Towards a lightweight mobile semantic-based approach for enhancing interaction with smart objects

Josué Iglesias, Ana M. Bernardos, Luca Bergesio, Jesús Cano, José R. Casar

Telecommunications Engineering School, Technical University of Madrid (UPM), Spain

{josue, abernardos, luca.bergesio, jcano, jramon}@grpss.ssr.upm.es

Abstract This work describes a semantic extension for a user-smart object interaction model based on the ECA paradigm (Event-Condition-Action). In this approach, smart objects publish their sensing (event) and action capabilities in the cloud and mobile devices are prepared to retrieve them and act as mediators to configure personalized behaviours for the objects. In this paper, the information handled by this interaction system has been shaped according several semantic models that, together with the integration of an embedded ontological and rule-based reasoner, are exploited in order to *(i)* automatically detect incompatible ECA rules configurations and to *(ii)* support complex ECA rules definitions and execution. This semantic extension may significantly improve the management of smart spaces populated with numerous smart objects from mobile personal devices, as it facilitates the configuration of coherent ECA rules.

Keywords Smart objects, smart spaces, user-object interaction, mobile middleware, embedded reasoning, ontology-based modelling.

1 Introduction

The concept of *smart object* includes any kind of device with sensing or/and processing capabilities that is capable of reacting or adapting its functionalities depending on external stimulus or users' requirements, while preserving their traditional physical interaction paradigm. Then, s*paces* become *smart* as they host several smart objects with heterogeneous functionalities. Within this scenario, smart spaces exploitation not only involves acquiring data and controlling smart objects, but a common strategy to enable the user with capabilities to coordinate them in an intelligent way. To this end, the ECA paradigm appears as a simple formalism used to implement a particular perspective for user-smart object intelligent interaction.

The ECA paradigm is composed by a structure of reactive rules working over an event-driven architecture. Each ECA rule may have three kinds of 'atoms': the *event* is the signal that triggers a set of rules; the *condition* is a logical test that, if satisfied, makes the execution of the rule to continue; and, finally, the *action* identifies the execution of a process. A set of rules has the form: **ON** *event* **IF** *condition(s)* **DO** *action(s)*.

In [1] we have developed a first prototype to validate this interaction paradigm, which uses the mobile device as mediator to handle ECA rules built on the sensing and action capabilities of smart objects (details in Section 3). It happens that, the more the smart space ecosystem grows (including new smart objects), it becomes exponentially difficult to manage from a user's perspective; at the same time, the continuous rule checking process also becomes very demanding in terms of mobile processing. So, in this work we propose to extend the interaction model's capabilities from a semantic perspective, in order to enhance *(i)* user experience and *(ii)* mobile device performance.

The paper is structured as follows. Section 2 reviews the state of the art of embedded lightweight semantic tools and semantic-based ECA model approaches. Section 3 introduces the interaction scenario and the original architecture that has been extended. Section 4 presents the proposed models and addresses the enhancements obtained. Finally, Section 5 concludes the work with future lines for research.

2 State of the art

Since 1999, when works such as the one carried out by Biegl [2] first addressed how to link and control different kinds of devices, interaction paradigms for physical interaction between devices (or digitally augmented objects) have been increasingly used to implement the smart space concept.

Trying to fill the gap between existing developments for smart spaces configuration and state-of-the-art initiatives for enhancing these paradigms (and always from a ECA model based point of view), this Section focuses on analysing *(i)* the enabling technologies for embedded context information representation and reasoning and *(ii)* some of the most up-to-date projects for semantic enhancement of ECA-based interaction models.

From a functional point of view, machine processable representations are necessary to organize, valorise and share the vast amount of information smart spaces generate (e.g., environmental conditions, users presence, available services, objects usage patterns, etc.). Within this scenario, information acquired from heterogeneous smart objects would need to be jointly processed, requiring a common and expressive enough data structure to support the smart space configuration. Personal smartphones are nowadays-common devices in people's everyday lives, becoming a potential candidate for centralizing smart spaces exploitation. Howev-

er, it is still an open challenge how to optimally represent information in resource-constrained mobile devices [3].

