1ALMA MATER STUDIORUM
UNIVERSITY OF BOLOGNA

ARCES - ADVANCED RESEARCH CENTRE ON ELECTRONIC SYSTEMS
FOR INFORMATION AND COMMUNICATION TECHNOLOGIES E. DE CASTRO

Semantic Service Architectures for Smart Environments

Alfredo D'Elia

PhD Coordinator
Prof. Claudio Fiegna

Supervisor
Prof. Tullio Salmon Cinotti

PHD. THESIS
January, 2009 – December, 2011

PHD PROGRAM IN INFORMATION TECHNOLOGY

ABSTRACT

Many industries and academic institutions share the vision that an appropriate use of information originated from the environment may add value to services in multiple domains and may help humans in dealing with the growing information overload which often seems to jeopardize our life.

It is also clear that information sharing and mutual understanding between software agents may impact complex processes where many actors (humans and machines) are involved, leading to relevant socioeconomic benefits.

Starting from these two input, architectural and technological solutions to enable "environment-related cooperative digital services" are here explored.

The proposed analysis starts from the consideration that our environment is physical space and here diversity is a major value. On the other side diversity is detrimental to common technological solutions, and it is an obstacle to mutual understanding. An appropriate environment abstraction and a shared information model are needed to provide the required levels of interoperability in our heterogeneous habitat.

This thesis reviews several approaches to support environment related applications and intends to demonstrate that *smart-space-based, ontology-driven, information-sharing platforms* may become a flexible and powerful solution to support interoperable services in virtually any domain and even in cross-domain scenarios. It also shows that semantic technologies can be fruitfully applied not only to represent application domain knowledge. For example semantic modeling of Human-Computer Interaction may support interaction interoperability and transformation of interaction primitives into actions, and the thesis shows how smart-space-based platforms driven by an interaction ontology may enable natural ad flexible ways of accessing resources and services, e.g, with gestures. An ontology for computational flow execution has also been built to represent abstract computation, with the goal of exploring new ways of scheduling computation flows with smart-space-based semantic platforms.

I – TABLE OF CONTENTS

II. Introduction

My doctorate in information technologies was dedicated to the investigation of the smart environments domain, with particular attention to software architectures for smart services based on semantic technologies.
The results are solutions at different levels of abstraction, sometimes bound to specific application scenarios and sometimes more general but always with the objective of generating added value from the point of view of the perceived functional and non-functional service qualities.
The experimental approaches based on semantic context platforms applied to relevant scenarios reveal sometimes the potential to have an impact on the market, replacing localized approaches based on proprietary standards and focused on a confined domain, with approaches oriented towards synergic collaborations, interoperability, reuse and extendibility.
My contribution can be examined under different perspectives.
Semantic modeling of relevant context attributes was a key activity to share data in multi-agent concurrent scenarios.
Semantic modeling of Human-Computer Interaction may support interaction interoperability and transformation of interaction primitives into actions, and my contribution was to show how smart-space-based platforms driven by an interaction ontology may enable natural ad flexible ways of accessing resources and services, e.g. with gestures.
Semantic modeling of computational flows was done to represent abstract computation in terms of a chain of function calls, with potential advantages similar to those provided by Haskell Monads.
In general the above mentioned semantic models were conceived and implemented by continuously checking the tradeoff between expressivity and computability, in order to always provide an interoperable and machine understandable representation of information.

Beside semantic models for different conceptual areas another result of my work is the message that semantic smart environments are fighting to become a reality, and not just a research exercise. This message is based on the increasing maturity of two vectors: the application scenarios and the platforms.

Indeed ontology driven interoperability platforms are already showing their value in several european research contexts, such as, for example three ARTEMIS-JU projects, i.e. SOFIA, CHIRON and IoE to name only those where I'm involved in.

SOFIA (*Smart Environments for Intelligent Applications*, http://www.sofia-project.eu/) is the reference project for my entire work. It considers smart environments at different granularity levels, and particularly it is focused on a common solution to support applications inside a car, a building or city. Such common solution is a platform to share interoperable information in smart environments applications. The platform is called IOP (Interoperability Platform) and it implements the *Smart Space* concept. A *Smart Space* is a named information search domain, where information describes the objects existing in the environment, including the environment itself, together with their properties and the relations among them. Information may originate from heterogeneous legacy and embedded devices or may be produced by appropriate aggregators. The platform is very simple and it may be discovered and accessed as a Service (e.g. a Web Service, an OSGi Service). Information is represented in an application independent semantic format (RDF) and its interoperability and semantics are based on common ontologies. The platform is agnostic with respect to ontology, programming language, service and

communication levels. It is a very general platform that can be customized with specific semantic models, including those developed in this Doctorate.

The project Chiron *(Cyclic and person-centric Health management: Integrated appRoach for hOme, mobile and clinical eNvironments,* http://www.chiron-project.eu/ *)* has the goal to create an open tele-health platform for end-to-end health-care applications, and here SOFIA IOP is reused as the core interoperability component at patient's home.

The project" IoE" (*Internet of Energy for Electric Mobility,* http://www.artemis-ioe.eu/) has the goal of Integrating Data and Energy networks to extend the smart grid to the electric vehicle and its users. Here the relevant challenge is to extend the M2M service architecture adopted by IoE with Smart Spaces hosted by Semantic Information Brokers, again reusing SOFIA IOP.

Applications are expected to appear also in other domains including cultural heritage, tourism and agriculture. Not only the size of the application space increases, but also semantic platforms based on smart spaces are getting more and more mature, with a thematic action line of the EIT ICT LABS of the European Institute of Innovation & Technology dedicated to smart spaces and related platforms (http://eit.ictlabs.eu/).

Smart environments have been studied in their whole and in their components during recent years and they involve many branches of information technology and cognitive sciences. The aim of this work is to build on semantic web formalisms, description logics and smart environments and try to merge these sciences into software architectures and solutions which are as general as possible to fit a wide range of relevant use-cases.

The first two sections describe the state of the art and related works in the field of smart environments, context platforms and semantic web. Section three is a bout the semantic modeling of simple context attributes like sensor data and introduces the concept of smartification. Section four is about the semantic modeling of abstract context like the description of access control rights to resources, and the semantic modeling of computation: work that has been carried out during six month spent as a Nokia Intern in Helsinki. In Section five a prototype of software architecture providing interesting capabilities for pervasive computing is described and then conclusions are drawn.

SECTION 1

OVERVIEW OF SMART ENVIRONMENTS: RELATED WORKS AND TECHNOLOGIES

A smart environment is something we dreamt of since our first electronics lectures when we were exposed to the incredible size of the embedded systems application space and to the exponential progress rate of fundamental qualities of electronic systems.

To most of us happened at least one time to think about the reliability of some advanced functionality, and naturally got used to it as a well established reality a short time after. In a similar way, but on a more scientifically sound basis, at the beginning of the 90's Mark Weiser envisioned a new concept of human-computer interaction (HCI) where ubiquitous hidden machines collaborate for end user satisfaction, sometimes even without explicit user request[1]. In Weiser's vision, besides being autonomous, the electronic ecosystem was "interconnected by wires or radio waves and infrared", and it was so pervasive to be unnoticed by anyone. The time demonstrated the foundation of Weiser ideas: miniaturization followed and almost went beyond Moore's law by increasing performance and diminishing the recurring engineering costs of electronic devices. Low power connectivity technologies become available also to tiny or resource constrained devices, everyday life objects get more and more technologized, applications and services make use, when possible, of web resources to become aware of the user profile attributes, including their habits and tastes. Currently, most of the people are connected to the web through various devices (e.g. laptop, tablet PC, Smartphone) and with different communication channels (e.g. Wi-Fi, HSPA, fiber), complex electronic devices are commonly affordable and multiple vendors differentiate their products in the race to dominate their market area or to hold their position. In this dynamic scenario research plays a fundamental role in the attempt to realize our visions and progress towards our future. This thesis is about smart space based architectures supporting environment related cooperative services. This topic involves multiple technologies and disciplines, therefore in this first chapter I'll describe the related work and the state-of-the-art of smart space related technologies in order to create a base of knowledge to understand where we are and what we reasonably expect to have in the near future i.e. the motivations originating our vision.

1.1. Basic definitions, objectives and problem statement

Remembering that the target of an architecture for smart environment applications is its ability to support configurable and context dependent services, the first concept to define in the present discussion is the context.

Various definitions exist for the context but the most commonly used in recent years is the one provided by Dey[2]: "any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves". Shortly context is everything of interest, considering target scenario objectives; an application able to modify its behaviour depending from the context can be called context-aware. Context awareness is a requisite for smart environments since a user expects a "smart" environment to understand and take in consideration her identity, tastes, preferences, location, profile, past choices etc.

Another implicit requirement that smart environments must satisfy is the interoperability of its parts. The interoperability can be defined as the ability of different entities to understand each other, to communicate and to cooperate for a common purpose. Since interoperability is a wide concept it can be considered at different levels of abstraction: the interoperability at communication level regards the possibility of the interoperable entities to communicate through one or more transport layers, the interoperability at information level grants the ability of the different entities to understand the relative information while the interoperability at service level is about the cooperation and coordination of different services to obtain a common objective not possible by using only the available native functionalities. Context awareness can be obtained by appropriate programming of fully interoperable entities that, supposed aware of the context, are able to reason about it and work together to fulfill user requirements in a context dependent way.

The objective of this thesis is to go into detail of context aware applications and their software architectures, trying to use the previous and current research to perform a critical analysis of the state of the art and to find and delineate possible paths the research and industry can follow to progress in this field.

Other important concepts necessary to go further in our analysis and that I will shortly introduce are:

- the commonly used software infrastructures to provide context aware services in smart environments
- the issues to be faced in order to obtain context awareness
- examples of context aware services which can be improved with the adoption of new or different technologies inside the software infrastructure

- the concept of “semantic” or machine interpretable information, and its potential if applied in current software architectures.

1.2. Software architectures for smart environments

The context is a set of values characterizing the relevant entities (intended in the most general sense). Context attributes may be constant (e.g. the name of a person), slowly variable (e.g. the day of the week), or can sensibly change in a short time with variations that may be fundamental in the context interpretation (e.g. hart rate). In every case context values must be inserted into a shared framework or sensed by appropriate sensors able to share their data with all the interested entities. A software infrastructure for smart environment must so comprehend a set of components for sensing data or for inserting them into the system. Once raw information has been sensed is necessary to manage it in an information layer and to make it available when needed. In Fig.1.1 is showed this general, technology independent scenario from which a software architecture can be instantiated by choosing the HW/SW components and the communication protocols inside and between layers.

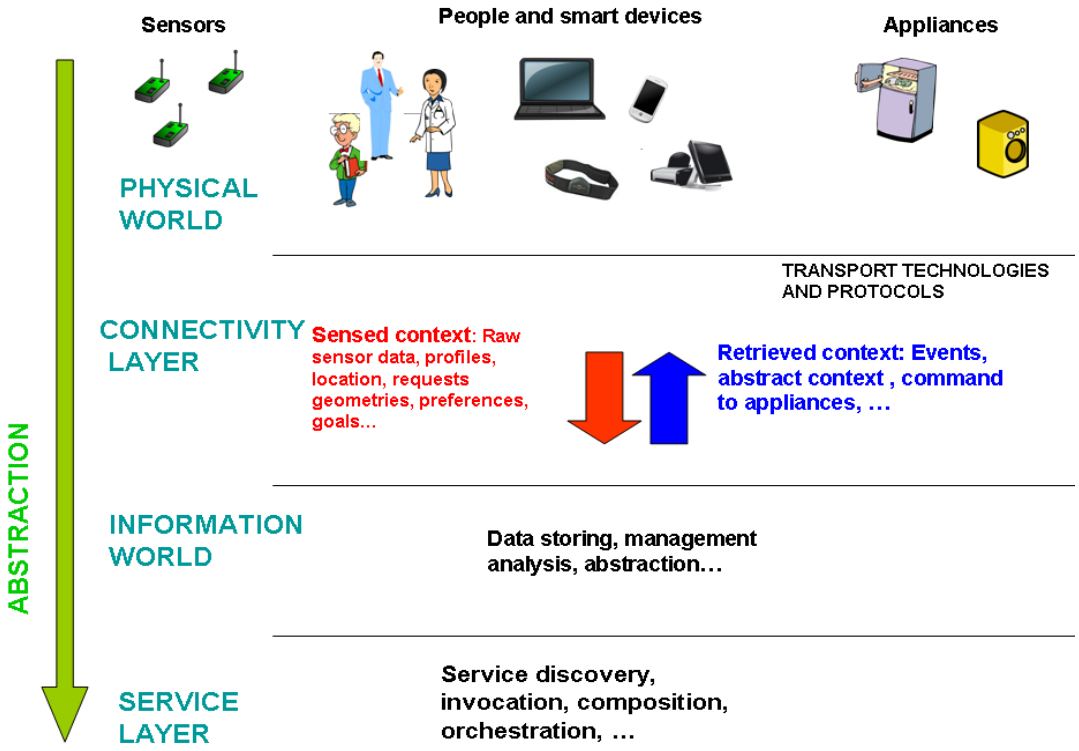


Figure 1.1: principal actors in smart environments

Sensors people and electronic devices produces raw or semi elaborated data which are sent through the connectivity layer to the information world where an infrastructure of HW/SW entities manages all the information. The data of the information world can be queried or augmented by the services; applications act vertically on one or more layers to provide functionalities to the end user. In the framework we have depicted many different architectural and technological choices can be made, we will start by a short description of each layer and from the introduction of the context toolkit whose concepts are present in the most common frameworks for context aware smart applications.

1.2.1. Sensors and physical layer

A deep analysis of sensors in smart environments is not needed in this discussion so I'll only mention some of the most important research topics which are internal to the sensor field:

1. communication technologies between sensors (e.g. Bluetooth, Zig-Bee, Simplicity, etc)
2. efficient information representation(e.g. binary XML[3], or proprietary formats and protocols)
3. reduce energy consumption, for example by reducing observation frequency or turning off unused sensors depending from the application requirements and current status
4. topology adaptation with not fixed sensors
5. …

It is important to analyze the binding between the sensing layer and the quality of the service provided, putting them in relationship with previous considerations about context and interoperability.

Context attributes can be either observed by sensors or introduced by humans manually or from an informative system, the recent advances in miniaturization and sensor networks allowed pervasive environmental sensors, but these are often not inserted in a standard framework and their usage is often not configurable for different scenarios. What happens is that proprietary solutions works only with a predefined number of modes, sensor data are interoperable with devices and software from the same vendor or from its partners in the given project. When there is the attempt to generalize the usage of sensors in a multi-industry not fully a priory defined scenario, there are problems of interoperability at all levels. Moreover the localized applications, correctly working in the target scenarios, are difficult to be ported to new ones and this is in clear opposition with the dynamicity an rapidity of development of environmental intelligence, because every time the interoperability has to be obtained again from scratch. In Oldes [4] a project regarding the monitoring of biomedical parameters, multiple sensors from different vendors where used. There where multiple protocols to be adapted and the representation of data was often optimized for communication, but was not interoperable at information level and so, when the data where put together, an

additional effort for converting all to a common language was needed. Another problem was that even if this effort allows before or later to reach the application objectives, again the data are not in a standard or conventionally determined format and so, if extensions or interaction with external services are needed, new effort has to be provided in order to make their data interoperable. Another example can be that of a large business building where different companies provide the management of different kind of sensors (e.g. presence, temperature, humidity, luminosity, etc). Often is difficult and very expensive to make the different management software interoperate, or this interoperability is not wanted at all by the management software creators for preserving their market area. What is important to be demonstrated by the new research trend based on interoperability is that sensors data, like many other information sets, can be inserted and managed in a common framework interoperable at information level. This innovation will change in some case the current model of business where single systems are delegated to single companies closed software infrastructures, but the changes allows more functionalities, services and efficiency. The reusability and the new potential will probably increase the productivity by reducing the costs and finally leading to Win-Win collaboration strategies between different business companies. A simple demonstration of this can be discussed trying to make hypothesis and to reason about the just mentioned example of the building. In a large building for business activity probably security software exists using cameras to track people and know presence information at any time in the different spaces. At the same time modern illumination systems use presence information to reduce illumination energy consumption in environments where there is no one. The two systems use the same type of information, provided by different sensors but are not able to cooperate and so happen that the sensors are replicated. Often cameras are not installed everywhere because of costs, while illumination presence sensors are based on cheap technologies (e.g. infrared sensors) which are put everywhere; compared to cameras they loose the images(not needed for illumination) but keep the presence information. The cooperation of the two apparatus could provide a more precise security system, informed also of the presence in environments where camera are not installed and a cheaper illumination because where camera sense presence there is not the need of infrared sensors.

The data of different management systems can also be merged to obtain new more useful information: often temperature sensors are connected in direct feedback with acclimatizers, but the temperature perceived by humans depends also on humidity. If the humidity and temperature informations are considered available from a system aimedation to provide more comfort, is possible to use environmental actuators to control perceived temperature instead of simply temperature. Illumination, climatization and surveillance systems are able to provide an even more comfortable environment if they know how many people are in a certain environment, and for this purpose information about presence, from the security or illumination systems can be used. If an office in a large building will be not used for one day i.e. because the tenant is hill, there is no need for illumination or heating, but to provide this service

interoperability with data regarding the tenants status is necessary. More similar examples can be found in this scenario and new ones are identifiable in different scenarios; Fig. 1.2 resumes these concepts showing a comparison between the majority of current systems(closed applications) and the ones based on the context intended as a whole. It is worthwhile how, besides the previously stated advantages, coordination and cooperation between different solutions are possible in the new framework in a controlled way by using the knowledge base as an information exchange point. The different management software doesn't need to be developed by the same teams and doesn't need exchange of code, but the only effort to create communication between totally independent systems has to be given only in their interface with the knowledge base(i.e. a standard portable interface). During this work I'll represent and make interoperable many kinds of information from different sources by using methodologies born for this purpose many years ago but still not so much applied by research or in commercial systems: those of the Semantic Web[5]. All of these information constitutes the interoperable context (or knowledge base) available to the software agents aimed to provide services or to increase the knowledge base itself.

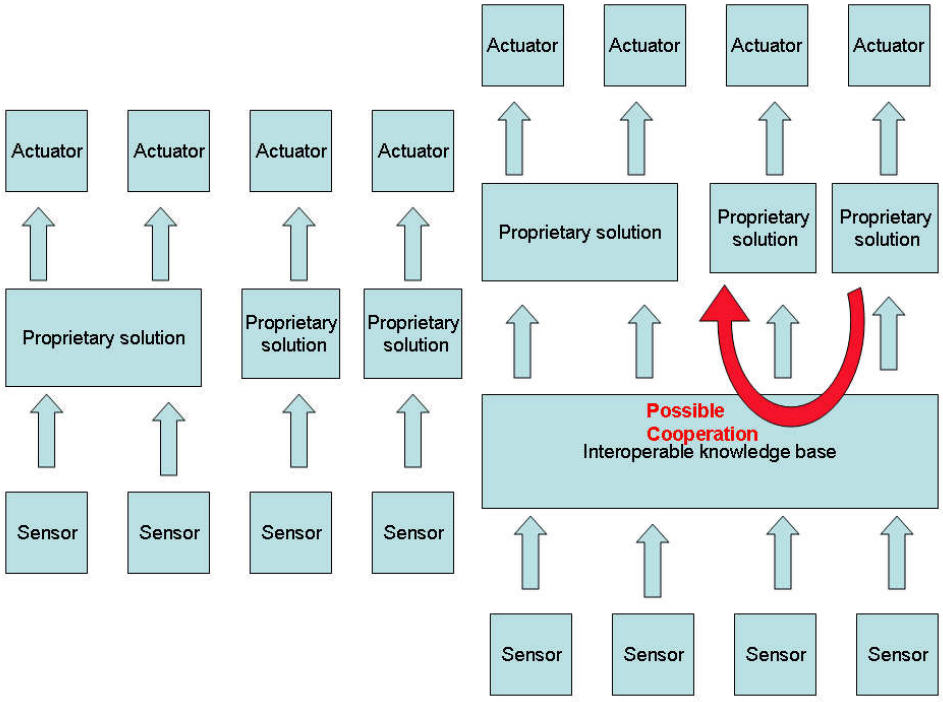


Figure 1.2:Typical management systems (left) versus a possible context aware solution

1.2.2. Information layer

The information layer has the objective to manage information and fulfills a certain set of requirements in order to be the center of a smart environment. The possible requirements a scenario needs to be respected by its information layer obviously depends from the scenario itself, but considering

common situations is possible to list the most relevant features of the information layer software management

- Connectivity: more protocols are available to interact with the knowledge base, more devices are capable of sharing their context and taking advantage of the information on the context infrastructure. A unique protocol is feasible but it should be as simple as possible because devices not natively supporting it must be always adapted to the information layer.
- Information representation: The information should be represented in a way that allows machine interpretability or continuous work must be done by developers to provide the meaning of the stored data to applications. Semantic technologies for data representation help in achieving this result as it will be discussed in section 2
- Performance: A context platform should perform its primitives as fast as possible in order to satisfy requirements of a larger amount of scenarios e.g. in telemedicine where sensor observations are often fast in order to provide more information about user health. Performance is a wide argument when dealing with smart environments and should be considered under different metrics[6] e.g. access in RW mode, loading a context made up of many data to initialize the system, degradation in condition of high traffic etc.
- Subscription notification: a common pattern in smart environment is the reaction to particular contexts. When a certain situation happens, expressed as a function of available context attributes, software agents perform the relative tasks. Subscription notification mechanisms natively supported by the context platform avoid continuous polling of context data diminishing the global traffic of data.
- Portability: a context platform which runs on many Operating systems and Hardware architectures allow more flexible smart environment construction
- Distribution: in a distributed context platform the information is not on a single physical location, but can be considered as a whole. In general this is an advantage, but its management is complex.
- Discovery: is important that applications are able to discover context repositories in order to interact with them in evolving smart environments.
- Persistency and reliability: the data should persist also to system failures and information should be always reliable
- Security and privacy: in certain scenarios is fundamental to avoid system intrusions, malevolent data corruption, or privacy breaches.

