Sharing Building Information with Smart-M3

Kary Främling
Aalto University
PO Box 15500, Espoo, Finland
e-mail: Kary.Framling@hut.fi

André Kaustell
Åbo Akademi University
Joukahaisenkatu 3-5, FIN-20500 Turku, Finland
e-mail: andre.kaustell@abo.fi

Ian Oliver
Nokia Mobile Solutions - Platforms
Helsinki, Finland.
e-mail: Ian.Oliver@nokia.com

Jan Nyman
Electrical Building Services Centre
Posintra Oy
Kipinätie 1, FIN-06150 Porvoo, Finland.
e-mail: jan.nyman@posintra.fi

Jukka Honkola
Nokia Research
Helsinki, Finland.
e-mail: Jukka.Honkola@nokia.com

***Abstract*— Semantic nets are a universal information structure that can be used for representing nearly any kind of information. This is why the semantic web has also chosen to use them as the universal format for representing data, usually using RDF (Resource Description Framework) as the syntax. Semantic nets are also suitable for sharing information between different domains, organizations, manufacturers etc. In this paper, we describe how a semantic net and agent-based shared storage called Smart-M3 has been implemented and can be used for such information sharing. The particular application domain studied is building automation, where interoperability between equipment made by different manufacturers is rare. This is a great challenge for implementing "ubiquitously smart buildings", where building automation systems, user interfaces and services could interact. The paper describes how the Smart-M3 concept can be used as an enabler of interoperability, where an ecosystem of supplementary services is created through manufacturer-agnostic agents.**

Keywords - Smart Buildings, Smart-M3, semantic net, ontologies, software agents.

I. INTRODUCTION

Creating smart buildings and smart environments in general has been a topic of research and development for a long time. However, such environments are still largely found only in experimental or pilot environments despite their potential to make people's lives easier, reduce energy consumption and environmental footprint, as well as improve the quality of life in general. In this paper, we describe a distributed information architecture that makes it possible to implement such smart environments on a large scale by integrating information access to and control of different building automation systems. We also show how smart buildings can be created as parts of smart environments.

Building automation is a domain where interoperability is a challenge due to conflicting interface and communication standards, e.g. KNX, LON, Modbus etc., in addition to a great number of prorietary solutions. Solutions to these interoperability challenges have been developed e.g. in the ongoing DIEM project (Devices and Interoperability Ecosystem, http://www.diem.fi) using device and protocol adapters that enable unified information access to them all on the Internet Protocol level and notably through Service Oriented Architecture (SOA) solutions. Such SOA-based solutions are good in the case where standards (real or de-facto) exist for the semantic representation of the information. In practice, there is a lack of universal standards. Meanwhile there tends to be many potential interfaces available developed by different organisations and projects, which are not interoperable. This lack of compatibility is a major obstacle for creating *Smart Spaces* where humans and devices could interact smoothly [1].

The Smart Spaces notation is heavily overloaded and has been used for describing a wide variety of things. In this paper, we use it to signify a geographical space where information is available about the space itself, the devices and services available in it, the people present in it and about other potentially useful information or services. Such a Smart Space concept has been initially proposed in [2] as a solution to enabling interoperability. As no standards exist that would cover the information representation needs of such Smart Spaces, we believe an incremental process will occur [3]. In the first phase, devices and systems will publish their available information and services using their current semantic notations (standardized or not). When the information becomes available, that makes it possible to

create new services that use the information, while augmenting it with information about the services themselves and information produced by them.

In the Sedvice/M3 architecture, information is expressed as subject-relation-object Triples that build up labeled, directed multi-graphs (one or more). In the rest of this paper we will call such graphs *semantic nets* even though graph theory and semantic net theory use partially different vocabularies and present other potential incompatibilities due to their background and history. The Triples are represented using Resource Description Framework (RDF) notations. Triples are stored and managed by the Semantic Information Broker (SIB), which can be distributed over many devices. The Smart Space Access Protocol (SSAP) is used for performing operations on the semantic net.

After this introduction, the paper gives a state of the art overview of building automation systems, semantic nets and Smart Spaces. In Section III we describe the whole system architecture and in Section IV we show the current level of implementation, followed by conclusions.

II. BACKGROUND

Building automation systems and Smart Spaces are currently two distinct domains with different technological and scientific backgrounds, which is the reason for splitting this section. We will provide an overview of the state-of-the-art for both domains, as well as some background for the work reported in this paper.

