Noname manuscript No.
(will be inserted by the editor)

3LConOnt: A Three-Level Ontology for Context Modelling in Context-Aware Computing

Oscar Cabrera · Xavier Franch · Jordi Marco

Received: date / Accepted: date

Abstract Context-aware computing is the ability of services and applications to adapt and react to context changes. Context modelling is a core feature of context-aware computing. Although a lot of research has been made in the field of context modelling, most of the context-aware computing proposals prefer to design their own customized context model instead of reusing an existing one. The main reason for this behaviour is that current context models present some problems concerning reusability, extensibility and adaptation. To contribute solving these issues, in this paper we present 3LConOnt, *a three-level context ontology* that can be easily reused, extended and adapted for specific or generic purposes. The proposed context model consolidates the context knowledge already available from a modular perspective yielding a clear schema of knowledge reutilization. To do so, we gathered context knowledge pieces from different ontologies to be integrated into standardized and well-defined levels of abstraction and modules. The proposal has been validated considering: 1) reusability, extensibility and adaptation by instantiating different smart scenarios; 2) consistency and reasoning by triggering queries to the proposed model based on some competence questions; and 3) reusability in existing ontologies by importing the needed module or level of the model. Additionally, we also illustrate its usability in context-aware services by modelling a context-aware framework architecture for supporting the whole context life cycle: acquisition, modelling, reasoning, and distribution.

Keywords Context-aware computing · service-oriented computing · context life cycle · context modelling · context reasoning · context ontology

Universitat Politècnica de Catalunya – BarcelonaTech, GESSI – UPC. C Jordi Girona 1-3, Omega Building S-208, 08034 Barcelona, Spain.

Oscar Cabrera
E-mail: ocabrera@essi.upc.edu
Xavier Franch
E-mail: franch@essi.upc.edu
Jordi Marco
E-mail: jmarco@cs.upc.edu

1 Introduction

Context-awareness captures the ability of services and applications of being aware of their surroundings, i.e. their context (such as location, profile, physical environment and time), by translating it into implicit or explicit situational information. Context-aware computing allows processing and changing the behaviour of services and applications given this situational information. According to Schilit et al. [48] and Hong et al. [31], a system with context-aware capabilities can inspect the environment in which it is running and react to changes in such context. From this perspective, context-aware computing has the aim of offering better services and applications to the society and for that reason it is a core topic in ubiquitous and pervasive computing, widely applied to domains such as the Internet of Things and Smart Cities and impacting social inclusion for the emerging information society [18].

An exemplary application scenario is given by Schmidt [49]. It describes the usage of context information in a restaurant service by providing different suggestions depending on the people who is walking by. If it is parents with children, the restaurant shows the children's menu; if a couple is looking at it in the evening, it shows the menu for a candle light dinner; if it is hot

The final publication is available at Springer via http://dx.doi.org/10.1007/s10270-017-0611-z

and sunny in the afternoon, the restaurant advertises the selection of ice cream. From this outline, Schmidt concluded that, by providing such a structured space, it becomes easier to link contexts in the real world to adaptations in services and applications.

To support this type of scenarios, a proposal of context life cycle has emerged in the context-aware computing field. It is based on four steps defined by Perera et al. [44] as follows: 1) *context acquisition*, referring to the techniques used to acquire context from different sources (physical of virtual sensors); 2) *context modelling*, referring to the representation and formalization of the context compliant to some modelling techniques; 3) *context reasoning*, referring to the method of deducing new knowledge (high-level context information) from low-level raw data; 4) *context dissemination*, referring to the exploitation of the context gathered and derived by providing methods to deliver it to the consumers.

Due to the relevance of context modelling in the context life cycle, we performed in [8] a state of the art and the practice of context modelling, and then it was extended in the form of a systematic mapping [11]. We surveyed a significant amount of papers, projects, prototypes, solutions, services and applications that have been developed in the context-aware computing domain. As a result, we mainly identified that there is a big lack in reusability, standardization and consolidation of resources for context modelling supporting the context life cycle in the area. Furthermore, as it happens in many other computing areas, it does not exist a single context model agreed by the scientific community; instead, several proposals have been presented for specific or general purposes. These proposals may diverge in various matters, among others: facets addressed; approaches employed; size, structure and ontological resources provided; underlying principles; semantic factors; engineering artefacts applied.

For overcoming these open issues, in this paper we propose a context model named 3LConOnt in the form of a three-level ontology. It has the aim to standardize and consolidate a body of context knowledge that can be considered as benchmark in the context-aware computing facilitating the tasks of capturing, managing and distributing context and therefore, improving the value delivered by services and applications. 3LConOnt comprises three levels of abstraction briefly described as follows:

- The *upper-level*. Its aim is establishing a basic taxonomy of high-level context classes well suited for reusing and extending the model. A first version of this upper-level was introduced in [8]; this paper consolidates such initial version using the results obtained in our systematic mapping [11]. Particularly, 1) the work that introduces the upper-level ontology is based on a state of the art where 30 papers were analyzed, in this contribution, the upper was consolidated through a systematic mapping where 138 papers were analyzed; 2) the upper-level derived from the state of the art was focused on modelling only context information, the upper-level presented in this work evolved such version to model separately entities and context information; 3) 11 context information classes comprised the first version of the upper-level, the version in this paper evolved to represent a more abstract view of context information and only 7 context information classes were considered due to the patterns found in the systematic study.
- The *middle-level*. It has the aim of consolidating and standardizing the ontological resources derived from an exhaustive study and analysis of existing contributions in context modelling. It acts as a bridge between the upper and lower levels with the aim of extending the upper-level ontology based on a prescribed process, and providing the resources required to be extended by domain-specific ontologies. A first version of this level was introduced in [9]; this paper consolidates and extends such initial version using the results obtained in our systematic mapping [11]. Particularly, 1) in this contribution the middle-level was consolidated and unified through a systematic mapping; 2) the work that introduces the middle-level partially covered the methodology used to perform the integration process conducted at this level, in this work we deepened into the methodology and its steps to consolidate the modules of the middle-level; 3) the perspective of the previous work of the middle-level was not focused on deepening into the details of any module as it happens in this work; 4) the feasibility of the proposed model was not treated in any previous work.
- The *lower-level*. It defines domain-specific ontologies which state a set of detailed classes highly dependent on the domain. Hence, a modeller can define different domain-specific context ontologies through the semantics represented in the upper and middle levels of the ontology. We illustrate in the paper this level with several application scenarios in different domains. Hence, this level represents a key validation of the main features of the proposed model detailed below. This level of 3LConOnt is first introduced in this paper.

The feasibility of the 3LConOnt is demonstrated by evaluating: 1) its level of generality, reusability, extensibility and adaptability; 2) its usage, consistency and

reasoning; and 3) its applicability in service-oriented computing. To conduct the first evaluation, the context model is validated considering a domain ontology developed here to structure monitoring data; a smart parking service scenario previously formulated in [8]; and some scenarios provided in the literature reviewed. Thereby, the context classes and individuals identified in the scenarios should be instantiated in the proposed model demonstrating both its capability to model and represent different context in a standardized way and its feasibility to be used in different context-aware computing projects. In the second case, the context model is validated through queries done from the perspective of a smart restaurant service [49]. Finally, in the third case the context model is validated in a conceptual context-aware framework architecture that has the aim of demonstrating how the model can interact with other modules that have a role in the context life cycle, i.e., it provides the components and relationships that are responsible for acquiring, modelling, reasoning and disseminating the context information. Hence, the conceptual architecture integrates the three-level context ontology as a reference model for processing the context information in the whole context life cycle.

The remainder of the paper is organized into sections as follows: Section 2 provides an introductory background in context modelling, context-awareness from a service-centric perspective and fundamentals of ontologies. Section 3 describes in depth the three-level context ontology. Section 4 validates the proposal. Section 5 illustrates the usability of the proposed model in context-aware services and applications. Finally, in Section 6, we present the conclusions and future work.

2 Background

2.1 Context modelling

Context is a broad concept and several definitions have been provided in the academic literature from different perspectives, leading to a misunderstanding of what is its meaning and how it can be effectively applied. In this regard, Bazire and Brézillon [3] have conducted a study of more than one hundred context definitions for pointing out their strengths and weaknesses, noting that these definitions fluctuate depending on the field of study. Similarly, Dey [22] has also evaluated 10 context definitions, remarking the restrictions that each of them has for identifying new context. Considering the restrictions pointed out in both studies, we have adopted in this paper the definition provided also by Dey [21]: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*. This highly-cited definition is relevant to our work since the design criteria of the proposed context model follows the philosophy of characterizing each entity that participates in a given process with the corresponding context information, allowing distinguishing between what is context and what is not.

Context modelling plays an important role to represent and to give meaning to the collected context data. According to Henricksen [30] *"a context model identifies a concrete subset of the context that is realistically attainable from sensors, applications and users and able to be exploited in the execution of the task. The context model that is employed by a given context-aware application is usually explicitly specified by the application developer, but may evolve over time"*. Context modelling is an effective method of gathering, representing and sharing context information across different information systems [14]. As stated by Bettini et al. [4], a well-defined context model will minimise the complexity of context-aware services and applications, enhancing their maintainability and ability to evolve.

As already pointed out, context modelling is relevant in the context life cycle for supporting and providing a well-defined structure of context information in conjunction with the remaining stages of the cycle. Given this importance, various modelling techniques have been proposed in the literature to define and structure context information. Recently, Perera et al. [44] presented a comparison of the six most popular categories of context modelling techniques briefly described as follows:

- Key-Value Modelling. It is the simplest and flexible data structure for modelling contextual information. In particular, key-value pairs are easy to manage, but lack capabilities for sophisticated structuring for enabling efficient context retrieval algorithms. Therefore, it is used only to model limited amount of data.
- Mark-up Scheme Modelling. It is a hierarchical data structure consisting of mark-up tags with attributes and content. Usually, the content of the mark-up tags is recursively defined by other mark-up tags. It is also flexible and provides a better structure to represent context more than key-value modelling.
- Graphical Modelling. It is a generic structure to model context mainly using UML and ORM diagrams. This richer structure supports the modelling of a high volume of data.
- Object-Based Modelling. It allows representing context employing the main benefits of any object ori-

ented approach (namely encapsulation and reusability) to cover parts of the problems arising from the dynamics of the context in ubiquitous environments.
- Logic-Based Modelling. A logic defines the conditions on which a concluding expression or fact may be derived (a process known as reasoning or inference) from a set of other expressions or facts. To describe these conditions in a set of rules a formal system is applied. In a logic-based context model, the context is consequently defined as facts, expressions and rules. Common to all logic-based models is a high degree of formality.
- Ontology-Based Modelling. It describes the concepts and relationships of the context and entities in the environment. It provides reasoning capabilities and data structure for data sources.

The analysis and evaluation of the previous techniques for context modelling given by Perera et al. [44] and Strang and Linnhoff-Popien [51] indicates that the most appropriate technique to manage context is the ontology-based modelling. According to Sudhana et al. [55], the main purpose of ontology-based context models is to enable semantic interoperability and to provide common understanding of the structure of context information among users. The ontologies are believed to be a key feature in the making of context-aware distributed systems because they support knowledge sharing, reasoning and interoperability [42,16]. For all these reasons, we have adopted the ontology-based modelling to develop the context model proposed in this work.

2.2 Context-aware computing from a service-centric perspective

Context-aware computing refers to the development of systems with the ability to gather, manage and apply the information related to context. According to Hong et al. [31], one of the goals of context-awareness is to acquire and utilize information to provide services and applications that are appropriate to particular people, place, time, event, etc. Furthermore, Gu et al. [26] note that to avoid increasing complexity and allow users concentrating on their tasks, services and applications must be aware of their context and automatically adapt to context changes. Abowd et al. [1] consider that a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

Badidi and Taleb [2] have highlighted the importance of context from a viewpoint of services allowing identifying the intervention of different entities that can be translated into context perspectives. From this point of view, the context information can be read as circumstantial situations of an entity. For example, the identity of the client who invoked the service, whether it is a person, or another service; location and time at which the client invokes the service; activity that the client is carrying out at the time it invokes the service; device (laptop, PDA, smartphone, etc.) that the client is using to invoke the service; etc. Given these benefits, we have adopted a generic service-centric perspective that can consider a body of context knowledge affecting other perspectives including user, provider, interaction means, etc.

2.3 Fundamentals of ontologies: classification and development methodologies

According to Gruber, an ontology is an *"explicit specification of a conceptualization"* [24]. This well-known and highly referenced definition was complemented by Borst [6] as *"a formal specification of a shared conceptualization"*. Later, Corcho et al. [17] stated that a formal specification means that the ontology specification should be machine-readable. From this point of view, different comprehensive methodologies for developing ontologies have been proposed so far. These methodologies describe the stages and activities that should be performed to develop and maintain ontologies. Some of the most known and accepted methodologies found in the literature are those proposed by Uschold and King [56], Gruninger and Fox (named TOVE) [25], Fernández-López et al. (Methontology) [23], Brusa et al. [7] and De Nicola et al. (UPON) [20], among others. In this work, we have consolidated the ontology building process of our approach considering the recommendations of these methodologies, especially from Methontology due to its evolving prototyping life cycle that allows going back from any state to any other if some definition is missing or wrong.

Commonly, the methodologies for developing ontologies specify an activity focused on the reuse of existing knowledge in order to avoid terminological ambiguities and improve the efficiency of the ontology building process. Since we have special interest in consolidating the existing context knowledge already defined in the literature, we focus on the reuse of existing ontologies. As stated by Pinto et al. [45], there are two different reuse processes: merge and integration. In a merge process, the ontology is built in one domain by reusing two or more different ontologies belonging to that domain. Hence, the source ontologies are unified into a single one, being usually difficult to keep traceability in the unified ontology, e.g. identifying which regions were taken from the merged ontologies and were left

unchanged. In an integration process, the ontology is built in one domain reusing one or more ontologies in different domains (which may be related). Hence, the source ontologies are aggregated, combined and assembled together, to form the resulting ontology, possibly after reused ontologies have suffered some changes, such as extension, specialization or adaptation. In our approach, we have followed an integration approach because we are more interested on unifying modules than complete ontologies. For this purpose, we have adopted the integration process defined by Pinto and Martins [46] since they have compiled integration activities from different methodologies.

Ontologies are often classified given the design and structure adopted during its building process. In this regard, two usual criteria are generality and expressiveness. The generality criterion supports the adoption of a layered view of ontologies [41,57], and has the main purpose of specifying general classes towards top levels (abstraction) and more specific classes towards lower levels (granularity) [28]. The expressiveness criterion indicates the level of detail of an ontology. Usually ontologies are classified into lightweight and heavyweight [17]. Whilst lightweight ontologies include concepts, taxonomies, relationships between concepts and properties that describe concepts, heavyweight ontologies add axioms and constraints. In this paper, we have adopted an ontology-layered view and a lightweight expressivity with the aim of providing a model easy to use and adapt in different use cases.

3 3LConOnt: A three-level ontology for context modelling

3LConOnt is articulated around three-levels of detail for context modelling, namely upper, middle and lower level ontologies. The *upper-level* provides high level classes that we have consolidated from the context models reviewed in a systematic mapping detailed below. The aim of this upper-level is to provide a basic taxonomy of context classes that represents very general context concepts like time, location, agent, etc., which are independent of any particular problem or domain. Every context model should appraise this taxonomy in order to prevent terminological and conceptual ambiguities. The *middle-level* represents a bridge between the upper-level and the lower-level, and provides an easy way to reuse and to extend ontological resources of existing context models and other consolidated ontologies from a modular perspective. As a benchmark of this level, we propose reusing a set of ontological resources that represent structured modules selected using different strategies (detailed in the next subsection). Finally, the *lower-level* represents a set of detailed classes highly dependent on the domain. The aim at this level is that domain-specific ontologies proposed in existing contributions or developed from scratch can be defined and structured by extending the appropriate classes of the modules specified in the middle-level ontology. Such situation was also an issue found in the proposals reviewed.

This level-based modelling strategy has been adopted for covering some gaps found in the antecedents of the work, particularly related to reusability issues and the lack of consolidated and standardized ontological resources. From this perspective, we consider the following main benefits:

- The model specifies vocabulary at three levels of detail, which facilitates its reusability [6]. Hence, an ontology designer is able to define and structure further vocabulary in the proper level of abstraction. Such capability will be validated in the paper by extending the vocabulary proposed at each level of the model with the vocabulary taken from the smart parking service scenario [8] and other scenarios provided in the reviewed literature.
- The upper and middle levels of the model and all the modules of the middle-level ontology can be reused independently of the entire model. Hence, an ontology designer can integrate in an existing ontology the level of abstraction that is required without the dependency of other levels of the model. From the same perspective, an ontology designer can reuse any module of the middle level ontology without inconsistencies since the semantic of each module is independent of each other (e.g., the module of time can be reused without the integration of other modules since it was formulated to represent only the semantic of time). Such capability is validated in the paper by illustrating into the Protégé tool[1] the reuse of a level of the model in an existing ontology.
- Lower level ontologies and modules of the middle level ontology can be integrated with other modules or domain-dependent ontologies maintaining the reasoning capabilities defined in upper levels. For this purpose, the ontology designer should follow a formal integration methodology as the presented in this paper to avoid inconsistencies in the generated axioms. Such capability is validated in the paper through the integration process of modules and levels of the ontology and by means of queries that involve the reasoning of the domain-dependent modules and upper levels.

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

Finally, from the service perspective of this work, the model has been designed for answering generic questions mainly related with entities participating in a process of service provisioning and consumption such as: Which are these entities? Which are the features of the entities that delimit the value delivered? What is the context information that can be used to characterize the situation of each entity? What is the context information that affects negatively the consumption of certain service? What is the context information that characterises interaction means affecting positively or negatively the service consumption?

