Making Smart Buildings and Personal Systems Cooperate via Knowledge Base Overlays

Making Smart Buildings and Personal Systems Cooperate via Knowledge Base Overlays

Ester Giallonardo
Dept. of Engineering, University of Sannio, Italy
egiallonardo@unisannio.it

Francesco Poggi
Dept. of Communication and Economics, University of Modena and Reggio Emilia, Italy
francesco.poggi@unimore.it

Davide Rossi
Dept. of Computer Science and Engineering, University of Bologna, Italy
daviderossi@unibo.it

Eugenio Zimeo
Dept. of Engineering, University of Sannio, Italy
eugenio.zimeo@unisannio.it

ABSTRACT

Reactive IOT applications often have to deal with the data source Babel arising from their need to operate on context information originated from different data sources.
Semantic knowledge bases can be fruitfully deployed to alleviate this problem: they provide a unified access point for context information including both long term (such as the structure of the environment) and transient (such as sensor readings) data thanks to their ability to host elements responding to different schemas within the same container.
In past works we introduced an architecture to create reactive IOT systems based on a semantic knowledge base that also hosts the definition of their behavior and on an accompanying reactive machinery.
In this paper we introduce the use of knowledge base overlays, i.e. containers providing a live, unified view over (parts of) different underlying knowledge bases, as a mechanism to enable interoperation between multiple IOT semantics-based systems. Specifically we explore the benefits of this approach in a case study in which a semantic IOT system governing a smart building interacts with the personal semantic systems of the people entering the building[1].

CCS CONCEPTS

• Computer systems organization → Embedded and cyber-physical systems → **Sensor networks**; Information systems → Data management systems → **Information integration**; Information systems → World Wide Web → Web data description languages → Semantic web description languages → **Web Ontology Language (OWL)**.

KEYWORDS

Context modeling, Context-awareness, Semantic modeling, Semantic Sensor Networks, Ontologies, Models@runtime, Reactive systems.

1 INTRODUCTION

IoT, smart cities, cyber-physical and sensor networks applications are reactive and context-aware. A *reactive system* is a system that computes by reacting to stimuli coming from its environment. These stimuli are reported by an array of sensors whose readings trigger the application logic implementing the reactive behavior. Examples can span from simple open loop systems, such as a home automation application (e.g. a light switch actuator is fired when a light sensor reports a reading below a given threshold), to very complex systems, such as smart cities applications.

What these systems are able to sense (or to act on) represents their *context* which is usually reflected as a dynamic model within the system [3]. The structure of this model can be very simple (e.g., a collection of variables representing the latest observations reported by sensors) or very articulated (e.g. a megamodel, as the model of models proposed for self-adaptive systems

Goodtechs '20, September 2020, Antwerp, Belgium

in [15]). Depending on the relationship between context and application, in fact, we are presented with a spectrum ranging from simple reactive systems (the application logic is immutable but is able to change the context) to self-adaptive ones (the application logic can change dynamically according to the context) [2][10]. All these systems share the need to reason upon context at runtime, and can benefit from a flexible, expressive and queryable representation of context.

