the real world.
- ✓ Actuator: A device that has the capability to perform an operation on or control a system/physical entity in the real world.
- ✓ Processor: A device that has the capability to perform computation operations on data.
- ✓ Composite: A device that consists of at least 2 of the devices above.

In the following, we focus on modeling sensors, as they are representative of things and models of other devices adhere to the same conceptualization approach. Based on the current literature, we have identified several ontology concepts that are commonly used to model sensors (sensorML3). As shown in Figure 2, those concepts are:

- ✓ Manufacturer: The manufacturer of the sensor.
- ✓ Sensor component type: The sensor internal hardware components.
- ✓ Physical concept: The real world property measured by the sensor (e.g., temperature, wind speed, etc.).

This concept is the main link between the Device Ontology and the Physics Ontology.

- ✓ Sampling method: The way the sensor is triggered to sample its environment (e.g., periodic).
- ✓ Data transfer method: The way the sensor is triggered to transfer its readings (e.g., push).
- ✓ Transition function: The process used to convert the input phenomenon to a digital value.

We chose the entities above as we consider that a sensor can be properly identified given any of their respective values. With the exception of the last entity, which we introduce as it clarifies what and how phenomena or values are output by a sensor after a measurement is performed. This is needed so that a sensor's outputs can be meaningful to and usable by other applications.

3.1.5 Physics Domain Ontology

The Physics Domain Ontology [26] is created with two main goals. The first is to model real world entities as physical concepts so that any IoT middleware can extract knowledge about the real world, as this is a common task to be performed within the Internet of Things. The second is to model mathematical formulas and functions as they are the first alternative to be utilized when no device can provide needed services. The main classes of the Domain Ontology are:

- ✓ Physical concept: A real world object or property that can be measured.
- ✓ Physical unit: The output unit of the real world property measurement.
- ✓ Mathematical data type: The set of numbers that can represent a real world property measurement.
- ✓ Formula: Mathematical expression that computes a numerical value representing a real world property.
- ✓ Function: Formulas are implemented by functions that define the required input and output machine data types.

Our main contribution in this ontology is that we model and establish a direct relation between physical concepts, mathematical formulas and functions. We argue that this relation is essential as it allows services to be provided as mathematical computations over physical concepts. This process can be used by any middleware to substitute services of unavailable devices with alternative services that can be deployed on any other appropriate device, which is a very familiar scenario within the highly dynamic IoT. This relation further allows our ontology to be useful in any context requiring mathematical and physical knowledge by clearly modeling formulas that can compute mathematical values as measurements over a physical property.

However, a same physical concept can have several formulas that vary based on the units of measurement of input/output parameters. Hence, our second contribution is to introduce two, not previously described, first class entities: unit constraints and conversion formulas. The former allows users to specify if a formula can have only one output and one input unit per concept, or can have a defined set of such units, or it stands correct for any input/output units, linked to a physical concept. For instance, the formula speed = distance=time stands correct for any distance unit over any time unit. On the other hand, a wind chill formula for temperature in Celsius and wind speed in km/h is different than that of a temperature in Fahrenheit and wind speed in mph. As for the conversion formulas class, it allows users to model conversion formulas between one measurement unit to another. By adding the constraints class and conversion functions to our ontology, we introduce a higher degree of exibility as it allows any middleware or application using our model to dynamically adapt to unit constraints. A common reference model for representing and categorizing physical concepts is the DOLCE 4 representation. It is adopted by several works such as, as it has a well-organized vocabulary. However, it does not categorize entities by their physical properties but rather by the human perception of those entities. This organization is not in line with our representation of concepts that should be both intuitive and physics oriented. In our ontology, each physical concept is linked to:

- ✓ Sensor: All sensors that can measure its value.
- ✓ Units of measurement all units by which it can be measured.
- ✓ Mathematical data type: Set the mathematical values the concept can be represented.
- ✓ Formula: All mathematical formulas that can compute its value.

Regarding formulas, authors in provide an approach to modeling physics in an ontology. However, their model is only applicable for the biological domain as they focus on mapping laws of physics to biological processes. SPACE is another ontology that models physics but it only applies in the space physics domains. In our ontology, each formula is linked to:

- ✓ Mathematical expression: Mathematical equation.
- ✓ Input parameters: Measurements of physical concepts that will be used to evaluate another physical concept and their measurement units.
- ✓ Output parameter: Computed output and its measurement unit.
- ✓ Physical concept: The physical concept being evaluated.
- ✓ Unit of measurement: It is in fact the output's unit.

We argue that those concepts are well representative of physics and mathematical models and they specify all the parameters needed to define a mathematical equation. The Sensei project models a decomposition of physical concepts in to a set of other physical concepts. This decomposition can be similar to a direct link between our formula output and input concepts. However, the relation between their concepts is not clearly specified and therefore, their decomposition cannot substitute our formula model.

3.1.6 Estimation Ontology

The Estimation Ontology [27] is, perhaps, the most unusual among the three described here. This ontology is in charge of storing the different mathematical models that make up the mental toolbox carried by expert system designers in fields such as Robotics, Estimation, Sensor Networking, etc. However, in addition to simply storing these models, the Estimation Ontology must also organize them in a way that makes them machine-accessible. After all, our middleware must be able to discern (1) which models are appropriate for a given situation, and (2) which model is, in some sense, optimal. For this reason, the Estimation Ontology must provide a well-designed set of attributes for each model, as well as an intricate web of relationships between models, devices, and physical concepts. These rather indispensable elements, and how to best express them in this ontology, are something that we are currently investigating. For now, we limit ourselves to providing below a first-level set of classes that group the different types of mathematical models:

- ✓ Estimation & Prediction Models: These models are used during the automated estimation process, as well as before the look-up phase in probabilistic discovery. An example of such a model is a Kalman filter where each component in its matrices and vectors are functions of physical concepts from the Domain Ontology (for instance, an input vector for use in target-localization could be defined as a triplet of3D-acceleration, 3D-velocity, and 3D-position).

