Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2012, Article ID 845762, 14 pages
doi:10.1155/2012/845762

Research Article

Enabling Semantic Technology Empowered Smart Spaces

Jussi Kiljander, Arto Ylisaukko-oja, Janne Takalo-Mattila, Matti Eteläperä, and Juha-Pekka Soininen

VTT Technical Research Centre of Finland, Kaitoväylä 1, Oulu, P.O. Box 1100, 90571 Oulu, Finland

Correspondence should be addressed to Jussi Kiljander, jussi.kiljander@vtt.fi

Received 16 March 2012; Revised 17 September 2012; Accepted 19 September 2012

Academic Editor: Jae-Ho Choi

It has been proposed that Semantic Web technologies would be key enablers in achieving context-aware computing in our everyday environments. In our vision of semantic technology empowered smart spaces, the whole interaction model is based on the sharing of semantic data via common blackboards. This approach allows smart space applications to take full advantage of semantic technologies. Because of its novelty, there is, however, a lack of solutions and methods for developing semantic smart space applications according to this vision. In this paper, we present solutions to the most relevant challenges we have faced when developing context-aware computing in smart spaces. In particular the paper describes (1) methods for utilizing semantic technologies with resource restricted-devices, (2) a solution for identifying real world objects in semantic technology empowered smart spaces, (3) a method for users to modify the behavior of context-aware smart space applications, and (4) an approach for content sharing between autonomous smart space agents. The proposed solutions include ontologies, system models, and guidelines for building smart spaces with the M3 semantic information sharing platform. To validate and demonstrate the approaches in practice, we have implemented various prototype smart space applications and tools.

1. Introduction

The environments we live in (homes, cars, work places, etc.) are inhabited by a large and a constantly increasing number of electronic devices with huge amounts of information embedded into them. Smart space is a name for a physical place where these devices interoperate with each other in order to provide the user with services that are relevant in the given situation. In order to achieve this, it is necessary for a device to be able to "understand" its context. Here the term context follows the definition: "*Context is any information that can be used to characterize the situation of an entity.*" [1].

The smart space vision can be traced back to the beginning of the 1990s, when Mark Weiser presented his ideas of ubiquitous computing [2]. In addition to ubiquitous computing, smart spaces are a widely studied concept (with a slightly different area of emphasis) in pervasive computing [3], ambient intelligence (AmI) [4], and Internet of Things (IoT) [5] research. There have been many projects such as Buxton's Reactive Environment [6], Massachusetts Institute of Technology's Oxygen [7], Microsoft's EasyLiving [8], Hewlett Packard's Cooltown [9], and Stanford University's iRoom [10], just to name a few, focusing on different aspects of smart spaces. These research projects, among others, have produced many approaches for realizing various features of context-aware computing in smart spaces. However, to enable smart spaces on a larger scale, there is a need for Web-like infrastructure that provides a reusable base for building context-aware smart space applications.

Service-oriented architecture (SOA) frameworks and technologies such as Universal Plug and Play (UPnP) [11], OSGi [12], Simple Object Access Protocol (SOAP) [13], and Representational State Transfer (REST) [14] represent progress to the right direction as they provide reusable solutions for interoperability. However, SOA technologies are still heavily focused on using case specific a *priori* standardization that has its limits as it is impossible to anticipate all possible needs of future applications. Additionally, SOA technologies do not provide a standard language for presenting semantics of information in a formal structured manner. Because of this, it is difficult to develop SOA-based systems where devices share a mutual understanding of the context and are

thus able to interoperate with each other in order to provide relevant services for the end-users.

In 2001 Berners-Lee et al. presented a landmark paper titled "The Semantic Web" [15]. In the paper they presented a vision of the next generation World Wide Web (WWW), where the semantics of information would be presented in machine-interpretable format allowing autonomous software components, called agents, to execute tasks on behalf of humans. As semantic technologies provide, at the same time, a very flexible linked data [16] model and mechanisms for interchanging semantics of information in a structured format, they seem to be also a natural fit to the interoperability needs of context-aware smart space applications. This has been also proposed in various occasions. For example, in [17] Chen presents a Context Broker Architecture for Pervasive Computing (CoBrA) that utilizes Semantic Web technologies to provide context-aware computing infrastructure for physical places, and in [18] Wang et al. describe Semantic Web-based pervasive computing infrastructure called Semantic space.

The possibilities provided by the Semantic Web technologies to context-aware smart space systems would be plenty. Ontologies presented with semantic technologies such as the Resource Description Framework (RDF) [19], the RDF Schema (RDFS) [20], and the Web Ontology Language (OWL) [21] could be used to describe the properties, capabilities, and intentions of devices, persons, and other physical objects in the environment. This would make it easier to develop context-aware applications capable of providing more relevant services for users. The flexible linked data model of RDF would make it also much easier to combine information from various sources and even from different application domains in order to create a better view of the context in the environment. More importantly, RDF would allow new information to be added without a risk of breaking the existing systems. In the most advanced scenarios, the Semantic Web technologies would even allow smart space applications to utilize information about concepts they are unfamiliar with. This is made possible by the Semantic Web ontologies that allow unknown concepts to be described in terms of known ones much in the same way as humans use encyclopedias to describe words they do not know.

We agree with the vision that context-aware smart space infrastructure should rely on ontology-based information models and utilize Semantic Web technologies in ontology presentation. However, our vision differs significantly from existing approaches such as Task Computing Environment (TCE) [22], COCOA [23], Semantic Middleware for IoT [24], and Amigo [25], where Semantic Web technologies have been used to assist the user to better exploit the services available in a pervasive computing environment. We propose that the semantic technologies are not only used to enhance service discovery, composition, and utilization, but also as a way to share any kind of information in the smart space. By this we mean that the whole interaction model in smart spaces is based on semantic information sharing via common knowledge bases. When compared with the aforementioned service-based approaches, the advantage of our approach is that it makes it possible to fully exploit the semantic technologies in ubiquitous computing systems. In our approach, we also focus on two significant aspects that have been typically neglected in semantic smart space systems: low capacity systems and object identification.

In this paper, we will describe several approaches that make it possible to realize our vision of semantic technology empowered pervasive computing systems in practice. We first identify and then present solutions to some of the most important challenges we have faced in providing context-aware computing in semantic smart space systems. The work presented in the paper is built on top of M3 semantic interoperability platform [26] (that we have been developing in various projects) and includes various approaches, ontologies, and smart space applications and tools. When combined, these approaches and methods are significant because they enable the development of smart space applications which are able to fully exploit semantic technologies. The main scientific contributions of the paper can be summarized as follows:

(1) solutions for opening information of low-capacity embedded systems for semantic technology empowered smart spaces,

(2) methods to identify real-world objects in semantic smart spaces and to fetch information related to these objects from all over the world,

(3) methods for end-user to modify behavior of semantic technology empowered smart spaces,

(4) approach for file sharing between autonomous smart space agents.

The rest of the paper is organized as follows. In Section 2, we introduce necessary background information related to the M3 concept. Section 3 presents the related work and how ours differ from it. Section 4 describes the main challenges we have faced in building semantic technology based context-aware applications to smart spaces. In Section 5, we present several solutions and approaches to overcome these challenges. Section 6 describes prototype smart spaces applications and tools we have implemented to validate our approaches in practice. Finally, Section 7 concludes the paper and presents some future work directions.

2. M3: Semantic Information Sharing Solution for Smart Spaces

The goal of M3 is to exploit the Semantic Web ideas of ontologies and linked data to provide location-aware services to physical places. In practice this is achieved through ontology-based interoperability model and a functional architecture that specifies a device, application, and service domain in independent way to access the semantic data in smart spaces.

Semantic Web standards including RDF, RDFS, and OWL provide the core technologies for the ontology-based interoperability model of M3. RDF is a Semantic Web standard for modeling metadata in form of subject, predicate,

and object triples. For smart space agents, RDF triples provide a natural way to create linked data structures about the context information of smart spaces. RDFS and OWL in turn provide vocabularies to be used on top of RDF to describe relevant concepts in smart spaces as machine-interpretable ontologies. Excluding the RDFS and OWL ontologies, the M3 concept does not require any specific ontology to be used when M3 applications are developed. This allows existing Semantic Web and pervasive computing ontologies such as SSN [27], SOUPA [28], and ULCO [18] to be utilized when creating new M3 applications.

