BOnSAI: a Smart Building Ontology for Ambient Intelligence

Thanos G. Stavropoulos
Aristotle University & International Hellenic University
Thessaloniki, Greece
(+30)2310998433
athstavr@csd.auth.gr

Dimitris Vrakas
Aristotle University & International Hellenic University
Thessaloniki, Greece
(+30)2310998885
dvrakas@csd.auth.gr

Danai Vlachava
International Hellenic University
Thessaloniki, Greece
v.vlachava@ihu.edu.gr

Nick Bassiliades
Aristotle University & International Hellenic University
Thessaloniki, Greece
(+30)2310997913
nbassili@csd.auth.gr

ABSTRACT
This work introduces an ontology for incorporating Ambient Intelligence in Smart Buildings. The ontology extends and benefits from existing ontologies in the field, but also adds classes needed to sufficiently model every aspect of a service-oriented smart building system. Namely, it includes concepts modeling all functionality (i.e. services, operations, inputs, outputs, logic, parameters and environmental conditions), QoS (resources, QoS parameters), hardware (smart devices, sensors and actuators, appliances, servers) users and context (user profiles, moods, location, rooms etc.). The ontology is instantiated and put to use at the Smart Building setting of the International Hellenic University, enabling knowledge representation in machine-interpretable form and hence is expected to enhance service-based intelligent applications.

Categories and Subject Descriptors
I.2.4. [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *Ontologies, Representations.*

General Terms
Documentation, Design, Reliability, Experimentation, Languages

Keywords
Ambient Intelligence, Semantic Web, Ontologies.

1. INTRODUCTION

Both evolution in the Web technologies and hardware have resulted in two paradigm shifts that form a new era in computing. First, Web users are increasingly finding ways to get things done, instead of just looking up content. In other words, Web Applications and Web Services have emerged, along with standardized protocols that guarantee their interoperability. Alongside, Semantic Web technologies, that have been around for a while now, provide the means for annotating resources and services resulting in new standards that enable true semantic interoperability for services. Service computing is also interlinked with Ambient Intelligence, another vision of computing where smart environments of embedded computers surrounding the user fulfill his/her needs.

This work presents an ontology for enabling Ambient Intelligence in a Smart Building, named BOnSAI (Smart Building Ontology for Ambient Intelligence). There exist already domain-independent upper ontologies (not officially proclaimed standards yet) that enable the vision of Semantic Web services. Our ontology is domain-dependent and specializes such ontologies in order to model the domain-specific concepts of the AmI application. BOnSAI takes into account much related work, and actually imports and benefits from existing ontologies.

However, BOnSAI sets off to model many more concepts required in a Smart environment. The ontology is designed for the Smart IHU ambient setting whose goal is to provide automation and energy savings at the International Hellenic University (IHU) premises. This environment is equipped with sensors and actuators (so-called smart devices) in large scale, which interact with the rest of the system using the web service interface.

The next section includes a background study that clarifies both the field of application of this ontology and existing standards. The third section presents extensive related work i.e. existing ontologies for similar ambient systems. The fourth and fifth sections present the proposed ontology in detail and its instantiation in the Smart IHU environment, respectively. Finally, future work and conclusions from this work are presented in the final sections.

2. BACKGROUND STUDY
This section presents the background work that inspired the proposed ontology. For one, Ambient Intelligence is the domain of application of the ontology. Secondly, the two dominant protocols towards Semantic Web Services are presented, as service orientation is the most essential element of the target system.

2.1 Ambient Intelligence Overview
Ambient Intelligence is a vision related to ubiquitous computing, a computing paradigm of the modern era. The advancement of

WIMS'12, June 13–15, 2012, Craiova, Romania.

technology during the last decade has brought numerous changes in the everyday life of the ever-growing number of computer users. Computers are now affordable and potent to run demanding applications. More importantly, computing capabilities in smartphones and other handheld devices are making great leaps, enabling portability of processing power. In between laptops and smartphones, netbooks and more recently tablet PCs are raising in popularity and demand.

As a result, a paradigm shift from personal to ubiquitous computing is evidently occurring. Users are gradually removing themselves from in front of the personal, desktop computer and search for computing resources in their surroundings (e.g. touchscreens, tablets, and smartphones). Apparently, ubiquitous computing (UbiComp) or pervasive computing (PerComp) is amongst the leading technological paradigms of the future. Mark Weiser coined the term ubiquitous computing and envisioned many of its attributes [1].

Another vision, related to those developments, is the one of Ambient Intelligence (AmI). Ambient Intelligence extends the ideas presented in ubiquitous computing, by adding intelligent automations and intuitive interaction. Naturally, ubiquitous computing users, as they gradually move away from the desktop, require less and less interaction with the computer systems themselves. That is accomplished in two ways. First of all, human computer interactions become intuitive and use physical means. Sensor and actuator devices can contribute to this end. There is also the tendency to incorporate chips (e.g. RFID tags) into physical everyday life objects to identify them and use them as physical means of human-computer interaction. Secondly, Artificial Intelligence (AI) is incorporated into ubiquitous systems, to predict and carry out users' tasks, assist them and provide a higher level of comfort.

Ubiquitous computing and AmI are tightly linked to Web technologies. Ubiquitous computing envisions the embodiment of microprocessors in objects surrounding the users and their interconnection is made over Web. The notion of smart interconnected devices, objects or even people is known as the Internet of Things. Another relevant term is the one of the Sensor Web, which comprises of sensor networks and their data and actuator capabilities made available over web, sometimes through a web service interface to hide device and network heterogeneity.

AmI and Sensor Web are tightly linked to Service Oriented Computing (SOC) or the Service Oriented Architecture (SOA). Service orientation is a trend in computing where data and functions are considered as services. The benefits from this shift are interoperability, platform independence, and remote access for Web Services especially. AmI systems use a wide range of interconnected devices or software services that are heterogeneous in nature, primarily because of non-standardization and heterogeneity in the market. Thus, services benefit AmI systems with abstractions and decoupling from low level functions by providing uniform access to sensor data, device functions and software services. Finally, a goal of AmI is provision of complex user tasks which can be decomposed to atomic tasks that can be satisfied by atomic services. In other words this problem can be transformed to the one of Service Composition.

The use of ontologies in AmI can aid in both service computing and knowledge representation for semantic interoperability, in general. The semantic web technologies already provide the tools for complex taxonomy modeling, mainly using the Web Ontology Language (OWL)[1]. These taxonomies, widely known as ontologies, define a formal dictionary of entities and their relationships, either hierarchical or through properties. Ontologies for different domains are constantly being designed in the hopes of enabling the Semantic Web vision. Specifically, their purpose is to describe resources as entities on the Web (e.g. web page content) in a computer readable format so that software agents, a.k.a. semantic web agents, can discover and filter that content. Reasoning can also be carried out using that knowledge. AmI systems can also benefit from such representations that enable reasoning and the use of logic.