- ✓ Association & Correlation Models: These describe the numerous conditional probability relations that are used in Estimation Theory, relating one physical phenomenon as a function of another. For instance, the probability of the value of a temperature sensor given the value of a daylight sensor. In addition, these models can also be used to solve the Association Problem that occurs, for instance, with multiple-target tracking.

- ✓ Error Models: These models describe the different ways that uncertainties can be introduced into measurements and actuations. These are usually represented in the form of a stochastic model (for instance, a simple additive Gaussian noise model).

3.1.7. SensorML

SensorML [28] is specified as a generic data model in UML for capturing classes and associations that are common to all sensors. SensorML is part of an Open Geospatial Consortium (OGC) initiative to contribute to the development of a Sensor Web “through which applications and services will be able to access sensors of all types over the Web.” Instantiations of classes and associations provided by SensorML can be used to create specific sensor profiles, which facilitate the processing, geolocation and integration of observed data from a myriad of sensors. Profiles of individual sensors that use and/or extend SensorML concepts can be created and posted in a Web environment in which they can be tasked and queried by monitoring and processing systems.

SensorML is developed as a specification for efficient implementation by vendors that desire to implement OGC compliant sensor systems; therefore, SensorML strives to specify as few class and relation definitions as possible. However, ontological engineering focuses on rich semantic data and knowledge models; therefore, re-conceptualization, redefinition, or extension of some of the classes in SensorML is necessary.

SensorML has been realized using syntax from the extensible Markup Language (XML). The use of XML syntax alone to define classes restricts the potential scope of interoperability and reuse of its sensor profile instantiations. This is because of the lack of standard semantics of the XML syntactic constructs.

In general, given n sensor profiles constructed with XML syntax, n*(n-1)/2 mappings have to be constructed to translate among the profiles. Further, the lack of semantics of XML constructs alone precludes their use for formal definitions of sensor concepts in an ontology. According to Gruber, an ontology is a set of formally defined concepts and relations that are relevant to a knowledge domain. SensorML does not include formal definitions of the classes or relations it uses, that is, it provides no logical or axiomatic-grounded theory to account for its conceptualizations and therefore cannot be considered to be an ontology. However, SensorML does provide a generic data model for expressing knowledge and data about sensors and provides a good schema for developing a persistent data store for sensor metadata and sensed attribute values. Moreover, it provides a good organizational framework within which a sensor ontology can be defined.

OWL has been adopted by the World Wide Web Consortium (W3C) [29] as a standard formalism for the Semantic Web. Each OWL construct has formal semantics. Part of the OntoSensor [30] development effort includes mapping a subset of the SensorML concepts and relations to OWL [30] and adding additional concepts.

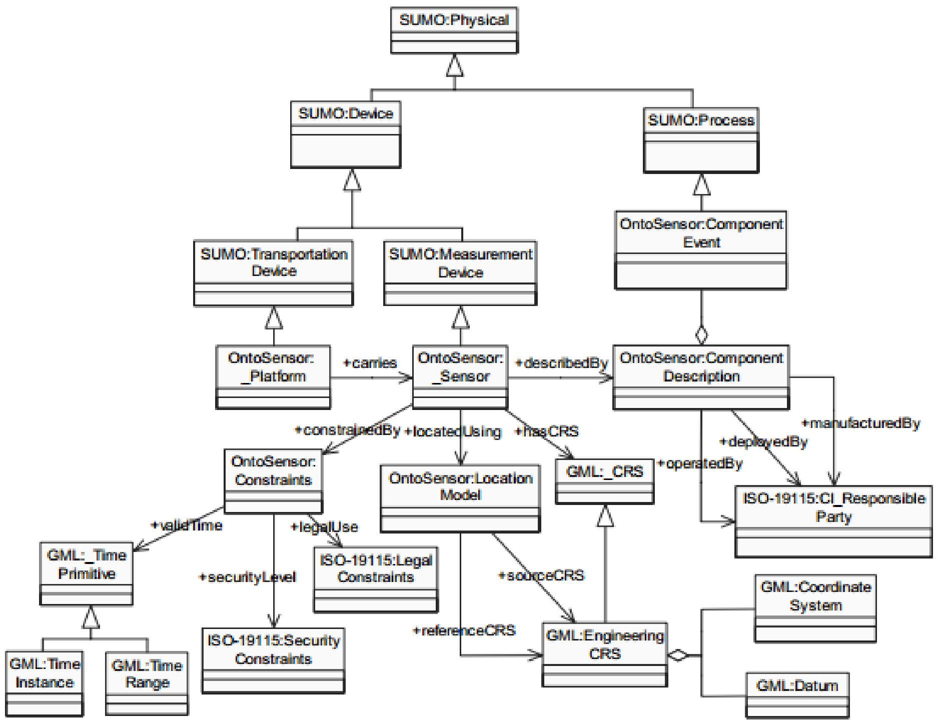


Figure 5-OntoSensor Extend [31]

As a first step towards utilizing high-level sensor ontologies for data/knowledge fusion on the Semantic Web, OntoSensor is used to build a prototype sensor repository. This repository is an instantiation of the sensor ontology and is coupled with a simple inference module implemented in Prolog to build applications. This prototype is the predecessor of more advanced systems, which must include advanced inference capabilities that can realize the potential of using high-level sensor ontologies in knowledge and data fusion processes.

To conclude, if we compare all the above ontologies the best and the most representative is the SensorML for many reasons.

- ✓ Use syntax of from the Extensible Markup Language (XML) [32].
- ✓ Developed especially for systems which use sensors.
- ✓ Can be used to create specific sensor profiles.

All the above are the reasons as we choose the SensorML to describe how acts the application. As it is the only ontology which is made to describe the characteristics of sensors.

3.2. Distributed Coordination

Nowadays the daily life require more communicating systems as a result this will create the need for new communication systems which will cover our needs and will represent our life with the best way. A peer-to-peer (P2P) network and Chubby and Zookeeper are characteristic examples of these communications systems.