The M3 functional architecture differs from other similar frameworks such as CoBrA and Semantic Space in its simplicity as it defines only two types of processes: Knowledge Processor (KP) and Semantic Information Broker (SIB). SIB is a shared blackboard of semantic information that provides KPs with an interface for sharing semantic data. KPs are software agents whose role is to provide the end-user with services by interacting with each other via the SIB. The KPs sharing information via a common SIB form a single smart space as illustrated in Figure 1. The Smart Space Access Protocol (SSAP) defines the rules and syntax of the communication between the SIB and KP. The SSAP is based on publish/subscribe paradigm providing event based interaction via the following operations: *join()*, *leave()*, *insert()*, *remove()*, *update()*, *query()*, *subscribe()*, and *unsubscribe()*.

The basic principle of M3 is to build on top of the existing communication and service level solutions. In practice this means that SSAP can be used as a payload in any communication technology and a KP can interact with a SIB if they share at least a single common communication method. A SIB can be also implemented as a service in any SOA solution allowing SOA-specific methods to be used by KPs to discover and interact with a SIB.

There are two implementations of the M3 concept available: Smart-M3 [29] and RDF Information Base Solutions (RIBS) [30]. Smart-M3 is Linux-based implementation that utilizes XML serialized SSAP format and supports both NoTA [31] and plain TCP/IP-based communication technologies. Additionally, the Smart-M3 SIB has also been implemented as OSGi framework service [32]. In the core of the Smart-M3 SIB is a RDF++ database called *PIGLET*. The *PIGLET* provides an interface for two types of query languages: simple template queries and Wilburs query language (WQL) [33]. The template query pattern consists of a one or more triples which are matched separately against an RDF graph in a SIB. The triples may contain a wildcard *sib:any* denoting any node in the RDF graph. In WQL a query is specified in terms of a start node and a path consisting of predicates to be traversed from the start node. In this paper, WQL path queries are presented using the following syntax:

WQL(startNode, path)

RIBS is a SIB implementation designed for security, portability, and performance. It is implemented in ANSI-C programming language and it supports both plain TCP and NoTA-based transports. The security in RIBS is based on transport layer security (TLS) and triple level access control mechanisms. To provide better performance and portability, RIBS utilizes a more compact SSAP serialization format, called Word Aligned XML (WAX). RIBS also differs from the Smart-M3 in the query languages it supports. In addition to the basic template queries, RIBS provides limited SPARQL [34] support. WQL is not supported.

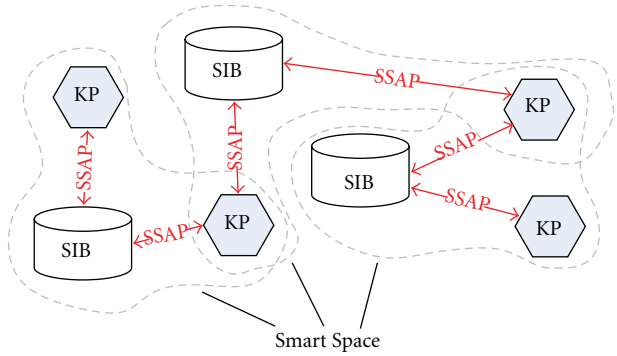

Figure 1: Composition of M3-based smart spaces.

3. Related Work

After the emergence of the Semantic Web paradigm, many semantic technology-based pervasive computing middleware systems have been proposed. The most typical way to utilize semantic technologies in pervasive computing systems is to use them to enhance traditional SOA-based systems. These approaches include, for example, TCE, COCOA, Semantic Middleware for IoT, and Amigo.

To the authors' knowledge, TCE is the first approach to combine Semantic Web and SOA technologies in pervasive computing domain. In TCE the term *Task Computing* refers to a computation that provides the end-user with required functionality by utilizing the services available in the given situation. The main goal of TCE is to allow users to concentrate on tasks the user wants to do, rather than the specific ways for accomplishing the task. A TCE system consists of one or more instances of the following components: Task Computing Client, Semantically Described Service, Semantic Service Discovery Mechanism, and Service Control (optional).

COCOA is a conversation-based solution for on-the-fly service integration in pervasive computing environments. COCOA is built on the top of OWL-S [35] and consists of three parts: COCOA-L, COCOA-SD, and COCOA-CI. The COCOA-L is OWL-S extension that provides means to specify the service capabilities, service conversation, and quality of service (QoS) properties. Service discovery and selection in COCOA are implemented in COCOA-SD. Both of the nonfunctional and functional capabilities are utilized in the discovery/selection process. The final composition of the required service is executed by COCOA-CI. The distinctive feature of COCOA-CI is that both the tasks and the integrated services are modeled as conversations.

The basic idea in the Semantic Middleware for IoT is to transform the existing specifications of standards such as Bluetooth and UPnP into Semantic Web services described with OWL-S. This enables users to create task by combining the capabilities of different devices without the need to know how the functionality is achieved in practice. There are three steps defined in the approach: device semanticization, task building, and device grounding. In the device semanticization phase, the middleware framework utilizes the standard specific discovery mechanism for locating the devices and services. Then the middleware extracts the necessary data and creates the OWL-S description at a run-time. In the task building phase, the Task Computing framework assists the user in the creation of new tasks. The last phase is the device grounding where the actual task is executed. In this phase, the technology-specific service realization is invoked to execute the required functionality of the services.

The Ambient Intelligence for the Networked Home Environment (Amigo) Project has developed an interoperability platform that enables interaction between heterogeneous services and devices in a home network domain. The Amigo architecture is based on the SOA paradigm and supports various different SOA technologies such as UPnP, Service Location Protocol (SLP), and Web Service Description Language (WSDL). Additionally, it is possible to enrich any Amigo service with Amigo-S semantic service description language which continues the work of COCOA. The advantage of utilizing Amigo-S is that it provides much more flexible and effective service discovery based-on the context than can be achieved with traditional SOA-based service discovery mechanisms. The Amigo-S is based on OWL-S and extends it with functionality needed for describing context and nonfunctional properties such as the quality of the service. In addition, the OWL-S is extended with mappings to new types of groundings than just the WSDL supported by the OWL-S.

Our vision of semantic smart spaces differs significantly from the aforementioned solutions. In our approach, the semantic technologies are not only used to enhance service discovery, selection, and composition, but also as a way to share any kind of information in the smart space (i.e., the whole interaction model is based on sharing of semantic data via common knowledge bases). This kind of approach is closer to the original vision of the Semantic Web and provides better interoperability as it is not required to directly interact with various technology specific services. To concretize, the sensor information can be accessed and actuators controlled just by modifying the semantic data in the smart space. Our work also differs from the aforementioned systems by taking into account that the typical devices in smart spaces are resource restricted. Additionally, in contrast to other semantic pervasive middleware systems, we use ucode-based URIs to identify real-world objects. The ucode-based approach is more suitable to various pervasive computing scenarios such as retail, than the traditional URL-type URIs used typically in the Semantic Web-based systems.

4. Challenges of Context-Aware Computing in Semantic Technology Empowered Smart Spaces

4.1. Challenge 1: Accessing Data of Embedded Systems. There is a huge amount of data embedded into various devices that inhabit our everyday living environments. It would be beneficial to open this information for other devices and applications in a common machine-interpretable format. This would enable devices, agents and applications to access much larger pool of data and thus obtain a better view of the context in the smart space. Consequently, smart space agents, could provide more relevant and sophisticated services for the user.

Because of the heterogeneity of devices and communication methods, it is not easy to utilize semantic technologies in real-life smart spaces. Especially the most resource-limited devices and networks are a big challenge. This is because the technologies of the Semantic Web have not been designed for low-capacity embedded systems. For example, the standard serialization formats for RDF and OWL are based on Extensible Markup Language (XML) which does not suit resource-restricted communication channels well. XML is also difficult to parse on low-capacity embedded systems. Additionally, typical ontologies are difficult to process and store in resource-constrained systems. This is because ontology designers use human readable vocabulary which is sparse and causes thus overhead to the system.

