A Development Methodology to Facilitate the Integration of Smart Spaces into the Web of Things

Iván Corredor, Josué Iglesias, Ana M. Bernardos, José R. Casar
Telecommunications School, Universidad Politécnica de Madrid, Spain
{ivan.corredor, josue,abernardos,jramon}@grpss.ssr.upm.es

Abstract—**How to create or integrate large Smart Spaces (considered as mash-ups of sensors and actuators) into the paradigm of "Web of Things" has been the motivation of many recent works. A cutting-edge approach deals with developing and deploying web-enabled embedded devices with two major objectives: 1) to integrate sensor and actuator technologies into everyday objects, and 2) to allow a diversity of devices to plug to Internet. Currently, developers who want to use this Internet-oriented approach need have solid understanding about sensorial platforms and semantic technologies. In this paper we propose a Resource-Oriented and Ontology-Driven Development (ROOD) methodology, based on Model Driven Architecture (MDA), to facilitate to any developer the development and deployment of Smart Spaces. Early evaluations of the ROOD methodology have been successfully accomplished through a partial deployment of a Smart Hotel.**

***Keywords*-Smart Space, Web of Things, Model Driven Architecture, Development Methodology, Resource-Oriented Architecture**

I. Introduction

The emerging notions of Internet of Things (IoT) [1] and Web of Things (WoT) [2] are coming up with Weiser's premonitory theories [3] about ubiquitous and calm computing. Those early theories envisioned *Smart Spaces* as physical spaces digitally augmented by heterogeneous *smart objects* (bundles of *sensors, actuators, displays* and *computational components*). In particular, in this paper, we refer to Smart Object as '*a computationally augmented tangible object'*, which is 'aware' of its own situation and provide services according to its own context '*without compromising its original appearance and interaction metaphor'* [4]. The IoT paradigm suggests an IP-based internetworking schema in order to get resource-constrained devices ready to be plugged into Internet according to the WoT concept. Nowadays, both IoT and WoT are reaching an acceptable level of maturity allowing isolated sensor and actuator devices to be part of a connected network of heterogeneous Smart Objects. Therefore, in a short period, Smart Spaces made up of hundreds, even thousands, of networked daily Smart Objects, will become usual.

Nowadays, any developer who wants to tackle with a Smart Space has to have strong understanding on general and specific technology aspects as communication protocols (e.g. IEEE 802.15.4, 6LoWPAN, uIP…), platforms of wireless sensor and actuator networks (WSANs) or service discovery or security procedures, among others. Some research works have risen from the Model Driven Engineering (MDE) principles [5-7] with the aim of facilitating the management and organization of those knowledge areas, to enable rapid prototyping of large business systems. The IoT and WoT paradigms fit well with MDE-based approaches as this model:

I) Allow modeling every part of the system and real entities through high-level models independently of the underlying heterogeneous technologies.
II) Decouple consumers and providers of contextual resources (*sensor, actuators* and *logic processes*), enabling a real reuse of components.
III) Provide development tools to facilitate rapid prototyping of complex deployments even for non-expert users.

In spite of above mentioned characteristics, it is still difficult to get a common abstraction modeling which covers a wide range of scenarios involving a number of different sensors, actuators and logic processes. In order to solve this gap, this paper presents the Resource-Oriented and Ontology-Driven Development (ROOD) methodology. The ROOD methodology enables traditional MDE tools to rapid prototyping of Smart Spaces according to the WoT paradigm. The cornerstone of our proposal is the Smart Space Modeling Language (SSML), which allows modeling singularities of Smart Spaces, i.e. *resources*, *services* and *platforms*. The resource concept described by SSML is inspired in Fielding's seminar thesis [8]. The ROOD methodology manages two models that are defined from the SSML: the *Smart Object Model* (SOM) and the *Environment Context Model* (ECM). The ECM is focused on describing high level behaviors and context information of the entire Smart Space; the SOM models artifacts for specific Smart Objects, i.e. resources, services and business processes.