A peer-to-peer (P2P) [33] network consist decentralized and distributed network architecture. This network have individuals nodes known as peers which can act as servers and as objects. In this way we avoid the “classic” model of client-server. Furthermore, Chubby [34] and ZooKeeper [35] consist another way manage the large distributed systems. In this aspect all the group participants connect to a distributed hub and coordinate their actions through it. ZooKeeper-style services were developed to support the needs of mobile users and mobile applications as a consequence of soaring growth of these. All the above will present in details in the next paragraphs.

3.2.1. Peer-To-Peer Protocols

Peer-to-peer (P2P) networks connect many end-hosts (also referred to as peers) in an ad-hoc manner. P2P networks have been typically used for file sharing applications, which enable peers to share digitized content such as documents, audio, video, electronic books, etc. Later, appeared more advanced applications such as real-time conferences, online gaming, and media streaming which use such networks. So traditional client-server networks, where servers only provide content, and clients only consume content eliminate and in P2P networks, each peer is both a client and a server.
It has been observed that P2P file sharing applications dominate Internet traffic usage. In fact, a wide range of measurements, which were performed in 8 different geographic regions during the years of 2008-2009, show that P2P networks generated most of the traffic in all monitored regions, ranging from 43% in Northern Africa to 70% in Eastern Europe. The same study also identified that BitTorrent (Cohen, 2003) is the most popular protocol through the Internet, generating most of the traffic in 7 out of 8 regions ranging from 32% in South Africa to 57% in Eastern Europe. The recent years, streaming services and media delivery over the Internet such as YouTube and Internet videobroadcasting have emerged. These services have become very popular, as they can deliver video to a very big number of receivers in the same time at any given time. The content providers offer their services by using P2P network in order to reduce infrastructure, maintenance, and service costs, and provide more reliable services. While several designs for P2P systems have been successfully deployed for file sharing and real-time media streaming, key challenges such as the design of optimal resource reciprocation strategies among self-interested peers still remain largely unaddressed. However, such assumptions may be undesirable from the perspective of a self-interested peer, which aims to maximize its own utility. As a result, efficient resource reciprocation strategies need to be deployed, which can also provide incentives to the peers for their contributions. In BitTorrent systems, incentive strategies are based on the so-called tit-for-tat (TFT) [36] strategy, where a peer selects some of its associated peers, which are currently uploading at the highest rates, and provides them its content for downloading. This simple strategy is currently implemented in BitTorrent systems, and provides good performance. On the other hand, a disadvantage of this resource reciprocation strategy is that peers decide how to determine their resource reciprocation based on only the current upload rates that it receives from its associated peers, and does not consider how this will influence their upload rates in the future. In other words, the resource reciprocation based on the TFT is “myopic”. Since peers in P2P networks are generally involved in repeated and long-term interactions, such my topic resource reciprocation strategy can result in a suboptimal performance for the involved peers.

Overview of P2P system structures

P2P systems can be classified into two different classes: the structuredP2P systems and the unstructuredP2P systems. In structured P2P systems, connections among peers in the network are standard, and peers keep information about the resources (e.g., shared content) that their neighbor peers have. As a result, the data queries can be efficiently directed to the neighbor peers that have the desired data, even if the data is extremely rare. In structured P2P systems impose constraints both on node (peer) graph and on data placement to enable efficient discovery of data. The most common indexing that is used to structure P2P systems is the Distributed Hash Tables (DHTs) indexing. Similar to a hash table, a DHT provides a lookup service with pairs that are stored in the DHT. Any participating peers can efficiently retrieve the value associated with a given unique key. However, this may result in higher overhead compared to unstructuredP2P networks. Different DHT-based systems such as Chord, Pastry, Tapestry, CAN are different in their routing strategies and their organization schemes for the data objects and keys. Unlike structured P2P systems, in unstructured P2P systems, connections among peers in the network are formed arbitrarily in flat or hierarchical manners. In order to find as many peers that have the desired content as possible, peers in unstructured P2P systems query data based on several techniques such as flooding. Three different designs of unstructured P2P systems exist: centralized unstructured P2P systems, hybrid unstructured P2P systems, and decentralized (or pure) unstructured P2P systems. In a centralized unstructured P2P system, a central entity is used for indexing and get the entire system. In contrast to the structured approach, the connection between peers in the centralized unstructured approach is not determined by the central entity. ABitTorrent network is an example of a centralized unstructured P2P network. Napster, the network that gave the idea of P2P file sharing, is another example of a centralized design. In Napster, a server (or server farm) is used to provide a central directory. A peer in the network informs the directory server of its IP address and the names of the contents that it is available for sharing. So, the directory server knows which objects each peer in the network have, and creates a centralized and dynamic database that maps content name into a list of IPs. The main disadvantage of Napster’s design is that the directory server is a single point of failure. This means that, if the directory server crashes, then the entire network will crash. Furthermore, increasing the size of the network may cause a collision in the directory server, due to the need of responding to many queries and maintaining a large database for this metadata information. The collision can only be resolved by adding more infrastructure, such as servers, which may be expensive. The decentralized unstructured P2P network is an overlay network. An overlay network is a logical network. An edge in this network exists between any pair of peers that maintain a TCP connection. The decentralized unstructured overlay network is flat, this means that all peers act as equals. There is neither a central server that manages the network, nor are there preferred peers with a special infrastructure function. The network has a single routing layer. Gnutella is a characteristic example of a decentralized unstructured P2P network. In order to join the Gnutella network, a user initially connects to one of several known bootstrapping peers. The bootstrapped peers then respond with the information about one or more existing peers in the overlay network. This information includes the IP address and port of each peer. The peers in Gnutella are aware only of their neighbor peers. Peers that are connected with each other in the overlay network have a common virtual edge in the overlay network. In Gnutella, queries are distributed among the peers using a variation of a flooding mechanism. A peer that is interested in specific content sends a query to its neighbors in the overlay network. Every neighbor then forwards the query to all of its neighbor peers. The procedure continues until the query reaches a specific depth of search limit. Upon receiving a flood query, a peer that has a copy of the desired content sends a ‘query hit response ‘to the peer that originated the query, which is an indication of having the content. The response is sent on the reverse path of the query, using pre-existing TCP connections. The peer that originated the query then selects one peer from the responded peers, and downloads the desired content through a direct TCP connection from the selected peer. Although Gnutella design is simple, highly decentralized, and does not require peers to maintain information related to location of contents, it is often criticized for its non-scalability. This is because query traffic can grow linearly with the total number of queries, which in turn grows with the system size. In addition, another drawback of the protocol is that a peer that originated the query may not find the desired content especially if the content is rare. A hybrid unstructured P2P network allows the existence of infrastructure nodes, often referred to as super-peers (or super-nodes or overlay nodes). This creates a hierarchical overlay network that addresses the scaling problems on pure unstructured P2P networks such as Gnutella. A peer in such network can typically change roles over time. For example, a regular peer can become a super-peer that takes part in coordinating the P2P network structure. Finally, the fact that super-peers may have more responsibilities than ordinary peers can result in a bottleneck.