4.2. Challenge 2: Identification of Real-World Objects. In order to create context-aware applications to physical environments, it is necessary to be able to identify objects such as devices, sensors, actuators, locations, and users that form the context of the physical place. In practice this means that each object that contributes to the context of the environment must have a unique virtual identifier that can be used to refer to the object in the virtual world, that is, in the semantic database.

In typical Semantic Web-based systems, Uniform Resource Identifiers (URIs) [36] and later Internationalized Resource Identifiers (IRIs) [37] have been used as identifiers for various kinds of Web resources. This approach has been adopted from the traditional Web where the URL-type URIs have been successfully used for decades. The power of the URLs is that the Web architecture can be used to locate information related to URLs from all over the world. However; this approach does not always work in Semantic Web. This is because the agents that provide data to the Semantic Web do not typically interact with other agents directly but publish the data to semantic databases. The problem here is that because these semantic databases have a different hostname than the agent who created the resource, the Web architecture cannot be used to locate the information related to a URI. Another problem related to the traditional URL in both Semantic Web and smart spaces is that the Web architecture does not allow the same URL to point to various locations. In certain application domains such as retail and logistics, it is typical that information

related to a physical object is scattered to various semantic databases and it would be, therefore, required to be able to link the object identifier to multiple addresses.

Smart spaces also set their own challenges for using URIs as identifiers for physical objects. First, URIs are typically long strings that are difficult to process in resource-restricted devices. Second, resource-restricted devices in smart spaces are not necessarily connected to the Web and thus do not have unique global address that could be used as a base when creating new URIs for resources they publish to the smart space.

4.3. Challenge 3: Taking Human Needs into Account. The grand vision of smart spaces is to make the life of people better by providing useful services when necessary. The idea is to hide sensors, actuators, and other computing entities from the user so that she/he does not have to be bothered with unnecessary technical tasks. In order to achieve this, devices and context-aware applications need to be able to know what kind of services and functionality the user requires in different situations.

User profiles are the typical way to model the desires of the end-user in semantic technology-based smart space applications. User profiles provide a subtle and feasible solution to some use of cases of context-aware computing. However, it is difficult to design profiles that are, at the same time, generic enough and still suitable for all kinds of possible situations that may take place in smart spaces. In addition to user profiles, behavioral models and machine learning have been exploited in the field of ambient intelligence [4] to provide advanced methods for recognizing user needs in different situations. However, to fully realize the visions of smart spaces, humans need also to be able to be take more active role in smart spaces. Users need to able to express what kind of functionality is required from the smart space in a given context so that the services provided to the user are in fact useful for her/him.

4.4. Challenge 4: Autonomous Content Sharing. Typical smart spaces contain devices such as smart cameras that take pictures and record video of their environments. In modern computer systems, this kind of content is typically streamed as a live stream or stored as a file. Therefore, it is essential to have common methods to enable devices and agents to share content such as files and live media streams autonomously in context-aware smart spaces. By autonomously we mean that the devices interact with each other seamlessly so that the heterogeneity of devices, communication technologies, and file sharing methods are hidden from the end-user.

This is not a simple task in semantic technology empowered smart spaces, however. The main reason for this is that the semantic technology-based information sharing infrastructures such as the M3 are not feasible for storing "nonsemantic data" such as files and cannot be used to stream live media between agents. To perform this kind of functionality, the agents need to rely on existing protocols designed for this purpose. However, there is a big variety in the file sharing methods and communication technologies used to transmit data between devices. Because of this, it is difficult both to find all available file transfer services and to interact with them to execute the required functionality. In practice this means that a device which wants to ensure that a file it is sharing is accessible to as a wide range of devices and applications as possible must advertise the file with each communication technology and file sharing protocol it supports. This, of course, causes a lot of overhead when a device supports many different file sharing and communication methods.

5. Solutions for Semantic Technology Empowered Context Awareness in Smart Spaces

5.1. Accessing Information of Embedded Systems. One can think of two fundamental approaches to overcome the challenge related to opening the information of low-capacity embedded systems for context-aware applications in semantic smart spaces. The first approach is to build gateways that transform the raw data produced by various embedded systems to a semantic format. The advantage of this approach is that it is easy to implement on the embedded system side as there is no need to support semantic technologies which always introduce some amount of overhead to the communication. The drawback is that this approach is not very scalable or flexible as a new gateway needs to be implemented or existing gateways modified for each new type of device (or more precisely new type of data).

The second approach is to implement the software agent into the low-capacity embedded system. This approach is, of course, more flexible as all devices would utilize the same semantic protocol and no gateways would be needed. Another advantage of this approach is that the embedded system could utilize the semantic data produced by other devices and thus possibly improve its functionality in some way. The drawback is, however, that there are a lot of challenges in implementing a semantic technology empowered smart space agent into a resource-restricted embedded system.

There is no single solution for developing a semantic technology-based application for resource-restricted devices, but we attempt to present the most relevant issues a developer should consider when given such a task. In M3-based systems, the SSAP has a big impact in how easy it is to implement a KP in low-capacity device and therefore we propose the WAX encoded SSAP format to be used with low capacity devices. In WAX each XML tag is 32-bit long and the element contents are aligned to the word length of the CPU. This way the WAX-encoding is more compact and much faster to parse than the original SSAP/XML encoding. Another important advantage of WAX is that it does not require an XML parser and is therefore more portable to resource-restricted devices.

One of the most important issues when designing a KP to a resource-restricted platform is predictability. By this we mean that it is important to be able to define the needed computing resources such as memory at compile time.

Because of this, it is much simpler to implement a producer KP that just publishes information to the SIB. Fortunately, the majority of resource-restricted devices are sensors which simply publish information to the SIB. If a KP is required to fetch information from the SIB, the query and subscribe operations should be executed so that they do not return too large results, which may cause buffer overflows and are slow to process. However, this is not easy because the current M3 implementations do not provide feasible mechanisms for restricting the response size of the SSAP messages. With RIBS it is possible to utilize the LIMIT sequence modifier of SPARQL to restrict the number of results returned by the SIB, but this does not help when the results are of arbitrary length.

Usually the payload of a packet is the biggest factor in the memory requirements and performance of a networked system. In semantic technology-based smart space application, the payload shared between devices is defined by ontology. Ontologies targeted for context-aware pervasive computing applications such as SOUPA [28], COBRA-ONT [38], ULCO [18], and SeMaPs [39], have unfortunately adopted the Semantic Web style to use human-readable IRIs as identifiers for the concepts defined in ontology. This, of course, makes the ontology easier to design and understand by humans but is, in the end, completely unnecessary as ontologies are meant to be languages for devices. The human-readable ontologies introduce a lot of unnecessary overhead that could be avoided if the long human-readable IRIs would be replaced, for example, by unique 64-bit integer values. We propose an approach where a dual presentation for ontologies is used. A more compact binary format ontology can be created from the traditional human-readable ontology by replacing each ontology name URI and each concept URI in the ontology by unique 32-bit integers. This way we can create 64-bit identifiers where the first 32 bits are reserved for the ontology namespace and the last 32 bits for various concepts defined in a single ontology.

5.2. Object Identification. Our approach to identify real-world objects in semantic smart spaces is based on the uID technology [40] which provides mechanisms for both unique identification of real-world objects and for retrieving information related to these objects from all over the world. In uID 128-bit expandable codes, called ucodes, are used to identify real-world objects uniquely. The idea is that ucodes can be used with any kind of tags such as Radio Frequency Identification (RFID), Near Field Communication (NFC) and Quick Response (QR) Codes. The uID address sharing architecture defines how the information related to a ucode-tagged object can be accessed. The uID architecture consists of resolution and information servers. The three-tier resolution server architecture provides mechanisms for resolving the address of application domain-specific information server that hosts information related to a ucode-tagged object.

We propose an approach where real-world objects in semantic technology empowered smart spaces are tagged with ucodes and M3 SIBs are used as uID information servers which host semantic data about the ucode-tagged objects. The key idea in the approach is to use the ucode as the URI for the virtualized physical object. This way we can create a natural link between the physical object and its virtual counterpart in semantic databases. We utilize and a propose URI scheme with a prefix "ucode" to be used to identify the ucode-type URIs.