For providing the required benefits and functionality of the model, the remainder of this section focuses on the upper-level, middle-level and lower-level ontologies. In Section 3.1 the context knowledge pieces of different context models surveyed through a systematic mapping are integrated to develop the upper-level ontology. In Section 3.2 the ontological resources of the middle-level ontology (knowledge focused on context reasoning, and resources that facilitate the reuse of the model) are integrated and consolidated, with the ultimate goal of integrating the entire model since it represents the bridge between the upper and lower level of the model. Regarding the lower-level ontology in Section 3.3 we propose a domain ontology that is relevant for the contributions of this work and we provide the details that an ontology designer should take into account for building domain ontologies through the upper and middle levels of the model proposed.

3.1 Upper-level ontology

The systematic mapping mentioned above performed a state of the art in context modelling [11]. The study: 1) compiled different gaps reported by researchers in context modelling and identified through the analysis and evaluation of existing contributions; and 2) established, through the analysis of synonyms and hierarchies of the classes proposed in the contributions, a basic taxonomy of high level classes intended to serve as basis of the abstract level of a context model consolidating all these proposals.

As a result of this mapping, 138 primary studies were selected to answer the following research questions:

- What is the chronological overview of the research done so far in ontology-based context models?
- What are the characteristics of the proposed ontology-based context models?
- Which classes of context information and entities are the most addressed in ontology-based context models?
- What are the most consolidated classes of context information and entities in ontology-based context models?

Specifically, the selected contributions were studied in depth by analysing and assessing the following aspects:

- size measured by the number of nodes and levels of the context model;
- coverage of the definitions provided (as indicator of completeness);
- most consolidated definitions, based on the extent of the context information addressed.

The main results of the study are summarized as follows:

- only a small set of proposals (20%) have a unique and consistent definition for a majority (100% definition completeness) of their context information;
- definition completeness of properties is not optimal since only 4% of the proposals provide datatype properties between 70% and 100% of their classes, and 17% of them define object properties between 60% and 100% of their classes;
- restrictions were not included in most of the proposals (e.g., if a person hasAge xsd:Int greater-Than "18" then the person is classified as an adult);
- context information and entities oscillated in different degrees of definitions and their representation through the context models;
- due the diversity of proposals and the lack of a standard, it is not specified a common base of context resources to be reused and applied in different use cases.

Based on the results of the systematic mapping, the taxonomy of high level classes of entities and context information presented in [8] has been evolved into an upper-level ontology that represents the most abstract level in the three-level ontology proposed in this work. To populate and model this upper-level ontology, depicted in Fig.1, we have considered different aspects from the surveyed models. In particular: the most addressed classes belonging to the first three levels of their proposed hierarchy; the definition and semantic description of these classes; common patterns identified through the proposed schemas; the alignment with foundational ontologies that were partially reused by the contributions; etc.

We also considered the study on modelling styles employed in a context model and the alignment with the definition of context given by Dey [21] (see Section 2.1). From the perspective of Dey, Person, Place,

Object, User and an Application should be the general entities that participate in an interaction process, and context information the information used to characterize the situation of such entities. In our proposal, the upper-level ontology follows this logic and abstracts the mentioned entities and context information into different generic terms found by means the mapping study. For instance, we conceptualize and define an entity based on two big terms, Resource and Agent, where Resource is intended to provide the semantic needed to describe any resource such as an Object and an Application, and an Agent is intended to provide the semantic needed to describe any agent such as Person and User. The only variation in this alignment falls into the Place concept, according to the patterns found in the literature such concept beyond being an Entity it was represented as context information. Hence, such pattern is also considered in our proposal. With this consideration, we are able to say that the status of different entities is also affected by their location and another context information.

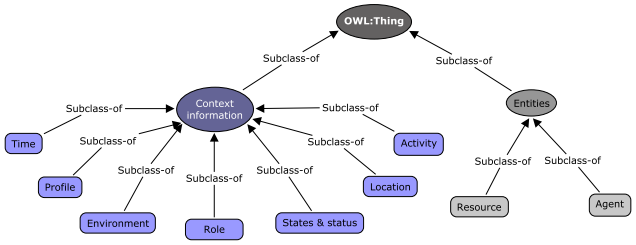

Fig. 1 Upper-level ontology

Summing up the previous considerations, the upper-level ontology (see Fig. 1) has been designed following a pattern that we called entity-independent in which entities and context information are clearly separated providing a more generic view of the model and its primitives (concepts, axioms and properties). Hence, entities into the Agent concept (e.g., Person, Organization, etc.) and Resource concept (e.g., Service, App, etc.) can be related, described or characterized through context information classes such as Profile, Activity, Environment, etc., that also can be extended by more specific modules or concepts. Hence, with the purpose of grounding the reuse and extension of the proposed upper-level ontology, context information and entities depicted in Fig. 1 are defined as follows:

- Context information:
  - *Time*. Temporal concepts and properties common to any formalization of time [16]. This means that the *Time* class fundaments the primitives of a module of time involving timeline of past, present and future. Hence, the situational characterization of an entity that is affected by the context information of time should be modelled through the modules and primitives that extend the semantic of such class. For instance, time at which an entity (e.g., client) invokes another entity (e.g., service).
  - *Profile*. Biographical sketch or an outline of something [40]. This means that the modules of the *Profile* class should provide the primitives needed to characterize the profile information of the entities involved in an interaction process. For instance, profile of an entity (e.g., client) that invokes another entity (e.g., service) to understand its preferences (a type of context information into the Profile class).
  - *Environment*. The surrounding conditions [40]. In the same way, the previous definition of the *Environment* class grounds its modules to provide the primitives needed to characterize all the environment factors of the entities involved in an interaction process. For instance, the social or environmental conditions (two types of context information into the *Environment* class) that affect an entity (e.g., client) to invoke another entity (e.g., service).
  - *Role*. Role of an agent can be used to characterize the intention of the agent [15]. In this case, the definition of the *Role* class fundaments its future modules to provide the primitives needed to characterize the functions of the entities involved in an interaction process. For instance, an entity (e.g., service) can be accessed by another entity (e.g., client) based on its assigned role.
  - *States and Status*. A state at a particular time [40]. As it can be seen, the suggested definition of the *States and Status* class is in terms of another context information (time). This means that the future modules and primitives of this class used to characterize the status information of the entities can be related with the primitives of the modules of the *Time* class to provide assertions of the status of an entity in the timeline of past, present and future. For instance, the emotional status (a type of context information into the *States and Status* class) of an entity when (it represents the time) invoking a service.
  - *Location*. By location context, we mean a collection of dynamic knowledge that describes the location of an entity [15]. The previous definition of the *Location* class fundaments its future modules to provide the primitives needed to char-

acterize the abstraction of a physical or logical location of the entities (agents or resources) involved in an interaction process. For instance, location where an entity (e.g., client) invokes another entity (e.g., service).
  - *Activity*. Represents a set of actions [16]. It means that the modules of the *Activity* class should provide the primitives needed to characterize any specific behaviour of the entities involved in an interaction process. Moreover, this type of context information can be related to another context information (e.g. time and location) to formalize the situational context of an entity through different dimensions that can generate more sophisticated assertions about the activity of an entity. For instance, the activity that an entity (e.g., client) is carrying out at the time it invokes another entity (e.g., service).
- Entities:
  - *Resource*. Resources describe anything used to perform an activity [47]. This definition fundaments the future modules and primitives needed to conceptualize any resource, i.e., through this generic class a modeller can specify different types of entities representing a service, application, tool, etc., that are needed by another entity (e.g., person) to perform something (e.g., activity). As it can be seen, the definition of the *Resource* class is in terms of a context information (activity), this also means that we can assert that the activity performed by an entity is also affecting, for instance, another context information (e.g., the status of the entity).
  - *Agent*. A representative who acts on behalf of other persons or organizations [40]. Contrary to the definition of the *Resource* class, this definition of the *Agent* class represents the basis to conceptualize different modules and primitives of different agents such as a person, people, a group, an organization, etc. that can act on behalf of other agents or by themselves. Note that the term Agent was selected due to the patterns found in the surveyed models to represent a person, user, client, etc. that generally represented this type of entities as a specialization of an agent. However, due to the generic purpose of the proposed upper-level ontology, i.e., it can be also extended at the same level of abstraction. A modeller can take the decision to represent such type of entities out of the scope of the suggested semantic of an agent. Hence, beyond representing two types of entities (Agent and Resource) a modeller can suggest to represent a person or user at the same level of abstraction. This type of modelling decision should not affect the reusability and extensibility of the proposed model while the modeller remains consistent with its proposal and the modelling considerations pointed out in this paper. Such principle apply also for the *Resource* class and the context information classes.

Note also that the previous definitions do not represent a selection of the "best" possible ones. In fact, due the lack of a standard, the study was conducted by evaluating their generality with the objective of obtaining a set of high level classes which are semantically coherent and generic enough to be extended by modules and primitives needed to conceptualize different entities and context information as detailed above. For instance, the *Time* class can be extended, specialized or assembled by means of modules related to time, i.e., sub-ontologies specifying essential concepts of time such as hour, day, etc., needed to reason about the events triggered from the past through the present into the future. Similarly, the essential concepts for the rest of the upper-level classes should be consistent among them and provide the semantic needed to be extended.

3.2 Middle-level ontology

The middle-level ontology is defined and structured in a modular way for supporting reuse by following the integration process prescribed by Pinto and Martins [46]. The extension and consolidation tasks addressed in this work are carried out by deepening into the analysis, selection and combination of many useful vocabularies from different existing proposals. In this regard, the integration process allows identifying and gathering parts of different ontologies to be integrated systematically into modules. From this initiative, we addressed some gaps of a generic context model by allowing the definition and instantiation of new or existing context knowledge in a unique and simple way. Therefore, the main objective of the middle-level ontology is to provide a set of modules easy to reuse, extend and link among them and with upper and lower levels of the ontology.

It is worth noting that the middle-level ontology was introduced in a previous work [9], in the present work we consolidate, extend and refine the integrations tasks defined by Pinto and Martins such as assumptions and ontological commitments, knowledge represented in each module, study and analysis of candidate ontologies including the study of properties, and integration operations including new operations, facts, modules and

extensions. The complete integration process is applied below.

Identify integration possibility. In this step, we are interested on importing specific modules from existing ontologies in an easy way providing a clear schema of reutilization, and connection with the upper level classes. To do so, it is necessary to select a tool to support the ontology construction. Following the criteria provided by Su and Ilebrekke [54], we have selected Protégé as ontology development tool. This implies that, in cases where the ontologies selected are provided in a different framework, we will translate the selected modules into the semantics of Protégé. It is clear thus that the selection of this tool is greatly influencing our proposal. This is unavoidable because we want an ontology that can be used in an engineering context in order to provide tangible value in the development of contextual software and services.

Identify modules. We select as modules of the middle-level ontology the context classes established in the upper-level ontology (see Fig. 1). Hence, the subontologies/modules should comprise the context knowledge pieces aligned with the semantic of the high level classes defined in the upper-level ontology, i.e., time, profile, states and status, environment, role, location, activity, resource and agent.

Identify assumptions and ontological commitments. To cover this task, we define the assumptions and ontological commitments that each module should comply among them and the resulting model, based on the specification requirements of the future ontology, i.e., the three-level ontology as suggested by Pinto and Martins. For this purpose, we follow the guidelines provided in Methontology [23] specifically from the specification phase that allows to produce the specification document that describes the domain, purpose and scope of the future ontology. Such specifications are described below.

The domain of the ontology is the context than can be aligned with the definition given by Dey [21] who sates that *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*. Taking relevance, the context causing either positive or negative effects on entities involved in the value structure given by services.

The purpose of the resulting model and modules is to represent and provide a unified, consolidated and standardized context knowledge easy to be reused, extended or adapted. Required in context-aware systems and services and for facing several gaps in context modelling such as dealing with heterogeneous context information, dependencies among the context knowledge represented in the models, a well-defined data schema for improving its dissemination and storage, and so on.

The scope of the ontology is delimited by the representation of context knowledge that characterizes the situation of different entities involved in the value structure given by services, and those identified in the review on context modelling conducted in the systematic mapping. The central idea is to visualize the context information that is always connected to an entity, i.e., the context information that characterizes the situation of each entity participant in the interaction given in a process. Future modules in the middle level of the ontology should comprise the specifications given here. Table 1 summarizes the ontology requirement specification supporting assumptions and commitments of the model.

Table 1 Requirements specification for supporting assumptions and commitments

Domain:	Context aligned with the definition given by Dey [21] and from a service-centric perspective
Purpose:	Ontology about the representation of context knowledge to be used when a unified, consolidated and standardized context information is required in context-aware systems and services, and for facing different gaps in context modelling
Scope:	– Representation of context knowledge and entities identified in the review on context modelling conducted in the systematic mapping; – Representation of context knowledge causing either positive or negative effects on entities involved in the value structure given by services – Provide a taxonomy of high-level classes to facilitate the extension of the model (upper-level); – Provide modules aligned with the semantic of high-level classes (middle-level); – Provide the initial criteria for grounding domain-ontologies (lower-level)
Sources of knowledge:	– Systematic mapping on context modelling – Brainstorming with experts on ontologies and services

Identify knowledge to be represented in each module. For the purpose of this task, we provide a list of essential concepts that should aid to compose the modules of the future ontology. As suggested by Pinto and Martins, the conceptual model of the ontology and abstraction capabilities can produce such list. In this regard, in [8] we have introduced some activities for specifying a glossary of terms belonging to the list. Such glossary of terms is complemented as depicted in Table 2 by considering both the study of about 138 existing contributions in context modelling found in the systematic mapping [11] and the brainstorming generated with service experts.

It is worth noting that the list of terms specified in the table is not intended to be a comprehensive list, as Pinto and Martins state it is only to have an idea of what the modules should contain in order to recognize whether available ontologies are adequate to be reused. In other words, the primitives (concepts and properties) of Table 2 represent both repetitive terms found in the study on context modelling and a basis of context knowledge that can fundament each module of the middle-level ontology. Hence, they are the basis to search ontologies that better fit and structure at the correct level of abstraction (also fixed under the pattern approach) these primitives. For instance, Air pollutant of the Environment module might be a middle-level class that can be extended and structured by different types of air pollutants (e.g. gas pollutants, radioactive pollutants, etc.) that can also be extended by more specific concepts (e.g., nitrogen oxides *extending* gas pollutants). This modelling consideration suggests that the concepts of a domain-level ontology should be carefully mapped with the concepts of the future middle-level ontology to maintain the consistency among the three levels of the proposed model. The complete structure and formalization of modules will be completed through the remaining tasks.

Table 2 Glossary of terms divided in modules

Activity module		Location module		Profile module	
Term	***Type***	**Term**	***Type***	**Term**	***Type***
Action	Concept	Indoor space	Concept	Community Prof.	Concept
Deduced activity	Concept	Coordinates	Concept	Device profile	Concept
Event	Concept	Outdoor space	Concept	Location profile	Concept
Process	Concept	Region	Concept	Network profile	Concept
Scheduled activity	Concept	Relative location	Concept	Object profile	Concept
Task	Concept	Building	Concept	Service profile	Concept
changes	Property	contains	Property	User profile	Concept
benefits	Property	coordinates	Property	aim	Property
causes	Property	claims	Property	depiction	Property
hasEvent	Property	existsIn	Property	dislikes	Property
hasPerformance	Property	hasPostCity	Property	account	Property
moves	Property	hasTenant	Property	depicts	Property
Environment module		Role module		Resource module	
Term	***Type***	**Term**	***Type***	**Term**	***Type***
Envir. conditions	Concept	Social role	Concept	Object	Concept
Social envir.	Concept	Civilian	Concept	Comp. entity	Concept
Regulation	Concept	Owner	Concept	Managed entity	Concept
Air pollutant	Concept	Provider	Concept	Proposition	Concept
forbids	Property	User	Concept	Service	Concept
isRegulatedBy	Property	plays	Property	fills	Property
hasAirTemp	Property	hasOwner	Property	consists	Property
States&Status		Time module		Agent module	
Term	***Type***	**Term**	***Type***	**Term**	***Type***
Current status	Concept	Time zone	Concept	Organism	Concept
Cognitive	Concept	Date	Concept	Commercial agent	Concept
Past status	Concept	Relative time	Concept	Group	Concept
Biological state	Concept	Day	Concept	believes	Property
Discrete state	Concept	after	Property	cohabitant	Property
Continuos state	Concept	before	Property	associated	Property
Future status	Concept	day	Property	defines	Property
hasStatus	Property	earlier	Property	desires	Property
isAffectedBy	Property	finishes	Property	employs	Property
isRelatedTo	Property	during	Property	executes	Property

Identify and get candidate ontologies. According to [46], this task first identifies candidate ontologies that could be used as modules of our middle-level ontology. We selected 64 context models coming from the systematic mapping [11] as possible candidates to be integrated in the modules of the middle-level ontology[2]. The selection criteria were: 1) the models provide several concepts and properties as the required in the previous step; 2) the models are based on ontologies; and 3) the ontological resources offered by the models provide context knowledge that matches the modules identified in the middle-level ontology. It is worth noting that the rest of models were also analysed in order to identify possible lack of information in the selected models.