The rest of the paper is structured as follows. Section II gathers relevant works on approaches for modeling services and resources in IoT and WoT. Section III describes an overview of the major features of the ROOD methodology. Section IV presents the ROOD methodology through a case study of a Smart Hotel. Section V describes first results from an early implementation of ROOD. Section VI concludes the paper with future work around our proposal.

II. Related Work

A. MDE-based approaches

In summary, the objective of Model Driven Engineering (MDE) [9] is to create model-based development environments from which code is automatically generated for a number of different platforms. Therefore, MDE allows developing software components without having previous knowledge about specific programming languages or pervasive platform technologies. The Model Driven Architecture (MDA), proposed by the Object Management Group (OMG), is a proposed fulfillment of MDE. OMG provides all necessary standardized tools to design Domain Specific Languages (DSL) by which entities, relations, and behaviors of a system can be defined.

Recently, the MDA's principles have been the basis for some proposals. A few of them are focused on improving MDA through semantic technologies, in order to augment its capacities with regard to the definition of systems for many different scenarios and platforms. One of the first contributions in this field was the W3C's proposal called the Semantic Web Best Practices and Development (SWBPD) whose main contribution is an *Ontology Driven Architecture* [7]. Several research works have been performed over that early idea. For instance, [10] presents a MDA-based methodology to reduce the burden when using ontologies for pervasive systems. Authors focus their research on a model transformation mechanism for the generation of context-aware applications by using a M2 layer metamodel called Upper-Level Context Ontology Model (ULCOM). In [5], it is proposed a basic methodology, which is based on a synergy between Ontology-driven and Model-driven approaches. Conceptual models are used both at runtime and development for reasoning and code generation, respectively. In [6], an extension of MDA is performed. This research proposes an Ontology Driven Software Engineering (ODSE) which uses three groups of ontologies: *Domain ontology*, *Task ontology* and *Ontology of software*.

B. *Resources-Oriented Approaches*

Many research works are the result of the growing interest in modeling REST-based resources [8] for lightweight Resource Oriented Architectures (ROA). Some of those works are based on toolkits or frameworks that facilitate the development and deployment of RESTful applications. AutoWoT [11] provides a rapid integration of smart devices into the Web. It automatically generates both applications and server software components, enabling developers to focus on the use case. The research in [12] gathers a resource semantic model that describes sensors, actuators, and processing resources. Furthermore, it offers a framework based on that model for supporting queries and performing requests to actuators. On the other hand, [13] proposes a metadata framework inspired by EPCglobal network [14] to enable *plug and play* WSAN in Internet. The metadata managed by this framework allows discovering nodes and provides a list of available interfaces for query/actuating services as well as their application level message formats.

The ROOD methodology contributes to the above-mentioned research fields in several aspects. Firstly, it extends and improves traditional MDE-based approaches by using semantic technologies that validate models generated in each stage of the ROOD methodology. The major contribution of our proposal with regard to previous MDE-based approaches ([4-6] and [10]) is its capacity to model concepts and generate artifacts that perfectly adapt to REST architectural styles what leads to a convergence between embedded pervasive networks and the WoT. Furthermore, the ROOD methodology can be used for building friendly development environments based on graphical modeling to develop and deploy large Smart Spaces rapidly.

III. The Resource-Oriented and Ontology-Driven Development Methodology: An overview

This section presents the Resource-Oriented and Ontology-Driven Development (ROOD) methodology for rapid prototyping of Smart Spaces and its integration into WoT. Firstly, we show the viewpoint of the ROOD's metamodel (SSML) as well its models (ECM and SOM) according to the MDA. Secondly, we describe the different stages of the ROOD methodology and the role of each ontology.

A. *MDA's perspective of SSML, ECM and SOM*

OMG defines an architecture stratified in four abstraction levels (M0 through M3). M0 contains instances of data for a specific platform; M1 is where system's models are defined; M2 specifies the DSLs that take part in the definition of models at M1. Finally, M3 defines the Meta-Object Facility (MOF), which establishes the basis for different modeling languages.