On the other hand, adding semantic annotations on service descriptions can bring even more benefits for those systems that indeed employ web services. Primarily, service descriptions are rendered machine-interpretable, thus, semantically defined queries can be matched upon semantic service descriptions for more refined results. Additionally, semantic discovery can return results when syntactic discovery cannot. Due to the dynamic nature of ambient systems, service providers enter or leave the environment (e.g. smartphone service providers) or simply crash. When the desired services are not present, a semantic discovery agent can find alternative solutions according to reasoning conclusions (e.g. return a service that returns a relevant parameter). Clearly there is a lot of ongoing work to enable and standardize semantic web services to benefit from what it has to offer overall.

2.2 OWL-S

Some languages have flourished through work in the field of semantic annotations for services. OWL-S[2] is such a language. It is actually an upper ontology for services that strives to semantically enhance service descriptions to achieve interoperability. OWL-S derived from a previous attempt, namely DAML-S, by the same initiative. The OWL-S ontology contains three main classes that relate and describe a service: the Service Profile, the Service Process Model and the Service Grounding.

The Service Profile describes, sometimes vaguely, what the service does. Apart from that, it contains information about its inputs and its outputs. It is worth mentioning at that point that the OWL-S ontology acknowledges the fact that services may or may not have input or output data but they often have preconditions and results or effects i.e. perform some operation that changes the state of things. These notions are well known in the field of Artificial Intelligence as planning, which considers actions as operations that have preconditions and may change the world's state. Finally, services can be considered as actions and likewise be assigned a quadruple symbolized as IOPR or IOPE which stand for Inputs, Outputs, Preconditions and Results or Effects.

The Process Model unfolds the exact process of a service in workflow form. Using it, clients are able to analyze it, monitor its execution or when performing composition, orchestrate the execution of different services. In other words, the workflows of services can be intertwined into a complex composite service. An example of such work can be found in [2]. The Process Model itself, as a workflow, can be used to describe composite services.

---

[1] The Web Ontology Language, OWL:
http://www.w3.org/TR/owl-features/

[2] OWL for Services (OWL-S):
http://www.w3.org/Submission/OWL-S/

The Service Grounding, as the name implies, is the actual instantiation of a service, quite similarly to service binding in WSDL. It provides concrete implementation information for the services so that it can be invoked by clients. OWL-S is not restrictive and can support any type of Grounding but naturally it inherently already supports WSDL mapping.

Finally, OWL-S is complementary to WSDL. WSDL syntactically defines inputs and outputs of services while OWL-S adds semantic descriptions. OWL-S does not include implementation details to actually invoke a service, but its Service Grounding can map to such an implementation e.g. WSDL. OWL-S is also fully compliant with SOA, being compatible with known service registries (e.g. UDDI). The services can thus be advertised, discovered and invoked by clients. Although it was used widely in research or extensively influenced very similar custom approaches (in essence alternative service ontologies that extend it), OWL-S did not make it as a W3C recommendation.

2.3 SAWSDL

Another approach to adding semantic annotations in web service descriptions is SAWSDL[3] (Semantic Annotations for WSDL and XML Schema). SAWSDL is a W3C recommendation since 2007. Unlike OWL-S, this standard directly extends WSDL documents by adding ontology references in the document itself. These references to ontology models take the form of XML attributes, named *ModelReferences* and directly annotate interfaces, operations and faults within a WSDL file. *ModelRederences* can also be applied on XML Schema types, elements and attributes.

SAWSDL also offers two attributes to map data from semantic models to XML and vice versa. The *liftingSchemaMapping* specifies a transformation for an XML element to semantic data. The *loweringSchemaMapping* does the opposite, as it specifies a transformation for an element of a semantic model to XML data. These transformations are usually specified in well-known languages like XSLT, SPARQL and XQuery.

In comparison to OWL-S, SAWSDL follows a completely different approach. OWL-S is a top-down solution to the problem of semantic descriptions. OWL-S files are upper ontologies meant to include WSDL files, for instance, as groundings. Systems that need more domain specific concepts extend the ontology, raising the complexity of this approach. SAWSDL follows a bottom-up approach. WSDL files are enhanced, not replaced entirely as descriptions. The annotations within SAWSDL files can contain references to any number of ontologies to satisfy domain-specific demands. This approach is simpler and more straightforward to use.

3. RELATED WORK

Many existing approaches on Ambient Intelligence have introduced their own ontologies for enhancing knowledge interoperability in such systems. Most of these systems are indeed service-oriented. One use for ontologies in such systems is plain knowledge representation, querying and reasoning for data. Another use which is a more modern tendency is service annotation to enhance the discovery, (hence also matching and selection) and composition of web services. Towards that goal, the ontologies specify and disambiguate concepts for inputs, outputs, preconditions and effects of service operations. Several of them extend OWL-S. Then the ontology can be used just like OWL-S as a service description itself (it is just an OWL-S version with domain-specific classes). An example of such an ontology can be seen on Figure 1, where OWL-S (Service, Profile, Grounding) is extended with context classes [3]. The SAWSDL approach is yet not so popular in AmI. However, we believe that as the Sensor Web evolves, more and more web services will be available and the management of a large set of WSDL descriptions will call for semantic annotations in the form of SAWSDL, which is the most straightforward approach.

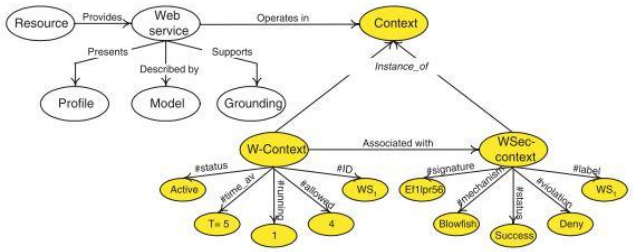

Figure 1. OWL-S extension by Maamar

All in all, whether just for knowledge representation or for service annotation and interoperability, ontologies are needed to specify domain-specific concepts. One can argue that true interoperability can only be reached if all systems use the same ontology. Currently each system has its own taxonomy and dictionary, so interoperability is achieved for all clients within that system. The standardization of an ontology in each domain, in that case the AmI domain, would offer cross-system interoperability. Some relevant systems indeed make steps towards that goal by importing other existing ontologies [4], [6].