3.2.2 Chubby and ZooKeeper

Chubby is a lock mechanism service for distributed systems. In most distributed systems, fault tolerance is a prime objective and so each device is built in such a way to have minimum dependence on one another in case there is a case of failure. Some distributed systems are also designed in such a way and this locking system is used for mainly these types of loosely coupled systems. There is a need for plenty of locking mechanisms over distributed systems on the whole. The use of shared resources and processors is becoming more and more common with the rapid growth of such systems. Such an access to common data might cause certain inconsistencies to appear in these systems mainly because the change made by one application is not reflected on another resource that the system will use later. Another reason for this failed updating is the level of redundancies that is being maintained in these systems due to the fact that data must be available in case of any particular processor or node failure. A lock system allows only one process or processor to use a particular resource and blocks off all other requesting or waiting processes until the executing process reaches conclusion. It manages these access rights in the same time in a manner to promote maximum process utilization and efficiency. In comparison to other lock systems, chubby system holds a lock for a matter of hours or days and not for anytime less. Chubby is a system that focuses more on reliability and fault tolerance and less on performance. It primarily achieves fault tolerance by replication of data. A Chubby instance or cell is known to serve up to ten thousand processor machines connected together using a 1Gbps Ethernet connection. Chubby can be used in a different way to make decisions regarding the master node in a particular host of servers. The BigTable [37] system uses Chubby because of this kind of service. Most systems use Chubby for the safe that offers. The question that create is how the primary or master node or system is decided from a group of equivalent other nodes. It use the Paxos algorithm [38] to resolve such consensus issues. Paxos is an advanced algorithm used for the above mentioned purpose. Its main feature is to resolve such consensus issues even when the participants in the resolving have failure prone communication media.

Design

The overall design of the Chubby system consists of Chubby clients running on the local machine which can link to its appropriate libraries. Each of the cells or instances consists of a small set of servers to improve fault tolerance. These servers communicate with the master using RPC (Remote Procedure Call). The master is decided based on the consensus resolution algorithm like Paxos. Once a master is chosen, it remains master for an specific amount of time in which it handles all requests. Slave clients identify the master using specific request calls. Once found that particular client routes all calls to the master it gets to the appropriate address. The interesting part is how read and writes requests are spread through the system. When the write request reaches a server through a consensus call, it is acknowledged when the request has spreader to a majority of the cells in the cell. A read request received is handled quite differently as these calls are handled by the master alone. If in case a particular server fails, a simple swap system selects a new machine and starts the lock server binary on it. Then the DNS updates the new IP to the currently selected server. A separate list of the new cell members is maintained in a special cell database which is kept consistent throughout. The master client architecture of Chubby is shown below. The client processes use RPC calls to interact with the master. These calls are made by special client side chubby libraries [15].

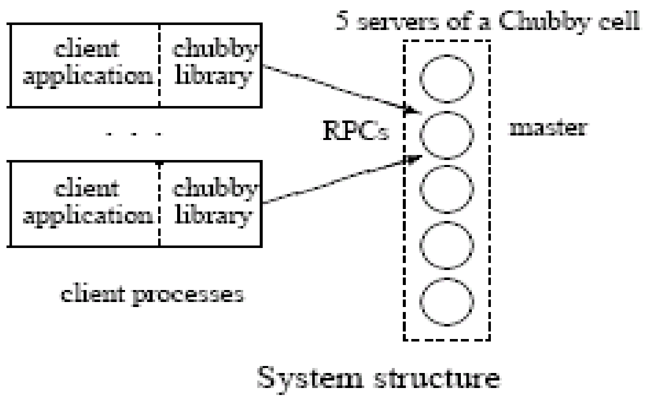


Figure 6-Chubby Structure [39]

Finally, the file system which the Chubby system maintains is similar to the UNIX system although very much simpler.

ZooKeeper is a fault-tolerant distributed coordination service for cloud computing applications currently maintained by Yahoo! and the Apache Software Foundation. It provides fundamental services for other cloud computing applications by enclosing distributed coordination algorithms and maintaining a simple database. The service is intended to be highly-available and highly-reliable, so several client processes rely on it for bootstrapping, storing configuration data, status of running processes, group membership, implementing synchronization primitives, and managing failure recovery. It achieves availability and reliability through replication, and is designed to have good performance in read-dominant workloads.
Total replication of the ZooKeeper database is performed on an total, i.e., a number of host servers, three or five being usual configurations, of which one is the leader of

a quorum. The service is considered up as long as a quorum of the ensemble is available. A critical component of ZooKeeper is Zab [40], the ZooKeeperAtomic Broadcast algorithm, which is the protocol that manages atomic updates to the replicas. It is responsible for agreeing on a leader in the ensemble, synchronizing the replicas, managing update transactions to be broadcast, as well as recovering from a crashed state to a valid state.