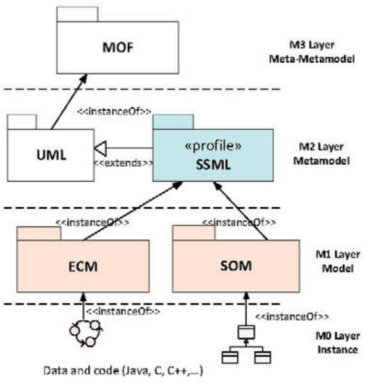


Fig. 1 MDA's perspective of ECM and SOML

Fig. 1 shows the location of SSML, ECM and SOM around MDA architecture. It is not the aim of this work to go over the specification of the SSML. Shortly, the SSML is hosted in M2 layer and extends the UML meta-model to create an own profile that defines every necessary element (entities, relations, interfaces, etc.). The SSML is a key piece within the ROOD's kernel. The ECM is an instance of SSML for modeling the behavior of a Smart Space in terms of *activities*, *transitions* between activities and *events* triggering transitions. The SOM is also an instance of SSML that allows modeling "things" or Smart Objects taking part in the Smart Space.

In summary, ECM and SOM are provided as modeling tools for Smart Spaces. Instances of those models are represented in the form of diagrams. The drawing schema of those diagrams is validated through ontologies, which determine what it can be represented according the participants (e.g. sensor/actuator devices and users) and resources (e.g. context information or functionalities offered by devices) that are available at a specific Smart Space.

B. *Ontologies for validation and transformation*

The strength of ROOD methodology is achieved by means of several validations and transformations whose final results are well-formed ECM and SOM models, as well as optimized code for specific platforms. Ontologies in ROOD represent knowledge about specific characteristic of the Smart Space to be developed. These ontologies are the following:

1) Domain ontology: defines the infrastructure entities and its relationships (e.g. spaces, human interfaces or devices).

II) Context ontology: defines the Smart Space context as it is intended in ROOD methodology. Basically, this ontology defines aspects related to "user" and "environment" (e.g. location, ambient conditions, state, etc.), the relationship between context entities, and axioms to reason on context information.

III) Service ontology: defines concepts such as tasks, business process and services as well its inputs and outputs.

IV) Resource ontology: define every entity necessary to map traditional services into RESTful style resources.

V) Platform ontology: defines relationships between entities represented in SOM and concrete software components for a specific platform.

Ontologies instances are stored in different Knowledge Bases (KB) as RDF documents. The structure of the ontologies is designed to support deductive inference for transforming and validating models along ROOD's stages using some ontology alignment techniques. The validation is performed both semantically and syntactically. Regarding the alignment, it is carried out between concepts in top-level ontologies to specific concepts in low-level, e.g. *temperature measure* in Context ontology is related to *environmental measurement service* in Service ontology.

C. *Phases of the development methodology*

The ontologies mentioned in the previous section are necessary to represent context consistently with the Smart Spaces we want to deploy. ROOD methodology offers ECM and SOM models to take advantage from context information along the phases defined in the traditional MDA methodology.

MDA development methodology consists of three main phases, which separate specific business logic aspects from platform features: *I)* Computation Independent Model (CIM); *II)* Platform Independent Model (PIM); *III)* Platform Specific Model (PSM). Particular transformations are performed from one phase to another, finally obtaining a platform-specific program code. This chain of model definitions and transformations has been adopted in the ROOD methodology. Moreover, this approach has been improved and consolidated by means of a mechanism of validations, which ensure well-formed models for a specific Smart Space.