To obtain the candidate ontologies in an adequate form, we analysed their knowledge and implementation levels as well as the documentation available. At this respect, although the knowledge level was found in most of the selected ontologies, generally it was not deeply detailed. Similarly, the implementation level of some of the ontologies was only partially defined and their availability in ontology libraries was almost inexistent. However, for each model we aimed at identifying and retrieving the ontological resources that we considered relevant to create or complement modules of the middle-level. The relevance has to do both with aspects such as frequent classes, common patterns identified through the proposed schemas, etc., and the considerations of the previous tasks to identify essential concepts. It is worth to mention that we considered not only the 64 context ontologies but also other 12 ontologies that were reused by them. We decided to establish a common base of candidate ontologies acting as reference point to structure the modules required. To carry out this task, from these ontologies, we selected those ones more referenced by existing proposals of context modelling considering their adequacy for representing knowledge of future modules as previously specified: CONON [57], SOUPA [16], SUMO [41], OpenCyc [19], FOAF[3], CCPP[4], OWL-Time[5] and OWL-S[6]. The remainder tasks of the integration process are focused on them.

Study and analysis of candidate ontologies. In this task, the candidate ontologies are analysed to identify possible problems in the integration process. We applied the SEQUAL evaluation framework formulated by Hella and Krogstie [29] which has been used for similar evaluations in a number of related areas such as goal models, requirement models, etc. In such cases, SEQUAL has been used by specializing the generic framework to the relevant domain and goal of modelling. There are 7 quality categories used to evaluate the reusability of the ontologies (see below). For each category, Hella and Krogstie propose some values that we have mostly kept, except for a couple of minor changes: 1) we added values that were not specified to describe some features of the current status of the ontologies, for instance we added the value "opens with certain problems" to specify some problems found when open an ontology as the too much spent time to charge the ontology, sometimes the complete ontology was not opened, etc.; 2) we added "RDF" as syntactic format; 3) we changed the "OK" values by the term "satisfactory". The results of the evaluation are depicted in Table 3 and the categories are described as follows:

- *Physical (Phy).* The ontology should be computationally available and it should be possible to make changes to it. Available (✓); available, presenting some problems to open in Protégé (✓ –); available, but too big to open (✓ – –); (not available X).
- *Empirical (Emp).* If a visual representation of the ontology is provided it should be intuitively and easy to understand. Satisfactory (✓); less satisfactory (✓ –).
- *Syntactic (Syn).* The ontology should be represented according to the syntax of a preferred machine-readable language. OWL full (✓); partial OWL (✓ –); RDF (✓ – –).
- *Semantic (Sem).* The ontology should cover the area of interest. Overlap, satisfactory validity (✓✓); partial overlap but not complete, satisfactory validity (– ✓); partial overlap but not complete, poor validity (– –); not overlapping (X). Since this category is too coarse to be applied globally, Table 4 shows its evaluation for the modules identified in the middle-level ontology. The analysis of completeness of the 138 reviewed ontologies can be found at [10].
- *Pragmatic (Prag).* It should be possible to understand what the ontology contains, and being able to use it for our purpose. Satisfactory (✓); not satisfactory (X).
- *Social (Soc).* The ontology should have a relatively large group of users. Mature and widely used (✓✓); assumed mature, not specified how much it is used (– –); not mature, but referenced (– ✓).
- *Organizational (Org).* The ontology should be freely available, accessible, maintained and supported. Free, accessible, and stable (✓✓✓); free, accessible, and

[2] The 64 references corresponding to the selected ontologies can be found at [10]

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

[4] http://www.w3.org/TR/CCPP-struct-vocab2/

[5] http://www.w3.org/TR/owl-time

[6] http://www.w3.org/Submission/OWL-S

probably stable (✓✓–); free, not accessible, and probably stable (✓X–).

Hella and Krogstie already provided in [29] the evaluation of some of the candidate ontologies selected in the previous task, concretely FOAF, OpenCyc and SUMO. We reused this evaluation reviewing the current status of the ontologies in order to check that the results obtained remain consistent; the only change has already been reported above (currently SUMO can be opened in Protégé). The rest of ontologies were evaluated from scratch. The results obtained are depicted in Table 3 and Table 4.

Table 3 Evaluation of candidate ontologies

Ontologies	*Phy*	*Emp*	*Syn*	*Prag*	*Soc*	*Org*
CONON	X	✓	✓-	✓	-✓	✓x-
SOUPA	✓	✓-	✓-	✓	-✓	✓✓-
SUMO	✓-	✓	✓	X	--	✓✓-
OpenCyc	✓--	✓-	✓	X	--	✓✓-
FOAF	✓	✓	✓	✓	✓✓	✓✓✓
OWL-Time	✓	✓-	✓	✓	--	✓✓-
CCPP	✓	✓	✓--	✓	--	✓✓-
OWL-S	✓	✓-	✓	✓	--	✓✓-

Table 4 Semantic evaluation of candidate ontologies organized by modules

Ontologies	*Agent*	*Resource*	*Activity*	*Time*	*Environment*	*Location*	*Profile*	*Role*	*Status*
CONON	-✓	-✓	-✓	X	X	-✓	--	X	X
SOUPA	-✓	--	-✓	✓✓	X	-✓	-✓	--	--
SUMO	-✓	-✓	-✓	✓✓	--	-✓	--	--	--
OpenCyc	--	--	--	--	--	--	--	--	--
FOAF	X	X	X	X	X	X	-✓	X	X
OWL-Time	X	X	X	✓✓	X	X	X	X	X
CCPP	X	X	X	X	X	X	-✓	X	X
OWL-S	X	X	X	X	X	X	-✓	X	X

Choosing source ontologies. Given the study and analysis of candidate ontologies the final choices must be made in this task. Pinto and Martins propose two stages. In a first stage, a critical look to the characteristics analysed in the previous task is made[7].

Although the schema presented by SOUPA and CONON for context modelling is widely referenced in the academic research, the major drawback of both ontologies is the resources provided: they are not fully available or cannot be imported into Protégé, they are not maintained and some context information and entities are not considered. Semantically, both SOUPA and CONON are small ontologies with few classes and relationships that partially characterize the situation of a few entities. Still, the design of the model presented here is partially inspired by the modular schema of SOUPA and the intuitive visual representation of CONON.

The rest of ontologies are computationally available. However, SUMO and OpenCyc are big ontologies difficult to import into ontology editors (this fact was already reported in [29]). In fact, the loading time of the OpenCyc ontology into the editors is too long due to its size. Empirically, SUMO, FOAF and CCPP provide a visual representation of their schema that is easy to understand. On the contrary, OpenCyc, OWL-Time and OWL-S do not present such an intuitive schema. In the semantic and pragmatic qualities, SUMO and OpenCyc are upper ontologies providing an extensive vocabulary; although these vocabularies can be used for purposes of context modelling, a large subset of this vocabulary is irrelevant for this purpose. The rest of the foundational ontologies are more concrete and provide a smaller set of vocabulary partially covering the context of an entity. For instance, FOAF provides simple vocabulary for describing people, what they do and their relationships to other people; OWL-Time provides a simple vocabulary for describing temporal content of web pages and temporal properties of web services.

Based on this assessment, the final decision is made in a second stage. Our aim is to select the parts of each candidate ontology that cover satisfactorily a module identified from the upper-level ontology; also, we consider the overall ontology evaluation to decide whether to include it in the result or not. The result is shown in Fig. 2. As it can be seen, in the middle-level of the model we propose different modules associated to the corresponding high-level classes of the upper-level ontology. These modules are selected from the candidate ontologies by means of the following considerations: 1) integrate modules fulfilling the conceptualization of a given entity or context information; 2) otherwise, a new module combining ontological resources from different sources is proposed. To support these considerations, we appraised the requirements specification stated in Table 1, the glossary of essential terms specified in Table 2 and the evaluation provided in Tables 3 and 4. We also illustrate how lower level, domain-specific ontologies are related to middle-level modules (see Section 3.3 and 4 for details).

Several situations have been found when selecting the modules. For instance, the *Object* module is selected from SUMO since it provides the overlap required to conceptualize this module. However, this ontology does not provide at all the required resources to conceptualize a computational entity, so we complement it with resources from CONON. Another case is presented in the *Environment* and *Role* classes from which we have

[7] Although Pinto and Martins use different criteria for evaluation, we find more natural to base the selection on the analysis previously made.

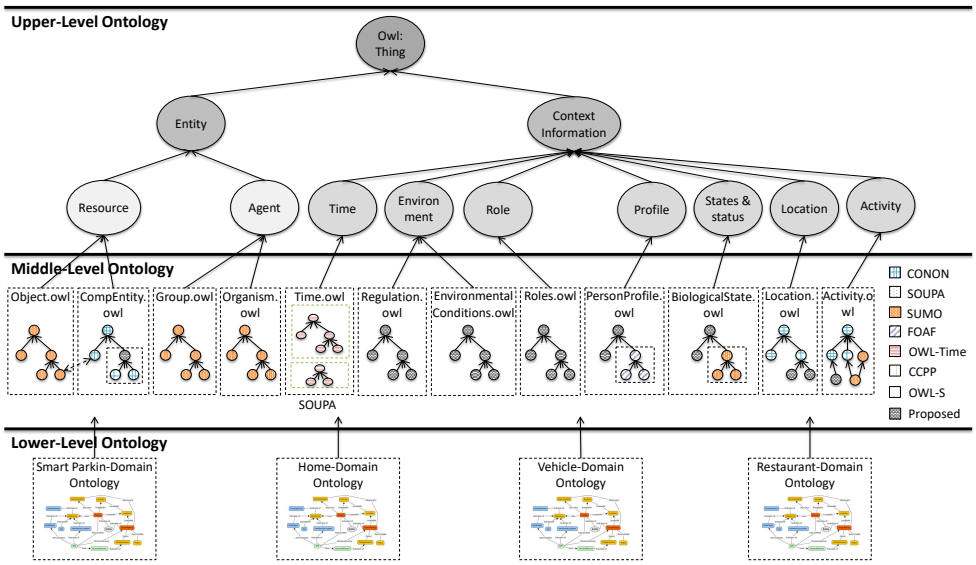

Fig. 2 Middle-level ontology and its relationships with the upper and lower levels

not found in the candidate ontologies the vocabulary that clearly overlap certain modules such as regulation, environmental conditions, roles, etc. In this case, we aim at selecting and combining vocabulary from all the ontologies evaluated to define new modules matching the semantic required by the parent classes. A third case occurs in the *Profile* class; although the FOAF ontology overlaps the semantics required to conceptualize a person's profile, it does not provide vocabulary to describe an object, a computational entity, etc. Specific details about the vocabulary used and combined are given in the next step where the integration task is performed.

Apply integration operations. Once the candidate ontologies have been filtered, the final task is to perform their integration (see Fig. 3).

Consider the following case. The upper-level class *Time* can be structured by using the semantics of SUMO, SOUPA or OWL-Time as it is depicted in the semantic evaluation of Table 4. However, according to the evaluation given in Table 3, it is difficult to identify certain resources from SUMO. In this regard, we took as a basis the semantics of time given by OWL-Time (formerly DAML-Time) because it is particularly focused on modelling time and for other features also evaluated in Table 3 (e.g., it is detailed and well-documented). Then, we adopted the following integration operations to provide the most suitable module of time: 1) we integrated in a module the OWL-Time as it is and then we make some modifications in the structure and vocabulary taking into account the next operation; 2) we identified the equivalent resources among the vocabulary and patterns presented in ontologies assessed where time was also modelled in order to consolidate, standardize and minimize semantic inconsistencies among these ontologies and OWL-Time ontology (see Fig. 3.A).

Consider a second case. The semantics of the *Location* class at the upper-level ontology has been matched with the conceptualization and structure of the *Location* class given in CONON; however, as shown in Table 3, this ontology provides a smaller set of vocabulary. Hence, we performed the knowledge augmented integration operation by semantic completeness, i.e., we retrieved vocabulary regarding location from candidate and consensus ontologies, integrating them following the semantic of CONON to minimize inconsistences (e.g., we integrated the *GeographicalPlace* class that is highly considered in the vocabulary of location as a subclass of *OutdoorSpace* specified in CONON). We also augmented knowledge considering the definition of all the classes involved in a module (e.g., the *RelativeLocation* class extent the structure of *Location* given in CONON and we also specified *Indoor* and *Outdoor* classes as a subclasses of this class as depicted in Fig. 3.B).

As a third case, modules to structure the *States* and *Status* class of the upper-level ontology were not found, i.e., the candidate and consensus ontologies do not provide a well-defined structure to conceptualize different vocabulary regarding states and status. At this respect, we retrieved from the mentioned ontologies all the vocabulary that defines a pattern and that overlaps with the semantics of this high-level class. For instance, we built from scratch the module of *BiologicalState* extending our structure by augmenting modules from other ontologies (e.g., SUMO provides a good overlapping with the *StateOfMind* as depicted in Fig. 3.C). Similarly, modules to structure the *Environment*, *Role* and

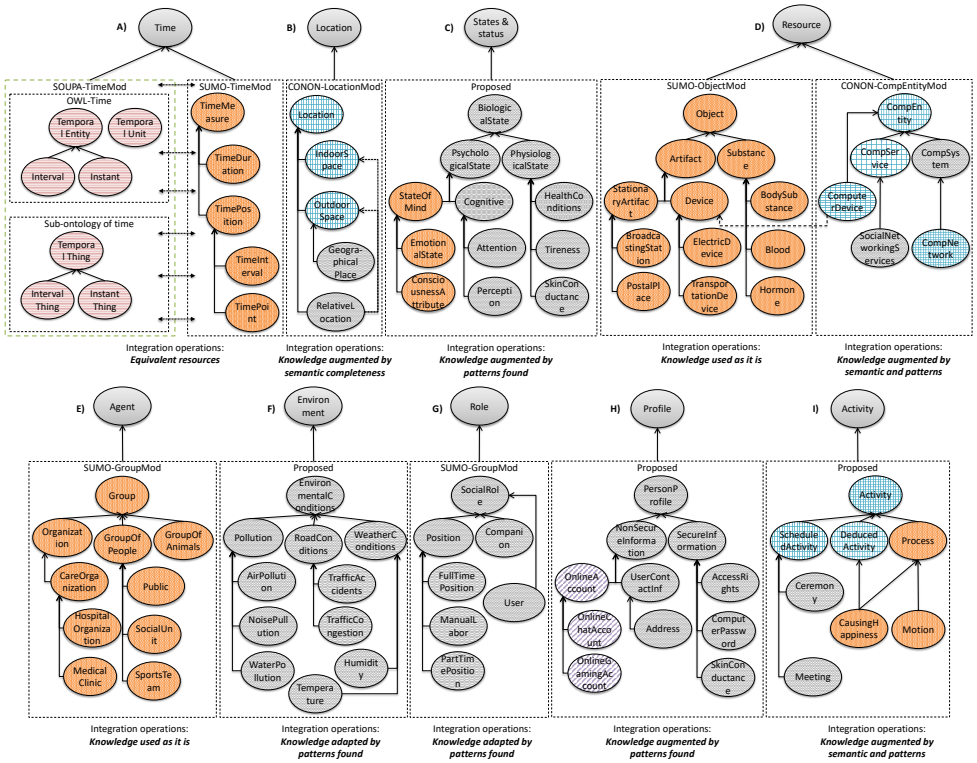

Fig. 3 Detail of the middle-level ontology

Profile classes of the upper-level ontology were not found. Hence, we also retrieve from the studied ontologies all the primitives aligned with patterns and semantics of these high-level classes (see Fig. 3.F.G.H). Note that in the case of the *Profile* class we augmented the knowledge of the *PersonProfile* module by considering the good overlapping of the FOAF ontology to describe a person's profile.

In a four case, we refer to the *Resource*, *Agent* and *Activity* classes (see Fig. 3.D.E.I) of the upper-level ontology. As it can be seen, in the *Resource* class we have 1) used as it is, the structure and knowledge given by the SUMO ontology to define the Object module since it provides a good overlapping with the semantic required and presented in the patterns of different proposals of context modelling; 2) augmented by semantic the knowledge of the *ComputationalEntity* module taking as a basis the CONON ontology and restructured by patterns the knowledge specified; and 3) search equivalences and generic types among the classes represented in the modules of the *Resource* class to decrease inconsistencies. Similarly, this process is applied to the modules of the *Agent* and *Activity* upper-level classes.

To sum up, we highlight the benefits of the proposed middle-level ontology for context modelling once the integration process is applied. Firstly, it is important to remark the restructuration and renovation of several ontological resources of different ontologies that we have reused in our model given the patterns found in existing contributions. For instance, we found that several data and object properties defined in different models are outdated and therefore they are not used, however these archaic resources are still provided (e.g., FOAF and OWL-time provide properties with this feature). In the same way, documentation and implementation of the model are also outdated and therefore they are not aligned between them. In the OWL-time ontology we identified that several datatypes are not supported by most of the reasoners currently available (e.g., gday datatype). In fact, only the Pellet[8] reasoner supports most of the datatypes provided in this ontology. In relation to this issue, we have updated and tested the usage of different ontological resources by means of their instantiation in different usage scenarios (see Section 4). We also have faced the standardization and semantics of the ontological resources in order to avoid inconsistencies among them by analysing their definitions and usage in different proposals (e.g., "interest" that is a property in FOAF was stated as "hasInterestOn" since most of the existing proposals use this term to refer the interest of someone about something; similarly, "topic" property was renamed as "hasTopic"; and so on).

[8] Pellet is an OWL-reasoner written in Java and provided as open source software supporting SWRL language to describe first order query rules [50].

3.3 Lower-level ontology

The lower-level ontology of the 3LConOnt context model represents domain-specific ontologies whose vocabulary, i.e. classes, properties (object, data and notation), etc., is highly dependent on the domain. According to Kishore and Sharman, lower-level ontologies pertain to bounded universe of discourses and are referred to in the literature as application, domain and task ontologies [37]. In the three-level approach, lower-level ontologies developed by the ontology designer are grounded on the upper and middle level ontologies, i.e., lower-level ontologies are created following a set of initial criteria and semantic principles given by the middle and upper levels of the model.