For example, although tuple-based (or key-value) models reduce management overhead and can be easily applied to 'legacy' mobile systems (e.g., [4]), they lack from validation and scalability capabilities and they are not suitable for handling context information ambiguity. A hierarchical structure and the automatic validation are, however, some of the strong points of markup scheme modelling, although XML has a high semantic redundancy and it is not fully adapted to the limited resources of embedded devices (several works points out that a better performance can be obtained with other techniques as, e.g., JSON[1]).

Ontology-based modelling [5] combines the advantages of object-based and logic models, i.e., encapsulation, extendibility and reusability, and formalism and inference capabilities, respectively. They facilitate information fusion from heterogeneous data and knowledge sources, also providing support for automated reasoning. OWL (Web Ontology Language)[2], the standard ontology language endorsed by the W3C, enables different applications to share a common model, providing common shared domain vocabularies and a consistent mechanism for information representation. According to [6], these features are particularly important in mobile and pervasive environments, in which different heterogeneous and distributed entities must interact for exchanging users' context information.

Formal information representation facilitates automated reasoning (e.g., concept and instance classification, model and knowledge base consistency checking, etc. [30]). Although scarce compared with general context management systems, some light tools enabling reasoning in resource-constrained devices have already been described in the literature. Following the conclusions presented in [3] (whose work is focused on lightweight ontology-based data models), *μJena* [7] and *Bossam* [8] (as ontology manager and ontology rule-based reasoner, respectively) were the only ones *(i)* capable of dealing with ontology data, *(ii)* using standard formats and *(iii)* working in resource-constrained mobile devices. Extending this work, it is worth mentioning *androJena*, a new development (the first version was released on May 2010) based on a subset of the popular Jena framework migrated to Android platforms, that also fulfils these requirements. *androJena* has been recently used in several works; [9] and [10] can be highlighted as they include performance tests. Although these tests only measure their own particular developments, they can be used to obtain an overall idea of *androJena*'s performance.

The state-of-the-art analysis reveals that there are still few developments of general-purpose lightweight ontological tools to be embedded in personal mobile devices with a promising success, and the existing ones are still far away from maturity. As previously said, although some performance tests can be found in the literature (e.g., for *μJena* [7], *Bossam* [8] or, more recently, *androjena* [9][10]), there is still a lack of experiments comparing their performances in common sce-

[1] http://www.json.org/

[2] http://www.w3.org/TR/owl-features/

narios. Finally, it should be noted that much of these developments are discontinuous research projects. In this sense, *androJena* seems to be the only exception to this issue nowadays.

Focusing now on semantic technologies, they may play an important role in enhancing ECA-based interactions for smart spaces. Ontologies can be used to formally model the information offered by the smart objects and their capabilities and particularities, information that can fed different semantic reasoning mechanisms in order to offer a consistent information layer ready to be exploited.

In this line, within the Rewerse project[3], r^3 prototype (Resourceful Reactive Rules) addresses ECA rules managing from a Semantic Web perspective. It implements a rule engine capable of dealing with this kind of rules defined in different languages (even each different component –event, condition or action– may be represented using different rule languages). Every resource involved in the reasoning process (e.g., rules, engines) is described in terms of RDF triplets based on an OWL-based model: the r^3 ontology [11].

The K^{4R} project[4], maybe the most notable extension of r^3 initiative, focuses on defining a RESTful (and ontological) interface for Knowledge Resource (including Knowledge Reasoners). Authors consider this approach broad enough to include ECA rules modelling support [12].

Finally, it is worth mentioning that RIF (Rule Interchange Format), a collection of rule dialects (i.e., consistent and rigorously defined rule languages) intended to facilitate rule sharing and exchange, also consider ECA rules support as a requirement to be developed [13].

Aligned with these researches, the work presented in this paper focuses on developing real world semantic-based mobile applications for managing smart spaces. Next Section introduces the scenario where our semantic-based ECA interaction model is to be deployed and its architectural particularities.