In the following I'll consider some context platform encountered during my PHD and I'll underline the most important differences considering the list above. A better understanding of context platforms and the

capacity to measure them basing on objective parameters, in order to be able to do the best system choice respect to the application requirements, can be seen as one of the most important outcomes of my PHD.

1.2.3. Service layer

Services are here intended as software entities with access to the information world in order to satisfy user needs. Lot o work has been done to provide an advanced service layer and many technologies exist to access and manage services. REST (REpresentational State Transfer)[7] is a set of principles defining how to address and use resources with simple HTTP. XML RPC (Remote Procedure call)[8] is a protocol to execute remote call to local services by using the internet. CORBA (Common Object Request Broker Architecture)[9] is a standardized mechanism for the realization of distributed service oriented Systems.

Many other relevant technologies could be cited, among which one of the most important and used is that of web services based on service registry, a markup for service definition language and a protocol to interact with service. UDDI, WSDL and SOAP[10] are the technologies trough which these kind of web services are published, discovered, described and utilized in a transparent way by end users. UDDI (Universal Description Discovery and Integration) [11] is a service registry i.e. an informative system providing access to web services offered by business companies. WSDL (Web Service Definition Language) is an XML based markup language to describe web services. The existence of a standard and a machine readable format for web service definition is very important for disseminating services by making simpler their usage through automatic code generators like [12]. In order code generators to simplify access to the web services is necessary not only to describe the web service, but also to have a standard way of accessing to their functionalities, to make code generators able to create stubs hiding all the low level details of service access to developers . In this respect the SOAP (Simple Object Access Protocol) protocol has the role of defining how messages should be exchanged between client and server with simple XML based messages.

Also the service layer is affected by the Semantic Web. Substitutes and evolutions of UDDI or WSDL have been proposed like for example a SUDDI module in [13]. The SAWSDL (Semantic Annotations for WSDL) [14] is currently a W3C recommendation. The service layer is not in the scope of this thesis, which is more focused on the information layer and on architectural solutions for making the device ecosystem possible, so I'll not go into further detail with services and their related technologies.

1.3. Context representation

One of the important issues to be faced in the information layer of context aware applications which need to be discussed in this work is that of context representation. Context representation may also be used as a way to categorize context aware systems; many approaches exist in literature. What emerges is that different approaches are preferable for different applications needs. Key-Value pairs are simple to be used and viewed and are good for service frameworks like [15] where the description of services is made through a set of capabilities which can be supported or not. Object oriented models for context representation are more complex, but when set up they can exploit features like encapsulation, reusability and inheritance. When using object oriented context representation the access to the context is provided through object interfaces, like also the possibility to perform context processing. When the information we want to represent is a profile like CCPP (Composite Capabilities Preferences Profile) and UAPROF (User Advanced Profile) for device profile and preferences, the markup scheme based model are a good choice. The markup models uses hierarchical data structures made up of markup tags, attributes, values and content; scheme languages like XML-schema and Relax NG are used to create schema definition for validation. Logic based models are formal models of context representation made-up of concepts, facts, expressions and rules. These models allow for inference, in order to derive facts not explicitly stated by reasoning on the formal properties of the logic considered. One of the newest context representation approaches is based on formal domain descriptions called ontologies. Ontologies are made up of Classes, properties, instances and statements; in most cases ontologies are based on an underlying description logic which offers sufficient formal properties to make reasoning like in logic based models. Since ontologies are also the semantic web way of representing a domain description, ontology based models for context representation will be the principal case of study in this work.

Depending from context representation techniques, the level of interoperability provided by a context framework drastically changes. If the naming convention used to represent context is totally free, the alignment needed to attain mutual understanding of independently developed software components could be an onerous operation from the developer's perspective. Hereinafter I'll focus on a context platform with a RDF based knowledge base: this context representation technique which is in strict relationship with the ontology based context

modeling introduced above. RDF is made up of triples corresponding to logical facts, also if RDF doesn't requires the formal rigor of logic models and so reasoning capability is limited. An RDF knowledge base always corresponds to an oriented labeled graph favoring the application of techniques and software component based on graphs like graph databases[16] and specific strategies for querying[17] and indexing[18]. RDF is also one of the firsts and more stable components of the semantic web pyramid, so applications based on RDF can be also considered under an ontological or a semantic web based perspective.

SECTION 2

RELATED WORK ON CONTEXT PLATFORMS AND SEMANTIC WEB

Context platforms are middleware solutions to manage the context i.e. the relevant information necessary to provide smart services. These solutions are under constant development both from the point of view of the functionalities provided and of the performance. Here I shortly describe two context management systems encountered and used for smart applications of different nature. Then I'll go into detail when describing smart M3 that will be the reference context platform in this work.

2.1. Mobicomp

Mobicomp[19] is a context platform for context aware applications developed at the university of Kent for supporting applications related to cultural heritage and archeological sites. It consist of a constext store that is suppossed accessible and by software modules accessing to it in order to utilize a shared knowledge base.

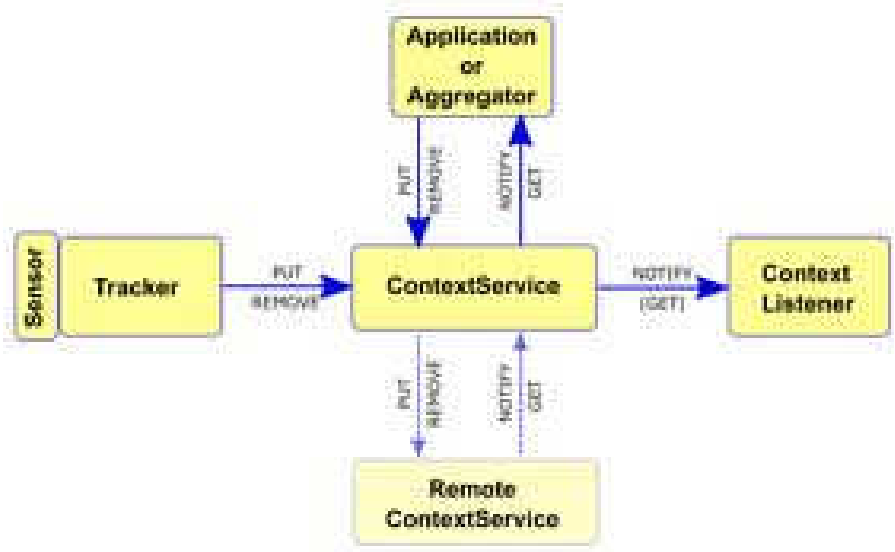


Figure 2.1: Mobicomp software architecture

Figure 2.1[20] shows the software architecture: software the agents which only puts information on the context service are called trackers, their counterpart, whose aim is only to read information from the shared knowledge base are the listeners, while a software agent able both to read and write to the store is called aggregator. Trackers are typically used to insert raw data from sensors on the store, listeners can make use of subscriptions in order to be notified of relevant context changes while aggregators are used in applications which need both to read and write o the store. The information is represent edas tuples of five elements: subject, predicate, object, privacy and source The context modeling approach is simple and expressive: information is intended as a set of triples related to "entities" an abstraction univocally identified by an ID. By knowing the ID and the properties of them is possible to construct also complex applications in which sensors and other trackers put new properties or updated their values with newly observed values while listeners use the information to allow context aware application logic. Sometimes, the application logic needs from the context store high level information that is not observable by sensors and that could be onerous to calculate on the run every time is needed. Aggregators solve this problem by performing the calculation of the high level context attribute and storing it in order to be used by all the listeners that need it. In a certain way these kind of aggregators can be seen as a kind of sensors and sometimes substitute them, so they also can be called virtual sensors.

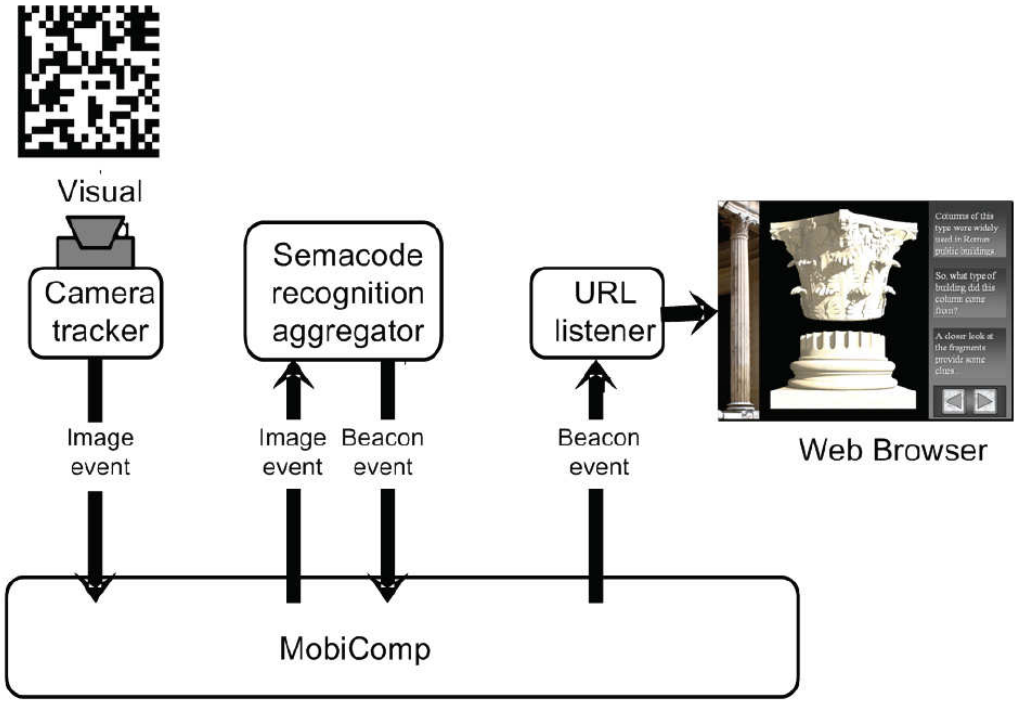


Figure 2.2: Mobicomp based smart space application

In Fig.2.2[21] is represented the information flow of part of the work presented for the final event of the EPOCH network of excellence [22] in 2008. As is possible to observe from the figure there are different kind of information source i.e. Semacode identification data, and camera events. This information is represented as a set of statements related to a certain set of entities decided by developers, according to the Mobicomp philosophy. Once all the software elements works following the programmers conventions it could be said that the context is really shared and the context aware applications can work properly. End users where provided by wearable ad hoc device performing pedestrian tracking basing on accelerometers and gyroscope values. The worn device also had a touch screen display, a GIS model of the museum, accelerometers and gyroscope for pedestrian tracking and to allow assisted navigation. Descriptions of the artwork resides on a different location and are retrievable by user devices knowing their ID. In the final applicationit was possible to provide the description of the piece of art whose semacode is read by the user, to assist navigation towards an artwork indicated by the user and, in general, to increase culture visibility through the help of electronic. In the described application the museum has become a smart environment in which multi source sensor data where fused to give advanced functionality; the constantly growing localization error, typical of pedestrian tracking made up with accelerometers, was reduced by the short range identification techniques, in fact since the semacode (but also RFID if needed) were readable only near the identifier location, the instant in which they are red correspond to the moment in which the user is in the precise location in which the semacode has been mapped; thank to this in this application has been possible an interesting data fusion allowing to use in symbiosis the pedestrian tracking with correction made up through the reading of identification codes. As I will deepen later, many step forward can be done starting from this situation and we will see that many of them imply the change of the philosophy with which the context is represented and structured. What we have described is in fact a system developed with internal protocols for communication and user defined context definition. Extensions or delegation of part of this work to third parties is difficult because is necessary to clearly explain the role of all the entities (usually human unreadable UUID and their properties, for which the semantic resides in the naming convention).

2.3. Context aware platform CAP

The context aware platform or CAP [22][23] has been realized by telecom italia lab, TLab to efficiently manage user data in order to offer advanced services. One of the most important motivations of its creation is the need to manage a large amount of data and events in a single framework. As previously mentioned, in fact, the increasing number available sensor data and the details about users (e.g. personal profiles, preferences, etc), constitutes a so large evolving knowledge base that is difficult to manage

efficiently. The high level software infrastructure of a CAP based context aware application is showed in FIG. 2.3.

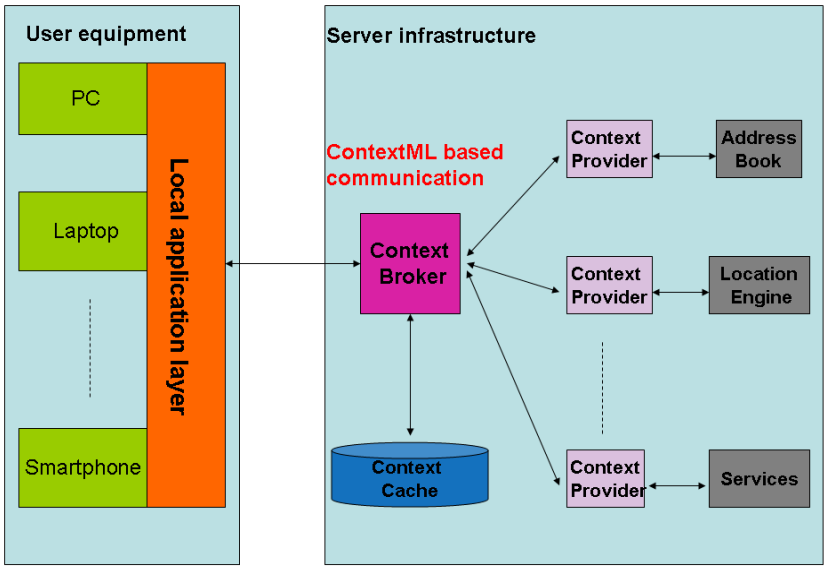


Figure 2.3: Big picture of TLab context aware platform based applications

User equipment and , in general, all the smart devices able to contribute and/or use context, run applications able to communicate with the *Context Broker*. The context that is inserted to the shared platform is sent to the *Context Cache* in order to be retrievable by interested agents. The *Context Providers* are able to perform processing to obtain abstract context like for example a integrated location based on multiple data sources(e.g. GPS, Cellular cell seen by the device, WiFi, …) so that when complex attributes are requested by applications, the broker asks to the right provider to use its resources to calculate it. All the communications are transported through a proprietary XML based protocol named ContextML[24] which is able to carry context attributes in the form of one or more *context scopes* Elements. *A context scope* may be atomic or complex: while atomic context scopes are the elementary context element which can be understood by the architecture, the *complex scopes* are composed of one or many atomic scopes. In this architecture the semantics and machine interpretability of context is higher than in Mobicomp because there has been an effort to formalize context and to represent it in a standard way. The standardization of context is done at infrastructure level and not by the single application developers, so the level of interoperability is higher. The naming convention is not free because there exist a clear vocabulary to express context and this favorites the development of context aware applications which are now based on a complete domain description. Problem arises when new functionalities are needed based on context attributes still

not formalized in form of scopes or of their composition. If extensions or evolutions of current services require new scopes, there could be a considerable effort to modify the existing ConteXtML version and the related software modules in a coherent way. When new conteXtML versions are going to be released it should be reasonable but not trivial to guarantee at the same time simplicity in the parsing and backward compatibility. But the real obstacle to the realization of smart environment seen as an ecosystem of interactive devices is the monolithic approach. The context representation and interpretation are conventions internal to the business company which owns context platform and provides the context aware services. The ubiquitous computing vision doesn't clash in principle with this model, but as we will see when dealing with the smart-M3 context platform a collaborative multi-industrial scenario with machine interpretable context definition and with interoperable modularized components is currently more concrete. Despite the validity and power of context platform like the CAP, it will be not the reference platform of this work because it is aimed mostly at mobile services and not to a pervasive smart environment scenario.

2.4. Smart M3

The smart M3 context platform perfectly suits the role of being the context store in a pervasive smart environment and to provide semantics to information, context reactivity, and a multiparty business model. The development of a context platform like Smart M3 has been the objective of the FP7 ARTEMIS project SOFIA[25] lead by Nokia which started in 2008 and ended in 2011. The opportunity to participate to this important European project allowed me to have close contact with semantic smart environments related technologies and with many industrial and academic partners interested in the development of vertical applications from raw sensor data observation to complex service erogation.

2.4.1. The big picture

Fig 2.4. represents SOFIA general vision of context aware smart environment. The SOFIA specific technologies are mapped to the previously defined general layers of a context aware applications showing that the attention is focused on the transport layer and on the information layer.

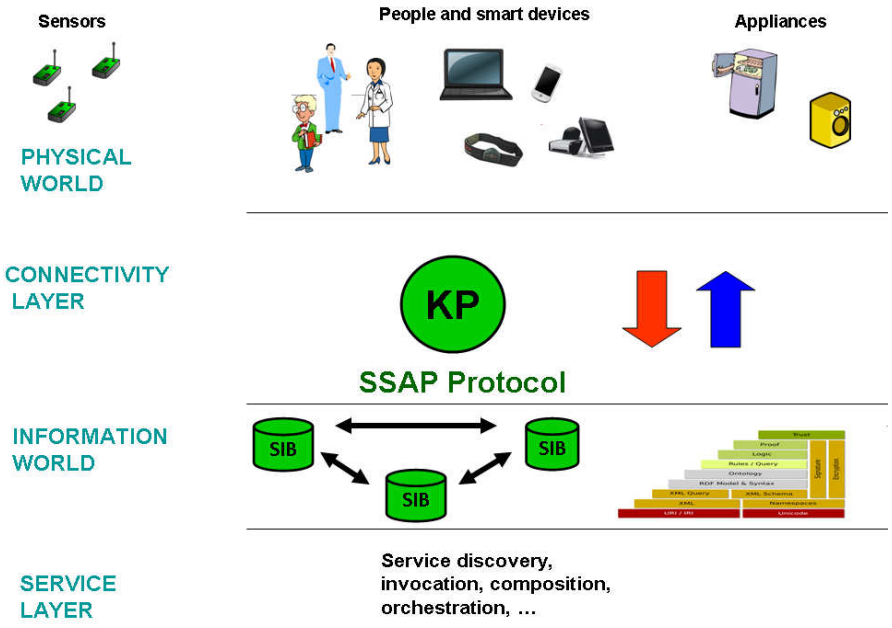


Figure 2.4: Smart environments in SOFIA vision

The *KPs* (Knowledge Processors) are generic software programs able to communicate using the information transport protocol called *SSAP* (Smart Space Access Protocol). The SIB (Semantic Information Broker) is the context platform internally exposing data through Semantic WEB representation techniques. The SIB is a specification more than a specific software solution, augmenting the level of generality without loosing contact with the SOFA vision we can say that a SIB is any hardware/software component able to properly manage and respond to SSAP requests. The semantic web (indicated in the figure with a miniature of the well known pyramid) plays a crucial role in the vision it provides ., in fact, a way through which representing data in the sharing knowledge base. Domain ontologies, realized through semantic web standards(i.e. OWL sublanguages), are a way to share a common terminology and domain vision between the active software agents. The service layer, bases its view of the context on raw and abstract data provided respectively by sensors and aggregators. Also for the service layer is very important a formal base built upon ontologies. Context reasoners are able to perform their task tanks to consistency and the time needed to perform a query is always finite if the underlying description logic is decidable. Also rule engines can be built on a ontology based view of context an well know existing solutions like Jess[26] can be applied to derive abstract context interpretations, to apply services compose them or to manage a proper service orchestration in order to better suite user requirements. From a more direct perspective the SIB (or sib aggregation) is the center where all the interested software agents (KPs) Store or query

information to perform their tasks, KP concurrency should be properly managed in order to avoid synchronization problems and by allowing transactional services.