An extensive review of relevant approaches can be found at [2]. The relevant ontologies found, use various specification languages ranging from plain XML, to DAML and OWL. Regardless of language, it is apparent that they share many concepts and hence standardization is possible. Furthermore, the numerous common concepts on these ontologies can be categorized into four main clusters depending on what they regard: functionality, context, QoS and hardware. Service related concepts can also be regarded as functionality-related. The following subsection presents some of these common concepts and the next subsection presents details of such ontologies.

3.1 Common Concepts

3.1.1 Functionality

The main purpose of functionality classes is to model all system's operations. For service-oriented systems this process is significant as services description will later be annotated in reference to these classes. Examples of such concepts can be found in [7] and [8] that define types of input and output, binary, physical data etc. Services themselves can also be included in the functionality category e.g. in [9], [11], [12]. Beauches et al. [13] define capability and data structures. Chakraborty et al. [14] define a hierarchy for groups of services. Ibrahim give some examples of service-concepts or actions, like "Take-A-Picture" and the related output "Image". Messer et al. [15] define classes for Media Types and Capabilities, as media is one of the most popular domains in AmI. The Hydra-middleware project extensively model functionality of each device. Then these descriptions are used by

[3] Semantic Annotations for WSDL (SAWSDL) Specification: http://www.w3.org/TR/sawsdl/

their ontology service compiler, Limbo [16], to dynamically generate services for embedded devices.

Ontologies for the Internet of Things (IoT Ontologies) are proposed in [17]. Three ontologies are proposed. A device ontology describes the hardware which is mostly sensors for the Internet of Things. A domain ontology defines Physics and Mathematics that can be used to describe and clarify service (and device) functionality. Finally an estimation ontology contains mathematical models that can be assigned to services. All in all, the ontologies extensively model most of the functionality that can be met in a service-oriented and sensor-enabled environment such as Internet of Things implementations.

3.1.2 Context-awareness

Context can be defined as a set of statements about a person, a place or an object [17]. Context-awareness refers to the ability of a system or method in general to perceive and take into account the current environment or world state. Naturally, this property adds up to AmI systems dynamicity as results become case-sensitive like service compositions according to user environment. Ontologies are the main tool for modeling context in research in order to achieve context-awareness.

Concepts related to context are mainly location, user-related, environment and time as seen clearly in [19]. Then a context can be a set of these parameters associated with an entity. E.g. a user can have a context of current time, location and room temperature. Context can also be the product of reasoning (e.g. absolute coordinates can point out a certain location or room) which is the case in [20]. Others include classes to distinguish context belonging to different entities. Iacob et al. [21] demonstrate a fragment of a domain ontology that includes context concepts. The Context class is associated with a GeoLocation a GSMCell and Schedule, which confirms the fact that usually context has to do with location and time. Further on, Users can separately be connected to a GSMCell, a GeoLocation, Home, Office and Schedule.

In [11] services, environment and user can all have context. Quite similarly in [12], context can be environmental, service-related or resource-related. Another context-related notion is the notion of Event (found in [9]). Santofirmia et al. [9] also regard context as a whole world state consisting of static and dynamic facts. A knowledge base and a semantic model are used for the world representation. Finally, Vallee et al. [8] puts context models into use by regarding context while performing service composition.

3.1.3 Quality of Service – QoS

Quality of Service considers added value parameters like service latency, response time or costs in general. Systems that take these parameters into account, offer optimum solutions. A typical approach to consider QoS is using an ontology that defines QoS concepts. Then, service QoS parameters can be registered and reviewed at discovery-time as service meta-data. Davidyuk et al. [22] indeed includes QoS metadata – non-functional properties of services in an upper ontology. Thus each service has known non-functional or QoS properties, that serve as selection criteria.

3.1.4 User Preferences

User preferences is another parameter similar to context but of distinct meaning. While context is case-sensitive, user parameters change less often and always accompany a certain user, characterizing him/her. The system also needs to consider these preferences alongside context during e.g. service composition. Preference related concepts can be Mood, PreferenceProfile and Role [20]. In practice, the system can be aware that a certain user always prefers mobile services, or black and white printing to color. In [20], users can set a streaming audio and video quality preference in a scale of Low, Medium or High. The system usually checks whether preferences can be met according to what is available at the time, and make settlements.

3.1.5 Hardware

AmI systems make use of unique and cutting-edge hardware to offer innovative and intuitive user interfaces, interconnect with physical objects and surround users. Hardware is also domain dependent. It can range from media streaming devices, to home automation and wellness devices. DEHEMS is a home energy efficiency application that introduces a wide ontology of home appliances and their energy profiles [23]. Messer et al. [15] define their multimedia devices. Also, hardware provides computing resources that are modeled in [12] and [24]. Other ontologies containing hardware concepts can be found in [25], [9] and [21].

3.2 Ontologies for Ambient Intelligence

3.2.1 The GAIA Ontology

The ontology infrastructure of the GAIA system is described in detail in [19]. Ontologies of the GAIA system mainly define either context information or entities in the environment. Context is represented in a predicate form that inherently suits the planning component (world states in Planning are sets of facts – predicates). E.g. the predicate Location (Chris, in, Room 2401) defines knowledge about a person's location. Other context-related predicates can be classified in physical context (location and time), environmental context (weather, light and sound levels), informational context (stock quotes, sports scores), personal context (health, mood, schedule, activity), social context (group activity, social relationships, whom one is in a room with), application context (email, websites visited) and system context (network traffic, status of printers). Ontologies are used to type-check arguments of these predicates (e.g. Chris and Room 2401). On the other hand entity-related ontologies define taxonomies and relations between devices, services, applications and users. GAIA also incorporates an ontology server that enables incremental addition of new ontologies. Classes and properties are then merged with the existing ones. However, the GAIA ontologies were not reused, because they contain many more concepts and relationships than needed for our purposes. In this work we strive to include a minimal set of concepts, in order to ease the use of the ontology. Furthermore, compared to GAIA, our ontology focuses mainly on the discrimination between the sensor and actuator role of devices (and services) in Smart Buildings and their corresponding functionality. Finally, another subtle reason for not re-suing the GAIA ontologies was the they could not be found online.