3 ECA interaction model for smart spaces management

3.1 ECA model based smart space management scenario

Our interaction scenario considers a space populated with different types of objects (with or without embedded processing capabilities). Within this scenario, the user's mobile device may be used to manage the identification, sensing, processing, etc. [14] of these smart objects, being able to act as an interaction mediator, delivering the functionality to *(i)* customize the responses to physical interac-

[3] http://rewerse.net/

[4] http://code.google.com/p/k4r/

tion with certain objects, *(ii)* make an object respond (physically or virtually) to a given order configured by the user, *(iii)* provide intuitive configuration of the smart environment through actions held in the mobile device and *(iv)* configure/activate features in the mobile device depending on environmental events.

The ECA interaction model aims at providing bidirectional interaction between objects and user's (mobile) devices, in order to make possible to configure the object's actions from the mobile device (note that within this general scenario, user's personal device can be also considered as a smart object). The ECA model is supported by implementing next functionalities:

1. *Module publishing*. Smart objects are able to publish their capabilities (both events generation or actions execution). Each 'module' implements the necessary logic in order to detect an event or perform an activity.
2. *Proximity detection*. Proximity will be the starting point for interaction. After proximity detection, a mobile device will be able to download the available modules belonging to the smart object it is close to.
3. *ECA rules configuration*. Mobile mash-up tools will allow the user to easily configure ECA rules as a combination of event, condition(s) and action(s).
4. *Rule-based reasoning*. Active ECA rules will be constantly evaluated in order to detect configured events, executing the associated actions if the conditions are fulfilled.
5. *Module life-cycle management*. Installed modules will be subject to continuous updates in order to detect unused or out-of-date ones.

So, when a smart object detects a mobile phone nearby, it offers to the phone the download of a set of modules that enables the interaction with the object. After agreeing to download the modules, the user can configure ECA rules with them as described before. When an event occurs in or is detected by a smart object, the corresponding module at the mobile phone receives a notification. The mobile application then checks the conditions in the ECA rules (if any) and, if satisfied, executes the configured actions, affecting other smart objects or the device itself.

3.2 Different architectural approaches to ECA-based deployments

Our particular approach employs the user's mobile device for interacting with the smart objects deployed in the smart spaces, i.e., for module installation and ECA rules configuration. However, ECA rules evaluation (event detection, condition assessment and action triggering) may be supported by different architectural approaches:

- *Server-based centralized approach*: once an ECA rule is configured in the user's mobile device this rule is sent to a centralized server managing the whole set of smart objects of a particular smart space. This server is the one in charge

of *(i)* monitoring all the smart objects (or automatically receiving state changes from them), *(ii)* detecting the configured events, *(iii)* assessing the required conditions and *(iv)* triggering the associated actions.

- *Smart object distributed approach*: in this case, each ECA rule would be distributed among the smart-objects involved in the rule. Smart objects would need to be intelligent enough to detect its own-generated events, to assess the required conditions and to execute the desired actions; they should also have communication capabilities (WiFi, Bluetooth, etc.) to automatically coordinate among themselves.
- *Mobile device approach*: this approach is equivalent to the server-based centralized one but, instead of having one server managing the ECA rules evaluation for each smart space, there would be one mobile device managing all the ECA rules configured by a particular user (regardless of the user location).

Although it has to be noted that these architectural approaches are compatible and they can coexists, this work addresses the real deployment of an architecture to be fully deployed in the mobile device.

3.3 Mobile-based architecture to enable ECA interaction model

Before addressing the semantic extension proposed in this work (Section 4), this subsection introduces the architecture of our (non-semantic) approach for smart spaces management based on an ECA interaction model [1].

For a practical implementation of the proposed scenario, smart objects are equipped with Bluetooth or NFC tags. When detecting an object for the first time (proximity interaction), the mobile device will retrieve the object's modules from a cloud server. The mobile application is the central element to manage interaction. It is divided into three main building blocks: *core*, *application interface* and *modules*. The core manages the modules' lifecycle: it dynamically retrieves them from the infrastructure and loads them into memory, manages the interaction between events, conditions and actions and, finally, provides the GUI. The application interface defines the data structure for the communication between the core and the modules. Finally, each ECA rule aggregates module events, conditions and actions to interact with a smart object or with the mobile itself. A module offering may include none, one or more than one of each (events, conditions and actions).