In Fig. 4, we depict a lower-level ontology designed from the upper and middle-level ones. This ontology has the aim of conceptualizing the input, output and general capabilities of monitoring tools. As depicted in the figure, middle-level classes have been extended by lower-level classes (domain classes). From this perspective, we can conceptualize different context information (e.g. time, social environment, location, etc.) and entities (e.g. applications, monitors, feedback mechanisms, etc.) that can be interrelated for conducting different activities (e.g. configure a monitoring tool given certain change in the context information).

As depicted in Fig. 4, a monitoring tool has been conceptualized to represent that *"any monitoring tool is a monitoring program which is in turn a computer program, a software and a computational entity"*. Such modelling can be applied to any monitoring tool for representing its inputs, outputs and capabilities. For instance, a feedback gathering tool can be modelled in the same way. It should extend the software class with the class or classes that can represent a feedback gathering tool class that can contain all the feedback mechanisms related. Following the conceptualization of the domain ontology we are able to represent that *"any instance of the SocialNetworksMonitoring class is a monitoring tool focused on monitoring different social network services such as Facebook and Twitter that have comments of people belonging to a social environment"*. For a generic description, we are able to represent that any instance of the SocialNetworksMonitoring class is related to only one instance of the SocialNetworksMonitoringProfile class for describing its profile information such as extended name, the URL prefix representing the URI for triggering requests to the server and more detailed description.

The input of a monitoring tool is conceptualized to represent that any instance of the SocialNetworksMonitoring class has a Boolean status On or Off indicating if the monitor is activated or not. Each instance of the SocialNetworksMonitoring class can be related to one or more instances of the SocialNetworksMonitoringConfProf class consisting of different parameters that can be configured such as the keywords that are going to be searched, the response format (json, php, xml, rss, csv), etc. Finally, the output of a monitoring tool (data monitored) is modelled for indicating that *"one or more instances of the SocialNetworksMonitoredData class is produced by an instance of the SocialNetworksMonitoringConfProf class. Each instance of the SocialNetworksMonitoredData class has a timestamp and one or more number of data items (instances of the SocialNetworksDataItems class) each of them consisting of different response properties such as id (unique hash id), message, link, timestamp, author, etc."*.

The benefits of the proposed lower-level ontology include the provisioning of a unified and representative schema of monitored data, as well as the provisioning of a clear schema of inputs and outputs of monitoring tools. In this regard, we intent to response some competency questions with the aim of evaluating that the model provides the set of axioms to represent and solve such questions. Hence, some of the competency questions that can be answered in the proposed domain ontology are the following:

- What are the parameters that can be configured in a monitoring tool?
- What kind of context information can be monitored by a monitoring tool?
- What are the monitored data that are related to a specific parameter (e.g. retrieve all the monitored data that are related to certain date)?
- What kind of response format is given by certain configuration instance?
- What instances of a monitoring tool are activated?
- What are the monitoring tools that can monitor a specific service or application (e.g. retrieve all the monitors that can monitor the Twitter)?

In summary, the upper and middle levels of the ontology intent to capture and model basic concepts and knowledge of context that can be reused by the ontology designer for building new domain-specific ontologies (lower-level ontologies) or for reusing existing ones following the same criteria. From this perspective, different domain specific ontologies can be developed and integrated in the proposed model to represent common concepts and relations that are typical in a particular topic. For more reference of lower level ontologies, we provide a set of concrete scenarios in Section 4 that validate the generality of the proposed context model.

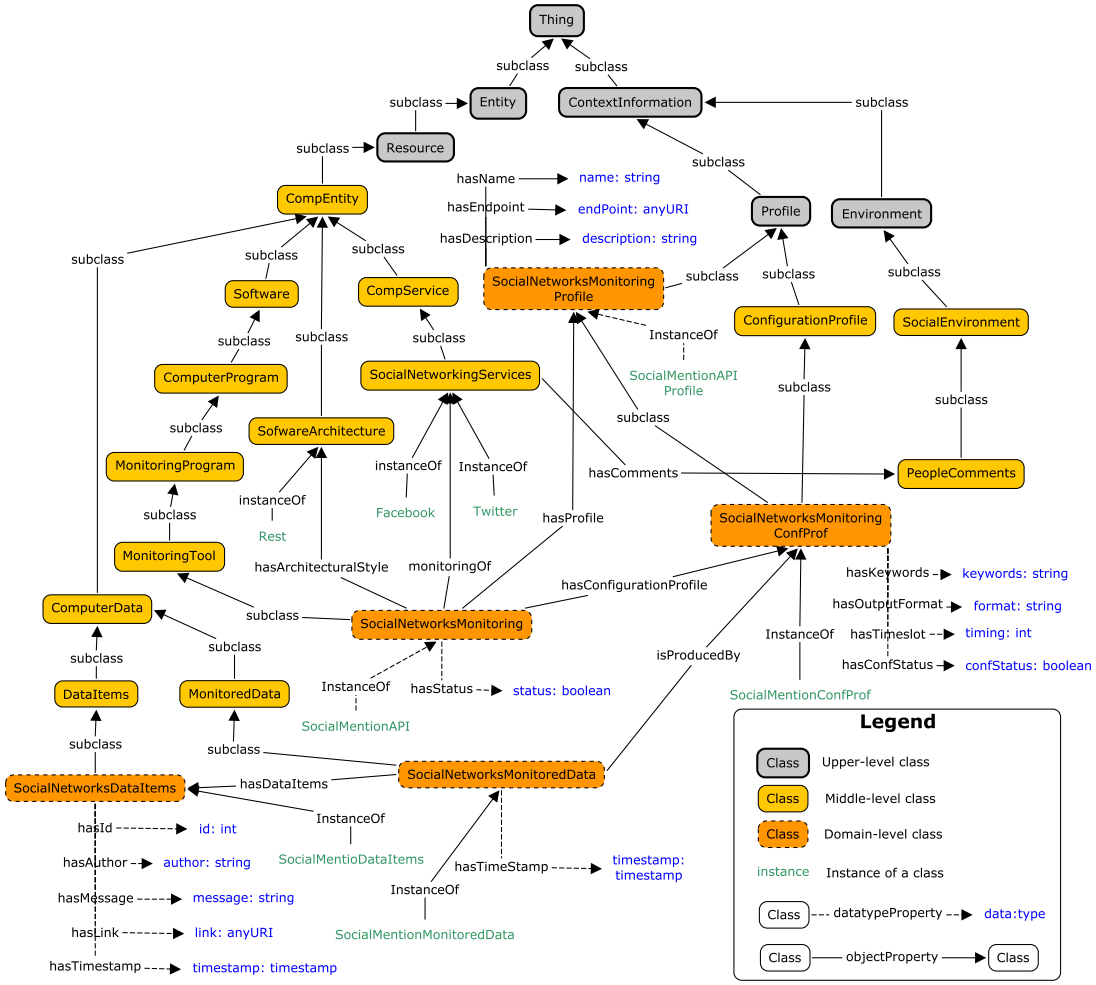

Fig. 4 Lower-level ontology for monitoring conceptualization

4 Validation of the proposal

In this section we validate the proposal considering: 1) its level of generality validating its reusability, extensibility and adaptation through different smart scenarios; 2) its consistency and reasoning by triggering queries to the proposed model based on some competence questions and from the perspective of a smart restaurant scenario; and 3) its reusability in existing context ontologies demonstrating that each level or module of the model is independent of the entire model; and 4) its correctness and completeness for demonstrating that the ontology is consistent and copes the vocabulary needed in the modules.

4.1 Evaluating level of generality: use case scenarios

For evaluating the level of generality and consolidation of the proposed model we analysed if some of the scenarios specified in the literature selected in the systematic mapping, are representable using the ontology. Such scenarios have been selected based on: 1) most referenced papers, 2) describing a kind of smart service, and 3) involving different entities and context information.

Scenario 1 – Smart parking scenario. "Chris is a 50-year old business man that needs to attend an event in the centre of Barcelona and he is thinking of using his car to reach there, but he is worried for the availability of parking spots and the route to arrive there because it is a very busy place. One of his friends advise him to use an application called Smart Parking Platform (SPP) that has different capabilities that improve the user experience, such as checking and booking free spaces in the area; searching simple route (basic use) or searching and being aware of other context e.g., strikes, accidents, etc. to provide the best route (advanced use); adapting or evolving its behaviour and interface since it is also aware of the user experience in using the app; etc. . . . " [8] (see Fig. 5).

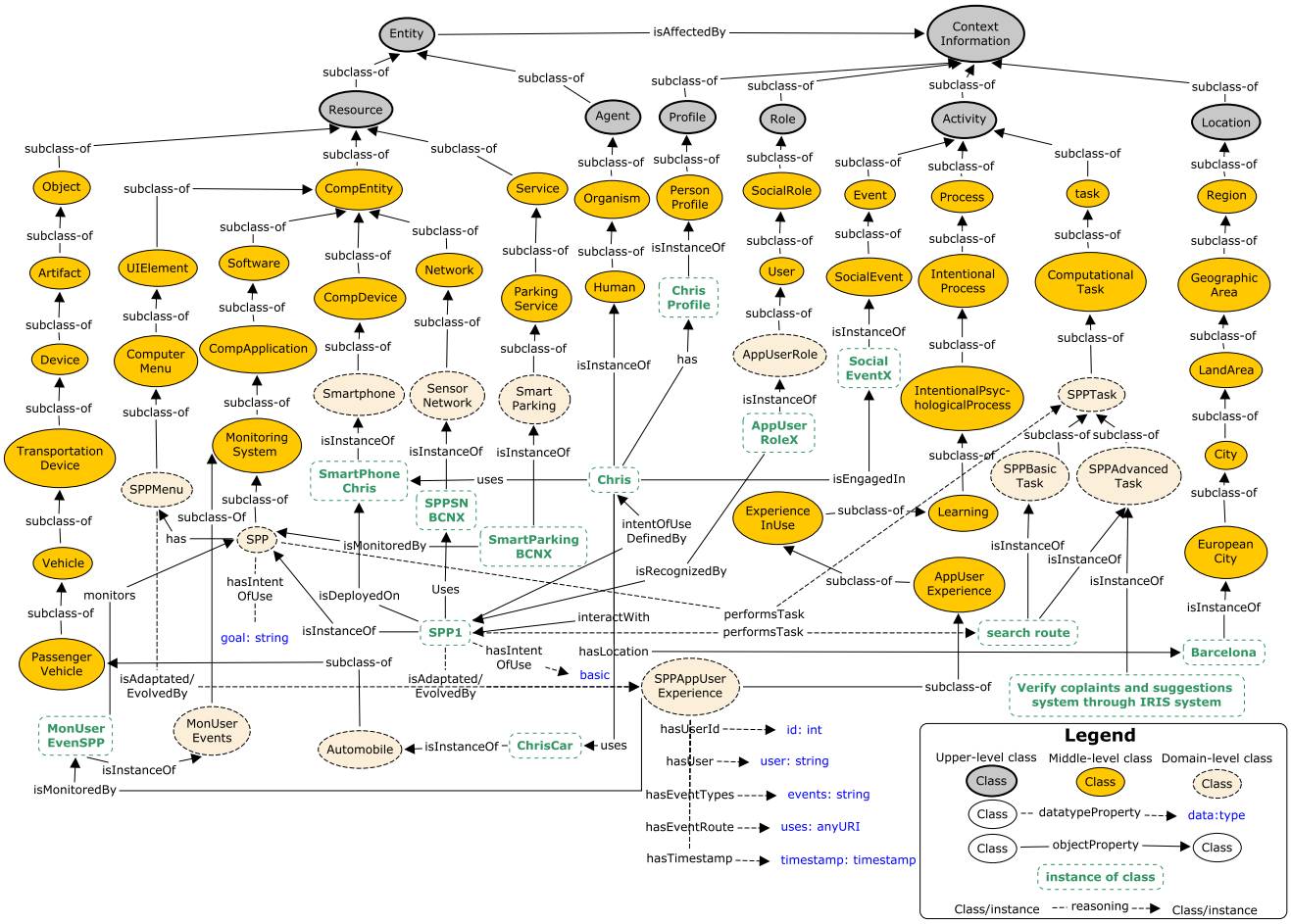

Fig. 5 Smart Parking – scenario 1

As it can be seen, the above scenario involves different primitives (classes of context information and entities, and objects and datatype properties). The lower-level ontology that models these primitives is illustrated in Fig. 5. As shown in this figure, the high-level classes of the upper-level ontology have been redefined by means of the classes of the middle-level ontology yielding the possibility to characterize domain-specific primitives of the Smart Parking scenario. For instance, the SSP class was possible to be characterized through the *MonitoringSystem* class which in turn is subclass of different middle-level classes belonging to the *Resource* class of the upper-level ontology. Furthermore, the individual *Chris* is an instance of the *Human* class that is synonym of person and that can be associated with different instances of other classes of the middle level ontology to describe the situation of this specific entity.

Regarding reasoning, the ontology can make inference about the user experience or intention of use to trigger adaptations or evolutions in the behaviour, interface (e.g., menu) or tasks of the SPP. For instance, through reasoning we can specify: if the intention of use of an instance of SPP is defined as *basic* by a user when interacting with such instance, then this instance should be adapted to fulfil some specific tasks. Note that the three levels of the proposed model are illustrated in each ontology representation of the scenarios. Note also that mainly the three-level classes remain static (defined at design time) and their instances can be created and evaluated dynamically (maybe at run time) in a process (e.g., adapt or evolve an application). The three-level classes are used to provide structure, semantic, meaning, consistency, etc. among the primitives involved in a scenario. This static view of the ontology is always important to provide such properties when creating an instance or even if a new class is also dynamically created, i.e., the meaning of such instance or class should be stated to increase the quality, interoperability, etc. of the entire model.

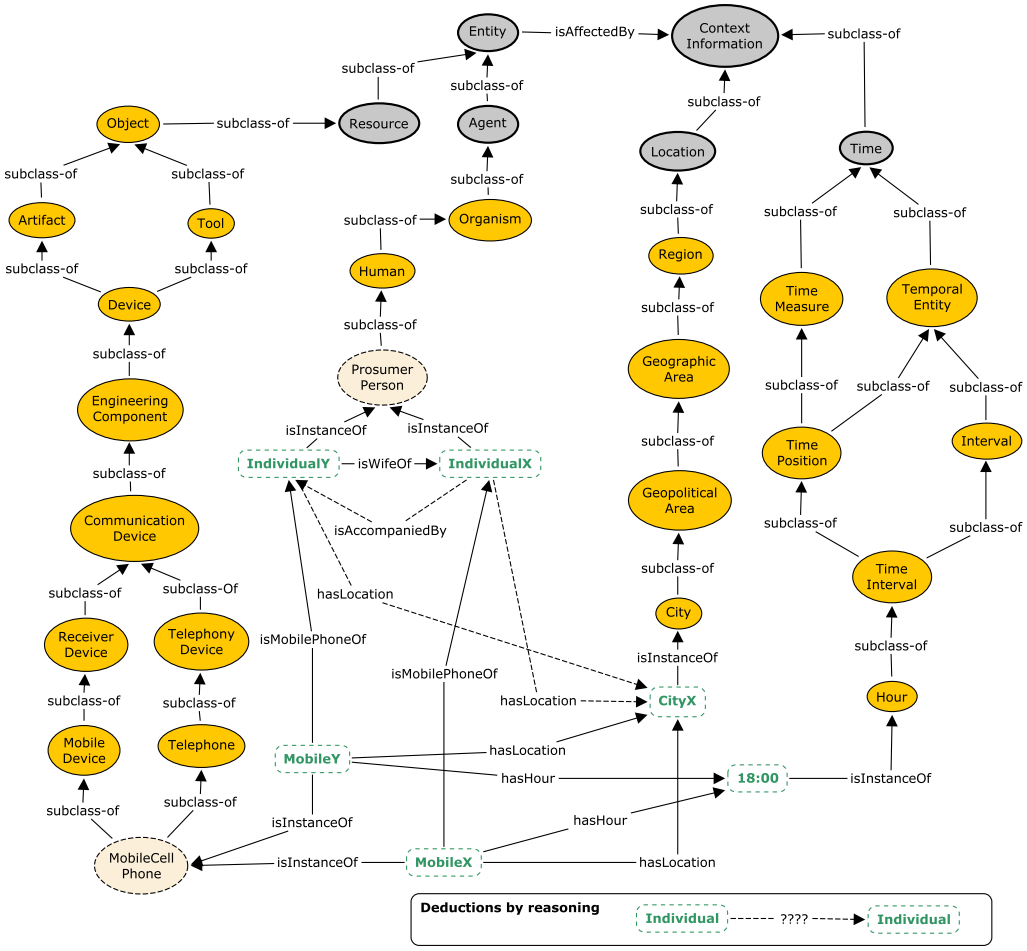

Fig. 6 Prosumer ontology - scenario 2

Scenario 2 - Prosumer scenario. "A particular individual has arrived in a city for the first time and that is travelling along with his wife. That information can be obtained from the location of the mobile devices and querying the context history database. Both devices have been located in the same bearings at the same time (the system concludes through reasoning that these two individuals are located together in the same place) . . . " [13] (see Fig. 6).

As it can be seen in the ontology representation of the above scenario (see Fig. 6), there are two domain classes (ProsumerPerson and MobileCellPhone) that were needed to represent some instances of the scenario, and other instances such as CityX and 18:00 were represented directly in middle-level classes. With this representation, we are able to deduce through the location of the mobiles belonging to the individual's X and Y their location and therefore, if they are together in a same place.