Background

A broadcast algorithm transmits a message from one process (the main process) to all other processes in a network or in a broadcast domain, including the main. Atomic broadcast protocols are distributed algorithms guaranteed either too correctly broadcast or to abort without side effects. It is widely used in distributed computing for group communication. Atomic broadcast can also be defined as a reliable broadcast that satisfies total order, that satisfies the following properties:

- ✓ Validity: If a correct process broadcasts a message, then all correct processes will eventually deliver it.
- ✓ Uniform Agreement: If a process delivers a message, then all correct processes eventually deliver that message.
- ✓ Uniform Integrity: For any message m, every process delivers m at most once, and only if m was previously broadcast by the sender of m.
- ✓ Uniform Total Order: If processes p and q both deliver messages m and m0, then p delivers m before m0 if and only if q delivers m before m0.

4. Approach

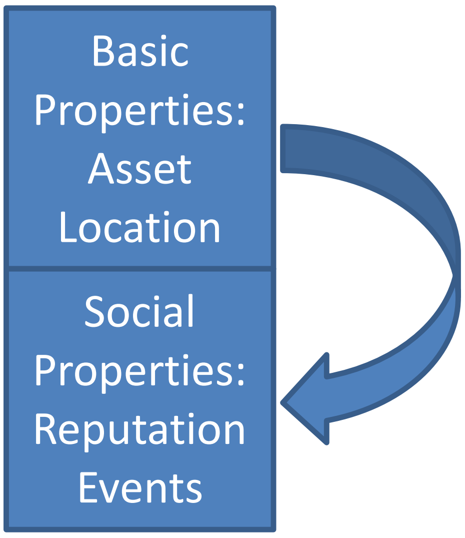
In this chapter we will analyze the ontology which was the base for the development of our code and the reasons that made us to select it.

4.1. Ontologies

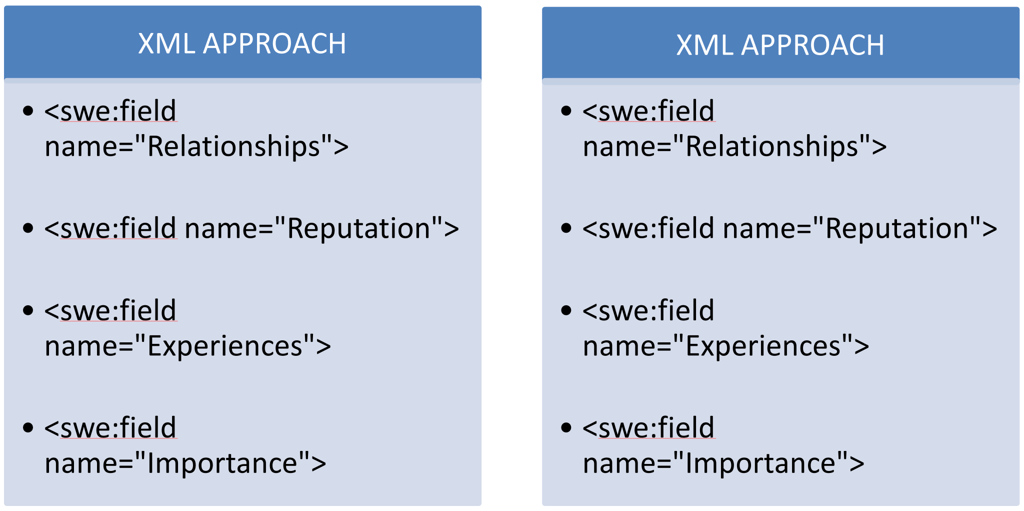
In previous chapter mentioned some ontologies which are very important about the state of the art of Internet of Things (IoT). The best ontology to describe this problem is the SensorML. The primary focus of the Sensor Model Language (SensorML) is to provide a robust and semantically-tied means of defining processes and processing components associated with the measurement and post-measurement transformation of observations. This includes sensors and actuators as well as computational processes applied pre- and post-measurement. The main objective is to enable interoperability, first at the syntactic level and later at the semantic level (by using ontologies and semantic mediation), so that sensors and processes can be better understood by machines, utilized automatically in complex workflows, and easily shared between intelligent sensor web nodes. Two very important improvements of this language is that there are further support for positions and dynamic state for example locations, orientation, velocity and acceleration and there is availability for real-time access to values and to data streams.

With the help of SensorML we aim to give some characteristics in objects which help them to get the necessary experience. Based on some already existing attributes we will add some additional social properties which will help objects with their experience. Some examples of social properties are:

- ✓ Experience
- ✓ Goals
- ✓ Relationships etc.


Basic Properties: Asset Location
Social Properties: Reputation Events

The scheme above describes the approach of our work. The basic properties are the already existing properties such as location and reliability. In our research we extended the basic properties and we create the social properties such as reputation, experience and events. The social properties help us to give better and more analytical characteristics to our things so as to create the idea of “facebook of things”.

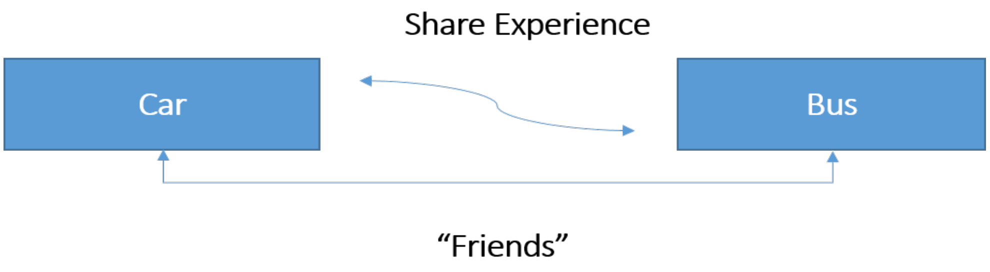


XML APPROACH

- <swe:field name="Relationships">
- <swe:field name="Reputation">
- <swe:field name="Experiences">
- <swe:field name="Importance">

XML APPROACH

- <swe:field name="Relationships">
- <swe:field name="Reputation">
- <swe:field name="Experiences">
- <swe:field name="Importance">



In the example above we see this idea where two objects became “friends” by sharing experience between them. So the point is to create a society of things which can communicate between them and exchange information.