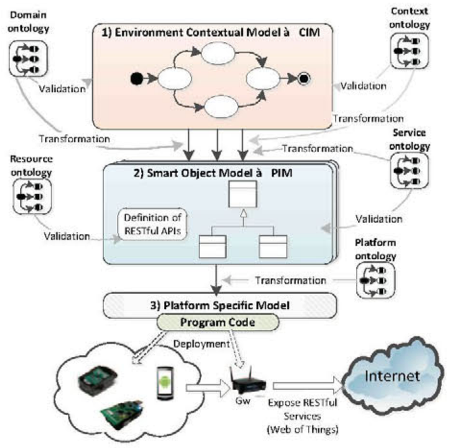


Fig. 2 The Resource-Oriented and Ontology-Driven Methodology

Fig. 2 shows an overview of the stages along the ROOD methodology and the involved ontologies in the previous section. Instances of those ontologies are used in *top-down* transformations in order to map ECM's concepts into specific SOM's entities, to finally achieve the automatic code generation for real platforms, through the needed Platform Specific Models (PSM). The main stages of the ROOD methodology are following described:

1) Modeling high-level viewpoint: the Environment Context Model

Before starting the ECM modeling process, both *Context ontology* and *Domain ontology* have to be instantiated: the information available to model the ECM (majorly *states, context information*, *events*, *actions* and *transitions*) will be validated according particular instances of Context and Domain ontologies. At this model, high level concepts are modeled, for instance control of environment parameters (e.g., light, humidity, etc.) or human-computer interfaces (e.g., displays or audio devices).

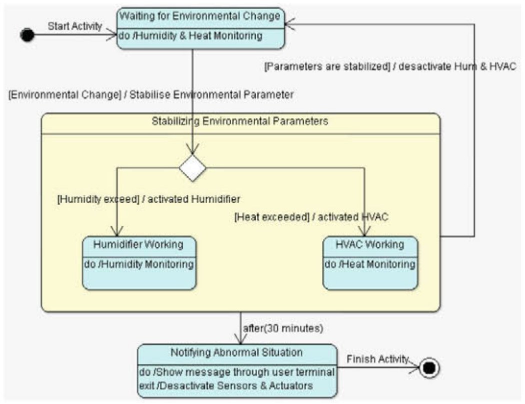

Start Activity
Waiting for Environmental Change
do /Humidity & Heat Monitoring
[Parameters are stabilized] / desactivate Hum & HVAC
[Environmental Change] / Stabilise Environmental Parameter
Stabilizing Environmental Parameters
[Humidity exceed] / activated Humidifier
[Heat exceeded] / activated HVAC
Humidifier Working
do /Humidity Monitoring
HVAC Working
do /Heat Monitoring
after(30 minutes)
Notifying Abnormal Situation
do /Show message through user terminal
exit /Desactivate Sensors & Actuators
Finish Activity

Fig. 3 An instance of the Environment Context Model

Fig. 3 shows an example of the ECM for modeling a Smart Comfort System that is able to control humidity and heat levels of a room. The ECM will provide one or more SOM models, which will model one or more Smart Objects involving actions and transitions within the modeled Smart Space at this stage.

2) Modeling Smart Things: the Smart Object Model

At the SOM stage, specific services and functionalities are modeled and mapped over each "thing", integrating the workflow that is defined at the ECM. At this point, Smart Object activity is modeled through the following elements: *I)* low level processes (Tasks); *II)* workflows of tasks (Business Process); *III)* high level interfaces for Business Processes (Services). This stage involves three ontologies for transforming from ECM to SOM: Domain, Context and Service ontologies. Moreover, Service and Resource ontologies are used for validating models that are generated at this stage.

Fig. 4 shows the general structure of the SOM. The Smart Object class, which is stereotyped as `<<Thing>>`, represents participant entities belonging to a Smart Space; they consume and/or provide resources within a specific domain following REST principles. Smart Objects can be associated in order to create a Composite Smart Object (stereotyped as

<<ObjectArchitecture>>). This collaboration allows providing complex services by means of aggregation of atomic services.