Our approach provides various interesting possibilities for context-aware smart space applications. First, it provides new kinds of ways for people to interact with the agents of semantic smart spaces. The user is able to read sensor measurements and actuate objects, for example, by simple touching them with a tag-reader equipped smart phone. Second, it allows context-aware smart space applications to offer better functionality by collecting and utilizing information related to a physical object from various sources located even in the other side of the world. A trivial example of this occurs when a repair assistant agent provides guidance for maintenance personnel by fetching necessary information related to a broken product from the SIBs of manufacturer, user, and other maintenance companies.

5.3. Enabling Users to Modify the Behavior of Smart Spaces. To provide a solution to the challenge related to taking human needs into account, we propose a rule-based approach that enables users to define how a smart space should behave in different situations. The approach allows users to define rules that consist of input condition and actions. The input condition is defined by combining selected events with Boolean operators. The idea is that all events and actions have a dual presentation. Human readable description is used for end-users and for KPs the events and actions are expressed with RDF query and update language patterns. This way even complex events can be expressed and computed using the built-in subscribe support of the M3 platform.

The approach is divided into ontology and system models. The event and action ontology (EA-ONT) describes the data shared by the components of the system model. The main classes of the EA-ONT are *Event* and *Action*. These classes have a human-readable name and a format presented either as basic triple, WQL, or SPARQL pattern. The ranges of possible values for events and actions are presented with by *NumericRange* and *EnumeratedRange* classes. The formal ontology model is presented with the Turtle notation [41] as follows:

```
#Ontology namespaces
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix event: <http://example.org/event/>.

#Class definitions
event:Event rdf:type rdfs:Class.
event:Action rdf:type rdfs:Class.
event:Format rdf:type rdfs:Class.
event:Range rdf:type rdfs:Class.
event:TripleFormat rdfs:subClassOf event:Format.
event:WqlFormat rdfs:subClassOf event:Format.
event:SparqlFormat rdfs:subClassOf event:Format.
```

```
event:EnumeratedRange rdfs:subClassOf event:Range.
event:NumericRange rdfs:subClassOf event:Range.

#Property definitions
event:hasFormat rdfs:domain event:Event,
    event:Action.
event:hasFormat rdfs:range event:Format.
event:hasRange rdfs:domain event:Format;
    rdfs:range event: Range.
event:hasSubject rdfs:domain event:TripleFormat;
    rdfs:range rdf:subject.
event:hasPredicate rdfs:domain event:TripleFormat;
    rdfs:range rdf:predicate.
event:startNode rdfs:domain event:WqlFormat;
    rdfs:range rdfs:Resource.
event:path rdfs:domain event:WqlFormat;
    rdfs:range xsd:string.
event:spaqrlQuery rdfs:domain event:SparqlFormat;
    rdfs:range xsd:string.
event:hasElement rdfs:domain event:EnumeratedRange;
    rdfs:range xsd:string.
event:minValue rdfs:domain event:NumericRange;
    rdfs:range xsd:double.
event:maxValue rdfs:domain event:NumericRange;
    rdfs:range xsd:double.
event:step rdfs:domain event:NumericRange;
    rdfs:range xsd:double.
```

The system model, illustrated in Figure 2, defines the functional entities that provide the end-user with a possibility to modify the behavior of semantic smart spaces. The system model consists of three entities: Event Provider, Rule Handler, and Event Browser. In the M3 level, these entities are implemented as KPs that utilize the SIB for sharing semantic data with each other.

Event Providers create events to smart spaces by producing semantic information as specified by the EA-ONT. Ideally Event Providers are located in devices that provide certain information to the smart space. For example, an Event Provider can be an embedded system that measures the temperature of the room and publishes this information to the smart space. It is also possible to use a generic KP for creating events and actions based on data published by other KPs. This way it is possible to (1) create events from information published by KPs whose developers were not familiar with our approach and (2) create combined events that model data produced by various KPs as a single event or action.

A central component of the system model is the Event Browser that provides the user with a possibility to define new rules to the smart space. The Event Browser is subscribed to the available events and actions and it is therefore capable of presenting the current view of the smart space functionality for the user. When the user creates a new rule, the Event Browser passes the required data about the rule to the Rule Handler that is responsible for processing the rules.

Once the Rule Handler receives a new rule, it subscribes to the events that define the input condition for the rule. This way the Rule Handler will be notified by the SIB when data related to these events is modified. When the SIB indicates that information related to the input events has been modified, the Rule Handler evaluates whether the input condition is satisfied. If the input condition is true, the Rule Handler executes actions by updating information to the SIB as specified in the triple pattern of the actions. It is noteworthy that because the events and actions are presented as query and update language patterns, the Rule Handler does not have to be familiar with the domain-specific ontology used by the KPs.

5.4. Autonomous Content Sharing in Smart Spaces. To provide a solution to the challenge related to providing autonomous content sharing in semantic smart spaces, we model necessary information about files and file sharing methods as a common ontology and use the SIB as a semantic communication channel where this data is shared between KPs. This enables software agents first to discover the available files in the smart space and then to negotiate about the most suitable methods for transferring the file between agents. The File Sharing Ontology (FS-ONT) modeled with the RDFS vocabulary can be presented with the Turtle notation as follows:

```
@prefix fso: <http://example.org/fileSharing/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

# Class definitions
fso:File rdf:type rdfs:Class.
fso:FileRequest rdf:type rdfs:Class.
fso:SharingMethod rdf:type rdfs:Class.
fso:TcpIp rdfs:subClassOf fso: SharingMethod.
fso:Bluetooth rdfs:subClassOf fso: SharingMethod.

#Property definitions
fso:filename rdfs:domain fso:File;
    rdfs:range xsd:string.
fso:format rdfs:domain fso:File;
    rdfs:range xsd:string.
fso:modifiedDate rdfs:domain fso:File;
    rdfs:range xsd:dateTime.
fso:size rdfs:domain fso:File;
    rdfs:range xsd:integer.
fso:targetFile rdfs:domain fso:FileRequest;
    rdfs:range fso:File.
fso:hasSharingMethod rdfs:domain fso:FileRequest;
    rdfs:range fso:SharingMethod.
fso:ip rdfs:domain fso:TcpIp;
    rdfs:range xsd:string.
fso:port rdfs:domain fso:TcpIp;
    rdfs:range xsd:integer.
fso:macAddr rdfs:domain fso:Bluetooth;
    rdfs:range xsd:integer.
fso:channel rdfs:domain fso:Bluetooth;
    rdfs:range xsd:integer.
```