In the next chapter we will analyze them with the help of SensorML.

4.2. Distributed Coordination

The growth in the number of communicating elements and the number of communicating subgroups is huge. A characteristic example of this challenge is the evolution of clustering and group communication in enterprise systems. Our concept is to bilk from the model of hierarchical structure and create an autonomous system. The idea of the main entity does not exist and the other entities act autonomous when they get the appropriate experience. In other words we will not have an entity which acts like a “leader” and give directions and information to sub entities because every entity will act autonomous when it will have all the information. All the above follow the Peer-to-Peer (P2P) model where all peers act as equals and we don’t have the “old classic” model of client-server. All of these make our system faster and useful without collisions and offer us security because if a peer crashes another peer will be the main entity and the system will continue to act without a problem.

5. Social-aware Ontologies

In this chapter we will analyze the above approach through a SensorML code and some paradigms which describes our idea [41].

```
<?xml version="1.0" encoding="UTF-8"?>
<sml:SensorML
xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/sensorML/1.0.1
http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd" version="1.0">


<sml:member>
<sml:System gml:id="basic">

<sml:identification>
<sml:IdentifierList>
<sml:identifier name="siteID">
<sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
<sml:value>urn:ogc:object:system:REAP:siteName</sml:value>
</sml:Term>
</sml:identifier>
</sml:IdentifierList>
</sml:identification>


<sml:capabilities name="specifications">
<sml:CapabilityList>
<sml:capability name="BasicProperties">
<swe:field name="Context">
<sml:description>Situations Conditions</sml:description>
<sml:value></sml:value>
</swe:field>

<swe:field name="Location">
<sml:description>Location static or not</sml:description>
<sml:value></sml:value>
</swe:field>

<swe:field name="Reliability">
<sml:description>Capability for constant operation</sml:description>
<sml:value></sml:value>
</swe:field>
```

```
<swe:field name="Trust">
<sml:description>trust</sml:description>
<sml:value>80</sml:value>
</swe:field>

<swe:field name="PhysicalEntity">
<sml:description>Connectivity     Mobility     BatteryLife     PowerDemands
Dimensions<sml:description>
<sml:value></sml:value>
</swe:field>

<swe:field name="Asset">
<swe:field name="Data">
<swe:Options definition="Accessibility">
<swe:description>Accessibility Timeliness Amount Accuracy</swe:value>
<sml:value></sml:value>
</swe:Options>
</swe:field>

<swe:field name="Services">
<swe:Options definition="Availability">
<swl:description>Availability Quality Cost</swl:description>
<sml:value></sml:value>
</swe:Options>
</swe:field>
</swe:field>
</sml:CapabilityList>
</sml:capabilities>
</sml:System>
</sml:member>


<sml:member>
<sml:System gml:id="social">

<sml:identification>
<sml:IdentifierList>
<sml:identifier name="siteID">
<sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
<sml:value>urn:ogc:object:system:REAP:siteName</sml:value>
</sml:Term>
</sml:identifier>
</sml:IdentifierList>
</sml:identification>
```

```
<sml:capabilities name="specifications">
<sml:CapabilityList>
<sml:capability name="SocialProperties">
<swe:field name="Intentions">
<sml:value>Similar Goals</sml:value>
</swe:field>

<swe:field name="Relationships">
<sml:value>Christos</sml:value>
<sml:reputation>39</sml:reputation>
<sml:value>Marilena</sml:value>
<sml:reputation>40</sml:reputation>

</swe:field>
```

```
<swe:field name="Experiences">
<sml:description>Experiences with respect to management of the IoT device</sml:description>

<!--LUMcondition = luminosity
BTcondition = battery condition
BRaction = brightness action
WF_OFF_ON_action = wi-fi on / off (1 enable - 0 disable)
BL_OFF_ON_action = bluetooth on / off   (1 enable - 0 disable)
ALaction = alert action (1 enable - 0 disable)
DATA_OFF_ON_action = data action  (1 enable - 0 disable) --!>


<sml:LUMcondition>25</sml:LUMcondition>
<sml:BRaction>25<sml:action>
<sml:SWaction><sml:action>
<sml:ALaction>1<sml:action>

<sml:LUMcondition>50</sml:LUMcondition>
<sml:BRaction>35<sml:action>
<sml:SWaction><sml:action>
<sml:ALaction>1<sml:action>

<sml:LUMcondition>200</sml:LUMcondition>
<sml:BRaction>200<sml:action>
<sml:SWaction><sml:action>
```

```
<sml:ALaction>1<sml:action>

<sml:LUMcondition>255</sml:LUMcondition>
<sml:BRaction>255<sml:BRaction>
<sml:ALaction>1<sml:ALaction>


<sml:BTcondition>20</sml:BTcondition>
<sml:WF_OFF_ON_action>0<sml:action>
<sml:BL_OFF_ON_action>0</sml:BL_OFF_ON_action>
<sml:DATA_OFF_ON_action>0</sml:DATA_OFF_ON_action>
<sml:ALaction>1</sml:ALaction>
<sml:BRaction>20</sml:BRaction>

<sml:BTcondition>100</sml:BTcondition>
<sml:WF_OFF_ON_action>1<sml:action>
<sml:BL_OFF_ON_action>0</sml:BL_OFF_ON_action>
<sml:DATA_OFF_ON_action>1</sml:DATA_OFF_ON_action>
<sml:ALaction>1</sml:ALaction>
<sml:BRaction>75</sml:BRaction>

<sml:BTcondition>50</sml:BTcondition>
<sml:WF_OFF_ON_action>1<sml:action>
<sml:BL_OFF_ON_action>0</sml:BL_OFF_ON_action>
<sml:DATA_OFF_ON_action>1</sml:DATA_OFF_ON_action>
<sml:ALaction>1</sml:ALaction>
<sml:BRaction>75</sml:BRaction>
</swe:field>
```

```
<swe:field name="Trends">
<sml:value>Trends</sml:value>
</swe:field>
<swe:field name="Importance">
<sml:description>Influence of objects</sml:description>

</swe:field>

<swe:field name="Administration">
<sml:description>Owner</sml:description>
<sml:value>1</sml:value>
</swe:field>

<swe:field name="PastExperience">
```

```
<sml:description>PastExperience</sml:description>
<sml:value></sml:value>
</swe:field>

<swe:field name="Objectives">
<sml:description>Similar Goals</sml:description>
<sml:value></sml:value>
</swe:field>

<swe:field name="Experience">
<swe:field name="Functional">
<swe:Options definition="jhkjjj">
<swe:description>PhysicalEntity Availability Healing</swe:description>
<sml:value></sml:value>
</swe:Options>
</swe:field>

<swe:field name="NonFunctional">
<swe:Options definition="jhkjjj">
<swe:description>Reliability Protection Trust Expectations</swe:description>

<sml:value></sml:value>
</swe:Options>
</swe:field>
</swe:field>
</sml:CapabilityList>
</sml:capabilities>
</sml:System>
</sml:member>
</sml:SensorML>
```