3.2.2 The DEHEMS Ontology

The DEHEMS project (Digital Environment Home energy Management System) proposes an ontology to address the knowledge representation and reasoning issues of the DEHEMS infrastructure. The project's overall goal is energy consumption awareness across multiple households and energy advice provisioning. This group of households can be considered as a Smart City. The DEHEMS infrastructure enables collecting data from households, inserting them into the ontology model and reasoning about them to warn about abnormal consumption of

energy and provide advice (tips). The proposed ontology, namely "the Home Appliances Ontology", is a SUMO[4]-compliant ontology that focuses on modeling extensive energy specifications about electric devices. The most generic concept of the ontology is the ElectricHomeAppliance concept. Other main concepts, like HomeOfficeAppliance, EntertainmentAppliance, Device and BodyCareAppliance are its subclasses. Each one of these leads to a different subtree in the hierarchy that can be seen as a cluster of relevant concepts. A basic property that gets a lot of focus in this ontology is the StarInfo slot, found in various appliance-concepts. It provides a star energy rating for each device as an index of energy efficiency. The ontology is evaluated for two reasoning problem instances: checking whether an energy consumption value is within normal range and providing tips. The DEHEMS ontology, however, is unsuitable for AmI systems, as it models extensive information about appliances only, and leaves out key-concepts like Service, Sensors, Actuators (i.e. other kinds of Devices), Input and Output parameters, which are vital for most of the functionality modeling. On the other hand, it could, be used to extend the branch of home Appliances in specific cases. In our case this was not required. Additionally, the ontology was not found online.

3.2.3 The CoDAMoS Ontology

Preuveneers et al. [20] define an ontology[5] in OWL that along with a context management system, is able to adapt services based on context. Concepts defined in this ontology revolve around the four main concepts of User, Platform, Service and Environment. These upper classes are interconnected as shown on Figure 2.

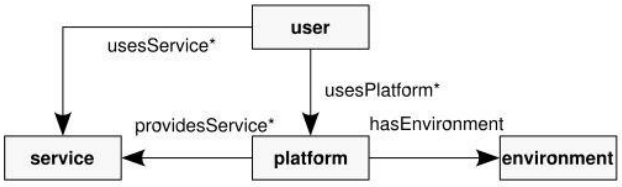

Figure 2. CoDAMoS upper ontology classes

At a glance, Environment has location, time and environmental condition data. A user has mood, profile, role and tasks (to complete) that include activities and use services in turn, as seen on Figure 3 shows the service-related classes, inspired from OWL-S and linked to CoDAMoS classes task and software. The OWL-S inspired concepts and linkage to the rest of CoDAMoS can be seen on Figure 4.

A platform provides hardware that relates to resources (power, memory, cpu, storage and network) and i/o devices, and software that provides services. Software can be an operating system, a virtual machine, a middleware or a rendering engine. Finally, the environment has location time and environmental conditions, as seen on Figure 5.The four main concepts are interconnected in many ways: a service requires a platform, a platform has an environment. The ontology for services is in fact OWL-S (that provides service profile, model and grounding) and is interlinked with the rest of the ontologies as tasks use services and software provides services.

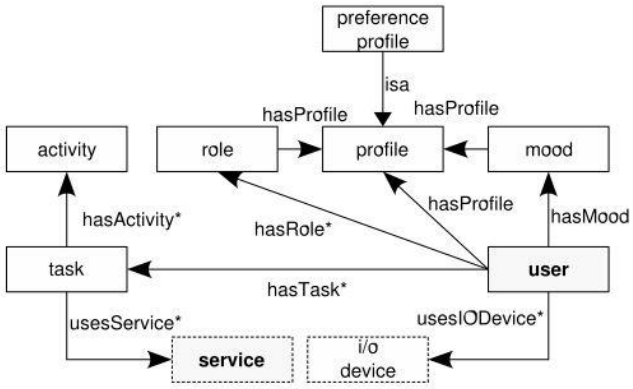

Figure 3. CoDAMoS user-related classes

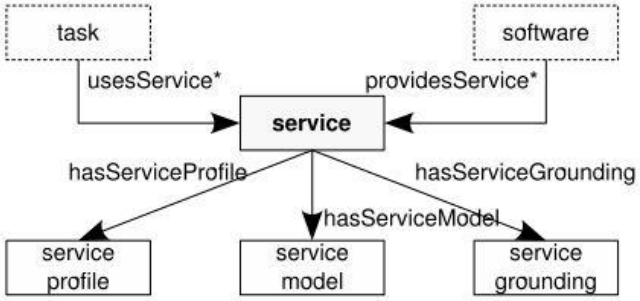

Figure 4. CoDAMoS service-related classes, OWL-S

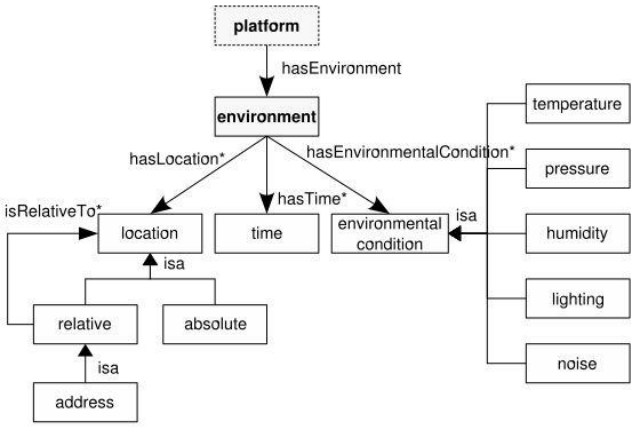

Figure 5. CoDAMoS environment-related classes

The proposed ontology directly imports the CoDAMoS ontology, and makes use of a lot of its classes. In detail, environmental condition and its subclasses are used and enriched, service is imported from OWL-S, and users can have moods and profiles. We consider the notion of context which here is named environment. However, we needed sensor and actuator oriented classes to represent hardware, while the CoDAMoS platform hierarchy is left unused. The users also do not need the property

[4]The Suggested Upper Merged Ontology (SUMO) http://www.ontologyportal.org/

[5] The CoDAMoS context ontology:

http://distrinet.cs.kuleuven.be//projects/CoDAMoS/ontology/context.owl

usesService and hasTask. Another main focus of our ontology is specifying the functionality of services (IOPEs).

Indeed the CoDAMoS ontology provides many useful concepts, but for our case we need to further specify service and device functionality. We need a hierarchy for smart devices and appliances. Furthermore, OWL-S here is not imported (but rather its concepts are redefined) so CoDAMoS-based systems are not really compatible to other OWL-S based systems. We also make a subclass of the OWL-S:Service, rather than linking properties to the OWL-S:Service directly, to ensure that other OWL-S based systems are not forced to include our concepts as well.