In our previous works [12,5] we explored semantic knowledge-based systems, reactive systems that make use of a unified knowledge base containing all contextual information expressed in RDF.
We leveraged the ability of RDF to "facilitate data merging even if the underlying schemas differ"[2] which makes it the ideal candidate for applications that have to deal with diverse information coming from different sources but that, ultimately, refer to the context of a system.
By adopting RDF we seamlessly allow all information to be stored in the same container making it easier to provide a unified information repository presenting all the data that the reactive logic needs to access.
In our case studies in the domain of smart buildings, we used the semantic knowledge base to store information about the structure of the context, such as the organization of the building in terms of levels, rooms, doors, etc., and transient information about relevant features of interest associated with this structure, such as the temperature of a room, the light level of a corridor, etc.
This latter knowledge "flows-in" into the system in the form of readings reported by a plethora of sensors placed within the building.
We adopted the W3C SSN ontology to represent this information, and we used different domain-specific ontologies for the remaining context elements.
In our proposal a further class of information represented in the semantic knowledge base is represented by the reactive elements that define the behavior of the systems, that we modeled with an extension of SSN called Logical Sensors and Actuators (LSA) ontology [4]. We envisioned that our architecture would accommodate different systems under the same umbrella by effectively merging all the elements in an overarching system of systems.
In the work we present in this paper we assume a different approach in which various systems interact while maintaining a strict identity. Specifically we focus on cyber physical scenarios in which systems such as the one managing a smart building and the ones hosted in personal mobile devices get in contact and how they can fruitfully interact and influence each other's behavior.
The remainder of this paper is organized as follows. Section 2 presents the related work from both research and standardization points of view. Section 3 introduces a case study and describes the reference architecture proposed for making reactive semantic systems cooperate. Section 4 describes the prototype developed to test our approach. Section 5 concludes the paper and highlights future work.

2 BACKGROUND ON CONTEXT MODELING AND SEMANTIC KNOWLEDGE

Various recent research works take the idea of using models as central artifacts to cope with dynamic aspects of ever-changing software and its environment. Szvetits et al. [14] comprehensively survey these kinds of approaches for adaptive context-aware systems highlighting the common idea of establishing semantic relationships between executed applications and runtime models based on monitoring events.

Most current research efforts recognize the need for runtime models of both system and context. These two kinds of models should be semantically related since a change in the context model should be associated with a variability alternative to introduce into the current configuration of the system. According to these requirements, several efforts have tried to propose semantics to easily model and handle dynamic context-aware applications, especially for ubiquitous and pervasive computing.

One of the first ontology-based approaches is SOUPA [1]. It is expressed in OWL and includes modular component vocabularies to represent intelligent agents, time, space, events, user profiles, actions, and policies for security and privacy. However, it does not focus on sensors/actuators and reactive systems but on smart meeting places.

Paper [11] surveys context awareness from an IoT perspective. IoT researchers are taking into consideration Web technologies (WoTs) to support context-driven system engineering. The goal of the WoT is to extend Web services to devices, allowing a Web client to access device properties, to request the execution of actions or to subscribe the events representing state changes [9]. The related ontology describes how to model sensors and actuators with the main objective of easing the binding with devices reachable through web protocols (REST, CoAP, etc.).

A different objective is pursued by the Semantic Sensor Network (SSN) ontology [7], an Open Geospatial Consortium (OGC)/World Wide Web Consortium (W3C)

[2] https://www.w3.org/RDF/

standard. It is mainly focused on the SOSA (Sensor, Observation, Sample, Actuator) pattern [8] to model reactive systems. It aims at supporting the definition of simple reactive behaviors that link observations, coming from modeled sensors, with the related reactions, performed by actuators. In order to link observations to physical or virtual properties, the SOSA pattern is extended with some system-oriented features. However, SSN does not directly support complex processing inside the knowledge base rather than asserting facts due to external sensing activities.

In our previous work [4] we introduced our extension to SSN named Logical Sensors and Actuators (LSA) ontology. This extension facilitates the development of reactive systems by introducing two main concepts: (software) *logical sensors* and *logical actuators*. A logical sensor (resp. actuator) is a sensor (resp. actuator) that generates observations (resp. actuations) by executing a software procedure that uses as inputs the observations produced by other (one or more) sensors or actuators. Logical sensors and actuators live only in the virtual space (e.g. a knowledge base), and are bound to real world entities by grounding (mainly via web services) to physical sensors and actuators. The behavior of the system can be specified by using sensing or actuating procedures tied to the logical devices provided by the semantic model. These procedures can act upon the knowledge base by generating new facts or by redefining the structure of the model. An example of how logical elements can be leveraged to implement reactive systems is the reference architecture described in Section 3.