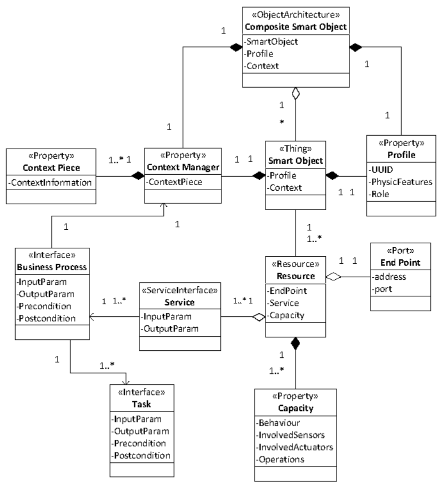


Fig. 4 Smart Object Model

The context of Smart Objects and Composite Smart Objects are characterized by Context Pieces (stereotyped as <<Property>>). Those context pieces are managed by the Context Manager. The Context Manager is intended for implementing a set of inferring rules that analyze context pieces and get the Smart Object ready to show different behaviors according to its contextual situation.

The Service class that is stereotyped as <<ServiceInterface>> defines specific inputs and outputs for a service primitive provided by a Smart Object. Those interfaces are then encapsulated into *resources*, which are modeled by means of classes stereotyped as <<Resource>>. This kind of entities characterizes every Smart Object's service as resources following a RESTful architectural style. This modeling process is carried out by mapping internal service interfaces into RESTful APIs; HTTP methods (GET, PUT, POST or DELETE) are defined to create or remove resources as well to query or modify its state. Additionally, classes stereotyped as <<Port>> define an end-point consist of a URI and a port. Thus, software artifacts that are generated from classes stereotyped as <<Resource>> enable an access point which allows integrating Smart Objects into the WoT following a RESTful architectural style.

3) Generating program code from abstractions: the Platform-Specific Model

The final stage of ROOD methodology is focused on creating a Platform Specific Model (PSM) and generating its corresponding program code. During this stage *business process* are mapped over *tasks*; those are finally executed by agents, i.e. atomic software pieces that manage business logic, being able to perform a task by themselves. This mapping process is carried out from *templates*, which are stored in software repositories. This matching mechanism generates program code accordingly to software pieces that can be found in the repository. Thus, the more software pieces are stored in the software repository, the more automatic code can be generated code automatically for a specific platform. If no matched components are found, then a stub is provided for the developer to complete it.

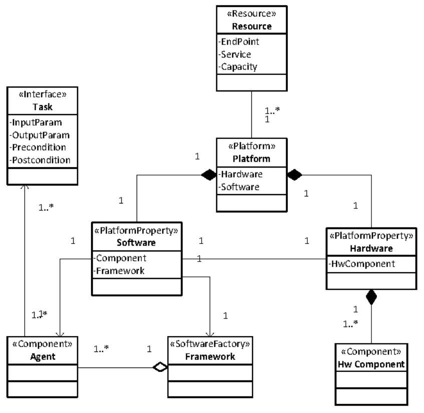


Fig. 5 Platform Specific Model

Fig. 5 shows the PSM used in ROOD methodology. A Platform (stereotyped as <<Platform>>) can host one or more resources, which are characterized by specific software and hardware (both stereotyped as <<PlatformProperty>>). Resources are related to tasks at low-level and used by agents (stereotyped as <<Component>>). The agent's life-cycle is managed by a Framework (stereotyped as <<SoftwareFactory>>).

IV. PROTOTYPING A SMART HOTEL: A CASE STUDY

Let us consider a real prototyping of a Smart Hotel to exemplify the use of ROOD methodology. This Smart Hotel involves a cluster of different Smart Objects providing and consuming specific RESTful resources and managing its own contextual information. Smart Objects have two major roles within the Smart Hotel:

I) *Ambient quality control*: This kind of Smart Objects deals with some environmental parameters (humidity, temperature and light) in order to generate a comfortable environment according to user's preferences.

II) *Simplified retrieval of tourism information*: The Smart Hotel is decorated with some RFID-tagged souvenirs, which enable direct retrieval of tourism information. Visitors are assumed to be carrying a device (e.g. tablet PC or PDA), which enables accessing objects' information through different visualization interfaces (e.g. augmented reality).