In the table [42] below represent the code with examples of conditions and what the results are if we have specific properties.

Properties	Conditions
Traffic Congestion	
accident	stop
traffic	slow down
no traffic / accident	normal
Smart Parking	
>10 parkings slots	free to park
<10 parkings slots	limited parking available
0 parkings slots	no parking available
Forest Fire Detection	
fire	alert authorities
early start fire	alert authorities
no fire	no alert
Waste Management	
75% - 100%	emergency evacuate recycle bin
25% - 65%	alert authorities
0% - 25%	no alert
Noise Urban Maps	
65% - 100%	call authorities
25% - 65%	alert authorities
0% - 25%	no alert

Case Study

Previously we describe how things can share experiences through a SensorML code. We will explain some parts of the code. In the example below we have:

Battery=20%
WiFi=off
Bluetooth=off
Data=off

Brightness=20%

This means that when we have 20% battery then Wi-Fi will be off, data and Bluetooth will be on and the brightness=20%

Those settings will be applied according to this code.

```
<sml:BTcondition>20</sml:BTcondition>
<!--if battery condition is 20% then turn off the wi-fi,the bluetooth and the data and take alert action and make the brightness 20--!>

<sml:WF_OFF_ON_action>0<sml:action>
<sml:BL_OFF_ON_action>0</sml:BL_OFF_ON_action>
<sml:DATA_OFF_ON_action>0</sml:DATA_OFF_ON_action>
<sml:ALaction>1</sml:ALaction>
<sml:BRaction>20</sml:BRaction>
```

These different actions present in the above table The following table will demonstrate the case scenarios that will happen in our example.

Battery (%)	Wi-Fi	Bluetooth	Data	Brightness (%)
20	off	off	off	20
35	on	off	off	45
50	on	on	on	60
100	on	on	on	100

Future work

Through this analysis we conclude that all the “things” (e.g. people, animals, devices, mobiles etc) can share experiences between them. Because the things are billions and the characteristics are too many we must classify them in categories. To extend this research in the future we must do some research about the characteristics of things and depend on them we must create groups that include only things with the same characteristics. For example the mobile phones will be a group, the devices will be another group which will have subgroups depends on the characteristics of devices (e.g. cameras, phones, house devices etc.). All the above we can embed them in Socail_IoTML which will consist for all the characteristics of “things” and according to circumstances will use the corresponding source of code. All of these will create a “smart city” where all the “things” will communicate and share experiences between them. A characteristic paradigm which describes our theory is to have sensors in the cars. This will help drivers to know the time that will arrive in their destination, to know where in the road has traffic or an accident happened etc. So the idea is to develop in the future a city where all the “things” can communicate and share experiences without the help of humans. To make this research more competitive we can add to our Social_IoTML rules that define the process profile (e.g. inputs, outputs, parameters, and metadata). Finally, this help the “things” to act better and with a specific way.

6. Conclusion

To conclude, the idea of Internet of Things (IoT) and how the “things” can share experiences between them consists a very important part of development because it is something new which will make our future easier. In our research we tried to approach how we can share experience. After an analysis through the Internet of Things (IoT) and what problems we must face (problem statement) we tried to resolve the current problem status through some solutions. The distributed coordination and the existing ontologies helps us to find ways to make different things communicate and share experiences. Through the distributed coordination we achieve the autonomous use of things with very positive results as well we minimized the errors and made the system safer. About the ontologies we find and analyze a lot of them but the ontology that represent better our analysis and it was much closer to our target was the SensorML. The SensorML helps us to develop an environment where the things can share experiences and all of these properties concluded in a SensorML code that give us results according to the data that takes as input. The SensorML code that we developed has the name Social_IoTML and it is an extend of Basic Properties of SensorML that adapted to our requirements. All the above in the future can be developed more and be implemented in a Smart City.