The Semantic Smart Sensor Network (S3N) ontology [13] is another effort that tries to specialize SSN for supporting the modeling of smart sensors. To this end the ontology introduces the concept of smart sensor, that is an aggregate element composed of embedded sensors, microcontrollers and communicating systems. The behavior is expressed by an algorithm selected among the existing ones on context basis. The microcontroller is able to select algorithms from the current context and to change the state of the whole smart sensor. Therefore, the main purpose of S3N is to support smart sensor description, but it is not able to support the execution of its logic.

3 CASE STUDY AND SYSTEM ARCHITECTURE

As stated in the introduction we focus on a cyber physical context in which systems such as the one managing a smart building and the ones hosted in personal mobile devices get in contact. In such a context it is not reasonable for the different systems to share the very same knowledge base.
This applies to all the cases in which the systems may want to maintain independence from their collaborators, because they express a high degree of asymmetry in terms of available resources, or even because of privacy concerns.
In this paper we use a case study to justify the rationale leading to the design of our solution.
The case study we adopt is depicted in Fig. 1 and can described as follows: a smart building is managed using a system implementing the reference architecture based on the semantic knowledge base (top of Fig. 1a). The building's knowledge base hosts static information, mainly pertaining the structure of the building (encoded as location-aware entities following the Spatial Data on the Web Best Practices), transient information produced by a sensor network (encoded with SSN) and behavioral information associating relevant contextual changes with corresponding activities that have to be undertaken when the changes take place (encoded with our LSA ontology).

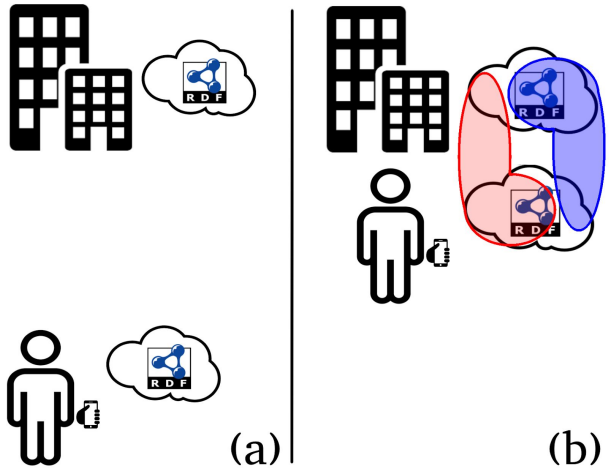

Figure 1: The case study consists of a smart building and a user, both equipped with a knowledge base (a). When they get in contact (b), they start to cooperate by sharing their information and creating two overlays: the user's one (in red), and the building's one (in blue).

These activities implement the management of the lighting system, of the heating/ventilating system and so forth.
People come and go in this building and carry with them personal mobile devices; we basically assume that whereas a single person can host multiple personal mobile devices their smartphone acts as a hub concentrating and distributing information to and from other *wearable things*.
Each smartphone hosts its own knowledge base (bottom of Fig. 1a) containing static information about its owner (such

as personal data, user preferences, etc.) and transient information in the form of observations collected by the sensors hosted in the smartphone itself and from other personal smart devices (heart rate straps, smartwatches, etc.). Reactive behavior can also be defined in the mobile knowledge base but, given the energy concerns characterizing these devices, it is reasonable to assume that activities are limited to showing notifications and other similar elementary tasks.

When these systems (the smart building and the smartphone) get in contact, that is when a person enters the building as depicted in Fig. 2b, we want them to cooperate.

The basic intuition here is that a new knowledge base overlay can be created by offering a uniform access point to all the static information about the building and the user and by collecting all transient information produced by the sensors hosted in the building and weared by the user. This way the users become aware of the structure of the building so that, for example, they can easily locate the nearest elevator, whereas the building becomes aware, for example, that the person (that now becomes a user) prefers a dim background light. Also: users have access to the readings from the building's sensors and vice versa, enabling an array of adaptive behaviors.