III) *Healthcare monitoring*: This kind of Smart Objects is responsible of monitoring vital signs of customers with special needs, through wearable personal biomedical sensors (e.g. hearth monitor belt, body humidity sensor or body temperature sensor).

According to ROOD methodology, the whole system will be modeled, including the involved physical elements and business processes in order to generate as much code as possible.

The first stage consists of managing high-level concepts through the Environment Context Model (ECM). For instance, Fig. 6 depicts the ECM for the tourism information service. This stage is driven by two ontologies: the Domain ontology and the Context ontology.

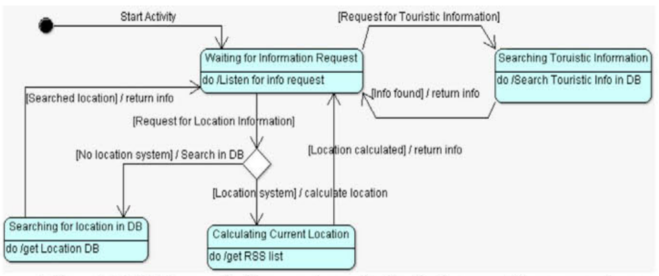
The Domain ontology defines concepts with regard to:

- *Structural elements*: e.g. walls, doors, monuments, etc.
- *Sensorial and actuator devices*: e.g. motes, analog transducers, switches or biometric sensor.
- *User terminals*: e.g. tablet PCs, mobiles and other information devices.

The Context ontology defines concepts as follows:

- *User preferences*: e.g. comfort temperature, light level or touristic points of interest.
- *Environmental parameters*: e.g. temperature, humidity, gases, light, cloudy, sunny, etc.
- *Device states*: e.g. current visual information, actuator states or battery level.
- *Axioms and rules*: e.g. *It is snowing* then *it is not recommended to leave from the Hotel*.

Instances of previous ontologies enable aligning concepts from both ontologies (e.g. monument to touristic points of interest). This alignment technique allows validating concepts represented in ECMs, i.e. states, context information, events, actions and transitions.



Start Activity
[Request for Touristic Information]
Waiting for Information Request
do /Listen for info request
Searching Toruistic Information
do /Search Touristic Info in DB
[Searched location] / return info
[Info found] / return info
[Request for Location Information]
[No location system] / Search in DB
[Location calculated] / return info
[Location system] / calculate location
Searching for location in DB
do /get Location DB
Calculating Current Location
do /get RSS list

Fig. 6 ECM modeling a touristic information service

Second stage begins when valid ECMs are generated; at this stage, ECM models are transformed into one or more Smart Object Models (SOMs). These initial SOMs are *stubs*, which have to be manually completed by the developer who has to decide how to map services into *resources* following a REST architectural style. The aforementioned process consists of encapsulating service primitives into HTTP methods (`GET`, `PUT`, `POST` and `DELETE`) as well as the definition of a URI schema for uniquely indentifying every resource hosted in each Smart Object.

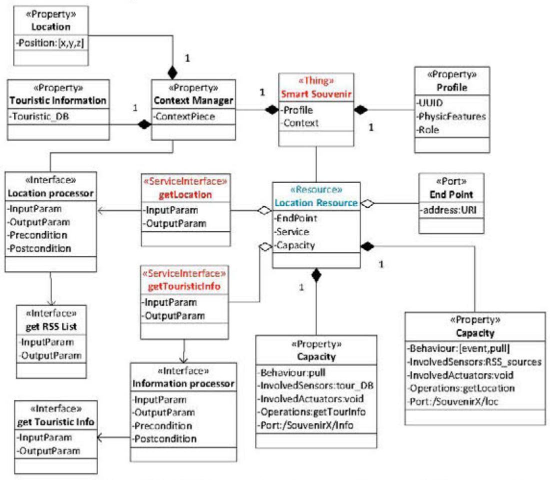


Fig. 7 SOM modeling a Smart Souvenir

Fig. 7 shows an example of well-formed SOM model, corresponding to a *virtual* Smart Souvenir tagged with a RFID tag. In this case, the service primitive `getTourictic Information()` is offered via a `GET` method that is invoked to the URI `http://SmartHotel.com/SouvenirX/description`. That invocation retrieves the touristic information in JSON format related to the tagged souvenir from a RESTful database.