3.2.4 The OntoAMI Ontology

Santofimia et al. [9] propose a general simplistic semantic model for universal use across AmI applications. This only includes basic concepts that cannot be left out in AmI which namely are "Service", "Device", "Event", "Action", "Object" and "Context". Figure 6 shows the relationship between these upper classes. Furthermore, as a showcase, they map this model to an OWL ontology, adding more domain-dependent concepts and relationships suited for their intrusion-detection implementation, like "Announce" and "Hazard".

The proposed ontology indeed conforms to many aspects of OntoAMI after only slight alterations. Devices do not actually provide services (sensor and actuators do not have embedded web servers) but are associated with them (Services expose certain device functions). Instead, Servers do provide Services. Actions are indeed performed by Services (Operations to be exact), while Actions are indeed linked to the Objects they affect (which in our case are facts – a more general notion). The notion of Event is left out of BOnSAI and Context is not associated with it.

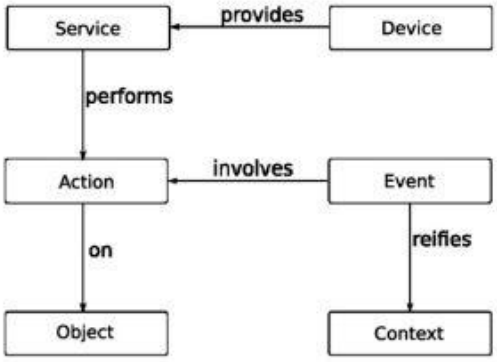

Figure 6. OntoAMI upper classes and their relationships

4. BOnSAI

The BOnSAI ontology (a Smart Building Ontology for Ambient Intelligence) is designed to enable the vision of Ambient Intelligence in large-scale service-oriented pervasive systems. The engineering method used was the Ontology Development 101 Guide [10], with emphasis on the intended use and application of the ontology (which is service interoperability). BOnSAI was implemented in OWL at Protégé.

BOnSAI classes can be categorized in context-related, service-related, hardware-related and functionality-related. Indeed, background study validates this categorization. Due to extensive work on this field, BOnSAI, takes advantage of, and imports existing ontologies available online. However, numerous concepts are either left out completely by existing ontologies or need modification.

While addressing the modeling of most AmI systems, BOnSAI's implementation purpose is the modeling of the Smart IHU system. BOnSAI sustains the generality of most AmI-related ontologies, as it includes all common general-purpose classes. Still, it is less generic as it contains many more domain-specific classes. A small number of extra classes is added then for the specific purpose of modeling the Smart IHU infrastructure. Finally, the classes are instantiated to represent Smart IHU existing entities. The ontology is thus put to use and verified during that process. It can be extended and instantiated in the same manner for similar AmI systems.

BOnSAI can be located online at the LPIS lab ontology repository[6]. Figure 10 presents an extensive class diagram of BOnsSAI. Subclasses are shown below their supercalsses. ObjectProperty relationships are also marked with an arrow. The two other ontologies that BOnSAI imports are shown within rectangles, and their relationship with BOnSAI concept is eveident through subclass and object property relationships. Please note that a version of CoDAMoS with working URIs had to be re-hosted at our website so that it can be imported.

In the next subsections, each cluster of BOnSAI classes is thoroughly presented and explained. The Smart IHU-specific classes are presented separately in a separate subsection. In the final subsection, the instantiation of BOnSAI with Smart IHU entities is shown.

4.1 Hardware-related Concepts

Hardware-related concepts should support both energy-awareness, functionality and service-orientation in the system. All Hardware is mainly divided into Appliances and Devices, which differ in their ability to offer services. The only thing common to Devices and Appliances is that they both have a Location so they can be located according to the place they function and affect (Location is detailed in the Context-related section).

Appliances are the non-service-enabled electric appliances in the building such as Radiator, Lighting, AirCondition, Printer, etc. The Appliance branch of the ontology can be extended to model extensive knowledge on appliance energy properties e.g. by using the DEHEMS ontology as long as the latter is standardized by SUMO and be made online available. Finally, interlinking with functionality properties, these appliances have the ability to alter the state of EnvironmentalParameters. This is modeled by the "afftects" property. E.g. Radiators and airconditioning affect Temperature and Lighting affects Luminance. The most interesting case of Appliances is the Computer, which provide computing resources (QoS-related). Servers are a subclass of Computers that are linked with Services via the "hostsService" property (inverse property of "hostedByServer"). All Appliances have a PowerState (which can be on or off), that can be affected by services. Additionally, their average power consumption is registered via the "consumes" property (with a PowerConsumption range), to facilitate making energy-saving decisions.

[6] BOnSAI online at ISKP repository:

http://lpis.csd.auth.gr/ontologies/ontolist.html

The Device branch is actually meant for the service-enabled, often so-called smart devices. These are mostly Sensors and Actuators, in general. BOnSAI also includes the classes of MultiSensor (i.e. a Sensor array) and SensorActuator, which is meant for devices of dual purposes. Devices are exposed by Services (and vice versa Services expose Devices), forming a Sensor Web. The variety of communication protocols is also modeled by the corresponding classes and categorized into Wireless, Wired, PLC protocols etc. The devices have also certain functionality. Sensors return certain parameters and Actuators support various Actions. This is further modeled in the functionality section.

For the case study of Smart IHU, all classes are instantiated and the extra class of SmartPlugs is added. SmartPlugs are a SensorActuator subcase that affect state of Appliances (turn them on or off) and also read Power and Energy consumption values. Additionally, a subclass of a specific brand and models of SmartPlugs purchased for the system is added along with their data properties. Additionally, another kind of Sensor is the SmartClamper, which monitors the whole building's consumption. For the purposes of our system, specific brand classes are added below SmartClamper and MultiSensor.

What is also interesting is that Smart IHU follows the service-oriented architecture. Thus, although devices directly function on the environment, and that knowledge is modeled, they are not invoked to do so. Rather, the services associated with them are sought, discovered, invoked and composed. Finally I/O devices do not play an important role on the system's function and are not currently modeled.

4.2 Context-related Concepts

Context is a very popular notion among ambient systems. Context-awareness adds much to the dynamicity of these systems as their behavior varies depending on circumstance. BOnSAI models context as a set of a single Location, a set of environmental parameters and a timestamp. An instance of the Context class can be associated with a User at a time. That enables taking decisions for users depending on context. E.g. different service composition can be delivered to the same user depending on his location and/or environmental settings. Location as a context element, also associated with every piece of hardware, can range from a specific point to a room, floor or to the whole building. Rooms also belong to floors, and floors in buildings. Another form of context of more explicit nature is the user's mood or his preference profile. This kind of explicit knowledge is often modeled in related work e.g. in the CoDAMoS ontology. This data is left out of BOnSAI for the sake of simplicity but can easily be added, in case it is needed for e.g. planning algorithms.