While this approach has the value of making it clear the advantages of the interplay between the knowledge bases, it is too simplistic for several reasons. To spot a couple: (i) users may be unwilling to share all their personal information but they prefer to share only a limited, selected dataset; (ii) the mobile devices risk to be saturated by the large number of readings coming from the building's sensors.

A reasonable refinement of the basic intuition can then be formulated as follows: when entering the smart building, users are asked to share specific elements stored in their mobile knowledge base, such as selected personal information and selected preferences. If users agree, these become part of the building's overlay. Conversely the smart building presents the static information it is willing to share and only the elements accepted by the user becomes part of the user's overlay.

Similarly users decide which of their sensors they want to be visible to the building whereas the building offers to the users the option to register for available sensors.

With this approach we will have a building's overlay composed by the building's original knowledge base enriched with users information (static and transient) and a mobile overlay for each user.

We can think of these overlays as the representation of what can be called the *elastic context* of these systems, that is a context that expands and retracts depending on the presence of other cooperating systems.

A bi-directional heartbeat protocol is used to ensure that a person is currently in the building, when the protocol signals that a user left the building, its part of the knowledge base is removed from the building's overlay and the building's sensors stop reporting their readings to the mobile system.

To better grasp the kind of scenarios enabled with this approach let us suppose that a user encodes in her mobile knowledge base her preferences about ambient heating: 21 °C when the activity level is low and 19 °C when the activity level is high. When the user enters the smart building she agrees to share these preferences with the building's system. She also agrees to share the readings from her smartwatch's heart rate sensor. Finally she agrees to let the beacon presence sensors locate her in the building.

Now this information becomes part of the building's overlay and is thus made available to its reactive logic.

The heating system is managed by logical actuators that are fired upon the periodical reports from the temperature sensors in the various rooms and places.

These actuators access the knowledge base to retrieve all the information needed to decide whether the heating for a specific place has to be increased or decreased. Since the knowledge base overlay now also hosts the preferences of the users and reports from their heart rate sensors, the heating logical actuators can factor into their decision for a specific room the preferences of the users hosted in this room. They can also consider users' activity level that can be derived from their heart rate, which is also available in the overlay knowledge base.

For example the reactive logic can query the knowledge base to retrieve (i) the temperature preferences for all the persons hosted in a given room, and (ii) their average heart rate for the last five minutes. It can then combine this information to decide whether the heating should be raised or lowered.

This proposed method allows a seamless interoperation between the different systems while keeping a unified consistent view of the context for the logic driving their behaviors.

This approach can be implemented using the general architecture for reactive systems presented in [6]. The main component of the architecture is the Semantic Engine (see Fig. 2). It extends a knowledge base with the machinery needed to interact with sensors and actuators. The knowledge base contains a model of the physical world it interacts with, which is enriched and modified with the data coming from the sensors, assuring consistency with the physical elements it represents.

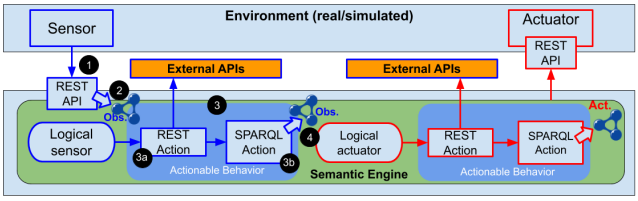

Figure 2: Architecture outline.

The engine connects to the physical world by exposing a service API used to *receive observations* from external sensors (see Fig. 2 on the left side) that can be real or simulated ones, and by invoking web service endpoints for *activating external actuators* (see Fig. 2 on the right side).
Whenever an external sensor notifies an observation invoking the engine's API (represented by the black circle with a "1" inside in Fig. 1), that observation is transformed in a semantic format and added to the knowledge base (2 in the figure).
If a logical sensor/actuator is interested in that observation, then its related software procedure is executed (3 in the figure) by running the actions (3a and 3b in the figure) that define the specific actionable behavior, producing new facts (observations or actuations, 4 in the figure). The actionable behavior can be composed of various action elements, such as SPARQL queries (e.g. to retrieve existing information and add new one), the execution of local (Java) code and the invocation of external services (depicted in orange) to increment the capabilities of the logical sensors and actuators. The two actions depicted in the figure are just a possible example. If new actuations are produced, they trigger external actuators through REST API invocations.