Scenario 3 – Meeting scenario. "John Pappas is an executive of an international construction company, who works at the company department that resides in Athens. John is informed that he will have to attend a meeting in Paris for a project he is currently involved in, so he activates his electronic agenda entering the meeting date, place and scope to check his availability . . . " [59] (see Fig. 7).

As it can be seen in the ontology representation of the above scenario (see Fig. 7), there are five domain classes such as ElectronicAgenda, Constructuction company, etc., that were needed to represent some instances of the scenario, and similarly to the previous scenario, other instances such as JohnPappas and ProjectX were represented directly in middle-level classes. With this representation, we are able to represent that a ProjectXMeeting has a profile ProjectXMeeting-Profile with the following information date, hour and place. Hence, different instances of FormalMeeting can be described through a profile and at the same time, an electronic agenda can manage such meetings.

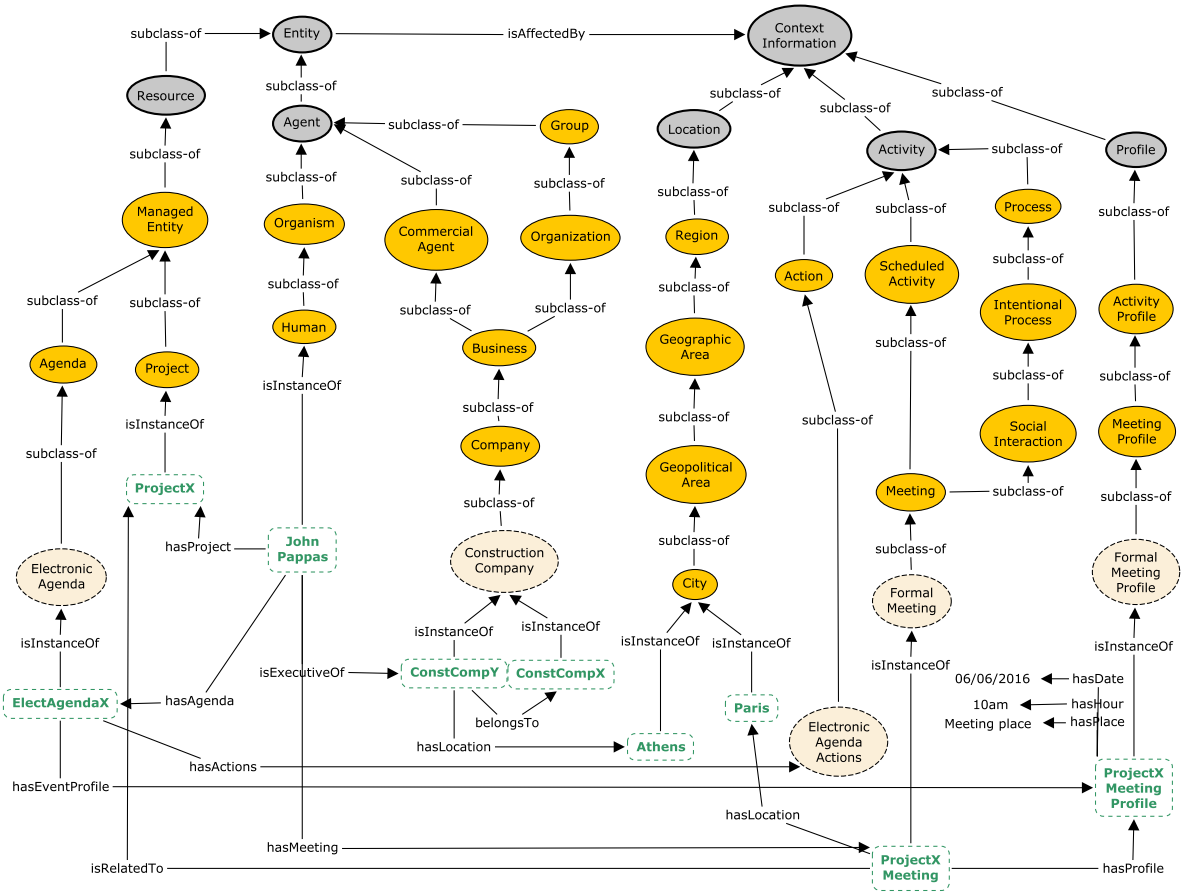

Fig. 7 Meeting ontology – scenario 3

Scenario 4 - Getting Up scenario. "Mr. Kim sets the getting up time at 6:00 am, and goes to bed late. He must go to his office early. The getting up application checks Mr. Kim's getting up time, and provides an alarm service at 6:00 am the next morning. Then the application opens the curtain to provide fresh morning air and sunshine. The application connects to a weather network service and receives local weather forecast. If it is not rainy, the application system opens the windows. If it is rainy, the system activates the air cleaning service and then provides a light service to supply enough brightness. This getting up application checks Mr. Kim's schedule, and displays it on an output device near Mr. Kim. The system also turns on a display device to show Mr. Kim a morning TV news program. The program information is referred from the preference list which stores Mr. Kim's favorite TV program list. [36] (see Fig. 8).

Similarly to the previous scenarios, domain-specific entities and context information playing a role in the above scenario were represented through the concepts and primitives of the proposed upper and middle level ontology. Although we are not pretending to represent extend and exhaustive scenarios with several variables that take part in a context decision, because the aim of the proposal is to provide a coherent model easily reusable and extensible, we considered the full version of the previous scenario as an example of completeness. As it can be seen, the model depicted in Fig. 8 represents different important aspects of the scenario such as how a window, air cleaning device and light system of a room can be automated and controlled through deductions by reasoning. For instance, if the getting up app detects air pollution at given time and location inferred by the latitude and longitude position of Mr. Kim and by the access to the environmental sensors of such location, then some resources of the Mr. Kim's room are turned on and others turned off or calibrated. In this case, the upper-level classes needed to provide the formal structure of the scenario are *Resource*, *Agent*, *Time*, *Activity*, *Profile*, *Location* and *Environment*. Note that some domain classes illustrated in Fig. 8 can be represented directly as middle-level classes, however it is a modelling decision taken by the modeller of the domain ontology. Specially, when a needed class is not yet represented in the modules of the proposed model.

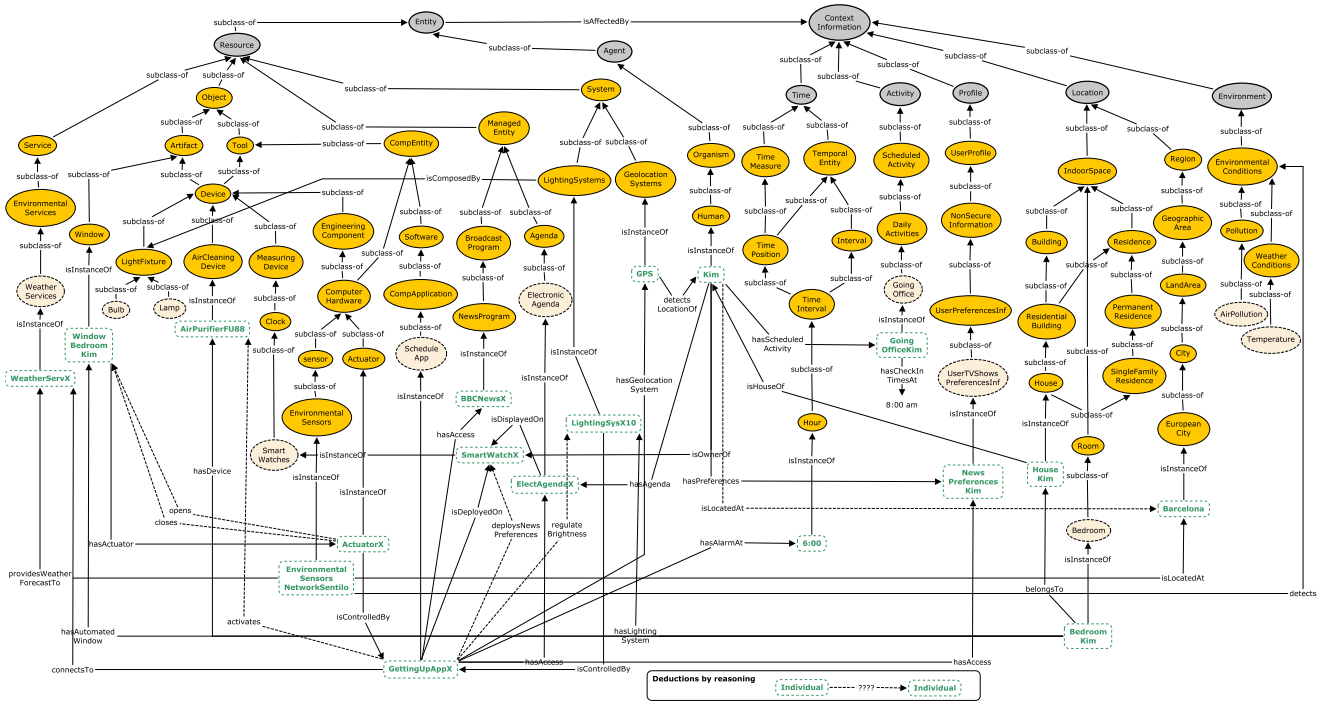

Fig. 8 Getting up ontology – scenario 4

Scenario 5 - DAIDALOS scenario. "... As soon as Bart is in sufficient range, his RFID unlocks the car and sets the car conditions to suit Bart. When he starts the car, the newscast session he was watching at home resumes in audio only mode at his in-car multi-media system ..." [52] (see Fig.9).

The instances of the above scenario were represented through middle and domain classes for automating some tasks of a smart car. In this case, the upper-level classes needed to provide the formal structure of the scenario are Resource, Agent, Location and Profile. In general, with these upper-level classes and their corresponding modules of the middle-level classes we are able to represent a multimedia system that is configured by a RFID belonging to specific persons that are owner of a car with a multimedia system.

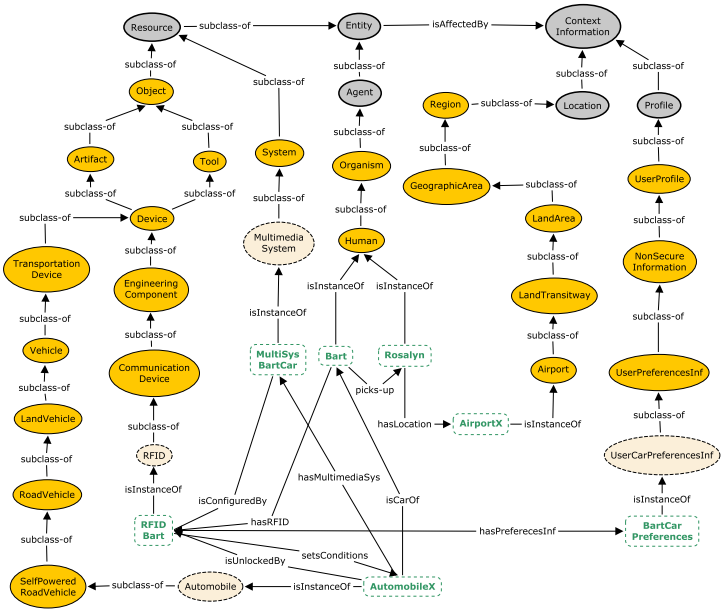

Fig. 9 DAIDALOS ontology – scenario 5

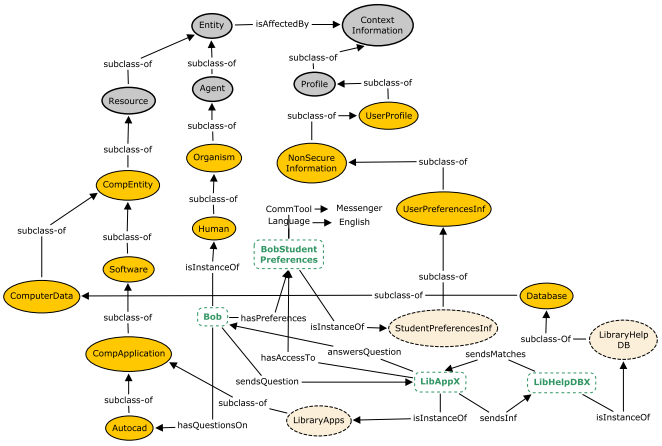

Fig. 10 Library service – scenario 6

Scenario 6 - An online collaboration service scenario. "The agent in context tag in the library checks his context and fetches some useful contexts e.g. language preferable (English), his module name, and appreciate communication tool (Messenger) etc . . . " [32] (see Fig. 10).

The ontology representation of the above scenario (see Fig. 10) uses 16 classes of the upper and middle levels of the proposed model to represent domain classes such as StudentsPreferencesInf, LibraryHelpDB, etc., needed to represent different instances of the domain that collaborate among them to automate a library service. In this case, a library app can access to the preferences of different students and based on this information provide a better library service.

Scenario 7 - Health scenario. "John is affected by a chronic disease. Her wife Barbara and his daughter Emily live with him and provide daily assistance services. John's home is equipped with a context-aware system, consisting of: monitoring devices (biomedical and environmental sensors), emergency and ordinary call buttons, and a PC which collects and analyses sensed data in order . . . " [43] (see Fig. 11).

The ontology representation of the above scenario (see Fig. 11) uses 39 classes of the upper and middle levels of the proposed model to represent instances of a resource, agent, etc., and domain classes such as PatientHealthPlan and HospitalBuilding needed to represent the health plan of a human and a hospital respectively. In general, the instances are related to illustrate the interaction of a PC placed in a home with a server placed in a hospital to generate a health plan of a patient.

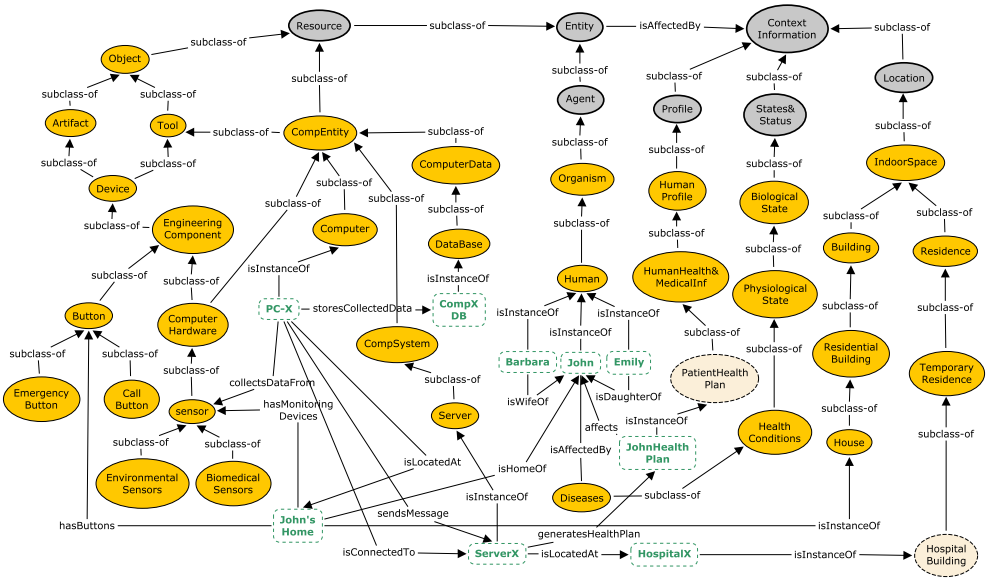

Fig. 11 Patient health and smart home – scenario 7

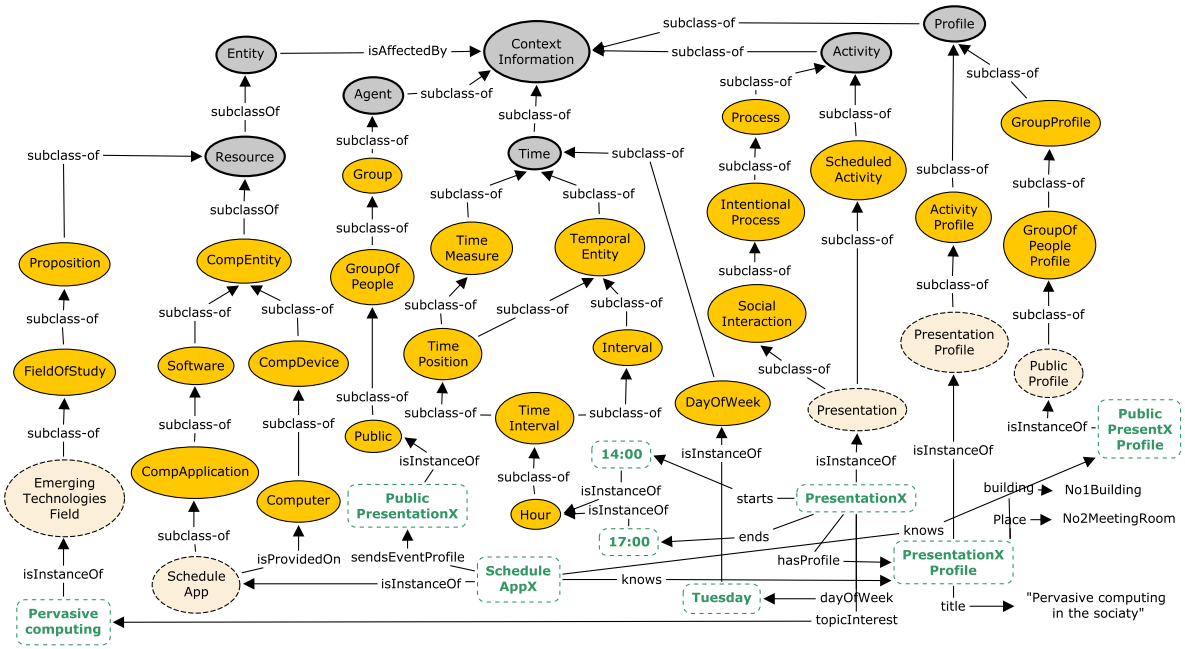

Fig. 12 Smart space ontology – scenario 8

Scenario 8 - A scenario of smart space (smart meeting room). "On Tuesday morning, a presentation about pervasive computing is scheduled to take place from 14:00-17:00 in the No.2 meeting room which is a smart meeting room in the No.1 building. The day before the presentation, the system has sent the meeting schedule including the title, the speaker, start time and location ..." [39] (see Fig. 12).