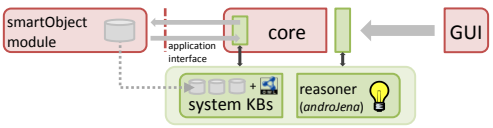

Fig. 1. ECA based architecture modules; non-semantic (red) and semantic extension (green).

This first prototype for the ECA model has been implemented on Android (v2.3, in a Google Nexus S smartphone which includes NFC technology).

Next Section shows how a lightweight ontology-based framework and rule engine (*androJena*) have been integrated inside the user's mobile device in order to exploit its semantic capabilities for *(i)* supporting the ECA rules configuration process and *(ii)* supporting the ECA rules evaluation (as depicted in Figure 1).

4 Semantic enhancing of the ECA-based architecture

This work extends with semantic capabilities the ECA model based architecture for smart spaces presented in Section 3. As depicted in Figure 2, three semantic models are used:

- *Smart object ontology* ($\mathbb{O}_{so}$): is used to model the characteristics of a smart object. This ontology models *(i)* the internal (sensing and acting) capabilities of a smart object (i.e., type of capability and valid values ranges, if applicable) and *(ii)* the relationships among objects capabilities (e.g, "fixed smart objects with proximity sensing capabilities can only detect mobile objects", etc.).
- *Smart space ontology* ($\mathbb{O}_{ss}$): models the relationships between a smart space and its smart objects. In this very first approach it can be considered just as a semantic map of the environment where the smart objects are deployed (e.g., "smartObjectX is currently located in roomA", etc.).
- *ECA rule ontology* ($\mathbb{O}_{ECA}$): is a formal definition of ECA rules. It can be used to *(i)* detect inconsistencies when configuring a particular ECA rule (i.e., among smart objects capabilities) and *(ii)* identify incompatibilities among the different sets of ECA rules configured for a particular user (future works will also consider restrictions among ECA rules defined by different users).

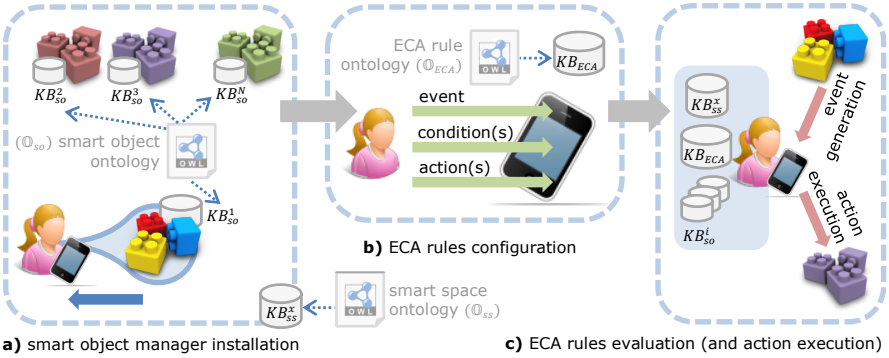

Fig. 2. ECA model based smart environment exploitation.

Each semantic model needs to be complemented with a rule base in order to address a comprehensive set of reasoning capabilities.

A preliminary version of these models (and rule bases) has been used in this work to *(i)* automatically detect incompatible ECA rules configurations (Section 4.1) and to *(ii)* support complex rules configuration and execution (Section 4.2).

Each smart object is defined by a smart object knowledge base KB_{so}^{i} shaped according a smart object ontology $\mathbb{O}_{so}$ (Figure 3.a). This ontology models a smart object (*SmartObject* class) according its particular features (*Capability*) and the set of available configurations (*CapabilityConfiguration*) modelling its valid states. Although specific capabilities are defined (e.g., *Email*, *SMS*, *Photo*, *PhoneCall*, etc.), they can be grouped in the more abstract concepts *SensingCapability* and *ActingCapability*.

$\mathbb{O}_{ss}$ models smart spaces (*SmartSpace*), Figure 3.b. A smart space is composed by several physical spaces (*Space*) hosting different smart objects. These spaces may have relative relations (*nearTo*, *contains*, *disjointSpaces*, etc.), represented in Figure 3's model as *hasSpatialRelationWith**.