4 PROTOTYPE

We created a simple proof-of-concept prototype based on our Semantic Engine[3] to test the proposed approach. The prototype uses a simulated virtual environment generated with Freedomotic[4] describing a simple smart building composed by a corridor and three rooms.
Notice that in what follows we will refer to the sensors/actuators in the simulated environment as *virtual* sensors/actuators whereas by *logical* sensors/actuators we refer to the elements defined in the semantic knowledge base and responsible for implementing reactive behaviors.
Within the simulated environment a bluetooth-based presence sensor is deployed in each room. Virtual temperature actuators for the heating systems that accept a target temperature as parameter are also available in each room.
The building's virtual sensors produce observations that are reported in the form of actual REST API invocations to the Semantic Engine.
Whereas the system governing the building is a stand-alone application that interacts with the virtualized environment, the mobile systems for the users are fully virtualized (that is emulated and implemented as Freedomotic plugins) and limited to the publishing of preferences and heartbeat information. When a simulated user is instantiated in the virtual environment it (i) interacts with the semantic engine REST API to register its preferences, and (ii) starts producing observations by invoking another endpoint of the engine REST API (similarly to what the building's virtual sensors do).
The semantic engine creates its overlay by simply adding user preference triples to its knowledge base whenever a new user registers, and removes them when the user unregisters. An actual overlay is a live uniform representation of different underlying data sources, our solution for the prototype is a cheap trick to emulate the workings of a real overlay with obvious limits (for example if users change their preferences while in the building the emulated overlay is unaffected), but it is sufficient as a proof-of-concept.
Within the semantic engine a reactive logic is defined to manage the heating system. These logical elements interact (by invoking Freedomotic's REST API) with the virtual target temperature actuators.
The basic reactive behavior governing the heating system (changing the target temperature on a time-of-day basis) has been replaced to implement the preference-based logic previously described.
The structure of the reaction is as follows: each time a new user enters/exits a room the virtual presence sensor invokes the semantic engine API to post a new presence observation. This observation triggers two logical elements, one responsible for updating the presence map of the building (in the knowledge base), and another of retrieving (using a SPARQL query) the preferences of all the users now present in the room to determine a new target temperature. This fires a logical actuator that invokes Freedomotic's API to interact with the (virtual) temperature actuator.
When running the simulation, users can be moved within the virtual environment by clicking on a 2D map of the building. Users movements between different places activate the virtual presence sensors triggering the behavior described above, as displayed in Fig. 3: notice the presence

[3] https://github.com/ cars-team/semanticengine
[4] https://www.freedomotic-iot.com

sensors in the bottom left corner of the rooms and the thermostat actuator in the lower right.