The ontology representation of the above scenario (see Fig. 12) uses 31 classes of the upper and middle levels of the proposed model and such classes are extended by domain classes needed to instantiate different context information and entities that play an important role in the scenario. In general, the instances are related to illustrate how a smart meeting room manage a meeting. As it can be seen in the figure, a presentation has an interval of time and a profile describing the important information of the meeting such as place, title, etc. that can be accessed by a schedule app that can conduct the management of the meeting.

Scenario 9 - Healthcare scenario. "The scenario begins with patient Bob who is in the emergency room due to a heart attack. While not being Bob's usual treating physician, Jane, a medical practitioner of the hospital, is required to treat Bob and needs to access Bob's emergency medical records from the emergency room ..." [34,35] (see Fig. 13).

The ontology representation of the above scenario (see Fig. 13) uses 38 classes of the upper and middle levels of the proposed model, and 75 domain classes needed to instantiate different context information and entities that play an important role in the scenario. In general, the instances are related to illustrate the management of access rights to medical profiles such as who can provide access, how can be conducted the access request, where the access rights are deployed, what are the policies that restrict the access rights, etc. As it can be seen in the figure, Bob is a human located in an emergency room because he has suffered a heart attack. Jane that is a doctor requires access to the Bob health profile in order to assist him. For this purpose, Jane makes an access request to the application called "ARMapp1" that manages the access rights of the personal working on the hospital. If this access that is restricted by the policies of the hospital can be assigned to Jane, then Jane can treat Bob. Note in Fig. 13 that such fact is asserted by the ontology through its reasoning capabilities. For instance, it can be asserted through semantic rules that can evaluate dynamic context (e.g., policies that can change dynamically given the changes of another context information, emergency in the state and status of the patient, etc.).

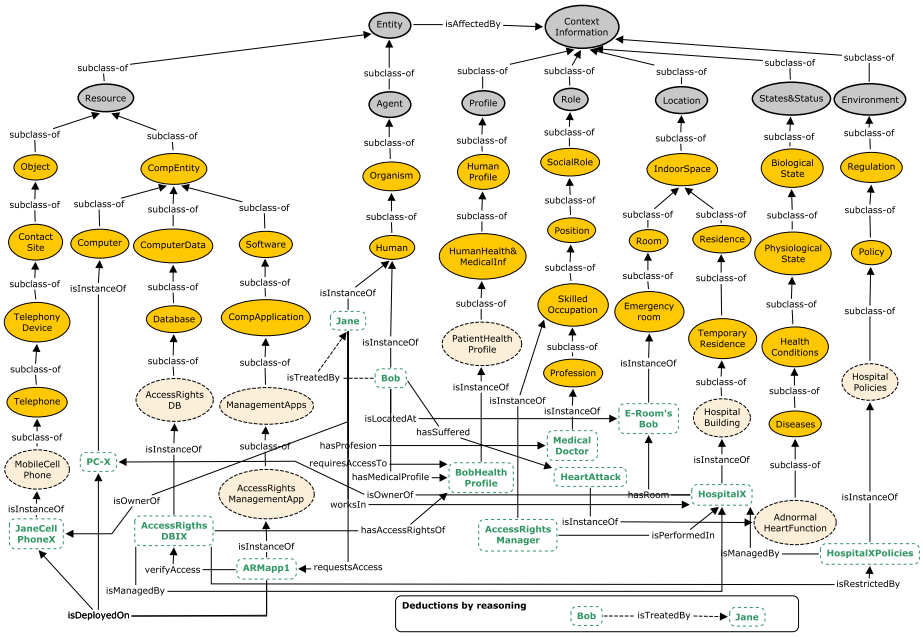

Fig. 13 Healthcare ontology – scenario 9

Scenario 10 – Smart call scenario. "The user is sleeping in the bedroom or taking a shower in the bathroom, incoming calls are forwarded to voice mail box; when the user is cooking in the kitchen or watching TV in the living room, the volume of the ring is turned up . . . " [57] (see Fig. 14).

The ontology representation of the above scenario (see Fig. 14) uses 30 classes of the upper and middle levels of the proposed model, and 5 domain classes needed to represent different domain instances that play an important role in the scenario. In general, the instances are related to illustrate the management of a call service. As it can be seen in the figure, a phone is controlled by a PC that collects data from a camera who detects the current state of a human that has a current state and location in his home.

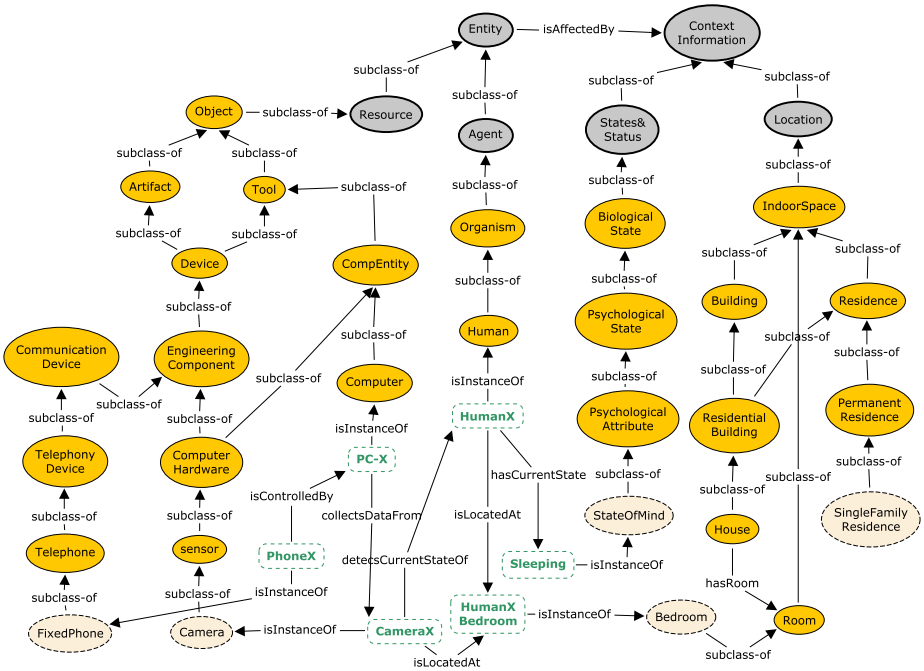

Fig. 14 Smart call ontology – scenario 10

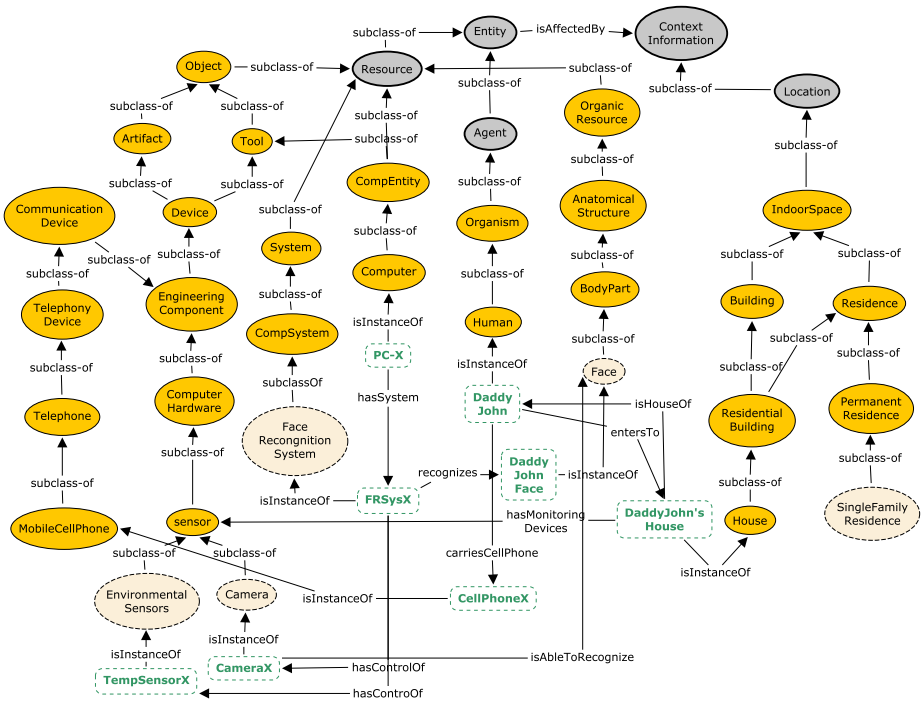

Fig. 15 Smart home ontology – scenario 11

Scenario 11 – Smart home scenario. "Daddy John carrying a cell phone has entered his house; the face recognition system senses his presence and his location information get updated. When John moves into the bathroom to take a shower or goes to his bedroom for a nap . . . " [27] (see Fig. 15).

Similarly to the previous scenario, the ontology representation of the above scenario (see Fig. 15) uses 29 classes of the upper and middle levels of the proposed model, and 5 domain classes needed to represent different domain instances that play an important role in the scenario. In general, the instances are related to illustrate the management of a smart home through faces recognition. As it can be seen in the figure, a PC has a system that controls different sensors and through them, the system can recognize the face of a human and based on this recognition the system can trigger different actions.

Scenario 12 – Smart restaurant service. In the introduction section of this paper we have referenced and described this scenario given by Schmidt [49]. The ontology representation of such scenario can be illustrated as depicted in Fig. 16. As it can be seen in the figure, a restaurant named Azurmendi is managed by a booking system that is installed in a PC-X. When a human makes a booking to the restaurant, the system requires some information provided by the human who made the reservation or it can be detected based on other source of information. As an example of usage, in the next section we made a query to this domain ontology to obtain direct or deduced context information.

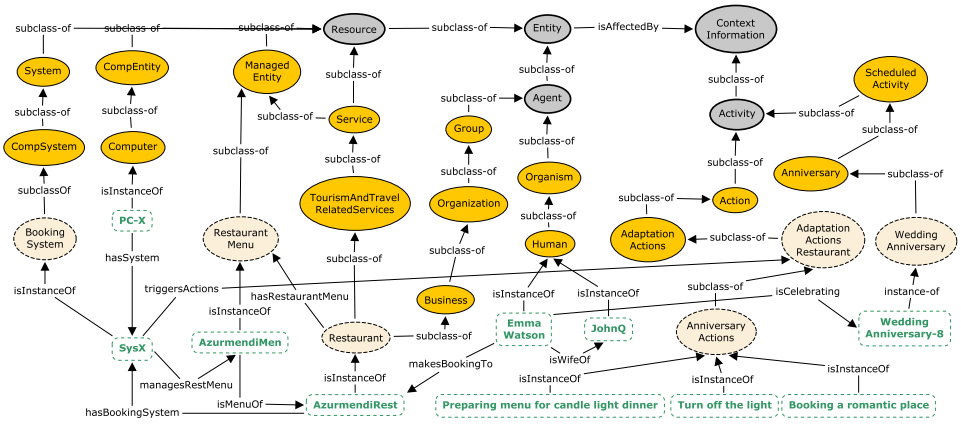

Fig. 16 Smart restaurant service ontology – scenario 12

Scenario 13 – Smart driving context. In this case, consider that an ontology modeler wants to characterize common primitives of a smart driving context. For this purpose, the modeler decides to reuse an existing domain ontology that can already provide the needed vocabulary. Hence, the modeler decided to reuse the OCM ontology [58] that in general has the aim to conceptualize traffic context and sensor capability. However, although the ontology provided the needed vocabulary at domain level, it was not aligned with a foundational or generic ontology in order to formulate and provide a consistent proposal that can increase the clarity of the terms used, the generality for improving the knowledge sharing, the uniformity for improving interoperability, etc. At this respect, the modeler decided to align the OCM ontology with the ontology proposed in this work for also increasing and unifying the semantic of OCM (see Fig. 17).

As it can be seen in Fig. 17 the type of scenario described above involves a bottom-up approach, i.e., an existing domain ontology is mapped to an existing abstract ontology. For this purpose, the modeler has the responsibility to 1) map the equivalent classes between both models to avoid inconsistencies, for instance, in the figure, 6 classes were mapped as equivalent classes of the proposed ontology; and 2) map the classes of the domain ontology as subclasses of the abstract ontology to increase the consistency of the final model, for instance, in the figure, 4 classes were mapped as subclasses of the proposed ontology. It is worth noting that although most of the modelers avoid this mapping process due to the time consuming derived mainly from the process of getting acquainted with an existing ontology, we consider that this cost of time can be balanced by increasing the interoperability and sharing knowledge of the final model among context-aware services and applications.

Through these 13 scenarios we have demonstrated the generality, extensibility and reusability of the proposed context model. Highlighting that although such scenarios of the existing contributions were represented in an ontology, they lacked of a structure representing a rich semantic of the entities and context information playing an important role in the scenarios. The scenarios also show that the proposed model, specifically the upper and middle level ontologies, can be adapted in several applications domains namely smart parking, smart home, smart health, smart restaurant service, etc. To do so, the model represents three levels of abstraction, where the upper and middle levels of the ontology specify context knowledge useful for representing and structuring domain ontologies in a rich and formal semantic increasing their reusability and applicability in different projects of the context-aware computing.

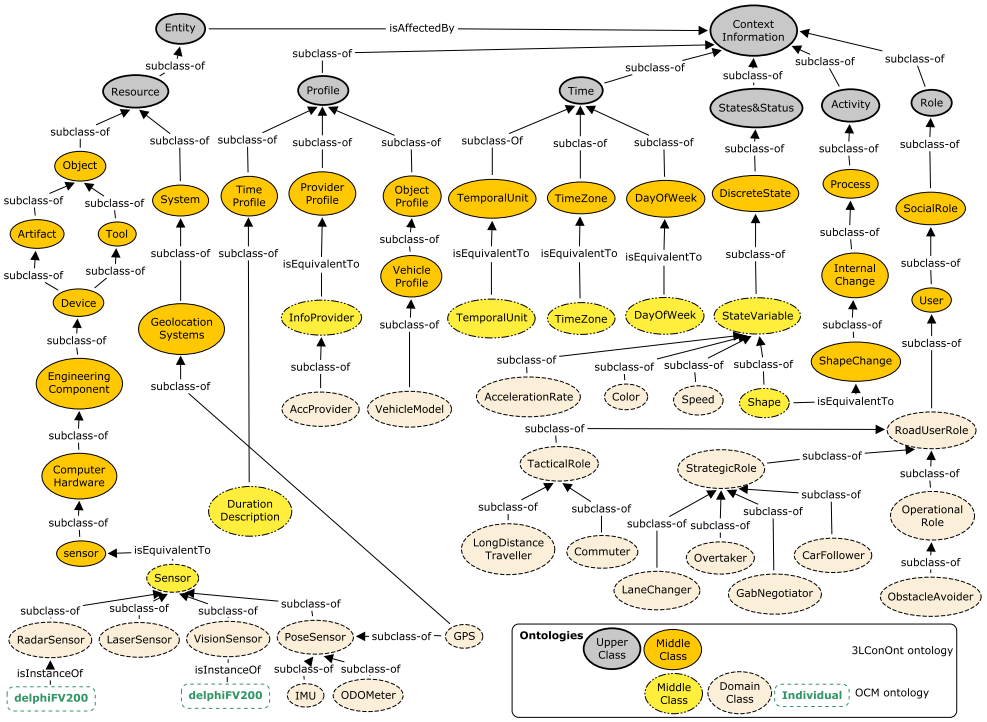

Fig. 17 Smart driving context – scenario 13

4.2 Evaluating usage, consistency and functionality

In this section, we validate the usage, consistency and functionality of the proposed model by triggering reasoning and queries over the model and modules. Specifically, to conduct such reasoning and queries tasks, we use some modules of the middle-level ontology such as Location, states and status, activity, etc. We also implement two domain ontologies to carry out these tasks such as the presented in Section 3.3 and the smart restaurant service illustrated in Fig. 16 because it manages different entities and context information that can be consulted in different use cases of a smart restaurant. The proposed model is implemented in Protégé and delivered as follows:

- The upper-upper level ontology and the modules of the middle-level ontology are implemented separately for allowing selective reuse of the ontology parts that apply to every particular scenario.
- The domain ontology depicted in Fig. 4 is implemented for illustrating the role of the lower-level ontology and supporting the validation addressed in this section. Hence, at this level of the model different domain ontologies can be implemented and linked to the middle and upper levels of the model.
- Each implemented level or module provides its specific context knowledge pieces, i.e., its classes, individuals, object and datatype properties that directly affect such level or module of the ontology.
- All implemented levels and modules are integrated allowing powerful context reasoning.

The OWL of each level and module of the proposed three-level ontology for context modelling can be found in https://github.com/ocabgit/Three-LevelContextOntology.git.