ECA rules (*ECArule* class) are modelled in $\mathbb{O}_{ECA}$ ontology (Figure 3.c) as a set of atoms (*ECAruleAtom*): *ECAevent*, *ECAcondition* and *ECAaction*. Each of these atoms is associated to a smart object capability (*involvesCapability* property) and to a particular configuration for that smart object (*hasConfiguration*).

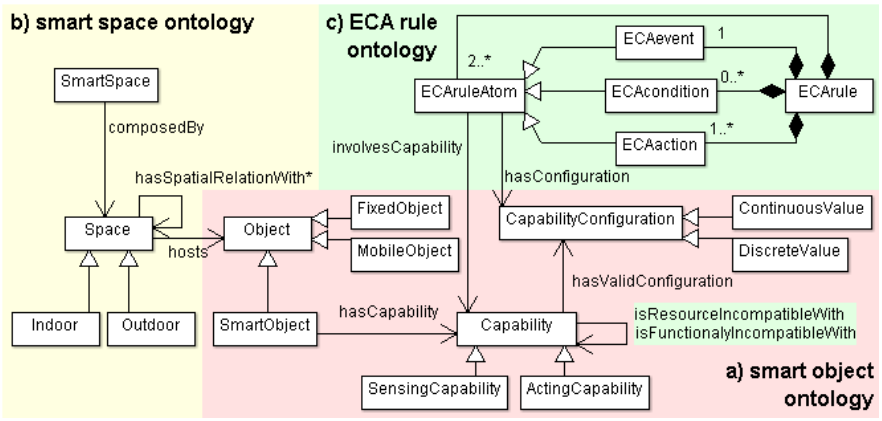

Fig. 3. ECA-based interaction conceptual model.

As stated before, this work follows a mobile centralized architecture for implementing an ECA model based smart space management system (see Section 3.2), so these semantic models are fully managed inside the user's mobile device (OWL-Lite has been used for ontology representation). Semantic models (and rule bases) management is performed using *androJena* as programmatic environment. The generic OWL-based rule engine offered by *androJena* has been also used to apply the rules bases to the semantic models defined.

4.1 Detecting semantic incompatibilities

Each ECA rule involves several options regarding the configuration of the available smart objects and each user may configure several ECA rules in his mobile device to personalize a particular smart space. This heterogeneity of resources available in the smart spaces (i.e., different smart objects with different capabilities) may lead to the configuration of inconsistent rules.

Semantic models described above may be exploited in order to automatically detect several kinds of inconsistencies. Although some of them may be directly detected from the formal definition of smart spaces and objects (e.g., when trying to configure certain value out of its valid range), a specific semantic has to be added to extend this for 'intra' and 'inter' ECA rules inconsistency detection (i.e., two configurations valid if isolated, may lead to an inconsistency if used inside the same ECA rule).

In the ECA rules configuration process, the information about the smart spaces modelled in this semantic way is used for:

- *ECA rules options filtering*: adapts the set of available options to be configured in the user's mobile device when creating ECA rules, just showing those smart objects whose controller module has been previously installed in the user's mobile device and adapting the configuration options of each smart object according to its features.

Additionally, four types of incompatibilities can be automatically detected using these models:

- *Exclusive resource incompatibility*: identifies those resources that cannot be employed at the same time in a ECA statement (e.g., a user cannot be located in disjoint places at the same time or s/he cannot both receive and make a call). The *isResourceIncompatibleWith* symmetric property in $\mathbb{O}_{ECA}$ ontology is used to identify these kinds of incompatibilities.
- *Smart object interaction incompatibility*: $\mathbb{O}_{so}$ ontology is used to identify inter-resource incompatibilities, as smart objects interact among them given certain restrictions. For instance, it would not be possible to set an event (or condition) like "*intelligentTray* detects *tvSet*" if the *tvSet* is defined as a fixed object and the *intelligentTray* is only able to detect mobile objects. $\mathbb{O}_{ECA}$ ontology adds the necessary semantic to extend this incompatibility detection at ECA rule level (e.g., "**ON** *userLocation=roomA* **IF** *userUses=carX* **DO** <*action*>" is an incompatible ECA statement in case *carX* cannot be inside *roomA* (this would be stated in $\mathbb{O}_{ss}$), although "*userLocation=roomA*" or "*userUses=carX*" are independently valid constructions).
- *Configuration incompatibility*: checks that each resource is configured according its valid set of values (e.g., "**ON** <*event*> **IF** *objectXtemperature<55º* **DO** <*action*>" would be a range-incompatible ECA statement if objectX temperature only range from -10º to +35º). $\mathbb{O}_{ECA}$ ontology adds the necessary semantic

to extend this incompatibility detection at ECA rule level (e.g., "**ON** *<event>* **IF** *temperature<20º* **AND** *temperature>30º* **DO** *<action>*" would be also a range-incompatible ECA statement).