2.4.2. SSAP Protocol

SSAP is the key integration point in the smart m3 software architecture, its is the protocol used to carry context data between the devices and the central knowledge base. Fig. 2.5. shows the graphical representation of an XML-schema designed to describe SSAP protocol. SSAP primitives are implemented by libraries called KPI (KP Interface) which are written for many programming languages like C[27], C#[28], Java[29], and others . If necessary new programming languages can be supported by the implementation of new KPI. The existence and the usability of KPIs is very important to hide from developers the low level details of protocol implementation.

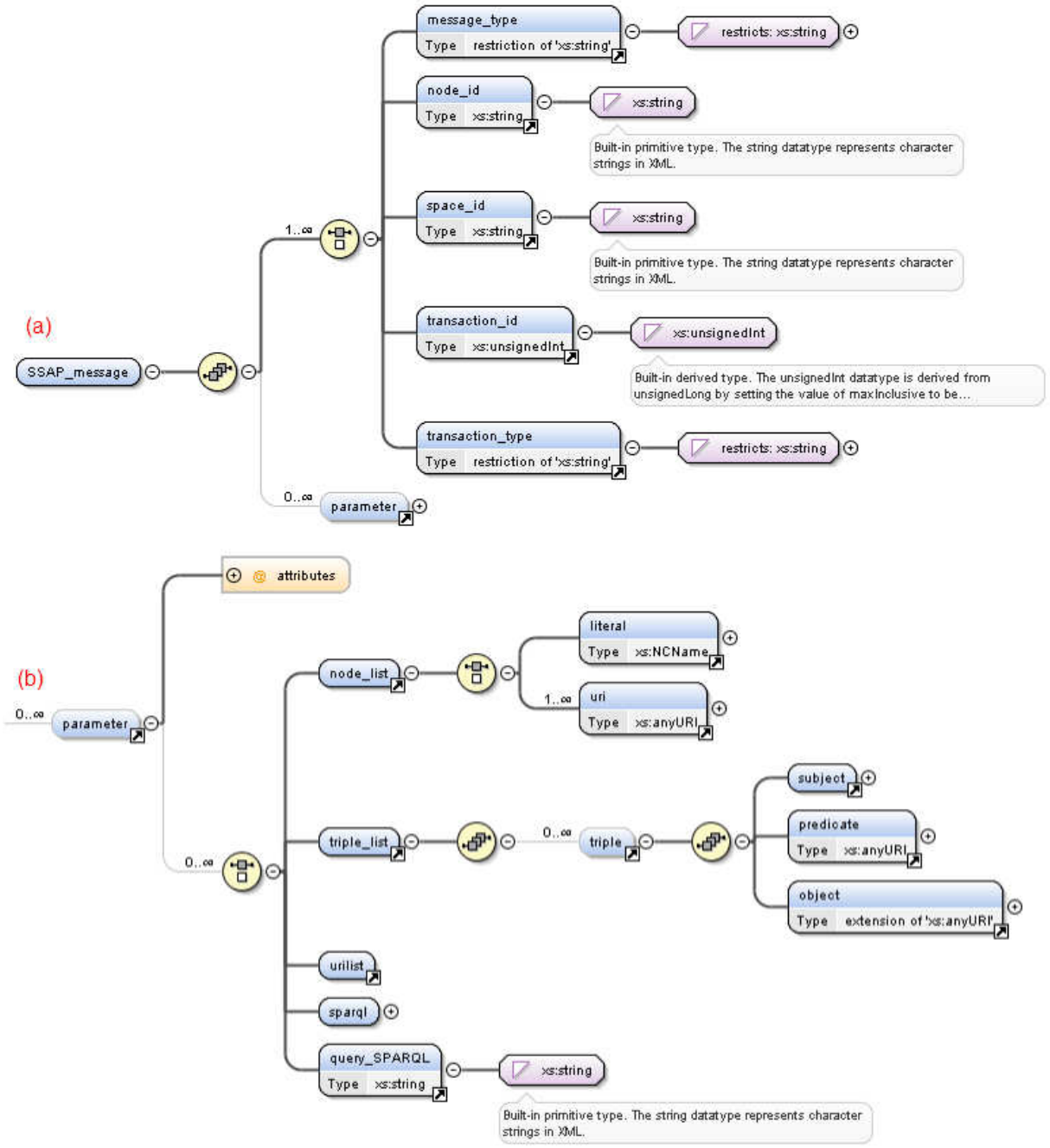


Figure 2.5: SSAP XML Schema graphical representation

The XML schema in the figure as been obtained from the analysis of a complete set of valid SSAP messages and then refined by hand to obtain a better match. In (a) the composition of a general SSAP message is showed. In particular each SSAP message is composed by an heading part and a set of parameters. In the heading part are elements and attributes stating the

nature of the message (i.e. *message_type* and *transaction_tipe*) and details regarding the KP (i.e. node_id)and the SIB (i.e. *space_id*)receiver. The transaction_id element specifies an incremental integer counting how many access have been done to the SIB by a certain KP. In (b) the *parameter* element is expanded to show the possible valid parameters which can follow the heading part: the node list element may contain literals or URI representing graph nodes. The SPARQL query is a string while the response is an XML document compliant with the W3C recommended XML format for query response[30]. The triple list element contains the set of triples to be inserted or removed in write mode while in query mode it contains the triple patterns used to perform queries on the shared graph. The possible SSAP messages can be classified in three main categories:

- Messages to specify that a KP wants to start or end interaction with the SIB. The primitives under this category are the *JOIN* and the *LEAVE*. Each KP can perform other operations only after Joining the SIB, the LEAVE primitive states that the KP has ended its interaction with the SIB, so, to perform other operations, is necessary to send a new JOIN message.
- Messages to access to the graph in read or write mode. The primitives to modify the content of the KB are INSERT, REMOVE and UPDATE. The former two respectively allows a KP to insert or remove triples from the store while the UPDATE primitive performs two consecutive remove and insert operations by a single primitive in which is possible to specify both the set of triples to be removed and that to be inserted. The triples can be specified in many different formalism for flexibility reasons: while the *triples* XML element is restricted to the RDF-M3 way of representing triples, all triples syntaxes are theorically valid from the schema and in particular the W3C recommended XML-RDF based syntax. Also if all triples specification syntaxes are possible, different SIB versions may support one or more of them, so the SIB profile concept and its interactions with the KPI libraries becomes very important.
- Messages related to the subscription/notification mechanism. Subscriptions specifies through triple patterns the intention of a KP to be notified of certain events at information level by the SIB. Notifications are messages sent by the SIB to subscribed KPs and specifying which triples has been added or removed from sub-graphs matching the patterns. Subscriptions are very important to improve smart environment capabilities by adding context reactivity without continuous polling on the knowledge base and so with a sensible traffic reduction.

2.4.3. Legacy devices

When the SOFIA project started and the smart M3 concepts where introduced there was a clear idea of not making a system totally new that was not able to be deployed effectively on market, nor to lose the impact on research with an effort not directed toward innovation. The real objective was to put the basis of an evolutionary revolution built on existing technologies, deployable on existing devices and appliances, but at the same time ready to be used in completely new applications. To perform this task the methods to interface the existing legacy devices to the new conception semantic knowledge base need to be simple and deployable on a wide range of devices, in particular that with limited resources. Fig 2.6. [31] shows how different kind of devices can be properly made part of a smart space. The adapters KP have been conceived to fill the gap between existing legacy devices and the information world. They send to the Knowledge base all the relevant information regarding the device status and they query for the external data to which the device is interested in order to better adapt its behavior to the evolving context. As previously stated KPs definition is very general: they are programs of whatever nature able to communicate with a SIB through SSAP protocol. The KP are technology independent and also their platform or the language in which there are implemented are a free choice. An adapter KP transforms the original internal conventions in ontology based ones and this operation make the exchanged information universally interpretable by software agent based on the same ontology. For the adapter Kps always two interfaces can be taken in consideration: *the legacy interface* is responsible for taking the raw data while the *smart space interface* aligns the information to the ontological reference and send them to the shared KB. For programmable devices, see (a) in figure, is possible to write the adapter KP on the device itself by transforming it in what we can call a “Smart Object” (SO). For many reasons is not always possible to change the way a legacy device behave and is necessary to run the adapter KP on a different host which is interconnected both to the device and to the smart M3 infrastructure. The situation represented in (b) happens, for example, when we want to “adapt” a proprietary sensor network. In this case the legacy should be able to share their context with the central Knowledge base, but at the same time is not possible to change the proprietary software and protocols. The simplest solution is so to run the adapter on the SIB host: from the legacy perspective nothing changes and the adapter is simply an application requesting its data, however, from the knowledge base perspective the chain legacy-adapter is effective and so the legacy has become a smart object. Solution (b) can be applied when the not programmable legacy has a stable connection with the SIB host and cannot be applied in mobile scenarios like that involving wearable sensors. The solution (c), has been successfully applied and demonstrated in [32], it

uses as host for the adapter KP in a programmable wearable device like for example a smart-phone. The legacy interface is based on the existing protocols supported by the legacy (e.g. Bluetooth or Zig- Bee) while the smart space interface has the additional task of discovering the smart space infrastructure in order to communicate the data in all the places visited by the adapter host holder. Situation d has been identified and described during this PHD. It is a particular kind of smart object typical of modularized hardware architectures: from an high level perspective we see a smart object communicating with a knowledge base ontology referenced data, but going into the details of the smart object only one module is programmable and can be the host of the adapter KP. Other modules of the smart object are not programmable and reasonably provide data in proprietary protocols pushed toward optimizing. To expand the SOFIA benefits in term of application development time, reusability and uniform programming approach also to this kind of legacy device the low level adapters (indicated with the yellow D letter in Figure) have been introduced into the scenario. These modules have still not a definitive specification because they are not part of the original vision, but they solve a specific need. The aim of the low level adapters is to have reusable code in the Adapter KP running on the programmable module of the modular legacy architecture. Usually, all KPs are reusable and also adapter KPs because their legacy interface is defined by the existing stable legacy device, while their smart space interface is reusable thanks to the stable ontology reference. This doesn't apply to the modular smart object because the nature of a modular device is to have the possibility to interchange modules with different features but offering the same functionalities. A clear example is that of a smart object which needs gesture recognition from an internal not programmable module. This not programmable gesture recognition board, in origin is interfaced with the programmable module with a certain protocol and conventions. From these conventions and protocols depend the adapter KP legacy interface for which we want an high level of re-usability. If we want to exchange the original gesture recognition module with another one, the adapter KP (or better its legacy interface) has to be rewritten from scratch. The low level adapters are an attempt to solve this problem by defining the low level semantic and access methods. Also for low level two interfaces are defined, the *Adapter KP interface* is stable and provides ontology referenced attributes from a static interface of access methods. The *external module interface* is aimed to the interaction with the specific module. Considering the adaptation to a smart environment of a modular programmable device, low level adapters, in my vision, could be the only pieces of software to be modified in case of internal module substitution by keeping the core logic of the adapter unchanged and simplifying the whole development process.

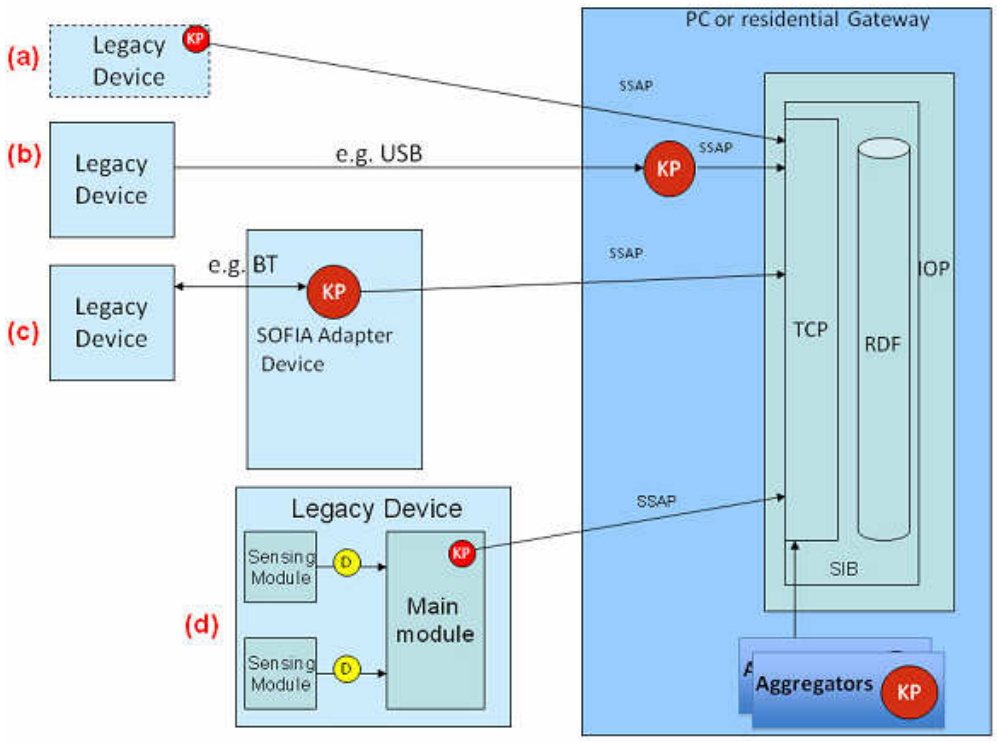


Figure 2.6: Different ways of adapting legacy devices to a smart M3 based software infrastructure

2.5. The Semantic Web

The semantic WEB activity is a collaborative movement promoted by the World Wide Web Consortium (W3C) to migrate from the current WEB of text to a WEB where information is accessible by software agents. Starting from the concept of semantic network defined since the sixties (e.g. [33] where multiple resources are connected with univocally defined properties, the semantic WEB term was coined by Tim Berners lee in [34] as a natural evolution (and not revolution) of the current WEB. Currently all the information accessible in the Web is in the form of static or dynamic web pages written with markup to show them in browsers. The HTML language provides a way to present a web page to the user, but does not allow, in principle, to make the meaning of content of the page accessible to software agents. This problem is faced through the use of search engines basing their behaviour on meta-information present in the web pages and on their textual content to run proprietary algorithms which find and rank results in

response on a textual user input. Search engines can be seen as the glue of the web but their behaviour is not the best possible one, actually most of web search engines are based on keywords, and is the user who have to use them properly and to correlate the results in search sessions to retrieve the resources to which is interested. Is unthinkable to delegate a software agent to solve this task because relatively simple objectives like “finding the work of art of Leonardo da Vinci” can be done with different search procedures, with different use of keywords and with navigation in the result pages. The results given by the most powerful search engines, even Google, are full of not pertinent sites and duplicated or similar answers. When this phenomena is less evident is often because workaround on results allow to hide unwanted items, but the problem is at the root and so in the key word based search. In the Semantic Web vision Resources are linked through semantically defined properties, so, it would be very simple to ask to a software agent to search for the resources linked with the resource “Leonardo da Vinci” through the property http://purl.org/dc/elements/1.1/creator.

The property used herein is part of the Dublin core initiative [35] and is so universally known. In order the example to work also “Leonardo da Vinci” must be a univocally determined resource, and not simply a string; as we will see the concept of identifying each relevant resource with a URI (Universal Resource Identifier) is one of the key features of the Semantic Web. An example of semantic Web engine related with arts won the Semantic Web context in 2006[36]

Search engines are only a clear example of a well known web application that can be improved by a different organization of web resources), but are not the only example and, moreover, there are totally new applications that can be built in a new web and that can positively impact the market and the user satisfaction. The Semantic web initiative is one of the most import undergoing projects to the establishment of a framework suitable for smart environments and context aware services. Here I’ll describe not going into too much detail, the affirmed Semantic Web technologies the ones that are still under discussion or under construction, and some example of their applications in research. The semantic WEB activity is often represented in a layerd graph similar to that reported in Fig 2.7.

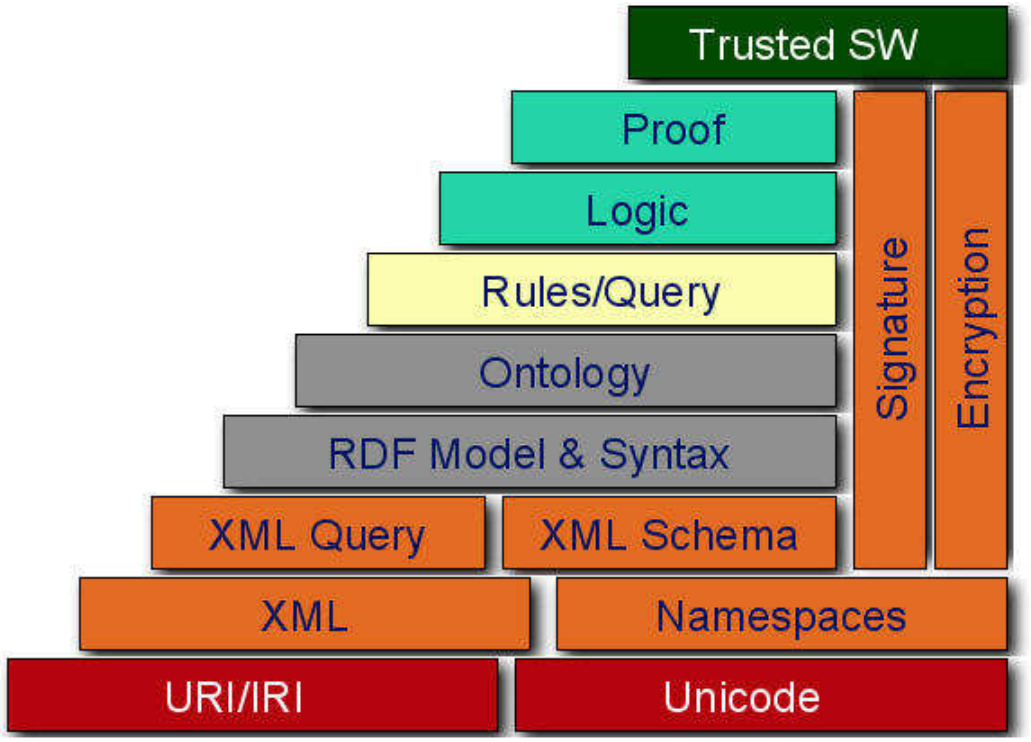


Figure 2.7 Semantic Web stack

The reason why the graph is similar to a pyramid is very important: since when it was conceived when the Semantic Web was so distant from the affirmed Web of that time, that its development could have required also many decades, and its deployment over a consolidated but different version were very improbable without a proper strategy. The idea was:

- to build the new web starting from standard technologies, i.e. XML and Unicode, not implying any interference with the existing and florid web. This is the evolutionary revolution introduced before, in fact XML and its derived standards were just omnipresent in the web.
- To make it step by step in a way that for each step was possible for the community to begin to "feel" the added values and so to have the will to use the new standards until the critical mass for that step finished.

As is possible to see in the semantic web pyramid there are a lot of technologies, currently the level of ontologies is stable, or at least clear recommendation exists for it, the RIF (rule interchange format) dialects have been released for rule representation and translation, but still there is ongoing effort in the integration between the rule layer and the ontology layer[37],[38]. The other steps towards the completion of the Semantic Web project are still under construction and will probably require many years.

2.5.1. XML

XML (Extensible Markup Language) is the first and simple step toward machine understandability of information. Through a nested structure of tags and attributes it is possible to build a tree based model of information. The syntactic rules of XML are very simple: there are no reserved tags and there is free naming of Elements. An xml document consists of a declaration line opening tags, closing tags, attributes and text. The xml elements begins with the opening tag (delimited by "<" and ">")and end with the corresponding closing tag (delimited by "</" and ">"). Each XML Element may contain other elements (i.e. nested elements or children), attributes or text. In each XML document there is only one root element which contains all the information of the document in the form of children, text or attributes. The attributes are name-value couple specified in the opening tags after the name of the relative element. The attribute name is followed by the equal sign and the attribute value either between single or double quotes. The attribute with name "xmlns:prefix" is special because it allows to specify namespaces with visibility in the element in which it has been declared and in its children. A namespace declaration assigns a prefix to the value of the attribute in a way that inside the element is possible to use the prefix followed by ":" as a shortcut. The namespace with no prefix is called default namespace: each element that have an ancestor which have specified the default namespace will belong to that namespace. Elements must be properly nested and so if *tag1* is opened before *tag2*, then *tag2* must be closed before *tag1*. With these basic and other less relevant rules XML documents correspond to trees where the root corresponds to the first element that is opened, the branches correspond to nesting elements or specifying attributes and the leaves are the values of the attributes or the text contained in the XML elements (Fig 2.8.).