The reasoning capabilities of a context ontology are defined as the ability of deducing new knowledge, and understanding better, based on the available context [5]. According to Wang et al.[57], the use of context reasoning has two goals: checking the consistency of context; and deducing high-level, implicit context from low-level, explicit context. Based on the second feature, we can infer the location of a person, the environment of a location, the state and status of a person or device, etc. We demonstrate this potential reasoning process on the proposed ontology considering two types of reasoning: ontology-based reasoning and rule-based reasoning. The ontology-based reasoning uses the existing reasoning rules already defined in the semantics of OWL (e.g., rdfs:subClassOf, owl:sameAs, etc.). To illustrate ontology-based reasoning, the transitivity rule of the isAffectedBy property infers the following situational context of a Human: if Chris is affected by the temperature of certain location and this temperature is also affected by the pollution of the environment, then Chris is also affected by the pollution of the environment (see Table 5).

Table 5 Ontology reasoning about the environment of an entity

Explicit context	*Implicit context*
<owl:ObjectProperty rdf:ID="isAffectedBy"> <rdf:type="owlTransitiveProperty"/> </owl:ObjectProperty> <Human rdf:ID="Chris"> <isAffectedBy rdf:resource="#Temperature"/> </Human> <EnvConditions rdf:ID=" Temperature"> <isAffectedBy rdf:resource="#Pollution"/> </EnvConditions >	<Human rdf:ID=" Chris"> <isAffectedBy rdf:resource="#Pollution"/> </Human >

On the contrary, rule-based reasoning is not included in the semantics of OWL and therefore, the rules should be explicitly defined by users. One of the main use cases of these rules is finding the match among them and the context information retrieved (e.g., from sensors) to perform an action or deduce complex context. In this regard, consider the following rules defined in Protégé:

Human(?h), Automobile(?a), hasLocation(?h, ?a),
performsActivity(?a, Accelerating),
hasStatusState(?a, On) -> performsActivity(?h, **Driving**)

Office(?o), MobileCellPhone(?m), Human(?h), Activity(?ac), Agenda(?a),
hasLocation(?h, ?o), hasStatusState(?m, Status-Off),
hasScheduledActivity(?a, ?ac) -> hasStatusState(?h, **Working**)

These rules infer the activities of a person, if all the statements are true, i.e., if all the context information retrieved match with the rule then it is deduced an activity. For instance, in the first rule it is specified that if a “Human” is located in his “Automobile” and the “Automobile” has the status ‘On’ but also it is “Accelerating”, then a “Human” is “Driving”. Similarly, the second rule specifies that if a “Human” is located in his “Office”, has his “Agenda” with an activity scheduled and his “Mobile” has status ‘Off’ then the “Human” is “Working”. Such rules can be specified in two ways, by using the Semantic Web Rule Language (SWRL)[9] into the Protégé editor or by means of Jena, a Java framework for developing Semantic web applications incorporating a rule reasoner API.

In this work, we formulate rules in Protégé and Jena for validating the capabilities of the context model. The rule-based reasoning is also supported by Jena since it provides a Java Rule object with a list of terms (premises), a list of head terms (conclusions), and an optional name or direction. From this perspective, the

[9] http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

syntax of Jena for defining rules is different from the syntax used in Protégé as previously specified. For illustrating this fact, we translate the first rule defined above by employing the syntax of Jena as follows:

[(?h rdf:type Human), (?a rdf:type Automobile), (?h :hasLocation ?a),
(?a :performsActivity Accelerating),
(?a :hasStatusState On) -> (?h :performsActivity **Driving**)]

For triggering queries to the model, we make available the 3LConOnt in the following URI: http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/module#resource. As it can be seen, such URI is composed by */module#resource*, where module is the name of the module that is needed in the query (e.g. ComputationalEntity), and resource is the class, instance, object and datatype property that is needed in the query. For instance, a specific URI for a query would be as follows: http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/ComputationalEntity#Software.

We also implemented a query engine by using the Jena API along with the SPARQL[10] query language. This query engine validates and assists the interaction with the three-level ontology by querying the context information that characterises the situation of entities such as services, user, provider, etc. Furthermore, the reasoning capabilities of the query engine are given by the Apache Jena framework, thus the context information gathered is also deduced by considering transitive and inferred relations. One of the methods for this purpose is the OntModelSpec. OWL_MEM_MICRO_RULE_INF containing a transitive reasoner which can be used to infer properties such as subClassOf and subPropertyOf. Therefore, the query engine is able to gather direct and indirect descendants of context information. In Fig. 18, a fragment of the console results when querying the subclasses of the Location module from the context model is shown. As it can be seen, if the query engine does not use reasoning capabilities, the subclasses of Location are only the direct ones, otherwise both direct and indirect subclasses are considered.

In addition, the context model proposed in this work is tested below by considering the use case scenario of the smart restaurant service described in Section 1 and 4.1. Such scenario can be addressed as follows: the query engine takes as input the name of a client "Emma Watson" that is required to be characterized by means of context information. Based on this input, the query engine requires the information from the context ontology that returns simple and deduced context information. Given the evaluation and analysis of the entities and its corresponding context information performed by the context ontology, the actions for adaptation purposes of the restaurant service are presented in Fig. 19.

[10] http://www.w3.org/TR/sparql11-query/

```
Without reasoning
----------------------------------------------------------
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#IndoorSpace>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#LocationCoordinates>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#OutdoorSpace>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#Region>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#RelativeLocation>
----------------------------------------------------------

Using reasoning capabilities
----------------------------------------------------------
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#IndoorSpace>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#LocationCoordinates>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#OutdoorSpace>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#Region>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#RelativeLocation>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#Building>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#Cave>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#Garage>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#Laboratory>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#Office>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Location#Automobile>
----------------------------------------------------------
```

Fig. 18 Reasoning capabilities of the query engine component

```
Entities and Context information
----------------------------------------------------------
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Resource#AzurmendiRest>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Resource#AzurmendiMen>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Agent#EmmaWatson>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Agent#JohnQ>
<http://gessi.lsi.upc.edu/threelevelcontextmodelling/ThreeLContextOnt/Activity#WeddingAnniversary-8>
----------------------------------------------------------
Actions for adaptation purposes
----------------------------------------------------------
<Preparing menu for candle light dinner>
<Booking a romantic place>
<Turn off the light>
----------------------------------------------------------
```

Fig. 19 Use case validation – smart restaurant service

Finally, we have executed some queries to the proposed model using the SPARQL into the Protégé platform. Such queries are related to the competence questions specified in Section 3.3 and the expected results should validate that layers and modules of the model, the extension and integration tasks, and the conceptualization of the domain ontology are correct and consistent. The queries are depicted as follows.

In Fig.20 is depicted a query to obtain all the monitoring tools that supervise the Twitter social network. As it can be seen, two types of monitoring tools are retrieved namely SocialMentionAPI and TwitterAPI. In Fig. 21 a variation of the query specified in Fig. 20 is depicted to obtain only the monitors that supervise Facebook. As it can be seen, only the SocialMentionAPI has such capability.

Fig. 22 shows the monitored data that can be obtained from a social network monitoring tool. As it can be seen, the timestamp is the generic monitored data that is related to a list of data items namely dataitemID, dataItemsLink, dataItemMessage, etc.

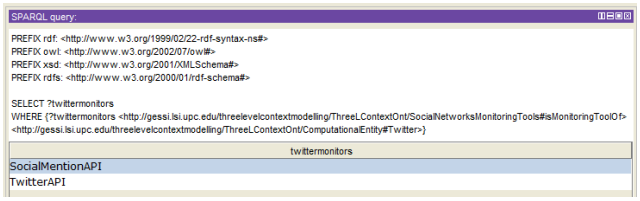

Fig. 20 SPARQL query of Twitter monitoring tools

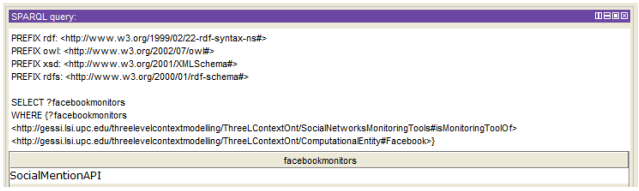

Fig. 21 SPARQL query of Facebook monitoring tools

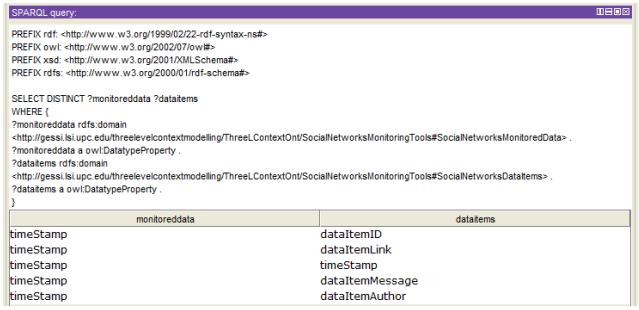

Fig. 22 SPARQL query of monitored data and data items related

4.3 Evaluating reusability

As previously mentioned, the levels and modules of the proposed model can be reused independently of the entire model. In other words, an ontology designer can integrate in an existing ontology the level of abstraction or module that is required without the dependency of other levels or modules of the model, since the semantic of each level and module is independent of each other. Such capability is validated here by illustrating into the Protégé tool the reuse of a level of the model in an existing ontology.

Consider the following case. The SOUPA ontology provides different ontologies that can be reused in different contextual domains. However, they lack of a semantic structure that can be given by an upper level ontology. For solving this issue, we suggest to reuse the upper-level ontology introduced in this paper. For instance, we opened in Protégé the Person ontology provided by SOUPA (see Fig. 23). As it can be seen in the figure, the Person ontology of SOUPA provide a vocabulary and a class hierarchy that can be complemented or restructured with a rich semantic. For this reason, we imported the upper-level ontology proposed in this work into the Person ontology of SOUPA (see Fig. 24). Hence, the class hierarchy depicted in Fig. 23 was restructured as depicted in Fig. 25 to provide better semantic of the proposed vocabulary in the SOUPA ontology. In this regard, we can say that the contact profile is a context information that belongs to a generic class named profile that can describe an agent through a resource.

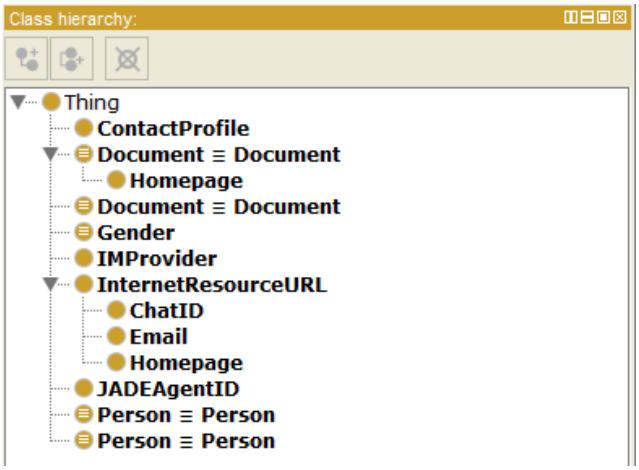

Fig. 23 Person ontology of SOUPA

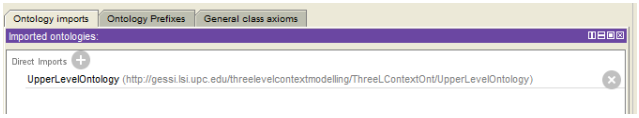

Fig. 24 Importing the upper-level ontology

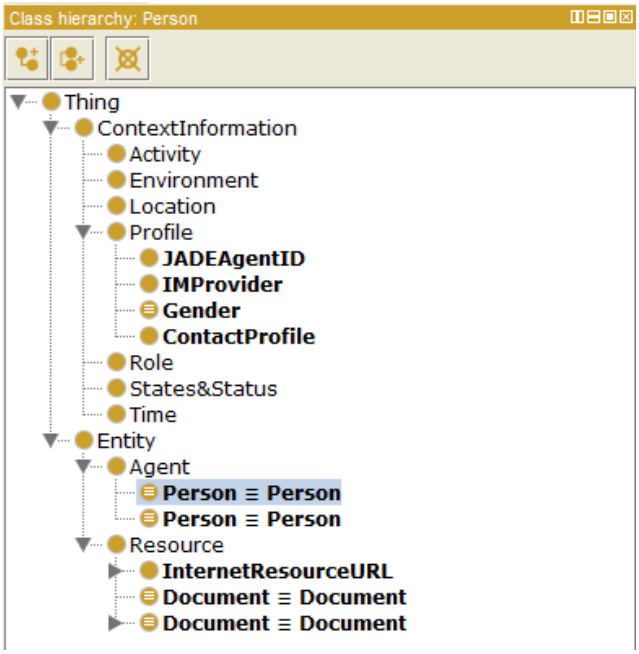

Fig. 25 Rich semantic of the person ontology by reusing the upper-level ontology

4.4 Evaluating correctness and completeness

The details and results of this validation are specified as follows:

A) Correctness. To carry out this validation we rely on the advises given in [53] that specifies that *"a basic requirement for a modular ontology to be correct is that each module is correct"*. This is the first perspective that we have adopted to conduct the correctness validation of the proposed ontology. Second, we have conducted a syntactic correctness, consistency among the specified primitives, and consistency between instances and specifications through automatic tools as it is also recommended. For this purpose, we have used 1) the Pellet reasoning engine available in Protégé (see Fig. 26). Note that some rules were used in this evaluation to improve the validation; 2) a validation service[11] provided by the W3C where the "triple" view (RDF) of the proposed model was also evaluated (see Fig. 27); and 3) RaDON [33] plugging of NeOn Toolkit[12] that has the aim of verifying inconsistencies and incoherencies in ontologies (see Fig. 28). As it can be seen, the results obtained in this validation process showed that none inconsistencies were found in any of the modules and layers of the proposed ontology. Note the dotted squares in each figure. In Fig. 26 "reasoner active" means that none inconsistencies were found by the reasoning engine.

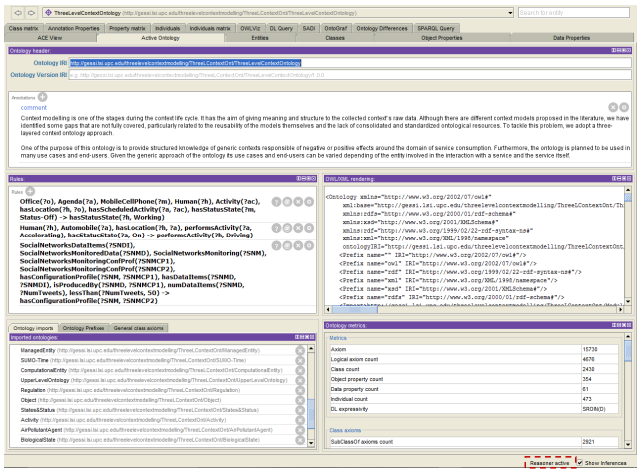

Fig. 26 Validating correctness through the reasoning engines of Protégé

B) Completeness. At this respect, it is important to highlight that although the proposal of this work takes as a basis 8 representative ontologies that were commonly referenced in the context modelling research area, the context knowledge, semantics and patterns of our proposal is stated also by considering the 138 contributions that were selected in the systematic mapping study. From this perspective, we consider that the completeness of the context knowledge represented in our proposal has been improved with respect to the knowledge represented in the analysed contributions. Hence, to carry out this validation we have performed a comparative analysis of completeness between our proposal and the 138 proposals of context modelling. Please find such report in the annexes of this paper at [10] since the comparative table is too big to be specified here. As we have expected, the results of such study show that our proposal has achieved 100% of completeness in the knowledge represented in the main modules of the upper-level ontology, i.e., we explicitly provided an structured vocabulary for Agent, Resource, Activity, Time, Environment, Location, Role, and States and Status.

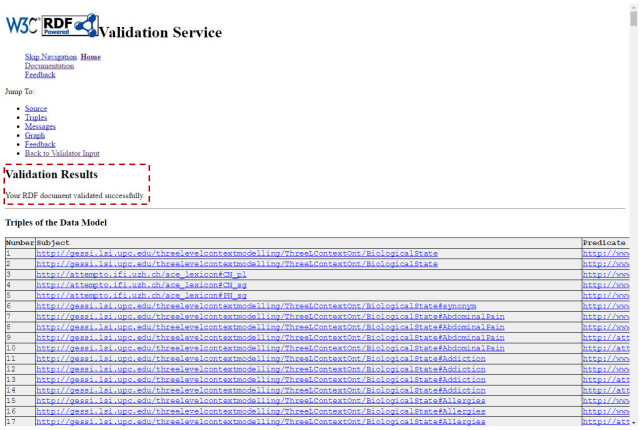

Fig. 27 Validating correctness through the validation service of W3C

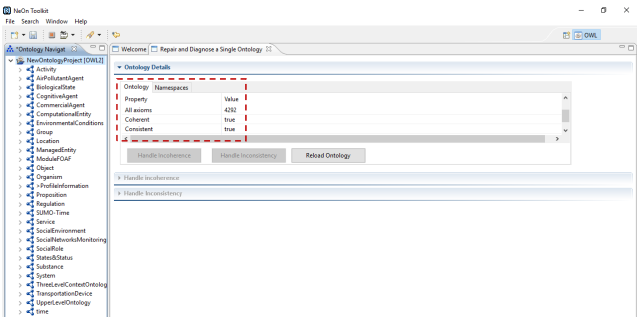

Fig. 28 Validating correctness through RaDON plugging of NeOn Toolkit

[11] https://www.w3.org/RDF/Validator/
[12] http://neon-toolkit.org/wiki/Main_Page.html

5 Putting the ontology into work: a context-aware architecture

As an added value to the usability of the proposed context model, we propose a context-aware architecture depicted in Fig. 29 for illustrating the role that can play the context model in different use cases of the service-oriented computing. This includes: structuring and unifying context data; configuration, adaptation and evolution of a monitoring infrastructure; ranking and adaptation of services and applications; evolution and adaptation of personalized software; improving the QoE of the user; etc. the context-aware architecture is intended to support the context life cycle taking as kernel the proposed context model. According to Dey et al. [22], context-aware frameworks should support acquisition, representation, delivery, and reaction. For this purpose, the architecture is designed around three main components, namely monitoring infrastructure, the manager engine and the context ontology (see Fig. 29).