- *Functional incoherence*: prevents inconsistent actions to be configured (e.g., the same set of events and conditions cannot trigger opposite actions: "**ON** *event1* **IF** *condition1* **DO** *turnRadioON*" rule is functionally incompatible with "**ON** *event1* **IF** *condition1* **DO** *turnRadioOFF*"). This reasoning employs the semantic encoded in $\mathbb{O}_{ECA}$ ontology (and its associated rule base), e.g., in the *isFunctionalyIncompatibleWith* property.

Finally, it has to be noted that, as $\mathbb{O}_{ECA}$ ontology is linked with the available capabilities of the smart objects which a user's mobile device is able to configure (i.e., with $\mathbb{O}_{so}$), this information can be used in order to load in memory a filtered version of the complete knowledge base, just containing the specifications of those smart objects participating in any of the active ECA rules.

4.2 Semantic rule-based reasoning support for ECA rules execution

The set of ECA rules configured by a user defines those events a mobile device should be aware of. Some smart objects (those more intelligent) may filter by themselves these events, only sending to the user's mobile device those events involved in any of the configured rules. On the contrary, in other cases, the user's mobile device has to be intelligent enough to detect those events itself. So, in the worst case, user's mobile devices should be intelligent enough to *(i)* detect only those significant events involved in any of the configured ECA rules and, afterwards, *(ii)* evaluate the set of conditions associated to a particular ECA rule, in order to decide whether to launch an action or not.

The semantic models presented in the previous Section may be also used to enhance these reasoning operations. Besides, having a general purpose rule engine eases the configuration of complex rules that would only need to be semantically defined but not programmatically implemented as in previous non-semantic versions of this ECA model based smart space managing system.

4.2.1 Context-based dynamic activation of ECA rules

Each ECA rule configured in the user's mobile device involves several resources (i.e., smart objects –including the mobile device–). Context related to these resources and the particular context of the user's mobile device may cause some rules to become obsolete (not applicable), or vice versa. It is possible to use the in-

formation about the smart objects encoded in the previously presented models in order to support the dynamic activation/deactivation of ECA rules depending on:

- *Smart objects status*: an ECA rule involving a particular smart object should be deactivated if the smart object is switched off, if it loses its communication capabilities, etc. (and vice versa).
- *User's mobile device communications status*: every ECA rule requiring the mobile device to access a remote object should be deactivated if the mobile device loses communication coverage (and vice versa).

This can be achieved because smart objects (and user's mobile device) status is stored in the $\mathbb{O}_{so}$ ontology and also because the $\mathbb{O}_{ECA}$ ontology offers a mapping between each ECA rule and the smart objects involved in the rule.

Future extensions of this work have to consider multiuser scenarios, where this automatic rules activation/deactivation process should be applied when detecting contradictory rules configured by different users.

4.2.2 ECA rules conditions execution order prioritization

ECA rules may involve evaluating 'online' conditions, i.e., those requiring to access an external resource. On the contrary, 'offline' conditions only need to access to parameters stored inside the user's mobile device. So it is quite common to configure ECA rules with both 'online' and 'offline' conditions.

Within this scenario, having information about the kind of conditions to be evaluated may be quite useful for saving mobile device resources: if the 'offline' conditions are first evaluated, once a necessary and sufficient condition is detected, no more conditions would need to be evaluated. This is shown in next simple example: "**ON** (<*offlineEvent1*> **OR** <*onlineEvent1*>) **IF** (<*offlineConditionA*> **AND** <*onlineConditionA*>) **DO** <*action*>". Having this optimization in mind, <*onlineEvent1*> and <*onlineConditionA*> would not need to be evaluated if <*offlineEvent1*> is true and <*offlineConditionA*> false, saving time and resources (e.g., processor time, battery, networking cost, etc.).