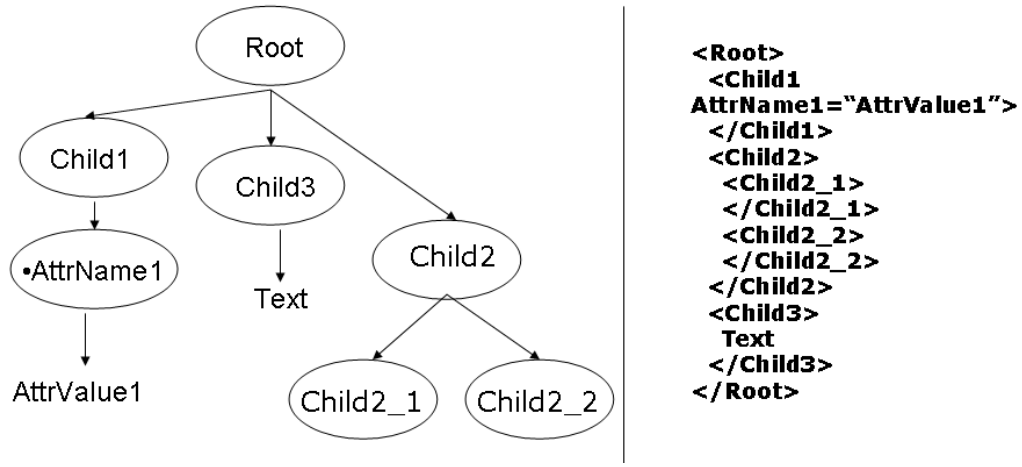


Figure 2.8 Example of XML Elements and their corresponding tree

An XML parser is a software written to read and partially manage the information contained in XML documents. Many approach are possible to parse an XML document: but typically SAX parsers and Dom models are used. It is commonly said that XML has very limited semantics, this happens manly for two reasons:

- Since the name of the elements are freely decided by software developers, there is no way these names can be understood by software agents without human intervention.
- There is no implicit meaning in the nesting of tags. [39]. Tag can be nested for many reasons so to specify different kinds of semantic relationships between parents and children. One can nest a full name and city to specify that the person with that name was born or is located or loves that city. At the same time is perfectly valid to nest a city name and a full person name to specify the same things.

XML is machine readable and so is very powerful with respect to plain text, the use of schema definition languages like DTD[40], XML schema and Relax NG [41] allows to reduce XML natural freedom and to create sublanguages valid in the context of specific application. Schema definition languages also make possible to build protocols between applications, to validate generic XML documents against the schema and, consequently to collaborate between distinct gruops with the schema acting as a reference point for information representation or transport. Despite the good features introduced by XML, the problem of the lack of semantic is crucial when we aim to autonomous semantic agents. Since the issues described before are structural and depend by XML itself (we can say that the biggest problem is the fact that the arcs

connecting the nodes in the XML tree are not labeled), the Semantic web builds over the layer of machine readability a layer that can be though as the first step towards semantic understandability: that of the Resource Description Framework (RDF).

2.5.2. RDF and RDF Schema

RDF perfecltly addresses the semantic issues introduced for XML. First of all RDF introduces the concepts of URIs to identify resources, the naming convention is still free, but the names created are unique and there is no possibility of misunderstanding same names given with a different meaning by independent developers. Applications built over the same set of URI will interoperate. With the introduction of URIs there is an advantage also versus schema definition languages because the scope of URIs is global while that of schema definition languages is local unless a namespaces is given for that schema. The other issue present in XML and solved by RDF is that of the meaning of the nesting. In RDF is not important the syntactical form with which the nodes are connected together, but the only important thing is that the information is specified in the form of triples <subject, predicate, object > or < s, p, o > where the subject is and the predicate must be identified by URIs while the object may be a URI or also a literal value of any of the basic types. This information organization results in a directed labeled graph, a more powerful model than the XML three. The great advantage is the obligation to identify the property with a URI to clearly and univocally determine the semantic relationship between the subject and the object. This identification was totally absent in XML where there was only a syntactic nesting between parents and children. As previously introduced RDF is not properly a language, but represents an information model independent from the specific syntax used to serialize it. For this reason many syntaxes exists for RDF like Turtle, Notation 3 and RDF-M3[43]. According to the semantic web stack showed in Fig 2.7. the syntax to be used for RDF in the semantic web and that is recommended by the same W3C is based on XML itself and is called RDF/XML based syntax for RDFHuman readability is low but at the same time the level of machine interpretability is high because all resources and their relationships are univocally identified. The directed labeled graph implied by an RDF model corresponds to the model of information of semantic networks but in principle every construct is feasible with given resources, what is used as predicate in a statement can be the subject of another, properties naturally functional like the one between a person and her fiscal code are not enforceable because too much freedom is given. RDF gives the right means to obtain semantics in application, but too much freedom is given to developers in order to allow automatic reasoning and inference. Inference is fundamental when not all the statements in an knowledge base are

supposed to be knew and when we want software agents to use information about the domain to have a smarter behaviour. RDF Schema partially addresses the need for a terminological and conceptual basis in domain definition. In RDF Schema are added the concepts of Classes and Properties: the subject or the object of a statement must not be properties which instead are used to connect the former two. Properties are also provided with domain and range definition. The domain referred to a certain Property is a Class or Class expression in term of union intersection and other set operators, defining from which set of entities a statement containing that Property may start. Analogously the range of a Property defines, through set operators on defined Classes, the superset of all the objects for that properties. An example of still primitive inference that is possible to obtain with RDF Schema is the grant that subjects of statements with a property that has Domain in class A, must belong to A, also if this assertion is not explicitly stated in the knowledge base. The objectives of the Semantic Web in terms of reasoning and use of domain structure in the process of inference are more advanced than the simple example just exposed, but more rigor is necessary in order to allow this. The ontologies and the OWL sublanguages have been thought to give a powerful instrument with basis in the logic theory to define the concepts an the properties of a certain domain.

2.5.3. OWL and Ontologies

OWL (Web Ontology Language) is built on RDF and RDF Schema as they are built on XML, according to the semantic Web big picture. It is not a syntactic, but a conceptual relationship which binds RDF with OWL, in fact many syntaxes exist to define an Ontology. An Ontology can be roughly defined as the specification of a conceptualization and it represents the terminological and structural definition of a domain of interest. In general terms the software agents aware of the ontology (or ontologies) they are using may perform inference and will create statements without breaking the domain consistency. OWL allows to define not only Classes, Properties, their domain and their range, but also other important features like Property cardinality, functionality, transitivity, chains and so on.

As previously mentioned the reasoning capability provided by OWL depends from the formal theory of logic and, precisely from description logic [42].A reasoning procedure is a computationally complex task for which one of the most important metrics is the decidability. A logic (and so a subset of ontologies) is decidable if each possible reasoning procedure will end in a finite time. Too much freedom in domain definition language means no deductive power or impossibility to construct the basis data structures needed to perform reasoning. This is the

problem of all that languages seen until know. Too much freedom is given in XML, but also in RDF and RDF Schema to grant decidability in a generic Ontological definition. At the same time more powerful constructs we use in our OWL domain definition, more complex is the algorithm to be performed by reasoners to arrive to a conclusion, the limit of this computational time is infinite and so undecidability. Since the OWL definition, but also in recent times, a big effort has been done to research which constructs in class and properties definition can be used to grant decidability and also to limit the worst case of computational time to not exponential trends. This effort resulted in a set of OWL sublanguages with different features, to be used depending from the application requirements. Chronologically at the beginning three OWL sub-languages have been defined:

- OWL-lite
- OWL-DL
- OWL-full

OWL-lite is the simplest OWL sublanguage, it has been defined to grant to its users decidability of the resulting description logic and fast query execution. An OWL-lite ontology is similar to an RDF-Schema knowledge base, but is possible to use restricted cardinality constraint (the max Cardinality allowed is 1) functional properties and a few other constructs which, reducing modeler freedom with language constraint, allow tableau algorithms[44] to be run for the inference process and for the consistency checking in a finite time. When the logic underlined by an ontology is decidable it belongs to the description logics sub-set. The logic underlined by an OWL-lite ontology is called SHIF, where S stays for basic logic operators like the existential and universal qualifiers; H indicates that is possible to define a hierarchy of Properties where the sub-property implies the super-property; I stays for Inverse and so tell us that inverse properties may be defined; F (functional) indicates that cardinality restriction for functional properties (max cardinality=1) can be defined. OWL-DL ontologies contains many more constructs and arbitrary cardinality in domain description, all the language rules grant decidability and a greater inference potential, but the resulting logic is more complex than that of an OWL-lite ontology and so the time needed to compute is major. The Logic of an OWL-DL ontology is called SHOIN where O (one of) indicates that concepts may be defined by the enumeration of its members while N means that property with arbitrary minimum and maximum cardinality can be defined. The arbitrary cardinality is not the most advanced that can be used because even in OWL-DL is not possible to have the so called ”qualified cardinality” which make possible to define cardinality distinguishing by Class. In example while in OWL-DL is possible to state that an individual belonging to the Person Class has at max two parents,

qualified cardinality allows to specify that the property “has Parent” has max cardinality 1 from the subclass of Person Man and 1from Woman.

OWL-full is an ontology definition language syntactically equal to OWL DL, i.e. the allowed constructs are the same, but with more freedom given in the statement creation. As previously mentioned, the lack of structural requirements in domain definition results in less mathematical rigor and for this reason OWL Full is not decidable belonging to a superset of description logics called first order logics. The most important difference between OWL Full and its decidable sublanguages is the clear distinction between classes and individuals. In OWL-DL and OWL-lite, the subjects and the objects of the statements can be only individuals belonging to some defined Class while in OWL-full also classes themselves can be used.

After the definition of the former three OWL sub-languages the research arrived to a new sublanguage OWL-2. OWL-2 is a subset of OWL-DL and is decidable which means that the effort has been given toward the way of decidability. More advanced constructs are available and the preserved decidability may be exploited on a more expressive description logic. It underlines a complex description logic known as SROIQ[45], where the R(Role chain) indicates that is possible to define chains of properties i.e. the chain <p1,p2,p3> is declared, then a software agent may infer the triple <z, p3,s> from the triples <f, p1, s> and <f, p2, z> ;the Q indicates that is possible to make use of the qualified cardinality constraints which have been previously introduced. All of the OWL 2 semantic results in finite but very long worst case computational times, so also for this ontology language a sub-division exists resulting in the so known OWL 2 profiles. In many advanced tools like [46] and [47], is possible to choose one of the former OWL sublanguages or OWL2 profiles, to build an ontology being supported by different graphical views and serialization options.

2.5.5. Rules SWRL RIF

The final relatively stable layer of the semantic web stack is the rule layer. The need for rules lays in the intrinsic complexity of description logics. As we have seen, adding syntactic sugar, and so expressive power, to a description logic results in a growing worst case response time until the undecidability. In frequent applicative tasks, we need software agents able to perform a more powerful reasoning than that possible with simple ontological reasoning. For humans it could seem obvious that if the available credit is less than the price to be paid, then a transaction cannot be done, but this and other kind of relationships between concepts are very difficult or impossible to be represented in decidable description logics. Rules are a very expressive system to express if-then statements. Their integration in an ontology is also simple

to be understood and there are approaches built to automatically derive rules from ontological assertions[48]. While in normal rule based framework variables, local constants and the supported operators are used, in a semantic web context the constants and the variables are part of the ontology and the operators are that supported by the semantic rule language used. Despite both the objectives and the motivation are clear, the integration of rule based logic and description logics is not trivial at all and research is investigating in subsets of rules called "safe" that make possible decidable reasoning if integrated in a description logic. Other works like [49] investigates on the theory and on valid transformations of description logics like the rolification to try to express as much as possible of an ontological knowledge base in the form of rules. Different rule languages have been created with the possibility to incorporate semantic annotation and to be attached to an ontology like RuleML and SWRL[50]. The SWRL (Semantic Web Rule Language) is very interesting for its property of being built over OWL always following the mentioned semantic web layered approach. An SWRL rule can be represented is made up of an antecedent and a consequent. The antecedent or left part or body of the rule is formed by one or more conditions (atoms) connected by conjunctive or disjunctive operators; the consequent (or right part or head) of the rule is a set of statements that must be true if the conditions are true. The use of SWRL is not simple for smart environment based applications and is impossible in conjunction with an ontological reasoner. An ontological reasoner (based on description logics) requires an ontology to be decidable in order to start its computation, but since SWRL is built using OWL full and so on an undecidable basis, each ontology containing SWRL rules, and so importing the SWRL ontology, results undecidable and so not suitable for ontology based reasoning. Two axiomatic statements of description logics, and precisely the open world assumption and the lack of the Unique name assumption are in contrast with what usually happens in rule based systems and are other issues in the integration of the ontologies with rules. The open world assumption states that the what is stated is only part of the available knowledge and that therefore there may be something unknown. If we state for example that a property has exact cardinality equal to four and we declare only three statements using that property for a single subject, the resulting ontology is still valid because the fourth statement may be unknown and may be delegated, to knowledge updates or to the reasoning process, the task of identifying it. In classical rule based systems stands the closed world assumption. The Unique name assumption, typical of rule based systems states that two entities with different identifiers are different entities. In OWL the OWL:sameAs construct is used to assign the same identity to individuals with different URIs and this is in clear discordance with the unique name assumption. Once the different approaches in logical definition taken by

description logics and rule based systems will be aligned in a single uniform and more powerful knowledge base, systems designer will have another powerful instrument for domain description, suitable for machine interpretability, automatic rule based and ontology based automated reasoning and for global interoperability.

2.6. Common semantic formats and standard

The semantic Web project had and have an important impact on the way developers represent and use information. During years many semantic formalism have been created for different conceptual areas, sometimes becoming de-facto standards and in other remaining recommendation still not spread at global level.

2.6.1. Dublin core

Dublin core is a metadata system made to describe a wide range of digital information accessible through web. The Dublin Core Metadata Initiative started its work after a meeting occurred in a conference in Ohio in 1995, between editors, personnel related with cultural heritage and technical experts of the internet. The conclusion was that a set of standard instruments to describe digital resources was needed to avoid interoperability problems. One of the possible simple mistakes that is common for automatic agents is to confuse an original resource with its surrogates, lets say the original Monnalisa that is conserved in Louvre and one of its digital copies. This can have consequences because the author of the original is different from the authors of the copies and copyright issues must be avoided. Applications are not a priori able to distinguish between resources and the effort of providing metadata with this purpose will be nullified, in a global perspective, if the metadata are not interoperable. The Semantic Web, and in particular RDF solves this task by providing simple properties to be associated to web resources in order to characterize them and, for example, to make possible to distinguish between an original and its copies. RDF uses URIs for defining properties, so, using the dublin core components (i.e. standardized RDF properties) there are not language issues and software developers with knowledge of the standard are always able to correctly add and read basilar meta-information about digital resources. In Fig. 2.9. the Dublin core elements are shown. There exist various element classifications but the one distinguishing between content, intellectual properties and identification is the most frequently used. The Dublin Core elements related to content describes information related to what the digital resource represents. The intellectual properties elements clearly define who owns the right for that resource. The

Identification Dublin Core Elements are used to specify information about the resource, in contrast with the content elements. Also if limited in number the Dublin core elements use is widespread because the usefulness of having a common machine interpretable base for describing digital resources is enormous. Moreover as for many other semantic web related standards, the usage is proportional to benefits and once a critical mass of developer make programs using Dublin Core the others are incentivized in order to follow the trend. The Dublin core initiative also thought to those applicative domains where more detail is needed in resource description. The Dublin Core metadata have been approved as an ISO standard (ISO 15836:2003), its extensions and specific profiles are still under development while the affirmed elements constitutes one the first cases in which the semantic web benefits are so evident to exponentially grow in a few years.

CONTENT	INTELLECTUAL PROPERTIES	IDENTIFICATION
Coverage Description Type Relation Source Subject Title	Contributor Creator Publisher Rights	Date Format Identifier Language

Figure 2.9 Dublin Core elements

2.6.2. CIDOC CRM

The CIDOC CRM (Conceptual Refernce Model) [51] is a standard at ontological level for he definition of cultural heritage resources. It is more expressive than Dublin Core because OWL DL is used to define a formal ontology in which the resources are instances of Classes and connected trough properties. The primary role of the CRM is to enable information exchange and integration between heterogeneous sources of cultural heritage information. When considering information sources related to the cultural heritage domain is common to be in front of a plethora of different information systems with different access methods and information representation. The cultural heritage documentation is an important process to increase the visibility of the art in business applications. A shared understating of cultural heritage

information is a key in the definition of accessible informative systems. The need of a formal description through ontologies resides in the need for a guide and good practice of conceptual modeling related to the world of the arts. After more than a decade of work by the CIDOC Documentation Standards Working Group it is now an ISO standard (ISO 21127:2006). The CIDOC CRM can be used also to model other domains not strictly related to the arts in [52] for example it is applied to linguistic to solve the so called exhibition problem, depending from the difference in ordinary linguistic communication between asserting a fact, and exhibiting the same fact. Fig. 2.10. is a screenshot of part of the CIDOC CRM taxonomy, in total there are more than hundred classes and a similar number of properties to be properly used in conceptual modeling. The adaptation of the cultural heritage documentation world to such comprehensive model is a process that requires years and education [53] but as happened with other semantic web related technologies, when a critical mass of users and application will put in evidence the benefits of interoperability, its usage will become natural as today are Exceland databases.

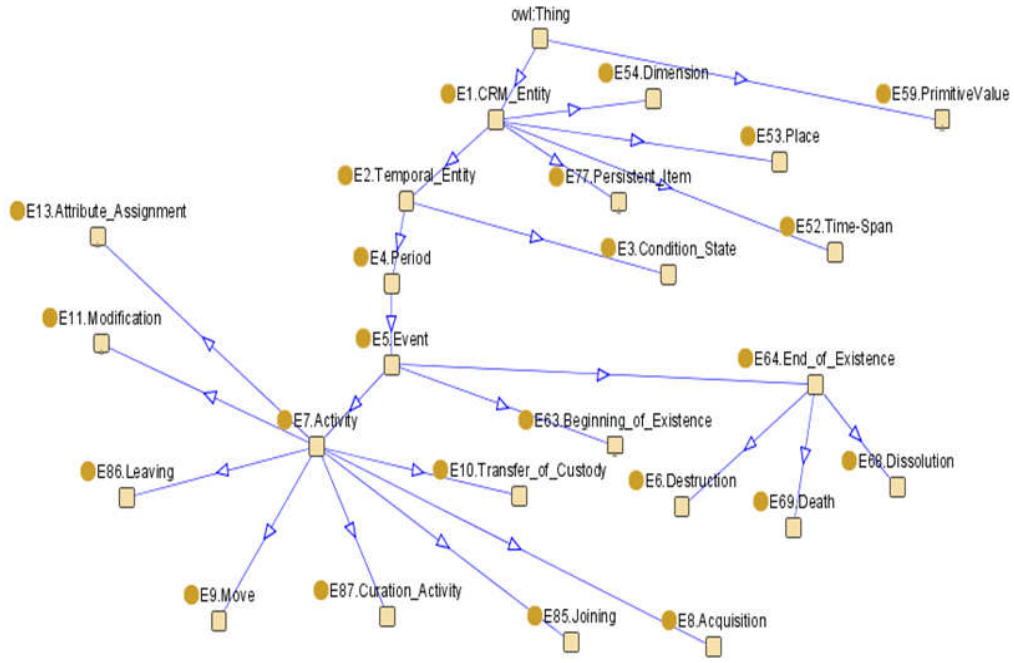


Figure 2.10: Part of the CIDOC CRM taxonomy

2.6.3. SOUPA and CoBrA

SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications)[54] is an ontology built for supporting with semantics the pervasive computing in smart environments. Fig. 2.11. taken from [55] shows a modular conceptual structure in which the *core* part comprehend fundamental concepts for smart environments like Events, location, temporal

relationships and persons. The SOUPA extensions are more application specific and, according to the modular approach taken, always imports, directly or indirectly, some of the modules contained in the core. The SOUPA ontology has been used as conceptual framework for pervasive computing applications related to the CoBrA architecture [56] . The CoBrA software architectures and motivations are very similar to those of which I speak in this thesis, but while the CoBrA project ended in 2004, here the technologies used are more consolidated, the research obtained new results in term of description logics and language definition and the business companies are interested to invest in the area of pervasive computing as put in evidence by the existence of European Projects centered on Ontologies and related hardware software infrastructures. Fig. 2.12. from [57] shows a conceptual and software architecture similar to that described for the SOFIA project. The Context Broker centralize all the information and has internal modules for enriching the knowledge through reasoning, rules, external sources and policies. Sensors, software agents, and appliances make use of the information on the store and, integrated together make possible the realization of smart pervasive scenarios.