The validation of a SOM is performed through both Resources ontology and Service ontology.

The Service ontology defines aspects for specifying traditional services-oriented architectures. These characteristics are the following:

- *Service methods*: procedures that are executed by specific business logic.
- *Data types for inputs and outputs*: data types that are managed by every service.
- *Preconditions and postconditions*: necessary conditions to complete a service execution cycle correctly.
- *Participants*: one for atomic services, and two or more for composite service.

The Resources ontology defines a reduced set of features for characterizing a RESTful API for functionalities hosted by Smart Objects. Those aspects are the following:

- *URI:* identifies uniquely a resource.
- *State*: indicates the current state of the resource: available, unavailable or temporally suspended.
- *Operations*: defines allowed operations (HTTP methods) for this resource (XML, HTML or JSON).
- *Output* representation: indicates the format of the returned data.
- *Sub* Resources: URLs of sub-resources depending on this resource.

Instances of the above mentioned ontologies are necessary to model all those singularities of resource-oriented architectures by wrapping service primitives characterizing traditional service-oriented approaches.

After having finished the modeling of every Smart Object in the Smart Hotel, ROOD methodology provides Platform Specific Models (PSMs) through instances of the Platform ontology. These PSMs allow generating code for different embedded platforms, e.g. motes (MicaZ[1], Sun SPOT[2], Shimmer[3]), Android-based platforms (android ADK[4], tablet PC) or IP cameras (Camera Axis[5]). Moreover, it is generated a set of specific components for managing every Smart Object through distributed Gateways. In summary, we finally have web resources according to the features modeled in SOMs.

V. First Evaluations of ROOD Methodology

The case study presented in previous section has been used to carry out a preliminary evaluation of the ROOD methodology. Currently, we have partially implemented our

[1] MicaZ: http://www.snm.ethz.ch/Projects/MicaZ

[2] Sun SPOT: http://www.sunspotworld.com/

[3] Shimmer: http://www.shimmer-research.com/

[4] Android ADK: http://www.cooking-hacks.com/index.php/arduino-mega-adk.html

[5] Camera Axis: http://www.axis.com/

proposal using the Obeo Designer[6], an Eclipse plugin that is based on the Eclipse Modeling Project[7]. A *ROOD visual editor* for rapidly creating Smart Object Models has been firstly implemented. In this version, the first ROOD's stage, involving the creation and validation of ECM and its transformation into SOM, is manually performed. For next versions, we foresee to extend our visual editor in order to automatically carry out such functionalities. By using our visual editor, it is expected to cover a wide range of devices to be deployed on our Smart Hotel. Shimmer nodes managing Bluetooth biometric sensors (hearth rate, accelerometer or body temperature) or Android ADK devices (e.g. offering visual feedback through its LEDs) may be modeled in order to create mashups of Smart Objects. Currently, ROOD visual editor only support mote platforms, i.e. MicaZ and Sun SPOT in our case study.

Apart from code generation for every device characterizing Smart Objects, our editor generates code for a web server based on *Restlet*[8], a flexible and lightweight REST framework. This web server is deployed on Gateways (usually a PC, Tablet PC or Android-based mobile), which connects Smart Objects to the Web of Things. It works as an interface exposing device functions in a RESTful way. Using *ROOD visual editor* we estimate that the process of developing and deploying Smart Objects for the Smart Hotel can be significantly reduced.

VI. Conclusions and Future Work

In this paper we propose a Resource-Oriented and Ontology-Driven (ROOD) methodology based on the OMG's MDA to develop and deploy pervasive applications for Smart Spaces. We have dealt with some semantic technologies to improve the MDA in order to provide friendly development tools allowing no expert developers to build complex Smart Spaces.