4.3 Functionality-related Concepts

Functionality elements are key to describing the operation of an ambient system. BOnSAI models functionality using two base classes: Parameters and Actions. Parameters are further classified into EnvironmentalParameters, Energy, PowerConsumption and Time. EnvironmentalParameters vary among Temperature, Humidity, CO_2 Level, Luminance and Pressure, similarly to CoDAMoS:EnvironmentalConditions. Parameters are useful to express the functionality of both Devices and Services. Specifically, Sensors directly link to Parameters(Sometimes EnvironmentalParameters) via the "returnsParameter" property.

Actuators are indirectly linked with parameters that they affect. Specifically, Actuators have Actions and Actions have Facts as results via the causesFact object property. Facts are useful for modeling logical conditions. Then, classes such as PowerState (linked to Applliances via the hasPowerState property) and EnvironmentalParameter (and its subclasses e.g. Temperature, Pressure etc), can be seen as Facts. To demonstrate the use of causesFact, a Thermostat device could have a causesFact property, with a value of a Temperature instance. That means that the Thermostat sets the temperature to that instance's value. Likewise, a light dimmer actuator that sets a luminance value for a room, can be modeled with a causesFact property and a Luminance instance as value.

For the purposes of modeling Smart IHU, an extra subclass of Action, the SwitchAction is specified. SwitchAction restricts the "causes" property to have a range from the class PowerState instead of its superclass, Fact. Moreover, SmartPlugs, as SensorActuators also return Energy and PowerConsumption of appliances. The Appliance that they affect is linked with the "attachedAtAppliance" property. The other devices are also linked with the parameters they return. SmartClampers return Energy and Power. The Sensors of the MultiSensor array returns one of Temperature, Humidity or Luminance.

As mentioned, parameters are linked both with devices and with the services that expose the same functionality, in a different way, and that is presented in the following subsection.

4.4 Service-related Concepts

The service class of BOnSAI is the most significant for service-oriented ambient applications. To comply with and benefit from existing standards in the field, the service class is imported from the upper ontology for services, OWL-S. Thus, inherently from OWL-S, a Service is describedBy a ServiceModel, presents a ServiceProfile and supports a ServiceGrounding.

However, BOnSAI adds annotations to Service operations as well. In literature, the notion of service is often confused with the notion of Service Operation. BOnSAI conforms to WSDL (and SAWSDL) consideration that a Service has many Operations (also many bindings etc.). Each Operation has different Inputs, Outputs, Preconditions and Effects. In an environment with Actuators, the concept of Effects is essential to model and categorize different service operations. Specifically, each Operation can have Inputs, and Outputs that belong to the class of Parameter. Preconditions and Effects on the other hand are Facts, such as a certain level of Environmental settings or states of appliances. Besides, effects are a synonymous notion to Actions as they result in Facts, e.g. the act of turning off the lighting. In the SmartIHU domain, the most useful Action is the SwitchAction (subclassOf Action), which specifically turns Appliances on or off. Thus, the SmartPlug Actuator has a restriction on the causesFact property to only cause PowerState Facts. That models the change of power state of Appliances using the smart plug actuators. Operations are classified further into SensorOperations (which return data and have no Effects) and ActuatorOperations (which cause at least one Effect). This classification aids the use of services instead of devices, and hence promotes service-orientation in applications.

Thus, for the purposes of extending OWL-S functionality and further linked with other classes (e.g. the Device class via the exposed/isExposedBy property), our Service class is implemented as a subclassOf the owl-s:Service. The Service class then can have many Operations (hasOperation property) and register the devices that it is associated with (i.e. the devices that a services exposes).

4.5 QoS-related Concepts

The BOnSAI ontology finally supports the modeling of QoS properties in various ways, to support optimized solutions in e.g. service composition or service selection. Primarily, the class Resource, directly imported from CoDAMoS along with its subtree, models all kinds of common computing resources e.g. memory, CPU etc. Further on, CoDAMoS:Resource is provided by Computers and Servers. QoS properties of devices are registered in instances of the CommunicationProtocol class and its subclasses (e.g. WirelessCommunicationProtocol, Zigbee, Z-Wave e.t.c.). These instances of CommunicationProtocol contain latency, nodesPerNetwork, range nad Data Rate properties.

4.6 INSTANTIATION OF BOnSAI

The instances of BOnSAI that model the current Smart IHU implementation are made in a separate taxonomy (separate OWL file). That ensures that the BOnSAI ontology is left unchanged and remains general-purpose so that other partners can use it. The Smart IHU instantiation can be found online at the ISKP group website[7].

Here we present some interesting samples of Smart IHU instances that also go to show how BOnSAI can be handled to model mainly smart devices and services.

Figure 8 shows many operation instances of the Smart IHU WSDL web services. SwitchOn which is in focus is shown to be linked with the PlugwiseServices via the belongsToService property and hasEffect of the PowerStateON (instance of PowerState with "ON" value).

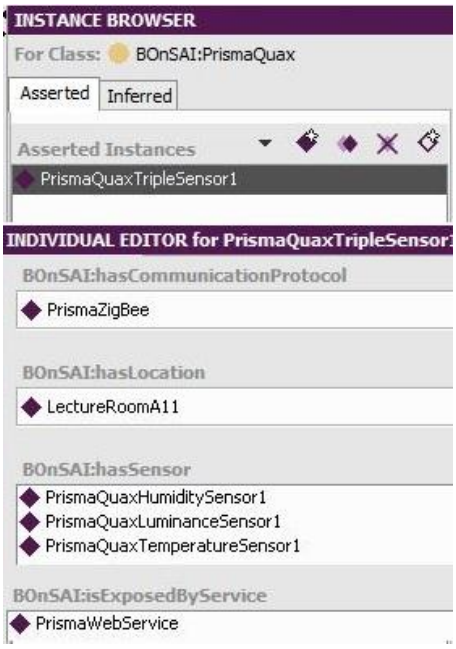

Figure 7. MultiSensor instance example

[7] Smart IHU instances:

http://lpis.csd.auth.gr/ontologies/bonsai/BOnSAISmartIHU.owl

Figure 7 shows a MultiSensor instance example of a sensor board that integrates three types of sensors. It is shown viw the hasSensor property that it embeds a Humidity sensor a Luminance sensor and a Temperature sensor. Although here it is evident by their name, these three sensors instances indeed explicitly model their returnParameter types. The MultiSensor also registers its location and the service through which users can manipulate the device. The CommunicationProtocol is a certain instance of the ZigBee protocol for smart devices.