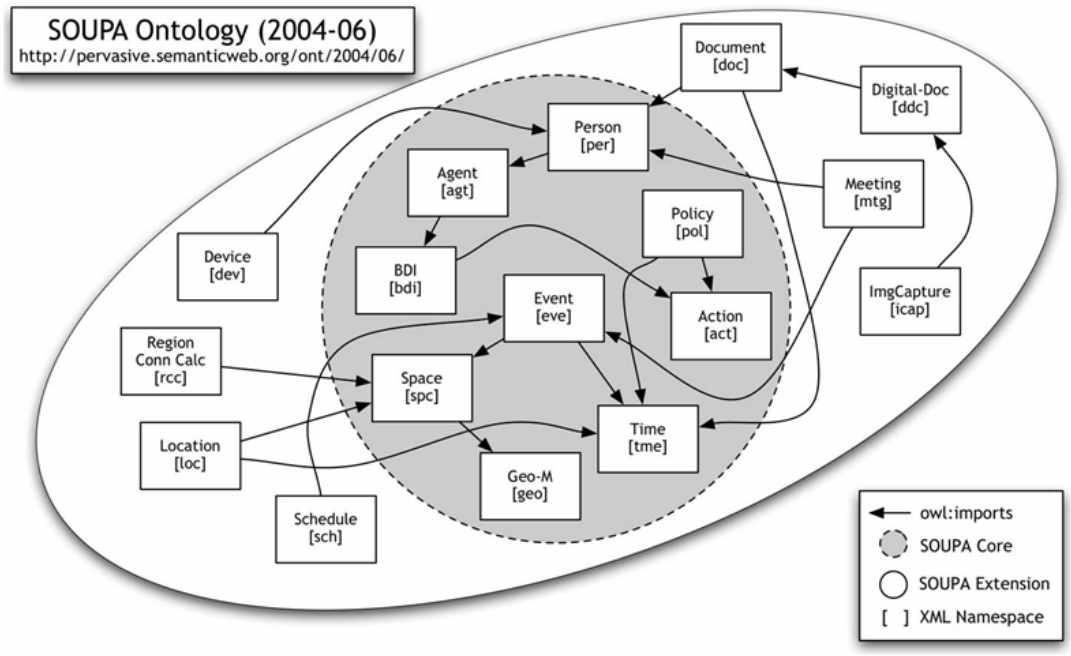


Figure 2.11: SOUPA ontology

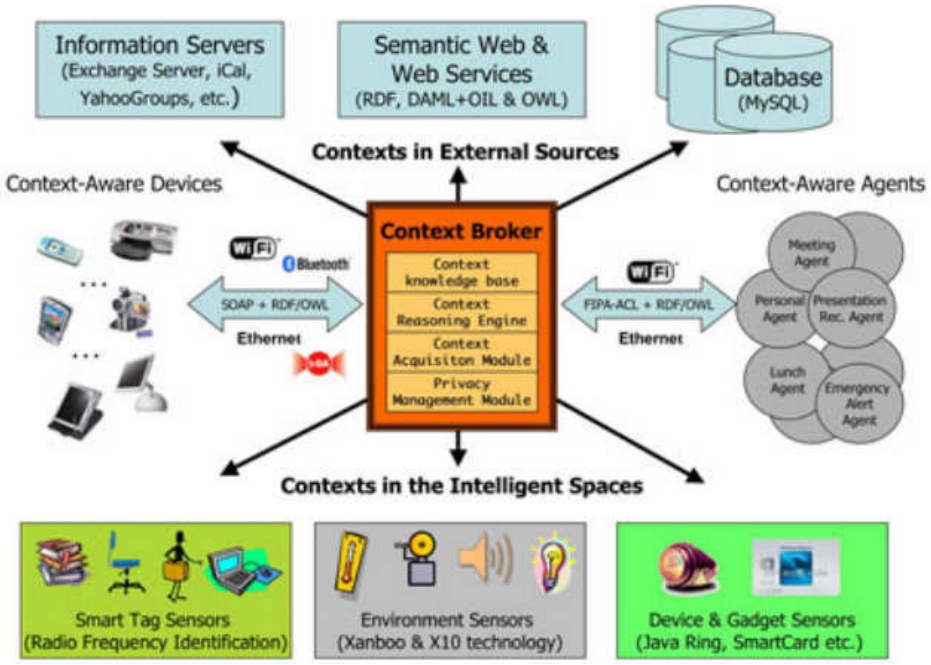


Figure 2.12: CoBrA smart space vision

2.6.4. DOLCE

Dolce ((Descriptive Ontology for Linguistic and Cognitive Engineering) [58] is an upper level ontology developed in the context of the WonderWeb project [59] . An upper ontology, also known as foundational ontology has the aim to identificate the meaning of formal relationships in order to be the basis for the construction of domain ontology. The domain ontology are usually built as result of a bottom up analysis which put in evidence the minimal terminology necessary for a certain community to support specific application requirements. The realization of dolce, like that of other upper ontologies like SUMO (Suggested Upper Merged Ontology) is a delicate task where the concepts and their relationship must be modeled depending from cognitive theory and with knowledge of the philosophical literature. In DOLCE for example there is a clear distinction between endurants and perdurants depending from the behaviour in time. Endurants are entities which exist in time, and exist in their totality while the level of existence in time of perdurants changes. It may be said for clarifying that while the endurants simply exist, the perdurants happen and so exist in different way during time. According to this distinction the objects in a smart environment should be modeled as subclasses of endurants because they are pure or elaborate matter while events are subclasses of perdurants. Other not trivial definitions preset in DOLCE and relevant for smart spaces are that of *qualities* and *quality regions*. *Qualities* are features we can perceive or measure like colors and sizes. They are not intended in the general way, but are specific of the entities we are taking in consideration. Two different rooms have different qualities representing their lengths. To each quality corresponds a value that is a different instance in the ontology and

belongs to the *quale* OWL class. A *quale* specifies the position of a certain quality in the measure domain, if two rooms have the same numerical length it means that two room instances exists, with two different instances for their length quality and with two different *quale* instances, which have the same numerical value. This definitions have been used when defining a top down foundational ontology for smart spaces based on Dolce, in particular when representing sensor data. Fig. 2.13. represents part of the Dolce taxonomy and shows the semantic collocation of some of the previously mentioned concepts.

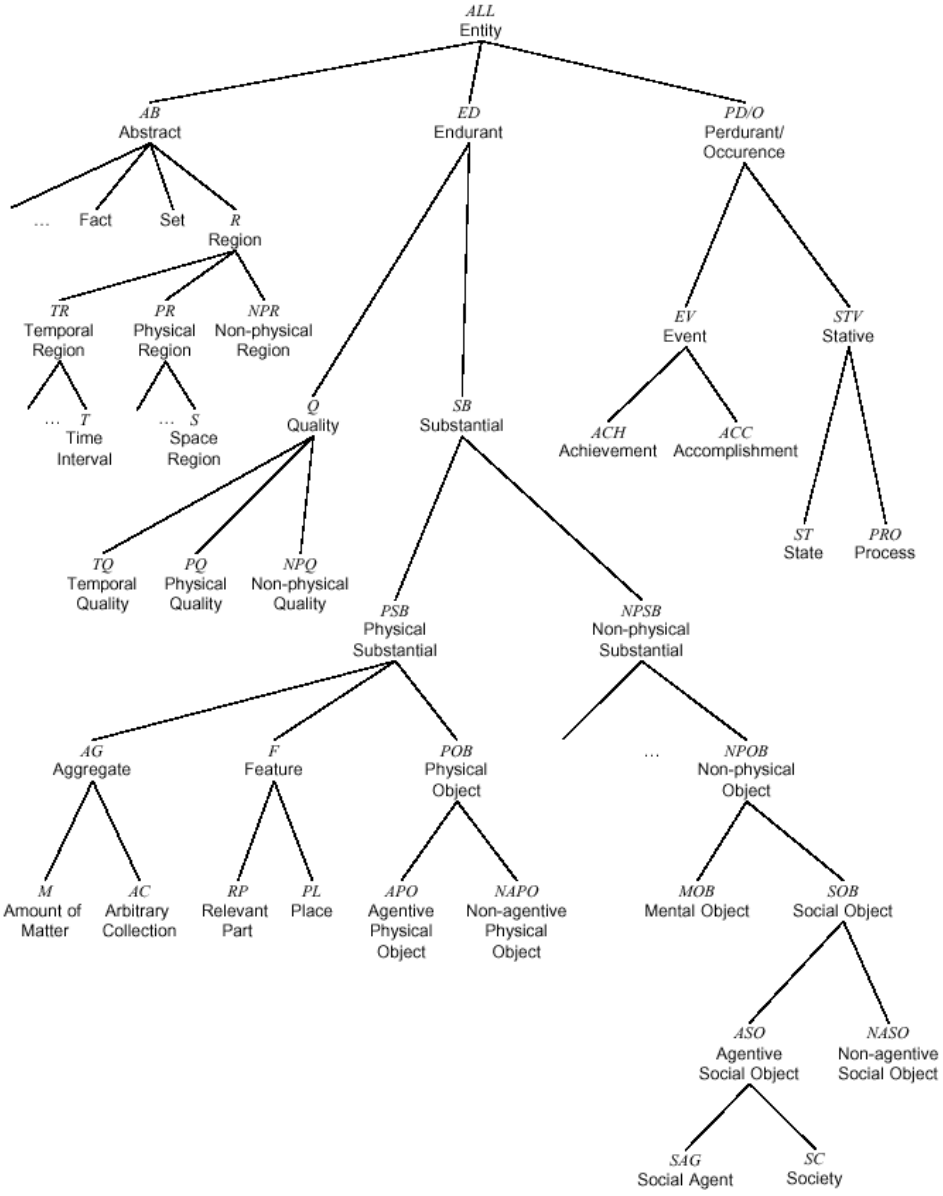


Figure 2.13 Taxonomy of DOLCE basic categories from Gangemi et al. http://www.loa-cnr.it/Papers/DOLCE-EKAW.pdf p.4

SECTION 3

SEMANTIC MODELING OF RELEVANT NOT ABSTRACT CONTEXT ATTRIBUTES

3.1. Introduction

After an introduction about state of the art, related works and existing research from different smart environment and semantic web based perspectives, I'll speak about original works and results that have been produced during my PHD. Hereinafter the discussion will be more technical because the motivations i.e. interoperability, innovation, portability and in general, the impression of smartness given through abstraction and service orchestration, that continue all to remain valid, have been fully introduced. I'll start with the semantic modeling of elementary relevant context attributes, to arrive in the end of this section and in the next one to discuss about the semantic modeling of complex context attributes and some software solution for service discovery, distribution of computation.

3.2. Sensor Data

It is possible to find numerous approaches to sensor data semantic specification. In each case the level of detail changes depending from the specific application requirements of the semantic modeler. To properly deal with this argument is necessary at least to mention and briefly examine two of these solutions that have been used in the SOFIA project: one is based on a top down approach relied on the DOLCE upper ontology, and one is built bottom up for major

usability and faster implementation. The bottom up approaches, also if not based on an upper ontology, can be aligned in a second time through the ontology merging and ontology alignment processes.

3.2.1. Sensor Data Top Down

Starting from the DOLCE conceptualization the objects are endurants provided with Characteristics. Sensors provides measures as result of their observations, but at the same time these results have to be semantically connected with the measured perdurants. Applications, in fact, are usually aware of the resource URI and want to perform graph based navigation through queries in order to know the current values of the perdurants characteristics. Fig. 3.1. shows a first attempt mapping of these concepts in an minimal sensor data ontology:

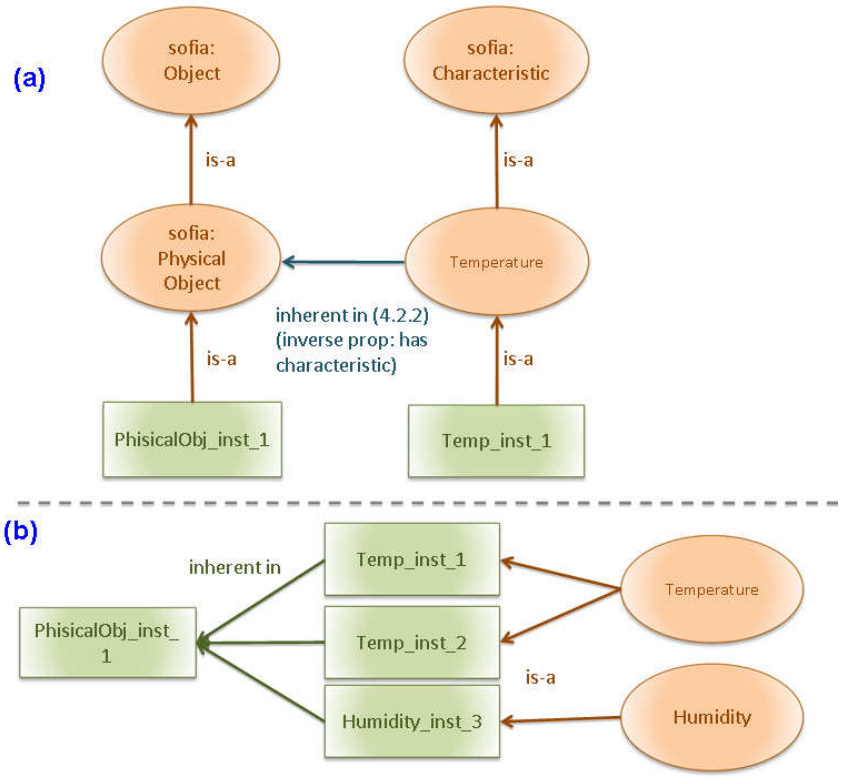


Figure 3.1: First attempt of semantic model for sensor data

In the figure we want to attach observations of different nature, e.g. temperature and humidity from different sensors, to a certain physical object, e.g. a room. Basically this model does not include the concept of Data. The value is attached to the Temperature instance (implicitly we assume that a Temperature instance is a piece of data describing temperature). In this case, Temp_inst_1(b) is not generically temperature, but the specific temperature of a physical object (e.g., a room). However it might be useful and correct to maintain an explicit notion of Data distinguished from the notion of a Temperature or Humidity instance. There is in fact a distinction between a temperature data (measured or somehow calculated) and the temperature of an object. The temperature of a Room may be the result of the mean operator applied to different available temperature data. It looks like both Data and Temperature are needed in order to distinguish the observations, more similar to perdurants (i.e. events), from the characteristics.

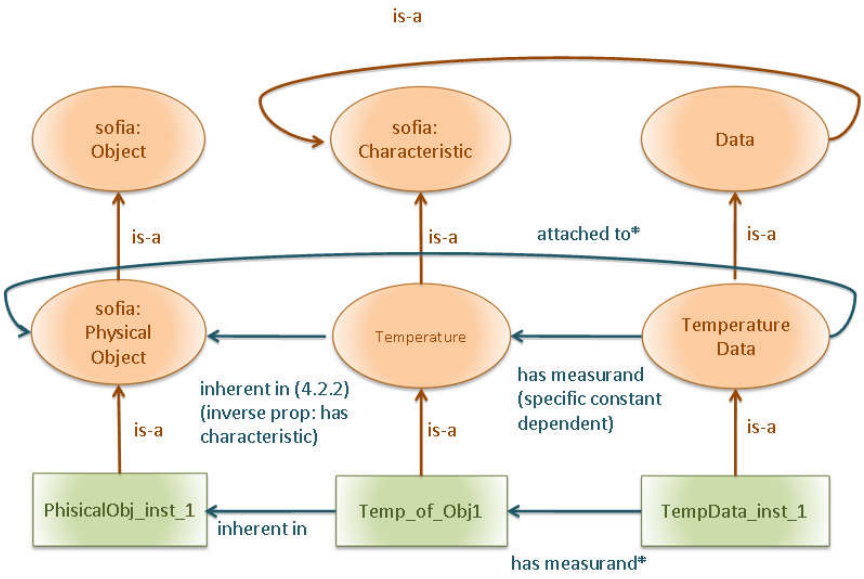


Figure 3.2: Sensor data semantic model after top down analysis

In Fig. 3.2. is shown the final model used in SOFIA Core ontology to describe sensor data at low level of abstraction. Here the Data Class has been defined as a subclass of SOFIA:Characteristics which, in its turn, is linked to the Dolce upper ontology. The Temperature Data, provided by the sensor, is linked through the has_Measurand property to the instance of the measured Characteristic of the Physical Object instance. It is important to notice that the semantics of the characteristic, i.e. the fact that is a temperature, resides in the specific subclass of Characteristic that is used to instantiate the characteristic itself. This will be the principal difference with the bottom up approach that has been developed independently.

3.2.1. Sensor Data Bottom Up

The bottom up approach consists of analyzing the domain and the specific purposes ,i.e. that of associating different kind of measures to objects by distinguishing between observations and characteristic, and mapping them to an ontology without taking too much care of philosophical validity of the result. In general this approach presents issues of extendibility and interoperability: if it is too much application specific. It may happen, in fact, that different applications which at the beginning are not supposed to collaborate, need a different level of detail in domain description. If in a phase successive to ontology creation it is needed collaboration and data sharing between two independently developed bottom up ontologies a mapping may not be possible because of too much difference in the level of detail and generality with which the two domains have been previously conceptualized. Fig. 3.3. shows the part of the bottom–up sensor data ontology corresponding with that analyzed before for the top down approach.

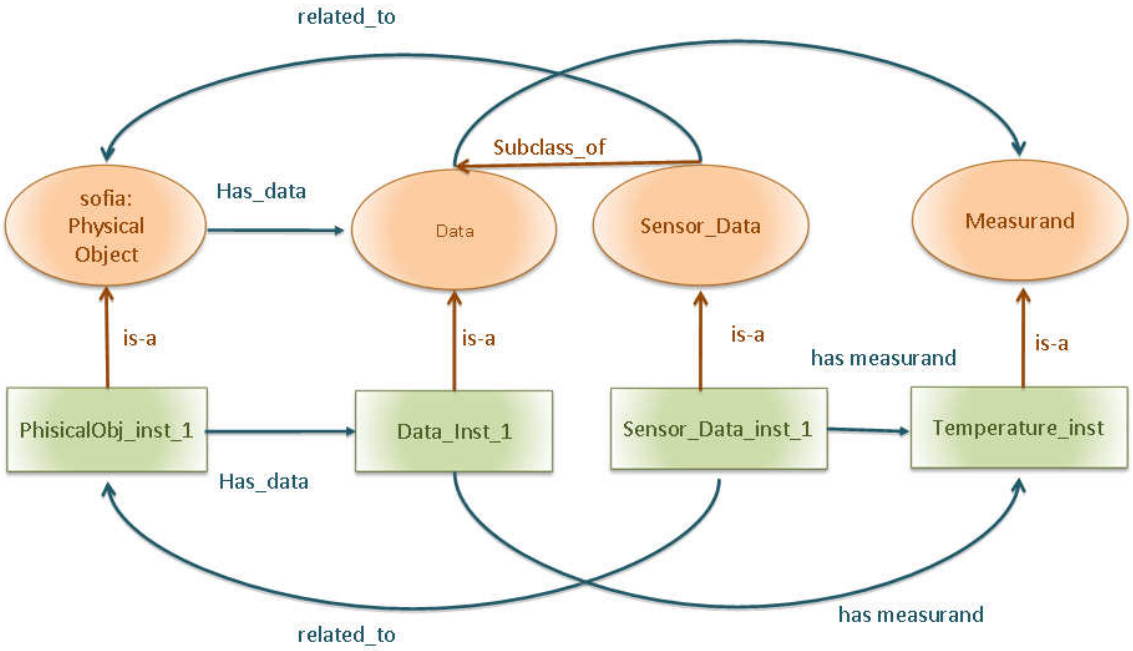


Figure 3.3: Sensors data semantic model after bottom up approach

First of all there is no mapping with the DOLCE Ontology (also if it could be determined). The sensor data Class has been created as a subclass of the generic Data Class. Observations are instances of Sensor Data and are provided by sensors while Data are relative to objects and are

associated to them by using the related_to relationship between sensor data and objects. The Measurand class is the class of all the general physical entities that can be measured. It is very different from the previously mentioned subclasses of Sofia:Characteristics, because while characteristics are specific to the measured object, the Measurand are intended in the more general sense. In the previous model two different sensor creates two different instances of the temperature subclass of Sofia:Characteristic while in this model create two sensor data with the same measurand. The semantics of the measure is in the instance of the Measurand class that is associated to the Data while the semantic of the measure association is identified by the related_to property. Despite the risks the bottom up ontology is simpler to be built and used and, if thought with good extendibility criteria it has good reusability. In the SOFIA project both bottom up and Top down ontologies have been used with success in collaborative multi-industry scenarios, the top down approach is preferable for compatibility reasons, but the effort necessary to produce coherent top-down ontologies has not to be underestimated. Sometimes also philosophy cannot help by giving a uniform and shared view of all the domain aspects and with also not so elevate levels of arbitrariness in taxonomy definition, bottom up approaches are preferable.