ROOD consists of three development stages similar to those of MDA. Thus the development of a smart environment is firstly tackled from a very abstract viewpoint (*Environment Context Model*) in which a flow of contextual data is modeled; secondly, from a more refined viewpoint (*Smart Object Model*), where roles and functionalities within Smart Spaces (*tasks*, *business process*, *service* and *resources*) are distributed among Smart Objects. Finally, a Platform Specific Model (PSM) is obtained - it is key to get program code ready to be deployed on physical devices.

We have presented a study case to illustrate the ROOD methodology through a Smart Hotel composed of multiple Smart Objects driven by specific physical devices. We have performed an early evaluation of the ROOD methodology through a visual editor based on Obeo Designer, whose first version partially implements the ROOD methodology. To this aim, we have partially prototyped a Smart Hotel, specifically the part of the system dedicated to environmental parameters management. This is a first step to test the potential of the ROOD methodology's concept to effectively facilitate the whole prototyping of Smart Spaces and their integration into the Web of Things.

Our future work is focused on researching an improvement of the expressiveness of the ROOD methodology by increasing the use of semantic technologies, as new ontology evolution and alignment techniques, to improve the integration of Smart Objects into virtual mashups.

Acknowledgment

This work has been supported by the Spanish Ministry of Industry, Tourism and Commerce and the European Fund for Regional Development under grant TIN2011-28620-C02-02, the Ministry for Science and Innovation under grant TIN2008-06742-C02-01 and the Government of Madrid under grant S2009/TIC-1485 (CONTEXTS).

References

[1] International Telecommunication Union, "ITU Internet reports 2005: The internet of things," 2005.

[2] S. Duquennoy, G. Grimaud and J. Vandewalle, "The web of things: Interconnecting devices with high usability and performance," in *International Conference on Embedded Software and Systems, ICESS '09*, pp. 323-330, 2009.

[3] M. Weiser, "The computer for the 21st Century," *Pervasive Computing, IEEE*, vol. 99, pp. 19-25, 2002.

[4] F. Kawsar, K. Fujinami and T. Nakajima, "Prottoy Middleware Platform for Smart Object Systems", *International Journal of Smart Home*, vol. 2, pp. 1-18, 2008.

[5] A. Soylu and P. De Causmaecker, "Merging model driven and ontology driven system development approaches pervasive computing perspective," in *24th International Symposium on Computer and Information Sciences, ISCIS*, pp. 730-735, 2009.

[6] A. Katasonov and M. Palviainen, "Towards ontology-driven development of applications for smart environments," in *8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 696-701, 2010.

[7] P. Tetlow, J. Pan, D. Oberle, E. Wallace, M. Uschold and E. Kendall, "Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering," vol. 2011, 2006.

[8] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000.

[9] T. Gherbi, D. Meslati and I. Borne, "MDE between promises and challenges," in *11th International Conference on Computer Modelling and Simulation*, pp. 152-155, 2009.

[10] N. Georgalas, S. Ou, M. Azmoodeh and K. Yang, "Towards a model-driven approach for ontology-based context-aware application development: A case study," in *Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, pp. 21-32, 2007.

[11] M. Simon, G. Dominique and T. Vlad, "Facilitating the integration and interaction for real-world services for the web of Things," in *Urban Internet of Things – Towards Programmable Real-Time Cities (UrbanIOT 2010); Workshop at the Internet of Things 2010 Conference (IoT 2010)*, Tokyo, 2010.

[12] C. Villalonga, M. Bauer, F. López Aguilar, V. Huang and M. Strohbach, "A resource model for the real world internet," in *Smart Sensing and Context*, P. Lukowicz, K. Kunze and G. Kortuem, Eds. Springer Berlin / Heidelberg, pp. 163-176, 2010.

[13] J. Sung, Y. Kim, T. Kim, Y. Kim and D. Kim, "Internet metadata framework for plug and play wireless sensor networks," in *Sensors Applications Symposium, IEEE*, pp. 320-324, 2009.

[14] GS1, "EPCglobal network," 2011. [Online] Available: http://www.gs1.org/epcglobal.