Figure 8. SwitchOn and other Operation Instances

Finally, Figure 9 shows a SmartPlug (SubclassOf ActuatorSensor) instance sample and lots of its object properties. As an Actuator subclass, it has a performAction property which a value of SwitchAction instance (according to the restriction on that property, for SmartPlugs). This device is also actually exposed by two different service implementations. The manufacturer of this device has also incorporated his own implementation of an encrypted ZigBee communication protocol. Naturally the location is also registered. Via the AttachedToAppliance property, one can

know which appliance the smart plug affects. As a Sensor, this device also returnsParameter of type PowerConsumption.

Afterall, the instantiation of BOnSAI can be used to verify its effectiveness as it was in fact built to facilitate applications. From a modeling point of view, reasoning was used to classify various instances. Operation instances that haveOutput but cause no effect are correctly classified as SensorOperations. Likewise Operation instances that cause an Effect are correctly classified under ActuatorOperation. An application was built to parse that information and invoke the desired services, but is outside the scope of this paper Future work includes such applications for selecting (matching etc) and invoking the services.

Figure 9. SmartPlug instance example

5. FUTURE WORK

Future work is the use of BOnSAI in various heterogeneous simple and intelligent applications. Primarily, the ontology will be put to use and benefit service discovery clients such as in [26]. Dynamicity and alternative options in the results can be provided by employing reasoning on the ontology. The users can then be provided with automatic composition of services that satisfy their needs. Additionally, these composite services can be context-dependent.

Semantic discovery can be incorporated in various applications. First of all, simple applications can benefit from semantic discovery to dynamically parse, add and remove functionality. iDEALISM is a desktop application presented in [27], where users monitor and manage all the services in the building. The application can be enhanced with semantic discovery to dynamically add and remove content without user intervention.

An expert system can be built, where users can specify rules or policies to manage the Smart IHU building. A multi-agent approach can be followed for better co-ordination, through negotiations. Rules can be used to achieve both energy savings in the building and increase user comfort.

The ontology model also enables exposing the data on the web under a universal schema. Publishing sensor data under e.g. the LinkedData[8] schema would enable universal manipulation of the data by different partners.

Finally, semantic description of services can also be used for more sophisticated service composition, via planning, such as in [28]. Existing planning algorithms can be applied on semantic services, which are already much similar to actions in planning (they have well-defined preconditions, inputs, outputs and effects).

6. CONCLUSIONS

Semantic Web technologies have sufficiently advanced and now provide the tools to enhance the Web user's experience. Content can be annotated and parsed by machines thus enabling semantic search. However, as a paradigm shift occurs, web users are more concerned to get things done on the Internet rather than get information. In other words, Web Services are emerging and several technologies and protocols are already standardizing their usage. There have already been attempts to apply Semantic Web technologies on services such as the upper ontology for services OWL-S and the SAWSDL W3C recommendation. Ambient Intelligence, another emerging computing paradigm, directly associated with service-orientation also benefits from Semantic Web Services.

This work introduces BOnSAI, a Smart Building Ontology for Ambient Intelligence, to be used in the Smart IHU Smart Building environment and any other similar AmI platform. The BOnSAI taxonomy includes concepts for describing services, functionality in the system, existing hardware and enabling context awareness. It imports and extends existing work, from the CoDAMoS project and the OWL-S upper ontology for services. Service interoperability provided by BOnSAI is due to be employed by service selection, description and matching algorithms. Further on, this infrastructure will enhance the development of an expert system and a planning infrastructure in the Smart IHU system.

7. ACKNOWLEDGMENTS

This project is funded by Operational Program Education and Lifelong Learning, OPS200056 (International Hellenic University, Thessaloniki, Greece). The authors would also like to thank the undergraduate student, Theo Mylonides, for his contribution.

8. REFERENCES

[1] Weiser M (1999) The computer for the 21st century. ACM SIGMOBILE Mob Comput Commun Rev 3(3): 3–11. doi:10.1145/329124.329126

[2] Thomson G, Bianco S, Mokhtar SB, Georgantas N, Issarny V (2008) Amigo aware services, communications in computer and information science, 1, vol 11. Constructing ambient intelligence part 7, pp 385–390

[3] Maamar Z., Narendra N. C., Subramanian S.: Towards an ontology-based approach for specifying and securing Web services. Information & Software Technology 48(7): 441-455 (2006)

[8] LinkedData: http://linkeddata.org/

[4] Sheshagiri M, Sadeh N. M., Gandon F. (2004) Using semantic web services for context-aware mobile applications. In: 2nd International conference on mobile systems (MobiSys 2004), applications, and services workshop on context awareness

[5] Stavropoulos, T.G., Vrakas, D., and Vlahavas, I. 2011. A survey of service composition in ambient intelligence environments. Artificial Intelligence Review (25 September 2011), pp. 1-24. doi:10.1007/s10462-011-9283-1

[6] Bottaro A, Bourcier J, Escoffier C, Lalanda P (2007) Autonomic context-aware service composition. In: 2nd IEEE international conference on pervasive services

[7] Masuoka R, Parsia B, Labrou Y (2003) Task computing—the semantic web meets pervasive computing. In: International semantic web conference, pp 866–881

[8] Vallée M, Ramparany F, Vercouter L (2005) Dynamic service composition in ambient intelligence environments: a multi-agent approach. In: First workshop on YR-SOC

[9] Santofimia MJ, Moya F, Villanueva FJ, Villa D, Lopez JC (2008) An agent-based approach towards automatic service composition in ambient intelligence. Artif Intell Rev 29(3–4):265–276

[10] Noy N. F. and McGuinness D. L. Ontology development 101: A guide to creating your first ontology. Online, 2001.