As it can be seen in the figure, the context ontology proposed in this work is responsible for structuring, reasoning and disseminating high-level context data. Thus, the model provides the data schema of the context data repository to unify and structure the acquired current context data coming from the monitoring infrastructure component. The manager engine that has two important roles namely analyser and enactor manages the context ontology and the context data repository to perform its tasks. First, the analysis and evaluation of the context data sent from the context-aware monitoring infrastructure. Second, based on the analysis and evaluation performed, the manager engine can decide and enact the needed actions in an application domain (e.g., supporting the configuration/adaptation of the monitoring infrastructure and other services or applications). Since queries and rules depend on the application domain, the user/agent plays an important role to specify them based on the functional requirements (e.g., competency questions) of a domain ontology. Hence, the manager engine can be able to extract direct or indirect information from the context ontology that provides new facts through its reasoning capabilities.

We also consider that the proposed context-aware architecture is generic enough to be aligned with other context-aware perspectives. For instance, the awareness layered view suggested in [38] can be processed and mapped in our proposal as follows: 1) the context middleware layer can be addressed by the interaction of the motoring infrastructure with the context ontology,

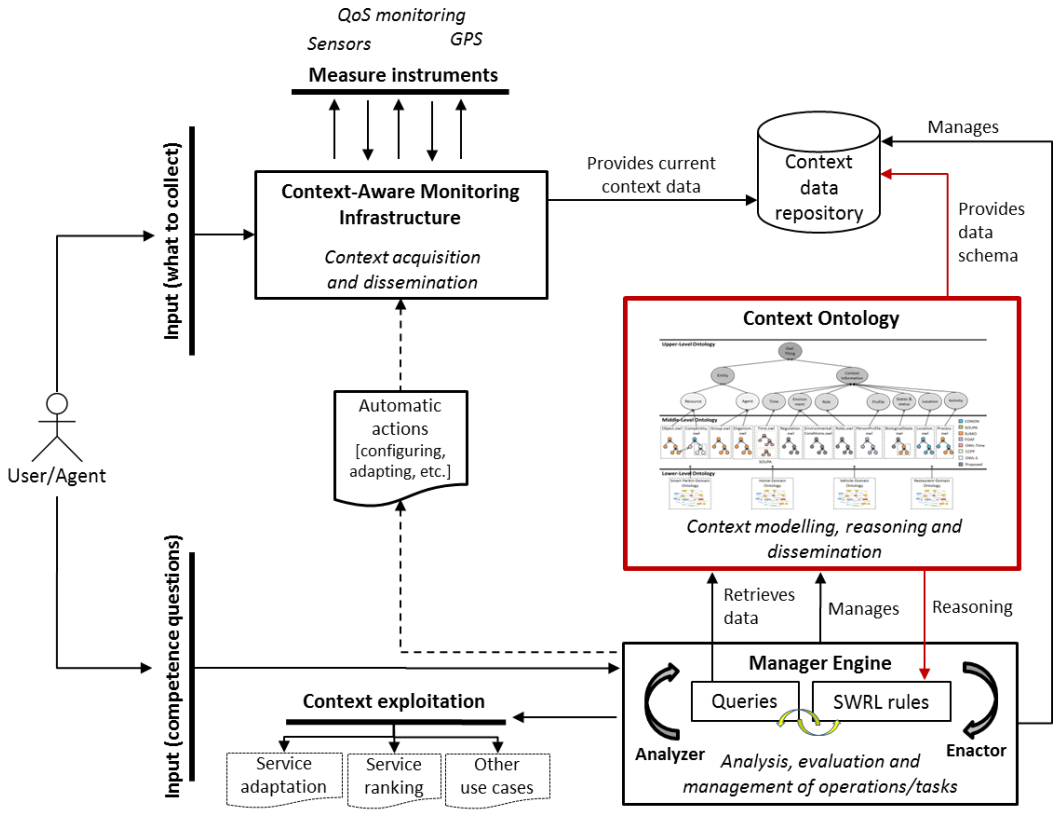

Fig. 29 A context-aware architecture

since these components collect and maintain the context data; and 2) the awareness and sensitivity layer can be addressed by the interaction of the manager engine and the context ontology, since these components can assess and classify the situational context of an entity as well as select, decide and execute the feasible actions that can be enacted in a context-aware service or application.

6 Conclusions

The contribution of this work is focused on 3LConOnt, a three-level context model with the goal of being easy to reuse, extend or adapt in different smart scenarios. This context model has been implemented and integrated in all its levels of abstraction following the next steps: 1) we consolidated a upper-level ontology through a systematic mapping which states high level context classes; 2) we extended, integrated and consolidated modules of context knowledge in a middle-level ontology following a detailed integration process; 3) we built different domain ontologies for validating and representing the role of the lower-level ontology in the proposed model.

Summing up, we aim at building a proposal aligned with existing context ontologies, reusing ontological resources semantically well-defined acting as a pattern repeated in different proposals. To this purpose, we reconcile the most appropriate aspects of existing contributions coming from the systematic mapping study.

We also validated the capabilities of the proposed three-level ontology by considering: 1) reusability, extensibility and adaptation by instantiating concrete classes or instances coming from different smart scenarios that have illustrated its level of generality; 2) consistency and reasoning by triggering queries from the perspective of a smart restaurant service and the proposed domain ontology that conceptualizes social networks monitoring tools (see Section 3.3); and 3) its reusability in existing context ontologies demonstrating that each level or module of the model is independent of the entire model. Additionally, as an added value we illustrated the usability of the proposed model in service-oriented computing by presenting a context-aware framework for supporting the whole context life cycle: acquisition, modelling, reasoning, and distribution. In this regard, the context model takes an active role for configuring a monitoring framework and for conducting the analysis useful to take decisions in different use cases.

We also adopt a service-centric perspective in the work because we mainly consider the value provided from services to customers and the capability to represent a generic body of context knowledge from different perspectives. In this sense, different circumstances of a service and other important entities can be understood such as the place in which a service can be executed, to delimit and understanding the behaviour and conversation among entities, to identify risks in the process of service provisioning and consumption, to understand customers, to extend and maintain the service life cycle, among others benefits.

Finally, the context model proposed in this work is being validated in the European project named SUPERSEDE[13] by providing the data schema and reasoning capabilities needed in the project. At the moment, we consider that the results obtained through the validation specified in the paper provide valuable results to expose the main benefits of the proposed model, fulfilling different capabilities of a consistent model such as generality, reusability, integration, extensibility, completeness, etc. Last, the usage of the results of this contribution could be instantiated in several use cases including Web service selection, monitoring frameworks and tools, static or dynamic evolution and adaptation of services and applications, among others. In fact, as future work we want 1) to evaluate the context model in different actual and trending cases of smart cities and internet of things to unify and increase the semantic and meaning of the data obtained from sensors and exchanged among different real context-aware services; 2) to evaluate the performance overhead of the ontology in practice; 3) to extend our Web service selection framework presented in [12] by considering context information in the selection process, and for reasoning context information in charge of maintaining highest quality standards regarding functional and non-functional features of services.

Acknowledgements This work is partially supported by the Spanish project TIN2016-79269-R and the SUPERSEDE project, funded by the European Union's Information and Communication Technologies Programme (H2020) under grant agreement no 644018.

References

1. Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: 1st International Symposium on Handheld and Ubiquitous Computing, pp. 304–307. Springer (1999)
2. Badidi, E., Taleb, I.: Towards a cloud-based framework for context management. In: 7th International Conference on Innovations in Information Technology, pp. 35–40. IEEE (2011)
3. Bazire, M., Brézillon, P.: Understanding context before using it. In: 5th International and Interdisciplinary

[13] https://www.supersede.eu/

Conference on Modeling and Using Context, pp. 29–40. Springer (2005)
4. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing **6**(2), 161–180 (2010)
5. Bikakis, A., Patkos, T., Antoniou, G., Plexousakis, D.: A survey of semantics-based approaches for context reasoning in ambient intelligence. In: European Conference on Ambient Intelligence, pp. 14–23. Springer (2007)
6. Borst, W.N.: Construction of engineering ontologies for knowledge sharing and reuse. Ph.D. thesis, Universiteit Twente, Enschede (1997)
7. Brusa, G., Caliusco, L., Chiotti, O.: Towards ontological engineering: a process for building a domain ontology from scratch in public administration. Expert Systems **25**(5), 484–503 (2008)
8. Cabrera, O., Franch, X., Marco, J.: A context ontology for service provisioning and consumption. In: 8th International Conference on Research Challenges in Information Science, pp. 1–12. IEEE (2014)
9. Cabrera, O., Franch, X., Marco, J.: A middle-level ontology for context modelling. In: 34th International Conference on Conceptual Modeling, pp. 148–156. Springer (2015)
10. Cabrera, O., Franch, X., Marco, J.: Online annexes of 3LConOnt: A three-level ontology for context modelling in context-aware computing. http://gessi.lsi.upc.edu/threelevelcontextmodelling/ (2017). Accessed: 2017-03-15
11. Cabrera, O., Franch, X., Marco, J.: Ontology-based context modeling in service-oriented computing: a systematic mapping. Data & Knowledge Engineering (2017). To appear
12. Cabrera, O., Oriol, M., Franch, X., Marco, J., López, L., Fragoso, O., Santaolaya, R.: Open framework for web service selection using multimodal and configurable techniques. Computación y Sistemas **18**(4), 665–682 (2014)
13. Cadenas, A., Ruiz, C., Larizgoitia, I., García-Castro, R., Lamsfus, C., Vázquez, I., González, M., Martín, D., Poveda, M.: Context management in mobile environments: a semantic approach. In: 1st Workshop on Context, Information and Ontologies, pp. 2:1–2:8. ACM (2009)
14. Cao, Y., Klamma, R., Hou, M., Jarke, M.: Follow me, follow you-spatiotemporal community context modeling and adaptation for mobile information systems. In: 9th International Conference on Mobile Data Management, pp. 108–115. IEEE (2008)
15. Chen, H., Finin, T., Joshi, A.: An ontology for context-aware pervasive computing environments. The knowledge Engineering Review **18**(3), 197–207 (2003)
16. Chen, H., Perich, F., Finin, T., Joshi, A.: Soupa: standard ontology for ubiquitous and pervasive applications. In: 1st International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 258–267. IEEE (2004)
17. Corcho, O., Fernández-López, M., Gómez-Pérez, A.: Methodologies, tools and languages for building ontologies. where is their meeting point? Data & Knowledge Engineering **46**(1), 41–64 (2003)
18. Coutaz, J., Crowley, J.L., Dobson, S., Garlan, D.: Context is key. Communications of the ACM **48**(3), 49–53 (2005)
19. Curtis, J., Cabral, J., Baxter, D.: On the application of the cyc ontology to word sense disambiguation. In: 19th International Florida Artificial Intelligence Research Society Conference, pp. 652–657. AAAI Press (2006)
20. De Nicola, A., Missikoff, M., Navigli, R.: A software engineering approach to ontology building. Information systems **34**(2), 258–275 (2009)
21. Dey, A.: Understanding and using context. Personal and Ubiquitous Computing **5**(1), 4–7 (2001)
22. Dey, A., Abowd, G., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction **16**(2), 97–166 (2001)
23. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: Methontology: from ontological art towards ontological engineering. In: Ontological Engineering AAAI Spring Symposium Series, pp. 33–40. American Asociation for Artificial Intelligence (1997)
24. Gruber, T.: A translation approach to portable ontology specifications. Knowledge Acquisition **5**(2), 199–220 (1993)
25. Grüninger, M., Fox, M.: Methodology for the design and evaluation of ontologies. In: 5th IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, pp. 14–24. IJCAI Inc. (1995)
26. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. Network and Computer Applications **28**(1), 1–18 (2005)
27. Gu, T., Wang, X., Pung, H., Zhang, D.Q.: An ontology-based context model in intelligent environments. In: Communication Networks and Distributed Systems Modeling and Simulation Conference, vol. 2004, pp. 270–275 (2004)
28. Guarino, N.: Formal ontology and information systems. In: 1st International Conference on Formal Ontology in Information Systems, pp. 81–97. ACM (1998)
29. Hella, L., Krogstie, J.: A structured evaluation to assess the reusability of models of user profiles. In: Enterprise, Business-Process and Information Systems Modeling, pp. 220–233. Springer (2010)
30. Henricksen, K.: A framework for context-aware pervasive computing applications. Ph.D. thesis, University of Queensland, Brisbane (2003)
31. Hong, J.y., Suh, E.h., Kim, S.J.: Context-aware systems: a literature review and classification. Expert Systems with Applications **36**(4), 8509–8522 (2009)
32. Hu, B., Moore, P., Chen, H.H.: A semantic context model for location-based cooperative mobile computing. In: 13th International Conference on Communications, pp. 326–331. IEEE (2007)
33. Ji, Q., Haase, P., Qi, G., Hitzler, P., Stadtmüller, S.: RaDON — repair and diagnosis in ontology networks, pp. 863–867. Springer, Berlin, Heidelberg (2009)
34. Kayes, A., Han, J., Colman, A.: Ontcaac: an ontology-based approach to context-aware access control for software services. The Computer Journal **58**(11), 3000–3034 (2015)
35. Kayes, A., Han, J., Colman, A.: An ontological framework for situation-aware access control of software services. Information Systems **53**, 253–277 (2015)
36. Kim, E., Choi, J.: An ontology-based context model in a smart home. In: 6th International Conference on Computational Science and Its Applications, pp. 11–20. Springer (2006)
37. Kishore, R., Sharman, R.: Computational ontologies and information systems i: foundations. Communications of the Association for Information Systems **14**(1), 158–183 (2004)

38. Kofod-Petersen, A., Aamodt, A.: Contextualised ambient intelligence through case-based reasoning, pp. 211–225. Springer, Berlin, Heidelberg (2006)
39. Li, M.: Ontology-based context information modeling for smart space. In: 10th International Conference on Cognitive Informatics & Cognitive Computing, pp. 278–283. IEEE (2011)
40. Miller, G.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
41. Niles, I., Pease, A.: Towards a standard upper ontology. In: 2nd International Conference on Formal Ontology in Information Systems, pp. 2–9. ACM (2001)
42. Noy, N.: Semantic integration: a survey of ontology-based approaches. ACM Sigmod Record **33**(4), 65–70 (2004)
43. Paganelli, F., Giuli, D.: An ontology-based context model for home health monitoring and alerting in chronic patient care networks. In: 21st International Conference on Advanced Information Networking and Applications Workshops, pp. 838–845. IEEE (2007)
44. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: A survey. IEEE Communications Surveys & Tutorials **16**(1), 414–454 (2014)
45. Pinto, S., Gómez-Pérez, A., Martins, J.P.: Some issues on ontology integration. In: 16th IJCAI workshop on Ontologies and Problem-Solving Methods, pp. 1–12. IJCAI Inc. (1999)
46. Pinto, S., Martins, J.P.: A methodology for ontology integration. In: 1st International Conference on Knowledge Capture, pp. 131–138. ACM (2001)
47. Prekop, P., Burnett, M.: Activities, context and ubiquitous computing. Computer Communications **26**(11), 1168–1176 (2003)
48. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: 1st Workshop on Mobile Computing Systems and Applications, pp. 85–90. IEEE (1994)
49. Schmidt, A.: Context-aware computing: context-awareness, context-aware user interfaces, and implicit interaction. The Encyclopedia of Human-Computer Interaction, 2nd Ed. pp. 1–28 (2013)
50. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical owl-dl reasoner. Web Semantics: Science, Services and Agents on the World Wide Web **5**(2), 51–53 (2007)
51. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: 1st International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp, pp. 1–8. University of Southhampton (2004)
52. Strimpakou, M., Roussaki, I., Anagnostou, M.: A context ontology for pervasive service provision. In: 20th International Conference on Advanced Information Networking and Applications, pp. 775–779. IEEE (2006)
53. Stuckenschmidt, H., Parent, C., Spaccapietra, S.: Modular ontologies: concepts, theories and techniques for knowledge modularization. Lecture Notes in Computer Science. Springer (2009)
54. Su, X., Ilebrekke, L.: A comparative study of ontology languages and tools. In: 14th International Conference on Advanced Information Systems Engineering, pp. 761–765. Springer (2002)
55. Sudhana, K.M., Raj, C., Suresh, R.: An ontology-based framework for context-aware adaptive e-learning system. In: International Conference on Computer Communication and Informatics, pp. 1–6. IEEE (2013)
56. Uschold, M., King, M.: Towards a methodology for building ontologies. In: 5th IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, pp. 1–13. IJCAI Inc. (1995)
57. Wang, H., Zhang, Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: 2nd Conference on Pervasive Computing and Communications Workshops, pp. 18–22. IEEE (2004)
58. Xiong, Z., Dixit, V., Waller, T.: The development of an ontology for driving context modelling and reasoning. In: 19th International Conference on Intelligent Transportation Systems, pp. 13–18. IEEE (2016)
59. Xynogalas, S., Roussaki, I., Chantzara, M., Anagnostou, M.: Context management in virtual home environment systems. Circuits, Systems, and Computers **13**(02), 293–311 (2004)