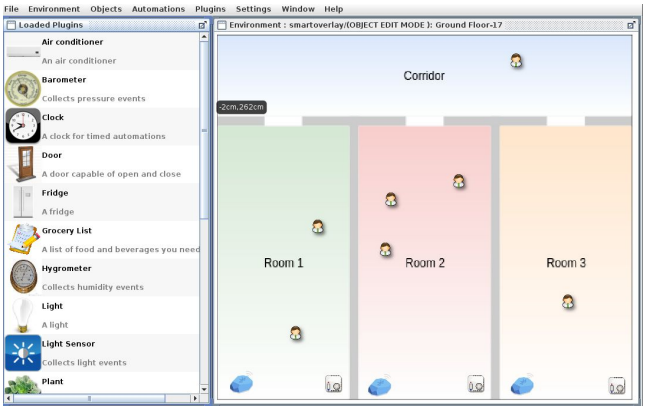

Figure 3: A simulated environment in Freedomotic interacting with the prototype

5 CONCLUSIONS

The logic of reactive IOT applications have to manage diverse data sources which can turn out to be a major issue in their development and maintenance. A unified knowledge base leveraging semantic technologies can mitigate this problem. We used this approach to propose an architecture in which also the behavior of the application is encoded in the knowledge base so that defining a new behavior or changing existing ones can be accomplished by knowledge base manipulations. This approach can be adopted for reactive IOT applications of different scale: from the large one governing a smart city to the diminutive one composed of personal connected wearable devices. In this paper we propose an approach to make these systems cooperate while maintaining the advantage of having a single information repository by defining dynamic knowledge base overlays. These overlays provide a view over the elastic context that is created when mobile systems get in touch one with the other allowing them to interact and reciprocally influence their behavior.

We implemented a proof-of-concept prototype by making a virtualized, simulated environment interact with actual running systems. Our demonstration shows how the behavior of a smart building can be extended to take into account the preferences and the state of its users without having to explicitly interact with them and relaying on a knowledge base overlay to retrieve the information pertaining its elastic context.

We are currently working to implement the "mobile side" of the prototype so that a running system can be deployed in a physical environment. We are also investigating other use cases in different domains.

REFERENCES

[1] Chen, H. et al. 2004. SOUPA: standard ontology for ubiquitous and pervasive applications. *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 2004)*, 258–267.

[2] Cheng, BettyH.C. et al. 2009. Software Engineering for Self-Adaptive Systems: A Research Roadmap. *Software Engineering for Self-Adaptive Systems*. BettyH.C. Cheng et al., eds. Springer Berlin Heidelberg. 1–26.

[3] Furno, A. and Zimeo, E. 2014. Context-aware Composition of Semantic Web Services. *Mobile Networks and Applications*. 19, 2 (Apr. 2014), 235–248. DOI:https://doi.org/10.1007/s11036-014-0494-y.

[4] Giallonardo, E. et al. 2019. Context-aware reactive systems based on runtime semantic models. *The 31st international conference on software engineering and knowledge engineering, SEKE 2019, hotel tivoli, lisbon, portugal, july 10-12, 2019* (2019), 301–403.

[5] Giallonardo, E. et al. 2019. Resilient Reactive Systems Based on Runtime Semantic Models. *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (Oct. 2019), 177–184.

[6] Giallonardo, E. et al. 2020. Semantics-driven reactive systems. *International Journal of Software Engineering and Knowledge Engineering*. to appear, (2020).

[7] Haller, A. et al. 2017. Semantic Sensor Network ontology. W3C recommendation. *World Wide Web Consortium*, (2017).

[8] Janowicz, K. et al. 2019. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*. 56, (May 2019), 1–10. DOI:https://doi.org/10.1016/j.websem.2018.06.003.

[9] Kaebisch, S. et al. 2017. Web of things (WoT) thing description. *First Public Working Draft, W3C*. (2017).

[10] de Lemos, R. et al. 2013. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. R. de Lemos et al., eds. Springer. 1–32.

[11] Perera, C. et al. 2014. Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys Tutorials*. 16, 1 (2014), 414–454. DOI:https://doi.org/10.1109/SURV.2013.042313.00197.

[12] Poggi, F. et al. 2019. Integrating semantic run-time models for adaptive software systems. *Journal of Web Engineering*. 18, 1 (2019), 1–42.

[13] Sagar, S. et al. 2018. Modeling smart sensors on top of SOSA/SSN and WoT TD with the semantic smart sensor network (S3N) modular ontology. *Emerging Topics in Semantic Technologies-ISWC 2018 Satellite Events*, 163–177.

[14] Szvetits, M. and Zdun, U. 2016. Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Software & Systems Modeling*. 15, 1 (2016), 31–69.

[15] Vogel, T. and Giese, H. 2014. Model-driven engineering of self-adaptive software with EUREMA. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*. 8, 4 (2014), 18.