5 Conclusion and future works

The work presented in this paper introduces the first steps to extend a mobile-based smart space interaction paradigm with semantic capabilities. This extension implies the integration of several ontology models and a semantic and rule-based inference engine inside mobile personal devices (e.g., a smartphone).

After introducing the ECA model concept and the (non-semantic) architecture of an already developed prototype [1], the employed ontology models have been detailed, highlighting how they can be exploited in order to *(i)* automatically de-

tect incompatible ECA rules configurations and to *(ii)* support complex rules configuration and execution.

Our current work includes a real implementation of this approach for Android-based smartphones. After validating this first version, the semantic models used should be further developed, defining particular restrictions and reusing existing ontology models. The final objective is to compare performance and usability, in order to come to conclusions that may support the choice of using compact reasoners in mobile applications with situational checking needs.

Acknowledgements
This work has been supported by the Government of Madrid under grant S2009/TIC-1485 (CONTEXTS) and by the Spanish Ministry of Science and Innovation under grant TIN2008-06742-C02-01.

References

[1] A.M. Bernardos, J.R. Casar, J. Cano, L. Bergesio. "Enhancing interaction with smart objects through mobile devices". In: 9th ACM Int. symposium on Mobility management and wireless access (MOBIWAC'11), pp. 199-202. Miami, FL, USA. November 2011.

[2] M. Beigl. “Point & Click – Interaction in Smart Environments”. Procs. of the First Int. Symposium on Handheld and Ubiquitous Comp., Springer-Verlag, pp. 311-313, 1999.

[3] J. Iglesias, A.M. Bernardos, P. Tarrío, J.R. Casar, H. Martín. "Design and validation of a light inference system to support embedded context reasoning". Personal and Ubiquitous Computing, pp. 1-17, 2011.

[4] F. Siegemund. “A context-aware communication platform for smart objects”. In: In Proc of the Int Conf on Pervasive Computing, pp. 69-86. Springer-Verlag, 2004.

[5] Ye, J., Coyle, L., Dobson, S., Nixon, P.: Ontology-based models in pervasive computing systems. Knowl. Eng. Rev. 22, 315-347, 2007.

[6] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni. “A survey of context modelling and reasoning techniques”. Pervasive Mob. Comput. 6, 161-180, 2010.

[7] F. Crivellaro. “μJena: Gestione di ontologie sui dispositivi mobile”. MsC Thesis, Politecnico di Milano, 2007.

[8] M. Jang, J.C. Sohn. “Bossam: An extended rule engine for owl inferencing”. In: G. Antoniou, H. Boley (eds.) Rules and Rule Markup Languages for the Semantic Web, Lecture Notes in Computer Science, vol. 3323, pp. 128-138. Springer Berlin/Heidelberg, 2004.

[9] A. Toninelli, A. Pathak, V. Issarny. “Yarta: A middleware for managing mobile social ecosystems”. In: Advances in Grid and Pervasive Computing, Lecture Notes in Computer Science, vol. 6646, pp. 209-220. Springer Berlin/Heidelberg, 2011.

[10] S. Hachem, A. Toninelli, A. Pathak, V. Issarny, “Policy-Based Access Control in Mobile Social Ecosystems”. IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY’11), pp.57-64, June, 2011.

[11] J.J. Alferes, R. Amador. “r3: A Foundational Ontology for Reactive Rules”. In Proc. OTM Conferences, part I, vol. 4803, pp.933-952, 2007.

[12] R. Amador, J.J. Alferes, “Knowledge Resources Towards a RESTful Knowledge”. To be published. http://k4r.googlecode.com/files/200903_kr2rk.pdf, 2009.

[13] M. Kifer, "Rule Interchange Format: The Framework". In: Web Reasoning and Rule Systems, Lecture Notes in Computer Science, vol. 5341, pp. 1-11, 2008.

[14] F. Siegemund, C. Floerkemeier, H. Vogt. “The value of handhelds in smart environments”. Procs. of ARCS, LNCS 2981, pp. 291-308, 2004.