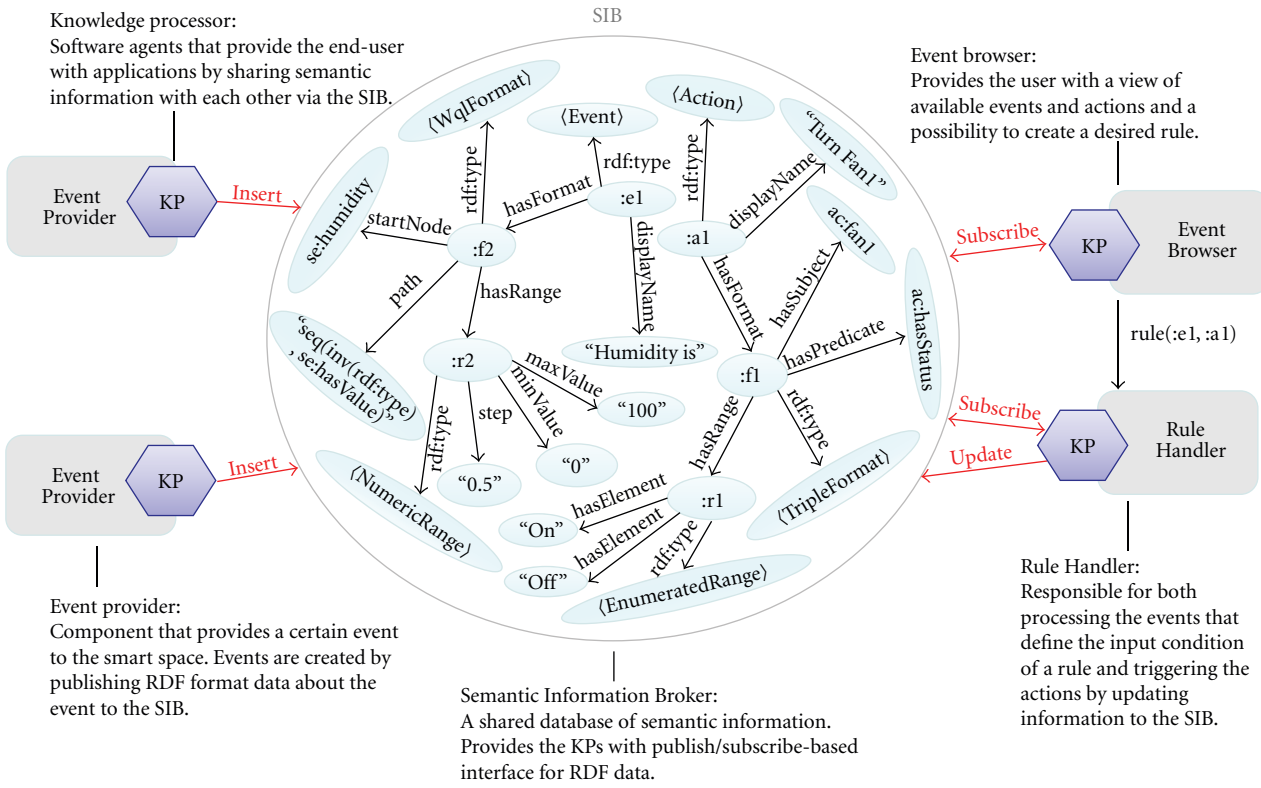

Figure 2: A system model for modifying the behavior of smart spaces.

The FS-ONT contains three main classes: *File*, *FileRequest,* and *SharingMethod*. The *File* class provides properties for describing concepts common for all kinds of files. These properties include the name, size, format, and modification date of the file. The idea is that applications that require more specific types of files import the FS-ONT and define new subclasses for the common *File* class. The *FileRequest* class represents an abstract concept of a request to a file. The class provides two properties: *targetFile* and *hasSharingMethod*. These properties associate the request with the requested file and the file sharing methods supported by the requester. The *SharingMethod* class is a base class for all protocols enabling file sharing. The FS-ONT defines two subclasses, namely, *Bluetooth* and *TcpIp*, for the *SharingMethod* class, and the idea is again that applications import the FS-ONT and introduce new subclasses when necessary.

In order to be informed about the available files and then to successfully negotiate about the most appropriate file sharing methods, KPs need to be able to perform four different SSAP operations. First, a KP that wants to share a file to a smart space needs to publish metadata about the file into the SIB as specified by the FS-ONT. Second, to be aware of the file requests, it needs to perform the following WQL subscription:

WQL(file, seq(inv(fso:targetFile),fso:hasSharingMethod))

where *file* is the URI of the *File* class instance and *seq()* and *inv()* are WQL-specific operations. The operation *seq()* denotes that the path consists of several RDF predicates and the *inv()* operation requires the predicate inside the brackets to be traversed in a reverse direction, that is, from object to subject. This operation subscribes directly to the file transfer technologies supported by KPs that request the file and thus enables the publisher KP to interpret whether the KPs share common file sharing methods.

Third, a KP that needs a certain file from the smart space needs to subscribe to all the available files in the SIB. A WQL subscription which informs when a new file is inserted to the SIB can be presented as

WQL(fso:File, inv(rdf:type)).

If needed, it is possible to make this subscription more specific when extra information such as the name or format of the file of interest is known by the requester.

Fourth, when a KP wants to request a file from another KP in the smart space, it must make a request by publishing information to the SIB as specified in the FS-ONT. As mentioned, this information must specify both the file of interest and the supported file transfer technologies. A practical example of autonomous file sharing between autonomous agents is given in Section 6.4.

6. Validation of Proposed Solutions

6.1. Smart Home Garden. The Smart Home Garden is a simple smart space where autonomous resource-restricted devices monitor the well-being of plants. It demonstrates

how especially the challenges 1 and 2 can be solved in real-life smart spaces. The demonstration setup consists of potted plants, low-capacity moisture sensors, and a smart phone.

We use RIBS as the M3 broker implementation and it runs on a Via Artigo A1100 Ubuntu PC. To provide a wide range of methods for KPs to share information via the SIB, we connected a Redwire LLC *Econotag* board to the Via Artigo platform. This way the KPs can communicate with the SIB using 3G, WLAN, and IEEE 802.15.4 radio technologies.

In M3-based pervasive computing applications, ontologies play a key role in the interoperability as they define the semantics of the data shared between devices. The ontology suite developed for the Smart Home Garden consists of sub-ontologies for concepts such as sensor, plant, location, and user and can be presented with the Turtle notation as follows:

```
#Ontology namespaces
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix se: <http://example.org/sensor/>.
@prefix lo: <http://example.org/location/>.
@prefix pl: <http://example.org/plant/>.
@prefix us: <http://example.org/user/>.

# Class definitions
lo:PhysicalObject rdf:type rdfs:Class.
lo:Location rdfs:subClassOf lo:PhysicalObject.
se:Sensor rdfs:subClassOf lo:PhysicalObject.
se:Measurement rdf:type rdfs:Class.
se:MoistureSensor rdfs:subClassOf se:Sensor.
se:HumiditySensor rdfs:subClassOf se:Sensor.
se:TemperatureSensor rdfs:subClassOf se:Sensor.
se:LightSensor rdfs:subClassOf se:Sensor.
pl:Plant rdfs:subClassOf lo:PhysicalObject.
pl:PottedPlant rdfs:subClassOf pl:Plant,
    lo:Location.
us:User rdfs:subClassOf lo:PhysicalObject.

#Property definitions
se:hasMeasurement rdfs:domain se:Sensor;
    rdfs:range se:Measurement.
se:hasName rdfs:domain se:Sensor;
    rdfs:range xsd:string.
se:hasValue rdfs:domain se:Measurement;
    rdfs:range xsd:double.
se:unitOfMeasurement rdfs:domain se:Measurement;
    rdfs:range xsd:string.
lo:hasName rdfs:domain lo:Location;
    rdfs:range xsd:string.
lo:hasLocation rdfs:domain lo:PhysicalObject;
    rdfs:range lo:Location.
pl:minMoisture rdfs:domain pl:Plant;
    rdfs:range xsd:double.
pl:maxMoisture rdfs:domain pl:Plant;
    rdfs:range xsd:double.
```

A low-capacity battery powered device called the Active Tag plays a central role in the Smart Home Garden application [42]. The Active Tag is an embedded system which measures the moisture of the soil and publishes this information to the local SIB. The main task of the Active Tag is to inform the user when the soil is too dry for the given plant by blinking an LED. Each Active Tag is identified by a unique ucode located in a NFC tag connected to the device (see Figure 3). We implemented the Active Tag on a Freescale MC13224V-based *Econotag* platform. The platform runs a Contiki operating system and provides a simple IEEE 802.15.4 compatible Rime communication protocol.

The total code side of the implemented Active Tag agent including the KP, Contiki, and Rime protocol was 39.7 kB. The plain KP code including the KP application programming interface and application logic resulted in a code size of 10.3 kB. The total RAM requirement was 25.3 kB of which 14.3 kB was consumed by the Contiki operating system and the Rime protocol.

The Smart Home Garden is best explained with the following scenario. Alicia is buying new plants to her home. She does not actually have a green thumb, however, and she wants to buy plants that can be identified by the new Smart Home Garden application she acquired the day before. Alicia decides to buy two potted plants which are equipped with QR-code-based ucodes. In the uID resolution server, the ucodes of these plants point to a greenhouse SIB which contains information of the given plants.