3.3. Smartification

The smartification [60] is a new concept patented by the University of Bologna during the SOFIA project[61] . After some experience in smart environment application programming it has become evident that a repetitive task was always necessary in order to provide an initial correspondence between what is present in the physical World and the information in the digital world. Relevant Context attributes like the available sensors, and actuators, the physical environments, the topology of the smart environment and so on, where to be inserted in the smart space in order to initialize the basic software modules which, in a second phase feed high level services and complex applications. With reference to 3.2.2. before a sensor measurement can be associated to a room is necessary to instantiate the Room, the sensor, its sensor data and the facts that the sensor data is related_to the room. In pedestrian assisted navigation is necessary to inform the smart environment of all the rooms and corridors, of their dimensions and of their relative location in order to start a GIS based graphical assistant. In Smart Collaborative scenarios it is common to see voting systems to be realized with RFID or Near Field Communication Techniques, but is necessary an initialization phase to inform the systems of the voter identities and of their association with a certain RFID code. In all these cases and in many others the initialization phase can be performed by directly inserting the information in the

system at the beginning, but this approach is suitable for demonstration and prototypes where the information is a priory known. The real need is that of a class of devices and services able to initialize the smart environment through the use of standard technologies and possibly with a natural user interaction, in order to simplify the work of smart environment management. This concept have been called Smartification. One of the key concept of smartification is the identification i.e. a correspondence between a standard identification technique like RFID or QR code and the detection of a unique URI of the digital smart space that is detected by the physical ID read. Fig. 3.4. shows the classes and example of instances in the identification ontology. It is a bottom up ontology perfectly integrated with the sensor data ontology. A subclass of Data, i.e. Identification_Data, has been defined, the value of the identifier, from whatever technology is connected to this instance as a literal. To interpret the semantics of the identification value, i.e. the identification technology, the value of the property has_Identification_type is used. The instances of the class Identification_Type are all the identification technologies supported. Thanks to this semantic model is possible to identify physical objects with different technologies at the same time: the software delegated to find the instance connected to a certain ID, must only know the identification technology that provided the literal value and will be able to query the knowledge base in order to properly perform the identification task.

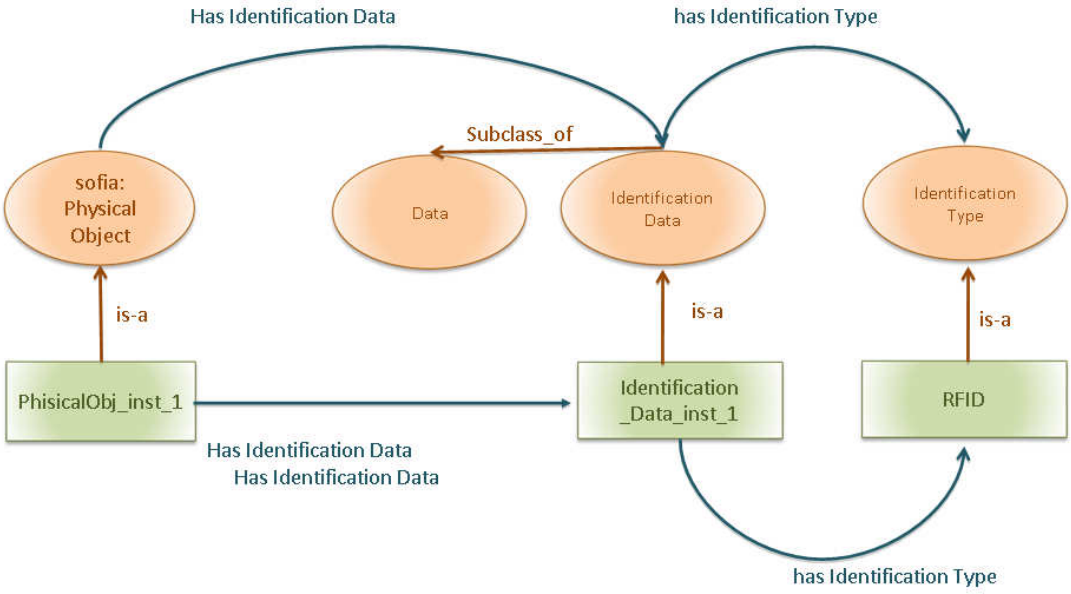


Figure 3.4: Semantic model for identification in smart environments

In developed scenarios different kind of identification technology where used to identify objects and rooms. Handheld devices provided by camera where able to recognize their location or to identify objects through QR code while resource constrained devices used RFID. In a complex maintenance scenario a maintenance operator starting from the human readable identifier of a room(e.g. room 4.2), was able to be guided from the hall of the building through the fault location using path solving [62] and then, by simply approaching to the room identifier was able to verify its location and to see it on a GIS model of the building. A prototypal smartificator device has been realized [63] to show how smartification process can be performed with low cost devices and by using natural interaction and a simple led based user interface.

3.4. Control and interaction

Human computer Interaction is a relevant science in pervasive computing because is one of the most important components providing the illusion of environmental smartness and contributes in hiding electronics. When building a semantic model for describing interaction, the most relevant scenario is that of a controller which have to give commands to some actuator by using a multimodal interface. The ontology represented in Fig. 3.5. has been realized to perform this task.

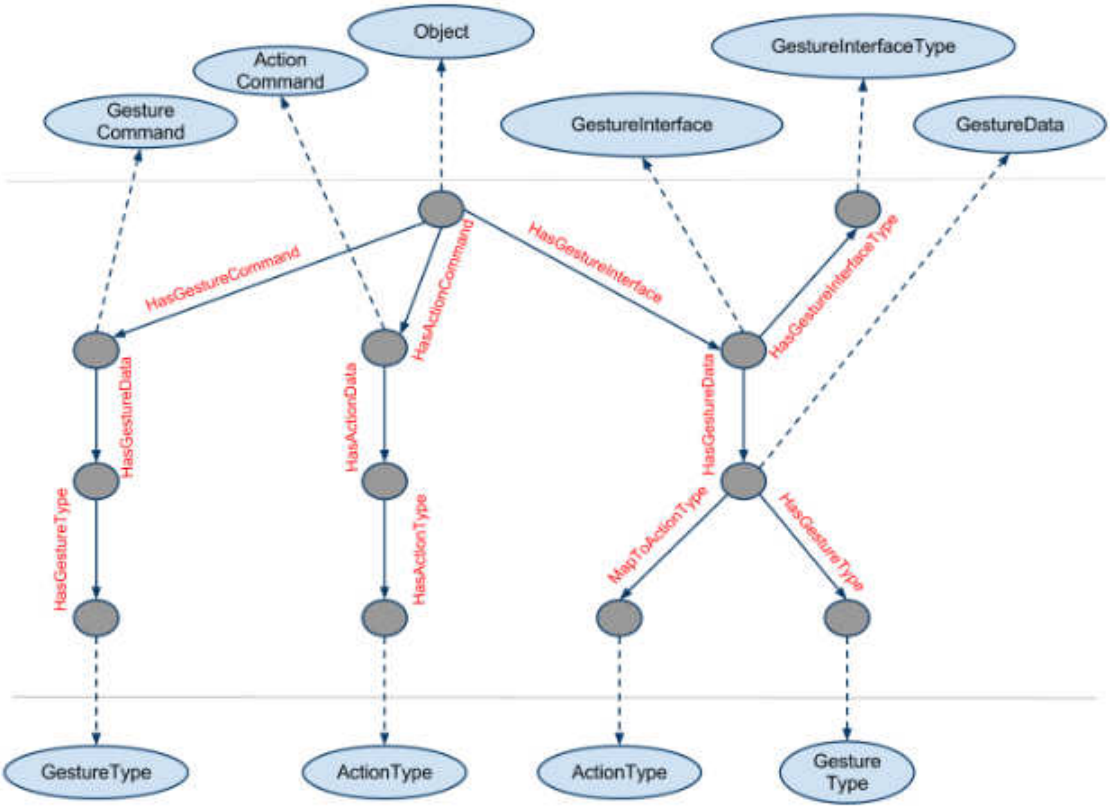


Figure 3.5: Semantic model for controlling an appliance. The instances of Gesture Command and Action Command are the interfaces for controlling the actuators

the grey nodes are instances of the Classes represented by ellipsis. The arcs represent OWL Properties with arrows pointing towards the statement objects. The properties of the represented model allow high level functionalities with a relatively simple semantic model. The scenario is that of a controller (not represented in figure) which may provide commands to an instance of Object.i.e. an actuator. The Actuator is smartified by providing its URI, its command_Interfaces (in figure an interface for actions and one for gestures are represented), one or more mappings between natural user interaction and the relative action to be performed. The software running on the actuator is supposed to be subscribed to its interfaces by being notified of every new command. When a command arrives the action is performed and then the command is deleted by giving the opportunity to receive new commands. The control is multimodal because, depending from the interface used, different ways of accessing the actuator are available. If the controller knows the specific commands of the actuator it is possible to use the action interface by writing action commands in the SIB. This modality is typical of controllers provided by GUI on which the end user may choose the action to be performed, e.g. by a touch screen. If instead the controller is provided by a gesture recognition module, it may use Gesture Commands or can previously query the Gesture Interface in order to understand the

mapping between gestures and actions, and then generate actions. Modifications to the Gesture Interface correspond to a reconfiguration of the gestural profile and so to the gesture-action correspondence. Finally smart objects may store internal semantic structure equal to the gestural interface sub-graph. In this way the mapping between gestures and actions is resolved in the controller code to obtain personalization. We worked also on similar semantic structures with controller implemented as Smartphone applications. The advantages are

- Ability to reconfigure
- Possibility to personalize the interaction
- Multimodality, i.e. gestures recognized by proper algorithms or touch screen
- Possibility to add new interactions, e.g. voice commands
- Exploration of a new market in which users are able to download software able to command her smart appliances

.

With the same objectives and similar results another interesting model based on events has been used for gestural interaction in the SOFA project. The model is based on [64] and is described in Fig. 3.6. where are also put in evidence the relationships between the appliance and the identification technologies.

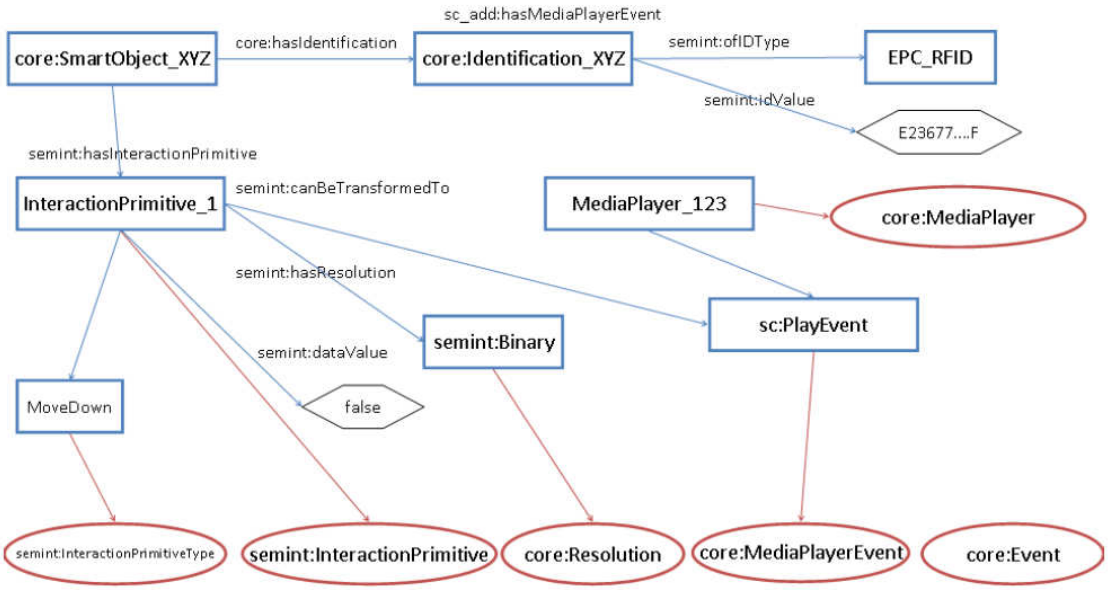


Figure 3.6. Semantic model for interaction used in SOFIA project

The main differences with the model previously described are the presence in the smart space of an instance relative to the controller (i.e. core:SmartObject_XYZ) and the

semint:canBetransformedTo property which plays the role of the previously mentioned Gesture Profile, by providing reconfigurability, mapping the interaction primitives like gestures to events. The software of the actuator has no more a direct explicit interface, but checks for events generated by controllers connected to actuator itself. To demonstrate the validity of this model also in a multiplayer context, where the actuator is a game application, a Connect 4 prototype have been realized and played in the context of a University course related to smart M3 [65] .

SECTION 4

SEMANTIC MODELING OF RELEVANT ABSTRACT CONTEXT ATTRIBUTES

4.1 - Data access control and synchronization

We have seen as a Smart Environment is often a multi-agent system in which concurrent software agents access in R/W mode to a shared Knowledge base in order to adapt their behaviour to the situation they observe. If a context platform doesn't natively support transactions and blocking operations there may happen synchronization issues on shared pieces of information. The problem has many implications and is related with the history of computing theory as diffusely explained in [66]. Access control and process synchronization in digital systems has been deeply explored since the origin of computer science at all levels of abstraction of conventional computer architectures [67] [68]. Also in operative and informative systems the possible alternatives of controlling access to resources , and so also to allow transactional behaviour, have been deeply explored. The discretionary models (DAC) [69] use a matrix relating access rights to the possible combinations of subjects and objects. The Bell La Padula [70] model adds a mandatory check to partially solve the problem of trojan horses. Lattice based access control (LBAC) allows the use of security labels organized in a lattice to enforce policies like the Chinese wall [71]. The study performed in this PHD has been focused on the definition of a simple extension of the semantic graph to obtain a powerful method to grant exclusive access to a small set of triples in an RDF

store. Besides proposing a solution, a lightweight implementation on the Smart M3 semantic platform has been coded and tested in a real use case. We started from the situation presented in Fig. 4.1.

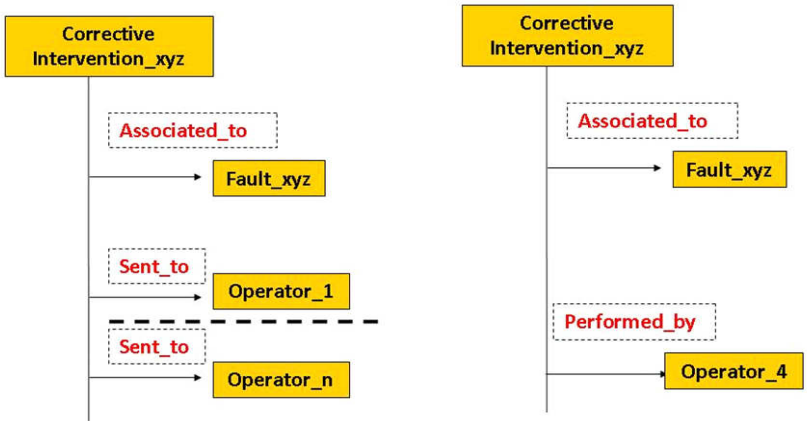


Figure 4.1: Starting (left) and ending (right) situation of the semantic graph generating conflicts in concurrent scenarios.

The right part of the figure represents a corrective intervention request relative to a certain fault that as been sent for approval to n different maintenance operators. The request is computed by maintenance operator devices resulting in a final graphical form which asks if the request is accepted or not. The wanted behaviour is that is the request is accepted by an operator, i.e. Operator_4, the initial graph is transformed to that represented on the right of Fig. 4.1. The graph transformations fire a set of subscriptions which result in the notification of the other operators that the intervention has been accepted and that is not possible to accept it anymore. When trying to implement this on the smart M3 architecture, since transactional primitives are not available, there is the possibility that synchronization issues arise. The possible applications behaviour are represented, for a more general situation in Fig. 4.2.

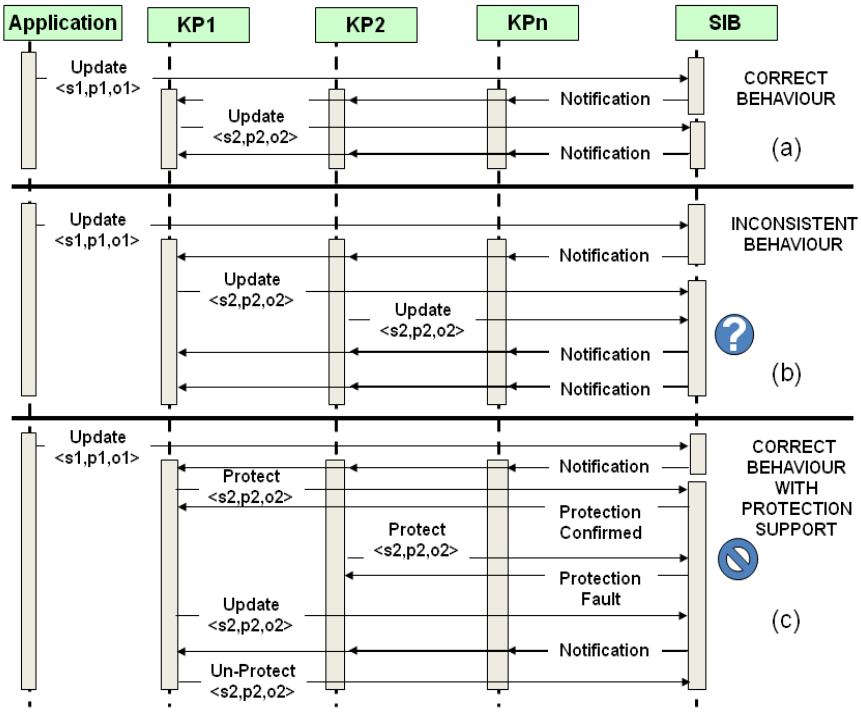


Figure 3.2: Sequence diagram of three possible evolutions of a concurrent scenario.

all KPs are supposed to be subscribed to the triple pattern <s1,p1,*>. and when notified they may insert <s2,p2,o2> (in the maintenance scenario it is the *Performed_by* operation) which in turn may notify other KPs.

Three different situations are shown. In (a), there is no access control enforced: KP1 is the only one process reacting to the notification; as no other process makes operations in the critical time, the overall behavior is correct but it is not safe. In fact, if two KPs perform the same operation during the critical time, unpredictable results may occur, as shown in (b) where KP1 and KP2, both receive a notification, but only one KP should perform the update. Fig. 4.2. (c) shows the behavior with access control in place: before updating the SIB, KP1 locks property *p2* and, when the concurrent process KP2 tries the same operation, it receives a Protection Fault due to the attempt to access to a locked pattern. Should KP2 try to do its sub-graph update without a prior protect request, it would still receive an access denied answer. KP1 releases its exclusive access right to the shared resource after receiving the notification of successful completion of the requested triple pattern update. The proposed solution is based on Fig. 4.3. and consists in adding at information level triples specifying the access control Rights. The protection entities P are connected to the node to which are attached the access control policies. The information owner and the properties for which the policies stand are specified through the AR_Owner and the AR_Target Properties.

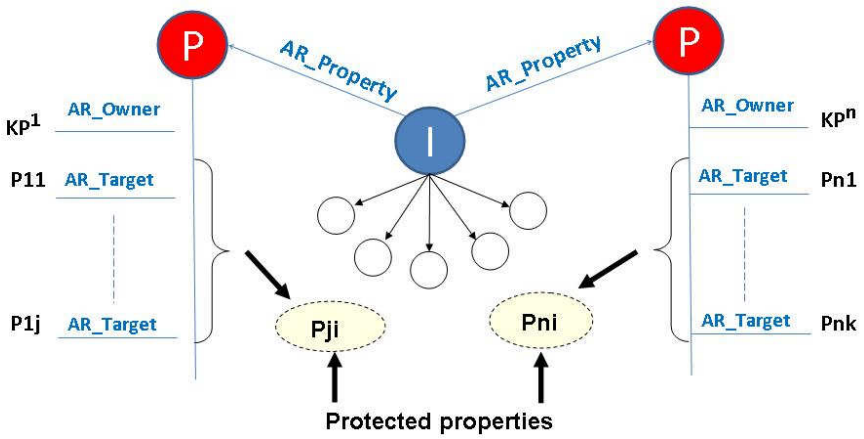


Figure 4.3: Semantic model for specifying access control at triple level.

When implementing the access control model on the Smart M3 platform it has been decided to limit the impact the SSAP protocol and on performances. The first objective has been achieved in a natural way because the model simply pretends the use of normal RDF triples and so is natively transported by SSAP. To limit the impact on performances it has been decided to avoid any query on the information store in order to enforce the access control policies. This has been obtained by basing the algorithm on a dynamic table called Lock Cache Table (LCT). The use of the LCT allowed to perform all the necessary access control checks on the incoming messages, but also all the updates of the LCT itself, on the simple checking the triples against the content of the table. The results obtained are shown in Fig. 4.4. where is possible to notice how the correlation between the time needed for a single insertion and the number of active triple pattern is minimal. The chart shows that even for insertion with a relatively high number of triples, hence involving many comparisons and accesses to the LCT, the total impact on performance with one thousand protections is a little percentage of the best case with no protections.