References

[1] Tai-Ran Hsu, “Reliability in MEMS packaging”, 44th International Reliability Physics Symposium, San Jose, CA, March 26-30, 2006

[2] M. Lang, “GlowCap maker Vitality bought by Soon-Shiong”, Boston Business Journal, 2011, http://www.vitality.net/glowcaps.html

[3] H. McCracken, “Nest’s Second-Generation Thermostat: Simpler and Smarter”, Time, 2012, https://nest.com/

[4] R.Pease, Internet of things: Trash talk signals mobile future, BBC, 2013, http://bigbelly.com/solutions/stations/smartbelly/

[5] Jayavardhana Gubbia, Rajkumar Buyyab, Slaven Marusic, Marimuthu Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions”, The University of Melbourne, 2013

[6] SENSEI Project, 2008, http://www.sensei-project.eu/

[7] A. Ajith Kumar S., Knut , Lars M. Kristensen, “An Industrial Perspective on Wireless Sensor Networks - A Survey of Requirements”, Protocols and Challenges, IEEE, 2014

[8] K. Domdouzisa, , B. Kumarb, , C. Anumba, “Radio-Frequency Identification (RFID) applications: A brief introduction”, Centre for Innovative and Collaborative Engineering, Department of Civil and Building Engineering, Loughborough University, Loughborough, UK, School of the Built and Natural Environment, Glasgow Caledonian University, Glasgow, 2006

[9] Dirk Krafzig,Karl Banke,Dirk Slama, “Enterprise SOA: Service-oriented Architecture Best Practices”, Pearson Education Inc., 2004

[10] Xingchen Chu, Open Sensor Web Architecture: Core Services, The University of Melbourne, Australia, 2005

[11] A. Bröring, J. Echterhoff , S. Jirka , I. Simonis , T. Everding, C. Stasch , S. Liang and R. Lemmens, “New Generation Sensor Web Enablement”, www.mdpi.com/journal/sensors , 2011

[12] http://www.opengeospatial.org/, February 9, 2008

[13] A.Hands, Ahnfelt, Beland, Brian Kendig, Cacycle, Chl, CodeMonk, “Uniform Resource Name”, 2011

[14] Al Underbrink, A. Potter, K. Witt, J. Stanley, “Modeling Sensor Web Autonomy”

[15] Gunther Schadow, Clement J. McDonald, “The Unified Code for Units of Measure”, Regenstrief Institute, 2013

[16] “MetadataTypeAttribute Class”, Microsoft, 2014, http://msdn.microsoft.com/

[17] Israel Mayk, D. Eng. Sc., Azad M. Madni, Ph. D., “The Role of Ontology in System-of-Systems Acquisition”, Intelligent Systems Technology, Inc, 2006

[18] Matt Calder, Robert A. Morris, Francesco Peri, “Machine Reasoning about Anomalous Sensor Data”, University of Massachusetts Boston, 2006-2010 , http://www.cesn.org/projects/semantic.php

[19] C. Rueda, N. Galbraith, R. A. Morris, L. E. Bermudez, R/ A. Arko, J. Graybeal, “The MMI Device Ontology: Enabling Sensor Integration”, American Geophysical Union, 2010,https://marinemetadata.org/

[20] H. Neuhaus, M. Compton, “The Semantic Sensor Network Ontology,A Generic Language to Describe Sensor Assets”, Commonwealth Scientific and Industrial Research Organisation, Australia, 2009

[21] Matt Calder, Robert A. Morris, Francesco Peri, 17. Matt Calder, Robert A. Morris, Francesco Peri, “Machine Reasoning about Anomalous Sensor Data”, University of Massachusetts Boston, 2009

[22] J. Madina, S. Bowersb , M. Schildhauera, S. Krivovc, D. Penningtond, F. Villac, "An ontology for describing and synthesizing ecological observation data", www.elsevier.com/locate/ecolinf , 2007

[23] J. Madina, S. Bowersb , M. Schildhauera, S. Krivovc, D. Penningtond, F. Villac, "An ontology for describing and synthesizing ecological observation data", www.elsevier.com/locate/ecolinf , 2007

[24] S. Hachem, T. Teixeira, V. Issarny, "Ontologies for the Internet of Things ", ACM/IFIP/USENIX 12th International Middleware Conference, 2011

[25] S. Hachem, T. Teixeira, V. Issarny, "Ontologies for the Internet of Things ", ACM/IFIP/USENIX 12th International Middleware Conference, 2011

[26] Daniel L. Cook, John H. Gennari, Jose L. V. Mejino, Maxwell L. Neal, "Ontology of Physics for Biology (OPB):Annotation of biological data and models", CellML Workshop, 2009

[27] Elena Paslaru Bontas Simperl, Malgorzata Mochol, "Cost Estimation for Ontology Development", Free University Berlin, 2006

[28] Ciaran Palmer, "SensorML on SenseTile", UCD School of Computer Science and Informatics, 2010

[29] World Wide Web Consortium, http://www.w3.org/, 2014

[30] David J. Russomanno, Cartik R. Kothari, Omoju A. Thomas, "Building a Sensor Ontology:A Practical Approach Leveraging ISO and OGC Models, The University of Memphis, 2000

[31] Jeff Heflin, "AN INTRODUCTION TO THE OWL WEB ONTOLOGY LANGUAGE", Lehigh University, 2004

[32] Alexander V. Besprozvannykh, "Extensible Markup Language (XML): Essentials for Climatologists", CCl OPAG 1 Implementation/Coordination Team, 2005

[33] Rodrigo Rodrigues, Peter Druschel, "Peer-to-Peer Systems", Communications of the ACM, 2010

[34] Mike Burrows, "The Chubby lock service for loosely-coupled distributed systems", Google Inc., 2005

[35] Patrick Hunt, Mahadev Konar, Flavio P. Junqueira, Benjamin Reed, "ZooKeeper: Wait-free coordination for Internet-scale systems", Yahoo! Grid, 2008

[36] George Bunn, Rodger A. Payne, "Tit-for-tat and the Negotiation of Nuclear Arms Control", Stanford University, 2000

[37] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data", Google, Inc., 2006

[38] Diego Ongaro, John Ousterhout, "In Search of an Understandable Consensus Algorithm (Extended Version)", Stanford University, 2014

[39] Mike Burrows, "The Chubby lock service for loosely-coupled distributed systems", Google Inc., 2005

[40] Flavio P. Junqueira, Benjamin C. Reed, Marco Serafini, "Zab: High-performance broadcast for primary-backup systems", Yahoo! Research, 2009

[41] Janet Fredericks, "Putting WHOI Data on the Map", AOP&E, WHOI, 2006

[42] Alicia Asín, David Gascón, "50 Sensor Applications for a Smarter World", Zaragoza (Spain), 2014, http://www.libelium.com/