When Alicia arrives home, the Home Garden KP in her Google Nexus S Android smart phone informs the smart space about her presence. Next, Alicia assigns the Active Tags to the plants as illustrated in Figure 3. To fetch necessary information related to the plants, Alicia points the camera of her smart phone towards the QR code in the potted plant. Once the ucode is read the Home Garden agent contacts the uID resolution server to obtain the address of the SIB which contains information about the ucode-tagged plant. After obtaining the address of the greenhouse SIB, the Home Garden KP fetches necessary information about the plants and copies the information to the local SIB in Via Artigo. Next, Alicia pairs each Active Tag with a potted plant. To do this, she touches the NFC tag in the Active Tag with her smart phone. After obtaining the ucode of the Active Tag, the Home Garden KP publishes the following RDF triple to the SIB:

```
@prefix lo: <http://example.org/location/>.
@prefix tag: <ucode:5554:2134:6442:>.
@prefix plant: <ucode:5254:2135:6325:>.
tag:1035 lo:hasLocation plant:2552.
```

This RDF triple specifies that the Active Tag identified by the ucode "5554:2134:6442:1035" is located in a potted plant identified by the ucode "5554:2134:6442:2552." The Active Tag needs this information to obtain the moisture preference of the plant. In addition, the Active Tag KP needs information about Alicia's presence so that it does not consume power unnecessary by blinking the LED when there is nobody to notify it. To know whether the Active Tag should blink the LED, it executes the following SPARQL ASK query to the SIB which returns a true or false response:

Figure 3: Active tag in a potted plant.

```
PREFIX lo: <http://example.org/location/>.
PREFIX pl: <http://example.org/plant/>.
ASK
{
  <5554:2134:6442:1035> lo:hasLocation ?pottedPlant.
  ?pottedPlant pl:moisturePreference ?preference.
  FILTER (?preference < value)
},
```

where *value* is the moisture value measured by the Active Tag. In addition to the query operation, the Active Tag updates the moisture value to the smart space in each cycle before going to a deep sleep mode. With 60-second wake-up intervals for query and update operations, we measured an average current consumption of 241 μA. If we do not take a battery self-discharge and LED blinking into account, this means approximate battery duration of 1.3 years with two 1.5 V and 2700 mAh alkaline batteries.

6.2. Smart Greenhouse. Smart Greenhouse is miniature version of a greenhouse where interacting agents assist a gardener in his/her daily routines. The Smart Greenhouse ecosystem illustrates how semantic technologies can be utilized to provide context awareness for more complex smart spaces than the Smart Home Garden [43, 44]. The Smart Greenhouse demonstrator, illustrated in Figure 4, consists of a miniature greenhouse, actuators, tagged plants, sensors, an autocontrol device, and a ubiquitous communicator as the gardener's personal device.

In the Smart Greenhouse, the ontologies used in Smart Home Garden are extended with new subclasses and properties for the plant class. Additionally, actuator ontology (ACT-ONT) was developed for presenting information about the physical actuators in semantic format. The ACT-ONT can be presented with the Turtle notation as follows:

```
#Ontology namespaces
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
```

Figure 4: Smart Greenhouse.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix ac: <http://example.org/actuator/>.

#Class definitions
ac:Actuator rdf:type rdfs:Class.
ac:Led rdfs:subClassOf ac:Actuator.
ac:Fan rdfs:subClassOf ac:Actuator.
ac:WaterPump rdfs:subClassOf ac:Actuator.
ac:NumericRange rdfs:subClassOf ac:Range.
ac:EnumeratedRange rdfs:subClassOf ac:Range.

#Property definitions
ac:status rdfs:domain ac:Actuator;
    rdfs:range ac:NumericRange.
ac:status rdfs:domain ac:Actuator;
    rdfs:range ac:EnumeratedRange.
```

In Smart Greenhouse, the interaction between the user and the smart space is provided by a device called the ubiquitous communicator. The ubiquitous communicator is equipped with an RFID reader and contains a Gardener Interface KP which subscribes to the sensor, plant, and actuator information to display this data for the gardener. It also enables the gardener to modify the state of actuators and provides methods for publishing information about the plants into the SIB.

The miniature greenhouse has two slots for plants. The habitat of these plants can be altered with embedded actuators such as fans, LEDs, and a water pump. The Actuator KP, located in the T-Engine Teaboard2-embedded platform [45], is responsible for controlling these physical actuators. To advertise the functionality it provides, the Actuator KP publishes semantic descriptions about the actuators, as specified in the ACT-ONT, to the SIB. Then it subscribes to the status of these actuators. This way it is aware when the state of the physical actuators needs to be modified. Each of the physical actuators is tagged with an RFID-based ucode tag which is used also as the URI for the given Actuator class instance in the SIB. In addition to providing a natural way to identify physical objects, this approach enables the gardener to modify the semantic data related to the actuators and thus control the physical actuators by touching them with his/her personal device.

Context information about the environmental state such as the humidity, luminosity, and temperature is published by the Sensor KP, located in a Stargate NetBridge sensor

platform. The semantic data published by the Sensor KP conforms the sensor ontology presented in section A. Similar to actuators are the sensors also tagged with RFID-based ucode tags. This enables the gardener to read sensor measurements by touching the physical sensors with the ubiquitous communicator.

In addition to the manual control, it is possible to give the responsibility of the greenhouse to an autonomous agent, called autocontrol KP. The auto-control KP, located on an embedded Linux-based Gumstix verdex platform, utilizes information about the sensor measurements and plant preferences to update new state values for the actuators available in the SIB.

6.3. ECSE: Configuration Tool for Smart Spaces. As the name implies event-based configuration of smart environments, ECSE is a tool that enables users to modify semantic technology empowered smart spaces. The ECSE tool [46] was developed to demonstrate and validate our approach for modifying the behavior of semantic technology-based smart spaces (presented in the Section 5.3). The ECSE tool is implemented with Qt framework which allows porting the same software to various computing platforms. The ECSE tool consists of the following separate modules: rule creator, RDFS browser, event/action creator, and rule processor.

The rule creator module is the main component of the ECSE-tool as it provides interface for the user to create new rules to smart spaces. The module utilizes the semantic descriptions of the events and actions in the SIB to present the functionality provided by the environment. Figure 5 illustrates the rule creator view on Linux laptop. In this figure, the ECSE tool is used in the Smart Greenhouse and the gardener creates a rule where the fans of the greenhouse are turned on when the humidity is above 75% and the temperature is above 30.5 Celsius degree.

The RDFS browser module provides a view to the RDF data in the SIB. When combined with the event/action creator module, it is useful for creating new events and actions, even at system run-time. This way it is possible to utilize information about devices which are not familiar with our approach and do not utilize the EA-ONT presented in Section 5.3. It should be noted, however, that this approach needs an expert who first identifies the events and actions from the RDF graph and then assigns them with appropriate ranges and human readable descriptions.

The rule processor module implements the Rule Handler entity of the system model presented in Section 5.3. To demonstrate the whole approach with a single tool, we decided to implement all the necessary components into the ECSE tool. As previously stated the functionality of the Rule Processor is quite simple. It subscribes to the input condition and when necessary executes the actions by updating RDF data to the SIB. For example, with the rule presented in Figure 5, the rule processor would execute the following WQL subscriptions:

```
WQL(se:HumidityMeasurement, seq(inv(rdf:type), se:hasValue))
WQL(se:TemperatureMeasurement, seq(inv(rdf:type),
se:hasValue)).
```

Figure 5: Rule creator view.

This way the rule processor will be aware when the humidity or the temperature changes and can check whether the input condition matches. Once the input condition evaluates as true, the Rule Processor updates information to the SIB as specified in the RDF triple pattern of the actions. For example, in the case of the rule presented in Figure 5, the Rule Handler executes the following *update* operation:

```
@prefix ac: <http://example.org/actuator/>.
@prefix atag: <ucode:4252:2534:1132:>.
REMOVE(atag:4413 ac:hasStatus "Off".
    atag:3211 ac:hasStatus "Off".)
INSERT(atag:4413 ac:hasStatus "On".
    atag:3211 ac:hasStatus "On".),
```

where "4252:2534:1132:1234" is the ucode of fan 1 and "4252:2534:1132:2345" is the ucode of fan 2. When this data is published to the SIB of the Smart Greenhouse, the Actuator KP will be notified about the new status and will modify the physical actuators accordingly.

6.4. Smart Meeting. The Smart Meeting is social smart space application that demonstrated how our approach for autonomous content sharing can be utilized in semantic technology empowered smart spaces [47]. It consists of various autonomous meeting agents that we have implemented to the following smart phones: Nokia N900, Nokia N97, Apple iPhone, and Google Nexus One. These meeting agents enable users to create and participate to meetings and share their contact information and documents with each other.

Smart Meeting imports and extends the FS-ONT presented in Section 5.4. We have defined new classes for the meeting and participant concepts and introduced new subclasses for the existing *File* class. The *Meeting* class presents necessary information about the available meetings in the smart space. The *Participant* class is a subclass of *foaf:Person* class [48] and it represents the contact information of meeting participants. Figure 6 presents how the meeting KPs in the smart phones share information about meetings, participants, files, and file sharing methods via the SIB.

The functionality of the Smart Meeting is best explained via the following scenario. A number of people are waiting

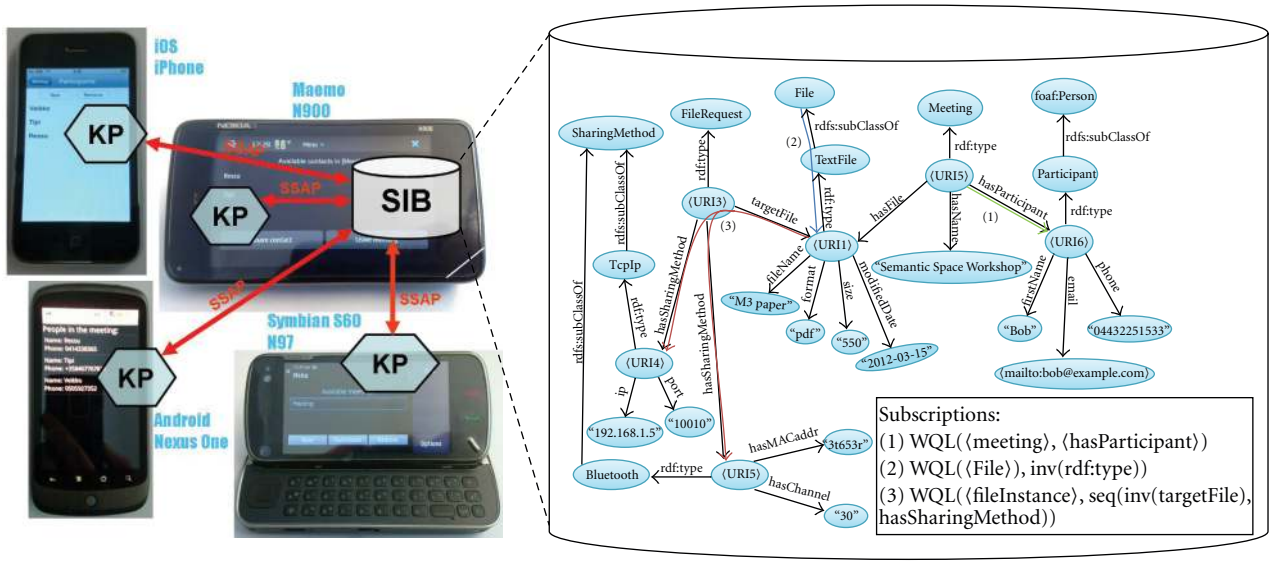

Figure 6: Meeting demonstration set up.

for a workshop on semantic technology empowered smart spaces to begin. One of them is Bob who is going to give a presentation about M3-based semantic interoperability solutions. Before the workshop begins, the session chair commands his meeting agent to set up a smart space and a virtual meeting to the WLAN network of the workshop. Once the smart space and the meeting have been set up, other people whose meeting KPs are connected to the WLAN network of the workshop are informed about the smart space and the meeting. When they join the meeting, their meeting KPs subscribe to the available contact and file information (see subscriptions 1 and 2 in Figure 6).

To advertise his research, Bob requests his meeting KP to publish his contact information and metadata about his scientific paper to the meeting. To do so, the meeting KP inserts the following RDF triples into the SIB:

```
@prefix fi: <http://example.org/fileSharing/>.
@prefix me: <http://example.org/meeting/>.

:bob rdf:type me:participant;
    me:firstName "Bob";
    me:phone "04432251533";
    me:email <mailto:bob@example.com>.
<exampleMeeting> me:hasParticipant:bob.

me:textFile rdfs:subClassOf fi:File.
:file rdf:type me:textFile;
    fi:filename "M3 paper";
    fi:format "pdf";
    fi:size "550";
    fi:modifiedDate "2012-03-15";
<exampleMeeting> me:hasFile:file.
```

Then to be aware when other KPs request the file, Bob's meeting KP subscribes to the sharing methods supported by the requesting KPs as presented in Section 5.4.

When Bob gives his presentation, the chairman takes a picture of him and requests his meeting agent to publish metadata about the picture to the virtual meeting. Soon after Bob's presentation, Mary arrives to the workshop and connects her meeting agent to the smart space. After joining the meeting, Mary's meeting KP subscribes to the available files and informs her about Bob's paper. Mary is interested about the paper and asks her meeting agent to fetch it for her. To request the paper, Mary's meeting KP publishes the following RDF triples into the SIB:

```
@prefix fi: <http://example.org/fileSharing/>.
:request rdf:type fi:fileRequest;
    fi:targetFile <m3paper>;
    fi:hasSharingMethod:bluetooth,:tcp.
:bluetooth rdf:type fi:bluetooth;
    fi:macAddr "3t635r";
    fi:channel "30".
:tcp rdf:type fi:tcp;
    fi:ip "192.168.1.5";
    fi:port "10010",
```

where *m3paper* is the URI of Bob's paper. The SIB notifies Bob's meeting KP about the request made by Mary's meeting KP (see subscription 3 in Figure 6). Similar to Mary's meeting KP, Bob's meeting KP supports both Bluetooth and TCP/IP-based communication and decides to save power by using Bluetooth for transferring the file. After Mary has read the paper, she would be interested to talk with the author of the paper about some interesting issues, but she does not know who the author is. She decides to ask help from her meeting KP who is fortunately able to fetch Bob's contact information and picture based on the linked semantic metadata in the SIB.

7. Conclusions

In this paper, we have described various approaches to support the exploitation of semantic technologies in context-aware smart space applications. When combined, the approaches presented in the paper are very important because they enable the creation of pervasive computing systems which can take full advantage of the semantic technologies. The developed approaches have been built according to the principles of M3 concept and validated with prototype smart space applications and tools.

First, we described approaches and methods for tackling the challenges of implementing a semantic technology empowered smart space agent in a resource-restricted low-power platform. The approach was validated by implementing autonomous agent called Active Tag with total code size of 39.7 kB and RAM consumption of 25.3 kB. Second, we described a uID-based method usable for identifying physical objects and locating information related to these objects from all over the world. This approach was demonstrated both in the Smart Home Garden, and Smart Greenhouse smart spaces. Third, we presented a novel approach that allows the end-users to modify the functionality provided by smart spaces. The main aspect of the approach is that it provides the user with natural human language description of the smart space capabilities so that the user is able to define what kind of functionality is required from the smart space. The approach was realized in practice by implementing a general-purpose ECSE tool with Qt-framework. Fourth, we described how content such as video, audio, and documents that are not feasible to be stored in the semantic database as such can be shared between the semantic technology empowered autonomous agents. The Smart Meeting application demonstrated this approach in practice.

By utilizing the approaches and methods presented in this paper, it will be possible to create context-aware applications for semantic technology empowered smart spaces. However, there is still a lot of work to be done before semantic technology-based context presentation can be utilized in large-scale real-life smart spaces. For example, security and privacy issues are important, and although the RIBS already provides some level of security there is still a lot to be done in this field. Other interesting future research areas in semantic technology empowered smart spaces include, for example, data maintenance and ontology governance.

Acknowledgment

This work has been funded by the Device and Interoperable Ecosystem (DIEM) and the Merging IoT Technologies (MIoTE) Projects.

References

[1] A. K. Dey, *Providing architectural support for building context-aware applications [Ph.D. thesis]*, Georgia Institute of Technology, 2000.

[2] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 94–100, 1991.

[3] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century," *Computer*, vol. 36, no. 3, pp. 25–31, 2003.

[4] E. Aarts, H. Harwing, and M. Schuurmans, "Ambient intelligence," in *The Invisible Future*, J. Denning, Ed., pp. 235–250, McGraw Hill, New York, NY, USA, 2001.

[5] N. Gershenfeld, "The internet of things," *Scientific American*, vol. 291, no. 4, 2004.

[6] J. R. Cooperstock, K. Tanikoshi, G. Beirne, T. Narine, and W. Buxton, "Evolution of a reactive environment," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '95)*, pp. 170–177, Denver, Colo, USA, May 1995.

[7] M. Dertouzos, *The Future of Computing*, Scientific American, 1999.

[8] B. Brumitt, B. Myers, J. Krum, A. Kern, and S. Shafer, "Easy-Living: technologies for intelligent environments," in *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC '00)*, vol. 1927 of *Lecture Notes in Computer Science*, pp. 12–29, Springer, 2000.

[9] T. Kindberg, J. Barton, J. Morgan et al., "People, places, things: web presence for the real world," in *Proceedings of the 3rd IEEE Workshop Mobile Computing Systems and Applications (WMCSA '00)*, p. 19, IEEE CS Press, 2000.

[10] B. Johanson, A. Fox, and T. Winograd, "The interactive workspaces project: experiences with ubiquitous computing rooms," *IEEE Pervasive Computing*, vol. 1, no. 2, pp. 67–74, 2002.

[11] Contributing Members of UPnP Forum, "UPnP device architecture 1. 1," http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf.

[12] OSGi Alliance web page, "OSGi—the dynamic module system for Java," http://www.osgi.org/.

[13] M. Gudgin, M. Hadley, N. Mendelsohn et al., *SOAP Version 1. 2 Part 1: Messaging Framework*, W3C Recommendation, 2nd edition, 2007.

[14] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," in *Proceedings of the International Conference on Software Engineering (ICSE '00)*, pp. 407–416, June 2000.

[15] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[16] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data—the story so far," *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1–22, 2009.

[17] H. Chen, *An intelligent broker architecture for pervasive and context-aware systems [doctoral dissertation]*, University of Maryland, Department of Computer Science and Electrical Engineering, Baltimore County, Md, USA, 2004.

[18] X. Wang, J. S. Dong, C. Chin, S. R. Hettiarachchi, and D. Zhang, "Semantic space: an infrastructure for smart spaces," *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 32–39, 2004.

[19] G. Klyne and J. J. Carroll, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C Recommendation, 2004.

[20] D. Brickley and R. V. Guha, *RDF Vocabulary Description Language 1. 0: RDF Schema*, W3C Recommendation, 2004.

[21] W3C OWL Working Group, *OWL 2 Web Ontology Language Document Overview*, W3C Recommendation, 2009.

[22] R. Masuoka, B. Parsia, and Y. Labrou, "Task computing—the semantic web meets pervasive computing," in *Proceedings of the 2nd International Semantic Web Conference (ISWC '03)*, pp. 886–881, 2003.

[23] S. Ben Mokhtar, N. Georgantas, and V. Issarny, "COCOA: conversation-based service composition in pervasive computing environments with QoS support," *Journal of Systems and Software*, vol. 80, no. 12, pp. 1941–1955, 2007.

[24] Z. Song, A. A. Cárdenas, and R. Masuoka, "Semantic middleware for the internet of things," in *Proceedings of the 2nd International Internet of Things Conference (IoT '10)*, pp. 1–8, December 2010.

[25] G. Thomson, S. Bianco, S. Mokhtar, N. Georgantas, and V. Issarny, "Amigo aware services," in *Communications in Computer and Information Science*, vol. 11, pp. 385–390, 2008.

[26] P. Liuha, J. Soininen, and R. Otaolea, "SOFIA: opening embedded information for smart applications," in *Proceedings of the Embedded World Conference*, Nuremberg, Germany, March 2010.

[27] M. Compton, P. Barnaghi, L. Bermudez et al., "The SSN ontology of the W3C semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 17, pp. 25–32, 2012.

[28] H. Chen, F. Perich, T. Finin, and A. Joshi, "SOUPA: standard ontology for ubiquitous and pervasive applications," in *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS '04)*, pp. 258–267, August 2004.

[29] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *Proceedings of the 15th IEEE Symposium on Computers and Communications (ISCC '10)*, pp. 1041–1046, Riccione, Italy, June 2010.

[30] J. Suomalainen, P. Hyttinen, and P. Tarvainen, "Secure information sharing between heterogeneous embedded devices," in *Proceedings of the 4th European Conference on Software Architecture: Doctoral Symposium, Industrial Track and Workshops (ECSA '10)*, pp. 205–212, August 2010.

[31] A. Lappeteläinen, J. Tuupola, A. Palin, and T. Eriksson, "Networked systems, services and information—the ultimate digital convergence," in *Proceedings of the 1st International Conference on Network on Terminal Architecture (NoTA '08)*, pp. 1–7, 2008.

[32] D. Manzaroli, L. Roffia, T. S. Cinotti et al., "Smart-M3 and OSGi: the interoperability platform," in *Proceedings of the International Workshop on Semantic Interoperability for Smart Spaces (SISS '10)*, pp. 1053–1058, IEEE Press, June 2010.

[33] O. Lassila, *Programming semantic web applications: a synthesis of knowledge representation and semi-structured data [doctoral dissertation]*, Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Software Technology, 2007.

[34] E. Prud'hommeaux and A. Seaborne, *SPARQL Query Language For RDF*, W3C Recommendation, 2008.

[35] The OWL Services Coalition, "OWL-S: semantic markup for web services," 2003, http://www.daml.org/services/owl-s/1.0/owl-s.html.

[36] T. Berners-Lee, R. Fielding, and L. Masinter, *Uniform Resource Identifiers (URI): Generic Syntax*, Internet Draft Standard RFC, 2396, IETF, 1998.

[37] M. Duerst and M. Suignard, *Internationalized Resource Identifiers (IRIs)*, Proposed Standard RFC, 3987, IETF, 2005.

[38] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *Knowledge Engineering Review*, vol. 18, no. 3, pp. 197–207, 2003.

[39] K. Hansen, W. Zang, J. Fernandes, and M. Ingstrup, "Semantic web ontologies for ambient intelligence," in *Proceedings of the 1st International Research Workshop on The Internet of Things and Services*, 2008.

[40] N. Koshizuka and K. Sakamura, "Ubiquitous ID: standards for ubiquitous computing and the internet of things," *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 98–101, 2010.

[41] D. Beckett and T. Berners-Lee, *Turtle—Terse RDF Triple Language*, W3C Team Submission, 2011.

[42] A. Ylisaukko-oja, P. Hyttinen, J. Kiljander, J. Soininen, and E. Viljamaa, "Semantic inteface for resource restricted wireless sensors," in *Proceedings of the IC3K 2nd International Workshop on Semantic Sensor Web (SSW '11)*, Paris, France, October, 2011.

[43] J. Kiljander, M. Eteläperä, J. Takalo-Mattila, and J. P. Soininen, "Opening information of low capacity embedded systems for Smart Spaces," in *Proceedings of the 8th IEEE Workshop on Intelligent Solutions in Embedded Systems (WISES '10)*, pp. 23–28, July 2010.

[44] J. Takalo-Mattila, J. Kiljander, M. Eteläperä, and J. Soininen, "Ubiquitous computing by utilizing semantic interoperability with item-level object identification," in *Proceedings of the 2nd International ICST Conference on Mobile Networks and Management (MONAMI '10)*, vol. 68, pp. 198–209, Springer, 2010.

[45] K. Sakamura and N. Koshizuka, "T-engine: the open, real-time embedded-systems platform," *IEEE Micro*, vol. 22, no. 6, pp. 48–57, 2002.

[46] J. Kiljander, J. Takalo-Mattila, M. Eteläperä, K. Keinänen, and J. Soininen, "Enabling end-users to configure smart environments," in *Proceedings of the International Symposium on Applications and the Internet (SAINT '11)*, pp. 303–308, 2011.

[47] J. Kiljander, M. Eteläperä, J. Takalo-Mattila, K. Keinänen, and J. Soininen, "Autonomous file sharing for smart environments," in *Proceedings of the International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS '11)*, pp. 191–196, 2011.

[48] D. Brickley and L. Miller, *FOAF Vocabulary Specification 0. 98*, Namespace Document, 2010.