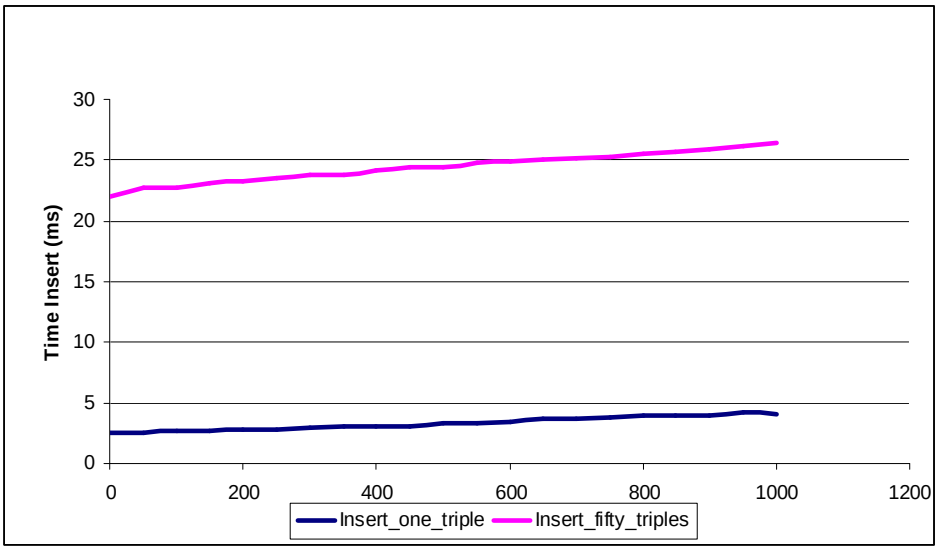
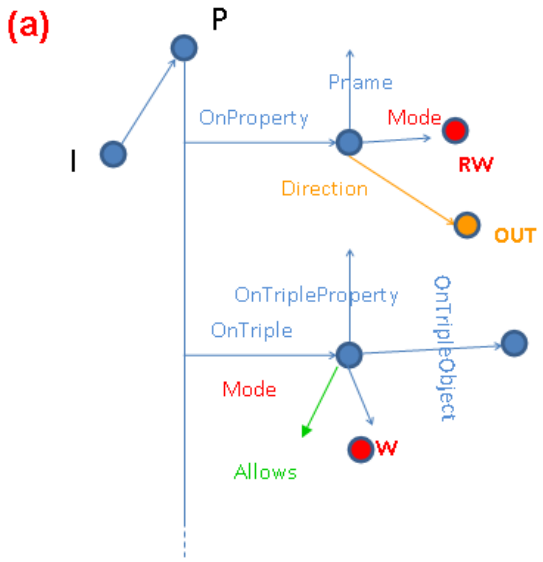


Figure 4.4: performance of insert operation of one (blue) or fifty (violet) triples with growing number of active protections. Each point is the mean of one hundred tests.

An extension of the work published is currently under study to allow more policies with minimal impact and maintaining the global approach. The new proposal is represented in Fig. 4.6. where (a) is the extended and modified semantic model while (b) is the corresponding LCT. The main additional features that are possible with this new model could be:

- Protection in read mode or read/write mode.
- Protection of a single triple; in the old model, in fact, is only possible to protect pattern like <s,p,*> which corresponds to a potentially infinite set of triples.
- Possibility to specify the protection entity attached to the object of the triple, feature that is very useful when dealing with symmetric properties. In the old model is possible for the agent A to avoid that B states <A, met, B> , but it could be problematic to do the same for <B, met, A>, that from a semantic point of view is the same. The direction field of the extended LCT is present for this reason.
- Possibility to specify a set of allowed software agents for each protection descriptor.



(b)

I	P	O	Owner	Mode	Allow	Direction

Figure 4.5: Proposed extension of the semantic model for access control (a) and LCT (b)

4.2 Computation

This sub-section is about the work I have done during the six months spent in Helsinki as a Nokia intern. The objective was to investigate the semantic representation of computational flow and the possibility to delocalize computation by having an infrastructure able to delegate functions or pieces of algorithms to external executors. The impact of this kind of research can be very high if efficient ways of moving computation are found, particularly in mobile scenarios where the usage of computational resources influences the device reactivity and the battery life.

An important related work is that of Marko A. Rodriguez [72] [73] who approached the problem by modeling a semantic network starting from the low level details of the microprocessor hardware. In my case another important reference were the Haskell monads [74] and the possibility to perform speculative execution like for example with the java.concurrent package[75]. In practice the complete objective was to realize a semantic model, and possibly a

demonstrator, supporting execution portability, like in Rodriguez work; at a functional level of abstraction, like is possible to do with Haskell monads; and trying to support speculative execution and parallelism.

To perform this task, after studying the initial material and the Haskell programming language, I have made additional research in flow based computing and hardware architectures, graphical models of computation, and hardware architectures for parallel and concurrent execution e.g. [76], [77]. My idea was that functional programs may be graphically represented in a dataflow fashion way to be then executed as we like by apposite executor modules, so exploiting parallelism and/or providing speculative execution and /or optimizing other execution metrics. Found the right graphical representation it is possible to check if a corresponding ontology can be built and then, given the ontology, and so the semantic representation of functional computation, building a prototype of executor would have been the minor problem. To make an important test (also if not exhaustive) of the validity of the mapping to Haskell I studied the State Monad and I tried to apply the graphical model, and so the ontology I invented to that. Hereinafter I will give a brief description of the practical work done.

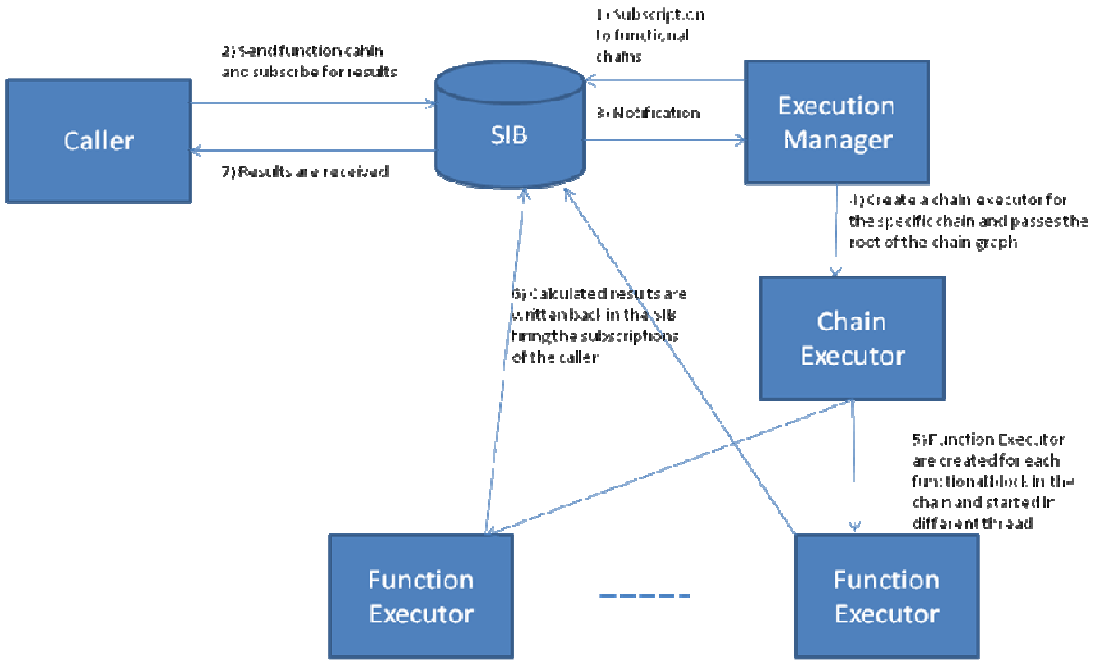


Figure 4.6: Example of software architecture and information flow in a distributed computation scenario.

Fig. 4.6. shows the high level software architecture and the information flow of the realized prototype. A caller sends to the SIB the semantic representation of a chain of functions. Then the Execution Manager module, that was subscribed to this kind of information, creates a

Chain executor thread which, in its turn generates as much Function Executor Threads as necessary. Each Thread passes to the successive the minimal information required in order to make it able to understand what to do. The Chain executor needs only the URI of the Functional Chain, then it has to find the inputs the outputs, the single functions that are part of the chain and organize them. Each function executor needs the URI of the function to perform and that of the input and output parameters. When function executors are able to compute, i.e. when all the input are available, they execute their functionality and write back the results on the SIB with statements related to the URIs to which the caller is subscribed in order to obtain the final results. It should be clear that the execution is out of order: each module that can compute executes and writes back on the SIB independently from the others, so at this level of abstraction the execution is speculative.

4.2.1. Semantic Model and Closure

We called “Closure” a concept corresponding to a functionality, its inputs and its outputs. A closure can be drawn as a box with slots called ports. The port is the instrument by which values are associated to the function as input or output. The Ports are associated to a parameter that is the container of a value or of a reference to it. The ports are also provided by a name whose function is to allow the correct reconstruction of the closure when it is queried from the SIB. A closure with valid inputs is ready to run in the sense that the functionality corresponding to the closure itself can be executed on the inputs to obtain the outputs. The first closure realized performed the functionality of summing two integers. So it was provided by two input ports (In1 and In2) and one output port (Out). The parameters without a value are conventionally called invalid, the value can be calculated by other closures in the same chain of functions or by external processes, whoever, when writing the value on the triple store, has also to take care of deleting the invalidity statement and putting a validity one for the parameter. In general a closure sent to the RDF store with at least one invalid parameter, corresponds to a waiting thread, when all the inputs will be valid the thread will run its functionality and will write the values on the corresponding output parameters. An invalid parameter can be connected to the input port of a closure and also to the output port of another. In this way a connection between the two closures is realized, i.e. a functional chain, and when the closure

with the parameter as output will write it, then the other one with the valid input will be possibly ready to run to bring on the computation. A simple example is showed below:

```
String[] inputs = new String[2];
String[] inputsRef = new String[2];
String[] outputsRef = new String[1];
inputs = new String[2];
Vector<String[]> VecInputs = new Vector<String[]>();
Vector<String> InReferences = new Vector<String>();
String[] inValues = {"1", "2", "3"};
VecInputs.add(inValues);
outputsRef = new String[1];
outputsRef[0] = "a";
addVecFunctionCall(OntologyVocabulary.AddVectorIntClosure,
VecInputs,InReferences, null, null, null, outputsRef);
inputs[0] = null;
inputs[1] = "2";
inputsRef[0] = "a";
inputsRef[1] = null;
outputsRef = new String[1];
outputsRef[0] = "b";
addFunctionCall(OntologyVocabulary.AddIntClosure, inputs,inputsRef,
outputsRef );
Vector<Vector<String>>triples= serializeTriples();
```

The code corresponds to the creation of a closure chain made up of two closures: one adds all the elements of the input vector and one simply sums two integers. The output of the first one is called “a” that is the same reference given for the first input port of the second closure, so there will be a single parameter “a” initially without value that will be written by the first closure by allowing the second one to execute. In Fig. 4.7. it is represented schematically what happens in the RDF store: in blue the single parameters, in orange the Vector-parameters with dimension inside, in red what is calculated at execution time.

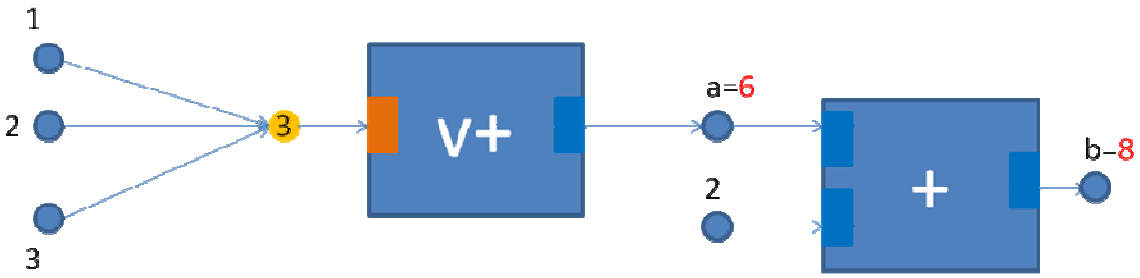


Figure 4.7: Graphical representation of the semantic graph of a simple closure chain

In the execution phase a graph made up of several triples representing in detail the computation is received by the SIB and the execution manager, that recognizes the closure chains, starts a chain executor. The chain executor discovers that there are two functions inside the chain and starts two function executors. One of the two begins to execute soon, the other one

has an invalid input parameter and waits. At a certain point the first executor calculates the results (i.e. a=6) and writes it back in the SIB by also updating the validity information of the parameter; now the second function executor discovers the availability of the new input, it queries for it and finally executes calculating the value and writing it back into the SIB. The caller, subscribed to the value of the parameter named “b”, is now able to know it.

4.6.2. Functional Parameters and Higher Order Functions

As previously mentioned an important, also if not definitive test, of general applicability of the model to a functional program is that involving mechanisms similar to that applied by the Haskell execution framework to monads. Restricting the analysis to the state monads the most important thing that is necessary to emulate is the currying and binding mechanism[78] which relies on possibility to call Higher Order Function (HOF) i.e. functions with other functions as input as the Haskell Map function [79]. In the following example we demonstrate the applicability of the semantic model created to the Map function in order to demonstrate its generality.

```
FunctionalChain fc = new FunctionalChain();
String[] inputs;
String[] inputsRef;
Vector<String[]> VecInputs = new Vector<String[]>();
Vector<String> InReferences;
String[] outputsRef;

inputs = new String[2];
inputsRef = new String[2];
outputsRef = new String[1];
inputs[0] = "3";
inputs[1] = null;
inputsRef[0] = null;
inputsRef[1] = "in";
outputsRef[0] = "f_x";
fc.addFunctionCall(OntologyVocabulary.AddIntClosure, inputs,inputsRef,
outputsRef );
inputs = new String[2];
inputsRef = new String[2];
outputsRef = new String[1];
inputs[0] = null;
inputs[1] = "4";
inputsRef[0] = "f_x";
inputsRef[1] = null;
outputsRef[0] = "f_y";
fc.addFunctionCall(OntologyVocabulary.SubIntClosure, inputs,inputsRef,
outputsRef );
FunctionalParameter fp = new FunctionalParameter();
fp.setContent(fc);
FunctionalParameter[] forInitialize = new FunctionalParameter[1];
forInitialize[0] = fp;
String[] inValues = {"1", "2", "3"};
VecInputs.add(inValues);
VectorParameter outParameter = new VectorParameter();
```

```
outParameter.setRandomURI();
//connectionTable.add
outParameter.setValid(false);
VectorParameter[] mapoutputs = new VectorParameter[1];
mapoutputs[0] = outParameter;
addFunctionalFunctionCall(OntologyVocabulary.MapClosure, VecInputs, null,
mapoutputs,null, null ,null,forInitialize , null);
Vector<SingleParameter> outs =
 AtomicFunctions.lastElement().getVectorialOutputPorts()[0].getSignal().
 getContent();
for (int i = 0;  i < outs.size();i++)
{
        connectionTable.put(outs.elementAt(i).getURI(), ("mapout_" + i) );
}
```

The code verbosity is mostly due to the prototypal level of the implemented software and to the fact that, in theory, this kind of code should not be written or read by humans, but automatically generated from an apposite framework. In Fig. 4.8. is represented the situation at information level: what is red or surrounded by red is not initially written into the SIB and in particular is written by the function executor of the map method and its sub threads. The green parameter and ports correspond to the flow of information of functional parameters. The functional chain input of the Map method has one input and one output, these parameters are not needed in the representation since they lack for the intrinsic nature of the map method. To clarify this we remember that the map method has as input, by definition, a function of one parameter i.e. a closure in which one input parameter is invalid. The map operation has as input an array and a function and gives as output an array containing the result of the application of the input function to each of the values inside the input vector. To represent and do this in our framework there are various possible way with pros and cons, but some consideration can be done. Each HOF has as input one or more functions and does something with them, the input functions are so used in two different ways during the process: at the beginning they are simply an input without any objective, and later their functionality has to be applied in some way. We call these two phases representational phase and executable phase after expansion. In the case of the map the execution correspond to the application of function to the right input and targeted to the right output. This objective can be done for example by repeating the functionality embedded in the functional parameter for all the input and putting the result in the right slots of the output vector. It is also possible to run in parallel the functionality on the various inputs by exploiting the parallelism implicit in the map operation. The solution that has been implemented currently is different to demonstrate that not only closure can be reconstructed by queries, but also user defined chain of function which becomes de-facto first class entities in our framework. When the function executor understand that it has to perform a map operation the input functional chain is reconstructed and then replicated as many times as the dimension of the input

vector. At this point for each replica the input and the output references are correctly bound (red arrows) and the obtained executable closure chains are simply sent again to the SIB firing three times the execution manager. Performances are not obviously the objective of this implementation because, as previously explained, the final scenario is quite different from the current execution environment and now the most important task is to understand what is possible to do with computation semantically represented to proceed in another moment to the refinement and optimization.

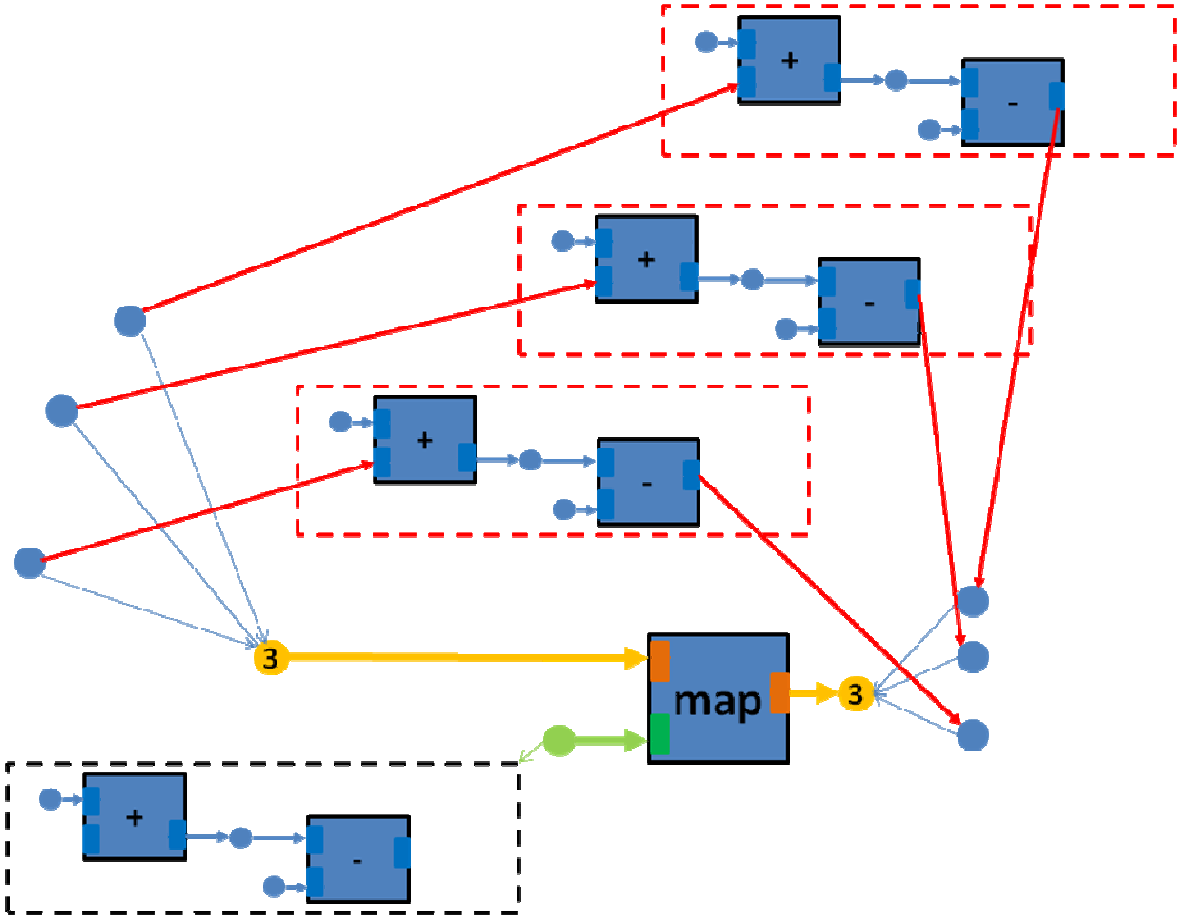


Figure 4.8: Graphical representation of the semantic graph of an HOF in red the executable phase after expansion.

To resume this section, in the six moths spent in Nokia research center, an ontology for computation has been realized starting from considerations about functional programming, dataflow architectures, and graphical framework for the representation of computation. The resulting model is sufficiently versatile because, being able to perform HOF, it is also suitable for currying and so to himitate the very general Haskell state monad and its binding mechanism. The most important features and consideration to be done reguarding this work, that has been filed as US patent [80] are:

- Possibility to semantically represent and transport computational flow information in the general form of a functional chain.
- Formal demonstration still lacks, but there are high probability that the model has the same level of generality of Haskell.
- Management of HOF.
- Ability to perform execution with different strategies depending from available resources.
- Speculative execution.
- The generality of the model is theorically applicable to different level of abstraction and in different frameworks: from web service based computation, to mobile scenarios with distributed computation, to parallel dsp programming and, a possible difficult but challenging task is to port the methodology to reconfigurable Hardware in order to exploit the innumerable execution units and their reconfigurability.

SECTION 5.