[11] Qiu L, Shi Z, Lin F (2006) Context optimization of ai planning for services composition. In: ICEBE 06: proceedings of the IEEE international conference on e-business engineering, pp 610–617

[12] Bellur U, Narendra NC (2005) Towards service orientation in pervasive computing systems. Int Conf InfTechnol Coding Comput 2:289–295

[13] Beauche S, Poizat P (2008) Automated service composition with adaptive planning. In: Bouguettaya A, Krueger, I, Margaria T (eds) ICSOC 2008. LNCS, vol 5364. Springer, Heidelberg, pp 530–537

[14] Chakraborty D, Joshi A, Finin T, Yesha Y (2005) Service composition for mobile environments. J Mob Netw Appl Spec Issue Mob Serv 10(4):435–451

[15] Messer A, Kunjithapatham A, Sheshagiri M, Song H, Kumar P, Nguyen P, Yi KH (2006) InterPlay: a middleware for seamless device integration and task orchestration in a networked home. In: Proceedings of the annual IEEE international conference on pervasive computing PerCom'06. IEEE Computer Society, Washington, pp 296–307

[16] Hansen K. M., Zhang W., Soares G.: Ontology-Enabled Generation of Embedded Web Services. SEKE 2008: 345-350

[17] Sachem S., Teixeira T., Issarny V., Ontologies for the Internet of Things (accepted for publication), available at: http://hal.inria.fr/hal-00642193/

[18] Abowd GD, Dey AK, Brown PJ, Davies N, Smith M, Steggles P (1999) Towards a better understanding of context and context-awareness. HUC, pp 304–307

[19] Ranganathan A, McGrath RE, Campbell RH, Mickunas MD (2003) Ontologies in a pervasive computing environment. In: Workshop on ontologies and distributed systems (part of the 18'th international joint conference on artificial intelligence (IJCAI 2003)), Acapulco, Mexico

[20] Preuveneers D, Berbers Y (2005) Automated context-driven composition of pervasive services to alleviate non-functional concerns. Int J Comput Inf Sci 3(2):19–28

[21] Iacob SM, Almeida JPA, Iacob ME (2008) Optimized dynamic semantic composition of services. SAC, pp 2286–2292

[22] Davidyuk O, Georgantas N, Issarny V, Riekki J (2010) Dans: MEDUSA: middleware for end-user composition of ubiquitous applications. In: IGI Global (ed) Handbook of research on ambient intelligence and smart environments: trends and perspectives

[23] Shah N., Chao K., Zlamaniec T., Matei A.: Ontology for Home Energy Management Domain. DICTAP (2) 2011: 337-347

[24] Mokhtar SB (2007) Semantic middleware for service-oriented pervasive computing. Doctoral dissertation, University of Paris 6, Paris, France

[25] Maffioletti S (2006) UBIDEV a homogeneous service framework for pervasive computing environments. Thesis

[26] Meditskos G., Bassiliades N., "*Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S*", IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 22, no. 2, pp. 278-290, Feb. 2010

[27] Stavropoulos T. G., Vrakas D., Arvanitidis A., Vlahavas I., A System for Energy Savings in an Ambient Intelligence Environment. ICT-GLOW 2011: 102-109

[28] Hatzi O., Vrakas D., Bassiliades N., Anagnostopoulos D., Vlahavas I., "The PORSCE II Framework: Using AI Planning for Automated Semantic Web Service Composition", *The Knowledge Engineering Review*, Cambridge University Press, 2010.

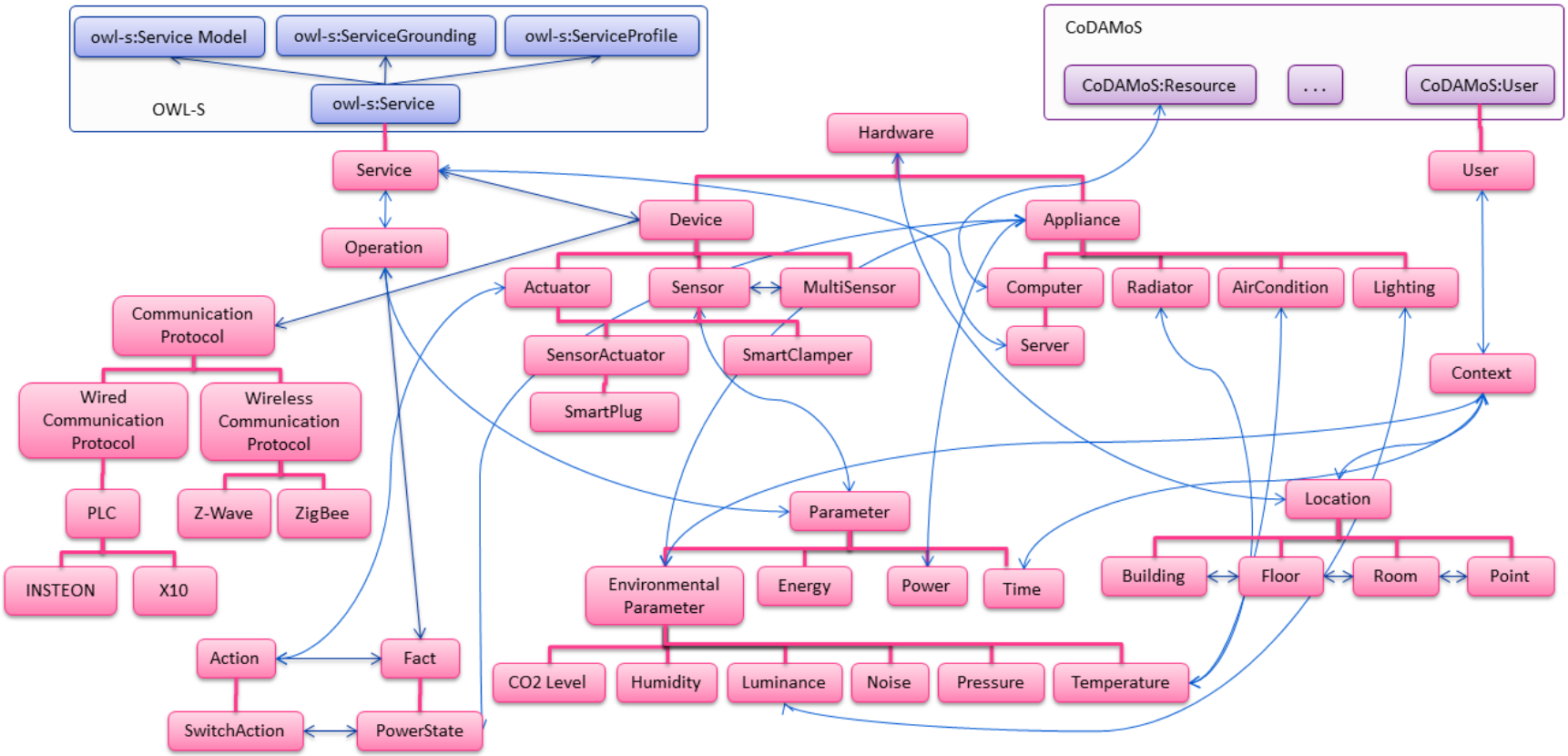

Figure 10. Extensive BOnSAI class diagram, subClass relationships and ObjectProperties