ONGOING WORK AND CONCLUSIONS

This section briefly describes work that is still progressing and that for this reason could not be properly characterized with the same level of detail of the other arguments, but that is important to mention. Also if my work has been mainly focused in the research, modeling and testing of more or less expressive semantic formalisms applied to an extremely heterogeneous scenario. I had also the opportunity to work and make architectural choices in an important activity that is still progressing: a multi smart space software architecture

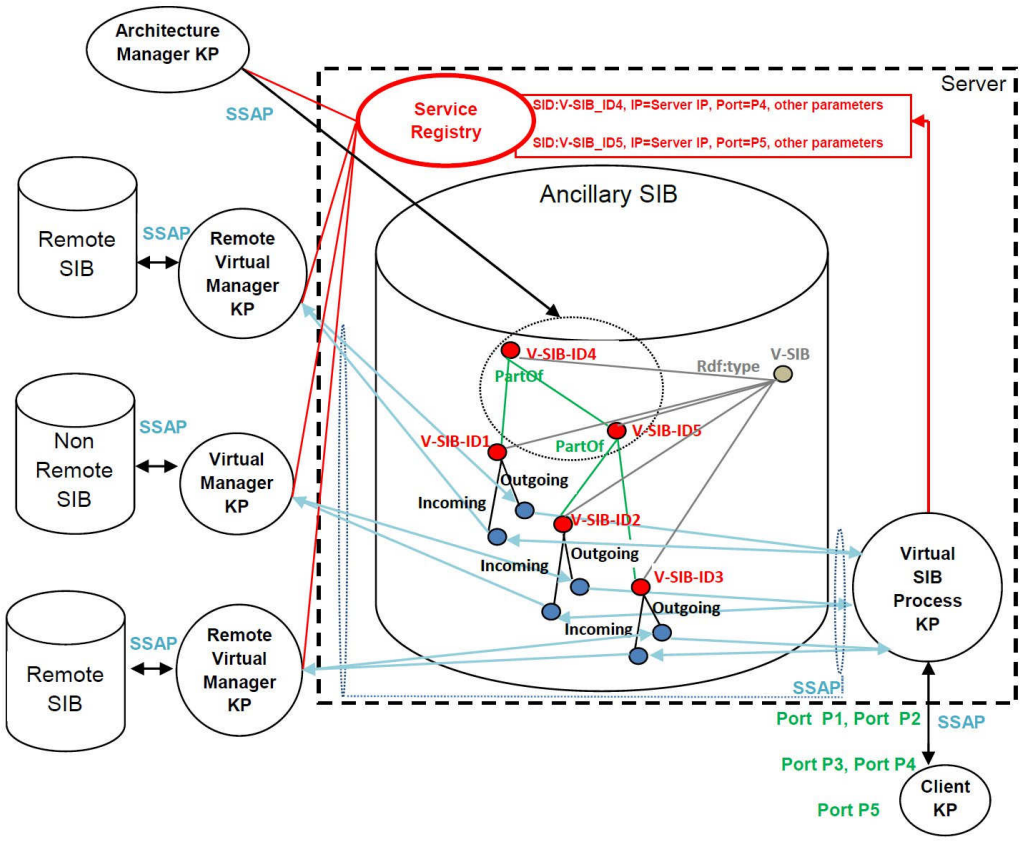


Figure 4.1: High level view of the Multi SIB software architecture for supporting discovery, remote smart spaces and logical aggregation of physically distinct smart spaces.

Figure 5.1 shows the HW/SW infrastructure of a platform for smart environments involving multiple SIBs. During the SOFIA project the demonstrators and the reference

architecture often involved one SIB leaving the multi smart space scenario with distributed knowledge as a future objective. In the last period of my PHD I helped in defining the software architecture and the behaviour of some of the represented modules, that, in their whole constitutes a Multi-SIB infrastructure. The most important features provided by the platform are

- Discovery mechanism through a web service: the KPs does not need anymore to know exactly the connection details of the SIB they want to interact with, but they simply need to know its features, which are semantically represented in the SIB profile definition. In this way the KPs can be coded in a more generic way an can be independent from the specific smart environment in which they run. With the discovery capability a KP can for example choose to connect to a SIB supporting a precise query language, providing a definite quality of service, or for which a certain additive functionality is present thanks to additive plug-ins or experimental software modules
- Remote connection: the KPs are able to discover and join SIB also if their IP is not directly reachable from their sub-network. This situation happens for example in a medical scenario where the KP of the doctor runs in the hospital and wants to join the SIB of a patient which instead runs on her mobile device. The fact that the SIB on the mobile device is connected to the internet through a proxy or a radio base station make impossible for the doctor KP to directly connect to it through TCP. The software and semantic artifacts represented in Fig 5.1., in particular the mapping SIB, the remote virtual manager, the virtual SIB Process KP and the instances in the semantic graph, allow to reach the patient SIB in an indirect way. The virtual SIB Process, in fact, take the responsibility of being, for the doctor KP as a real reachable SIB. It forwards the SSAP requests it receives in first instance to the mapping SIB and then to the patient SIB, by the help of the virtual manager KP running on the patient device. The virtual SIB process take also care of the SIB responses, which arrive to the mapping SIB, by sending them back to the doctor in a transparent way. The role of the instances in the mapping SIB is to allow the core services running on the platform to route the information to the correct host.
- SIB logical aggregation: the same mechanism used for SIB virtualization can be used with not much modifications and with a simple extension of the ontology in the mapping SIB, i.e. the addition of the V-SIB instances and of the part_of property, to create virtual SIB process which don't emulate a single SIB, but the union of the content of multiple SIBs. This is typically useful when multiple SIBs are necessary to cover a wide area that should be considered in its whole by applications. The additional overhead is loaded on

the Virtual SIB process which have to construct the tree of SIBs constituting the aggregation, separately forward the requests and, finally aggregate the responses.

This software architecture have a big potential considering all the possible use cases in which can be applied, but has still to be well characterized from a performance point of view. The subscriptions in aggregated scenarios generate a complex set of events to be managed and present a not negligible overhead, but are a powerful instrument in slowly variable scenarios, because they provide totally new functionalities

5.1. Conclusions

In the framework of pervasive computing and semantic smart environments many challenges exist and research is still working hard on different levels of abstraction. The work carried out during this PHD has been mainly focused on the semantic description of relevant context attributes of different nature: starting from sensor data and identification to arrive to very abstract context like access control, and computational flow. For every modeled contextual domain many choices where available, the possible solutions have been analyzed, compared and then validated with demonstrative software or with the realization of hardware prototypes. The SOFIA project, providing the Smart M3 semantic context platform and many collaborators from different countries, has been a perfect framework to develop new methodologies and criteria of software programming. The semantic Web technologies have revealed, as expected, optimal features of reusability, extendibility and in particular, have given the instruments to make possible real multi-industry-academic scenarios. The experience matured during the project allowed me to start from a knowledge on the theory of semantic web technologies and to arrive to implement or project software architectures and semantic web services in a natural way. The good practices and the lessons learned will be for sure an important added value in my culture, if, as it seems, the trend followed by industry and research, brings toward a world of services characterized by distribution of information and computation, context awareness and machine interpretability through the use of semantics.

References

[1] M. Weiser, “The Computer for the 21st Century”, Scientific American, September 1991.

[2] A. K. Dey, “Understanding and using context”, in Personal and Ubiquitous Computing, 5(1):4-7, 2001.

[3] R. J. Bayardo, D. Gruhl, V. Josifovski, and J. Myllymaki, “An Evaluation of Binary XML Encoding Optimizations for fast Stream based XML Processing”, in Proc. Int. World Wide Web Conf., pages 345–354. ACM Press, 2004. ISBN 1-58113-844-X. doi: http://doi.acm.org/10.1145/988672.988719.

[4] Oldes Project: http://www.oldes.eu/.

[5] Semantic web: http://www.w3.org/2001/sw/.

[6] K. Rohloff, et al. ,”An Evaluation of Triple-Store Technologies for Large Data Stores”, In: On the Move to Meaningful Internet Systems 2007: OTM 200 Workshops, LNCS vol. 4806/2007, pp 1105-1114.

[7] X. Shi, “Sharing service semantics using SOAP-based and REST Web services”. IT Professional, 8 (2). 18 - 24.

[8] M. Allman, “An evaluation of XML-RPC”, ACM SIGMETRICS Performance Evaluation Review, vol. 30, no. 4, p. 2 - 11, March 2003.

[9] S. Vinoski, “CORBA: Integrating Divrse Applications Within Distributed Heterogeneous Environments” IEEE Communications Magazine, vol 14, February 1997.

[10] SOAP Tutorial: http://tele1.dee.fct.unl.pt/rit2_2009_2010/teo/soap.tutorial.pdf.

[11] UDDI: http://en.wikipedia.org/wiki/UDDI.

[12] Web Tools: http://wiki.eclipse.org/Category:Eclipse_Web_Tools_Platform_Project.

[13] Q. Wang,et all, “Semantic Web Services based Data Exchange for Distributed and Heterogeneous Systems”, in Enterprise interoperability III, ISBN 978-1-84800-221-0, DOI 10.1007/978-1-84800-221-0_25.

[14] J. Kopeck´y, T. Vitvar, C. Bournez, J. Farrell, “Semantic Annotations for WSDL and XML Schema”, IEEE Internet Computing, 11(6):60–67, 2007.

[15] Jini Technology: http://www.jini.org/.

[16] Allegro graph: http://www.franz.com/agraph/allegrograph/.

[17] H. He, A. K. Singh, “Graphs-at-a-time: Query Language and Access Methods for Graph Databases”, Proc. of SIGMOD, 2008.

[18] J. Cheng, Y. Ke, W. Ng, and A. Lu, “Fg-index: towards verification-free query processing on graph databases”, In Proc. of SIGMOD, pages 857 - 868, 2007.

[19] N. Ryan, G.Raffa, P. Mohr, D. Manzaroli, L. Roffia, M. Pettinari, L. Sklenar, L. Stefano, and T.S. Cinotti. “On the Integration of Location Based Systems in Tourism and Cultural Heritage”, in D. Pletincx, editor, EPOCH Workshop (Brussels, Belgium, November 2006)., November 2007.

[20] N. Ryan, P. Mohr, D. Manzaroli, G. Mantovani, S. Bartolini , A. D’Elia, M. Pettinari, L. Roffia, L.Sklenar, F. Garzotto, T. Salmon Cinotti (2008), "Interoperable multimedia mobile services in cultural heritage site", in EPOCH Conference on Open Digital Cultural Heritage Systems, Rome, 2008, edited by David Arnold, Franco Niccolucci, Daniel Pletinckx, Luc Van Gool..

[21] EPOCH project: http://www.epoch-net.org/.

[22] N. Baker, M. Zafar, B. Moltchanov, M. Knappmeyer, “Context-Aware Systems and Implications for Future Internet”. In Future Internet conference and technical workshops, Prague, Czech Republic, May 2009.

[23] L. Roffia, L. Lamorte, G. Zamagni, S. Bartolini, A. D’Elia, F. Spadini, D. Manzaroli, C. A. Licciardi, T. Salmon Cinotti, “Personalized Context Based Services for Dynamic User Groups”, 2nd International Workshop on Social Aspects of Ubiquitous Computing Environments (SAUCE2009, held in conjunction with the 5th IEEE International Conference on wireless and mobile computing, networking and communication (Marrakech, Morocco, October 12-14, 2009).

[24] conteXtML: http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html .

[25] SOFIA Project: http://www.sofia-project.eu/.

[26] Friedman-Hill E. “Jess In Action: Rule-Based Systems in Java”, Manning Publications Co.; CT: 2003.

[27] C KPI: http://sourceforge.net/projects/kpilow/.

[28] C# KPI: http://sourceforge.net/projects/m3-csharp-kpi/develop.

[29] Java KPI: http://sourceforge.net/projects/smartm3-javakpi/.

[30] SPARQL: http://www.w3.org/TR/rdf-sparql-XMLres/.

[31] S. Bartolini, B. Milosevic, A. D'Elia, E. Farella, L. Benini. T. Salmon Cinotti (2011) “Reconfigurable Natural Interaction in Smart Environments: Approach and Prototype Implementation” Personal and Ubiquitous Computing Journal, Sep 2011, Springer-Verlag London Limited 2011, DOI 10.1007/s00779-011-0454-5.

[32] F. Vergari, S. Bartolini, F. Spadini, A. D'Elia, G. Zamagni, L. Roffia, and T. S. Cinotti, "A Smart Space Application to Dynamically Relate Medical and Environmental Information", in Design, Automation & Test in Europe (DATE 2010). Dresden, Germany: Kathy Preas. KP Publications, 2010, pp. 1542-1547.

[33] A. M. Collins, M.R. Quillian (1969). "Retrieval time from semantic memory", Journal of verbal learning and verbal behavior 8 (2): 240–247. doi:10.1016/S0022-5371(69)80069-1.

[34] T. Berners-Lee, J. Hendler. O. Lassila "The Semantic Web". Scientific American Magazine. (May 17, 2001).

[35] Dublin Core: http://dublincore.org/.

[36] E-Culture Demonstrator: http://challenge.semanticweb.org/.

[37] R. Rosati, "Integrating ontologies and rules: Semantic and computational issues", in: P. Barahona, F. Bry, E. Franconi, N. Henze, U. Sattler (Eds.), Reasoning Web, LNCS, vol. 4126, Springer (2006), pp. 128–151.

[38] A. Krisnadhi, F. Maier, P. Hitzler, "OWL and Rules", In: A. Polleres, C. d'Amato, M. Arenas, S. Handschuh,P. Kroner, S. Ossowski, P. Patel-Schneider, (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 382–415. Springer, Heidelberg (2011).

[39] G. Antoniou and F. van Harmelen, "A Semantic Web Primer", Cambridge MA:MIT 2004.

[40] DTD: http://it.wikipedia.org/wiki/Document_Type_Definition.

[41] J. Clark, M. Murata, "RELAX NG specification" (EDS). 2001. Available at http://www.oasis-open.org/committees/relax-ng/spec-20011203.html.

[42] I. Horrocks, O. Kutz, U. Sattler, "The Irresistible SRIQ". In Proc. of OWL: Experiences and Directions, 2005.

[43] J. Honkola, H. Laine, R. Brown, and O. Tyrkk¨o, "Smart-M3 information sharing platform," in The 1st Int'l Workshop on Semantic Interoperability for Smart Spaces (SISS 2010) in conjunction with IEEE ISCC 2010, Jun. 2010.

[44] F. Baader, U. Sattler, ""An Overview of Tableau Algorithms for Description Logics", Studia Logica 69(1), 5–40 2001.

[45] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible SROIQ, in: Proceedings of the KR 2006, Lake District, UK, 2006.

[46] Protegè: http://protege.stanford.edu/.

[47] Toolbraid composer: http://www.topquadrant.com/products/TB_Composer.html.

[48] F.A. Lisi, "Building Rules on Top of Ontologies for the Semantic Web with Inductive Logic Programming", Theory and Practice of Logic Programming, 8(03):271–300, 2008.

[49] F. Maier, A.A. Krisnadhi, M. Krotzsch, and P. Hitzler, "A better uncle for OWL: Nominal schemas for integrating rules and ontologies", In Proceed-ings of the 20th International World Wide Web Conference, 2011.

[50] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, (May 2004), "SWRL: A semantic web rule language combiningOWL and RuleML", available from http://www.w3.org/Submission/2004/SUBMSWRL-20040521.

[51] CIDOC crm: http://www.cidoc-crm.org/.

[52] E. Oyvind, “The Exhibition Problem, A Real-life Example with a Suggested Solution”, In Literary and Linguistic Computing, Vol. 23, No. 1, 2008.

[53] GIOCA course: http://corsi.unibo.it/gioca/Pages/default.aspx.

[54] H. Chen et al., "SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications", InProceedings, International Conference on Mobile and Ubiquitous Systems: Networking and Services, August 2004.

[55] CoBrA web site: http://ebiquity.umbc.edu/resource/html/id/75/An-Intelligent-Broker-for-Pervasive-Context-Aware-Systems.

[56] H. Chen et al., "Intelligent Agents Meet the Semantic Web in Smart Spaces", IEEE Internet Computing, November 2004.

[57] CoBrA http://cobra.umbc.edu/about.h.tml

[58] A. Gangemi et al., “Sweetening Ontologies with DOLCE,” Proc. 13th Int’l Conf. Knowledge Eng. and Knowledge Management (EKAW 02), Springer-Verlag, New York, 2002.

[59] Wonder Web project: http://wonderweb.semanticweb.org/deliverables/D17.shtml.

[60] A. Franchi, L. Di Stefano, T. S. Cinotti. (2010), “Mobile Visual Search using mart M3”, IEEE symposium on Computers and Communications - First International Workshop on Semantic Interoperability for Smart Spaces (SISS 2010). Riccione - Italy. June 22, 2010. (pp. 1065 - 1070). ISBN: 978-1-4244-7754-8.

[61] S. Bartolini, L. Roffia, T. S. Cinotti, D. Manzaroli, F. Spadini, A. D'Elia, F. Vergari, G. Zamagni, L. D. Stefano, A. Franchi, E. Farella, P. Zappi, A. Costanzo, and E. Montanari, "Creazione automatica di ambienti intelligenti” University of Bologna, Patent n. BO201A000117," 2010.

[62] D. Manzaroli, P. Lacchè, M. Pettinari, L. Roffia, A. D'Elia, and T. S. Cinotti (2008) "Enhancing Social Life with Path Solvers: Rendez-vous without Constraints on Meeting Place and Time," in 4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2008), Workshop on Social Aspects of Ubiquitous Computing Environments (SAUCE). Avignon, France, 2008, pp. 490–495.

[63] S. Bartolini, B. Milosevic, A. D'Elia, E. Farella, L. Benini. T. Salmon Cinotti (2011) “Reconfigurable Natural Interaction in Smart Environments: Approach and Prototype Implementation” Personal and Ubiquitous Computing Journal, Sep 2011, Springer-Verlag London Limited 2011, DOI 10.1007/s00779-011-0454-5.

[64] Niezen G, Van der Vlist B, Hu J, Feijs L (2010), “From events to goals: supporting semantic interaction in smart environments”, in: The IEEE symposium on computers and communications, pp 1029–1034.

[65] Smart M3 Lab: http://www.moodle.unibo.it/course/view.php?id=399.

[66] D’Elia, J.Honkola, D.Manzaroli, T.Salmon Cinotti (2011), “Access Control at Triple Level: Specification and Enforcement of a Simple RDF Model to Support Concurrent Applications in Smart Environments”, Proceedings of the 4th Conference on Smart Spaces, (Springer, LNCS6869), St. Petersburg, 22-23 August, 2011 (ruSMART 2011) , pp. 63-74.

[67] E.W. Dijkstra, “Solution of a problem in concurrent programming control”, In: Communications of the ACM, vol. 8, issue 9 (1965).

[68] E.W. Dijkstra, E.W.: “Co-operating sequential processes”, In: F. Genuys, editor, Programming Languages, pp. 43-112 (1968).

[69] B.W. Lampson, “Protection”, In: Proc. Princeton Symposium on Information Sciences and Systems, Princeton University, pp. 437-443 (1971). Reprinted in: Operating Systems Review, vol. 8, no. 1, pp. 18-24, (1974).

[70] D.E. Bell, L. J. LaPadula, “Secure Computer Systems: Mathematical Founda-tions and Model”, in: National Technical Information Service, Spring (1973) .

[71] R. Sandhu, “Lattice-based access control models”, In: IEEE Computer, vol. 26, issue 11, pp.9-19 (1993).

[72] M. A. Rodriguez, J. Shinavier, “The RDF Virtual Machine”. CoRR, abs/0802.3492, 2008.

[73] M. A. Rodriguez, J. Bollen, “Modeling Computations in a Semantic Network”, CoRR, abs/0706.0022, 2007.

[74] P. Wadler, “Comprehending monads”, In 1990 ACM Conference on Lisp and Functional Programming. ACM Press, New York, NY, USA, 61{78).

[75] Java concurrent: http://www.vogella.de/articles/JavaConcurrency/article.html#futures.

[76] W. R. Sutherland, “On-line Graphical Specification of Computer Procedures” IT PhD Thesis. Lincoln Labs Report TR-405. 1966.][MORRISON, J. P. 1994. Flow-Based Programming: A New Approach to Application Development. van Nostrand Reinhold, New York, NY.

[77] R. Ebner, A. Pfaffinger, “Higher Level Programming and Efficient Automatic Parallelization: A functional Data Flow Approach with FASAN”, in Proceedings of the ParCo97 Parallel Computing Conference, 16-19 September 1997, Bonn Bad-Godesberg, Elsevier Science Publishers, Amsterdam, 1998.

[78] Currying: http://en.wikipedia.org/wiki/Currying.

[79] Haskell Map: http://zvon.org/other/haskell/Outputprelude/map_f.html.

[80] A. D’Elia, J. Honkola, V. Luukkala, S. Boldyrev, “Method and Apparatus for computational flow execution” PAtent June 28, 2011, Serial No. 13/171,065.