A. Building Automation Systems

Systems integration in buildings has traditionally been about physical dimensions, voltage, plug dimensions etc. Control mechanisms usually control either one device only (e.g. a lamp, a refrigerator etc.) or power supply for security reasons (e.g. fuses, main switch etc.). Implementing integrated functions such as switching the power off from certain appliances, cutting off water supply and activating the burglar alarm with one single "leaving home" command has required a lot of dedicated cabling and custom devices, installed by professionals.

Different communication standards have been defined in order to provide more feasible solutions, such as LON (http://www.lonmark.org/), KNX (http://www.knx.org/) and ModBus (http://www.modbus.org/). However, none of these has become a global standard that all manufacturers would support. Many solutions based on these protocols also tend to be expensive to install, maintain and upgrade. Furthermore, they are not conceived in a way that would allow for easy integration between them; in fact, they may even on purpose be designed in a way that makes interoperability more difficult due to commercial reasons.

Meanwhile, remote monitoring and control of buildings has become a common functionality at least for bigger buildings such as shopping centers, office buildings, libraries etc. Remote monitoring services are becoming an increasingly important part of the business of traditional building companies as well as other companies. These systems tend to use internet as the information channel because it is cheap to set up and use. As people become increasingly connected to the internet from their homes, internet and the communication protocols associated with it have become an interesting option also for building automation solutions. The fact that many multimedia devices (including mobile phones) integrate internet connectivity by default, makes it possible to take systems integration and usability to levels that are not possible with "classical" building automation systems.

Figure 1 illustrates how different devices can be connected to a "protocol converter" that makes device information available through internet protocols. The "protocol converter" can be an ordinary computer or a cheaper and more energy-efficient solution, such as the Home Control Center (http://smarthomepartnering.com/cms/) initially proposed by Nokia. Device connectivity is implemented through adapters that convert the underlying protocols into a generic internet interface.

In practice, a generic internet interface nowadays signifies a browser-compatible format (HTML and others) for user interfaces and XML messages for machine-readable information. For successful machine-to-machine communication, the semantics of the XML messages have to be understood in the same way by both parties. The currently most used method for describing message semantics is XML Schemas. In building automation, the oBIX (Open Building Information Xchange, http://www.obix.org/) is an example of such a protocol. Devices Profile for Web Services (DPWS) is another initiative with similar goals. In addition to these, more generic messaging protocols exist that are intended for communication with any kind of devices (not only related to building automation). The PROMISE Messaging Interface (PMI) [4] is an example of such an interface. The IP for Smart Objects (IPSO) alliance (www.ipso-alliance.org) has similar objectives but it is unclear whether they have yet specified any messaging protocols. In practice, none of these has obtained global acceptance.

There are still technical and functional differences between the protocols. Currently, oBIX and PMI are the easiest protocols to compare because their specifications are readily available. For the moment, oBIX does not support real-time events due to the lack of callback functionality, which is included in PMI. On the other hand, oBIX is more well-known. oBIX is also REST-compliant [5], whereas PMI currently lacks the functionality of accessing resources (e.g. devices, sensor values etc.) directly via a URL. The lacking features would be easy to add both for oBIX and PMI but for the moment especially the lacking features in oBIX make it hard or impossible to implement some functionality that is essential in real-life applications. As a conclusion, no universal standard currently exists for representing information neither about smart objects in general, nor about building automation systems.

Figure 1. Connecting devices to the Internet.

B. Semantic nets for describing Thing-related information

Buildings are per definition complex products for which an extensive amount of documentation is produced both during the design and the manufacturing phases. CAD models and other technical information produced during the design phase are some of the most essential parts of the product information available when the building is taken into use. CAD models are a form of semantic networks that explicitly model "part-of", "depends-on" and similar relationships. The bill-of-materials (BOM) used during manufacturing is another important piece of product information that can be represented as a semantic network. However, the BOM is usually representing product information on a product-type or product-variant level rather than on a product-item level. The classical BOM is not sufficient for managing building product information, where each building is an individual product-item, where even the parts of the building have individual properties and where parts can be changed during the product lifecycle [6].

Semantic networks represent sets of named relationships between different nodes (or objects) in a network. Using a collection of pair-wise relations between nodes, where every relation may also have an associated "strength", can represent a semantic network. Relation strengths are particularly useful when semantic networks are used for reasoning, e.g. for diagnostic or prognostic purposes as those needed in many middle-of-life (MOL), i.e. in-usage applications of product items [7].

Solutions for managing semantic networks in a multi-organizational context are being developed under the name "semantic web". RDF and Web Ontology Language (OWL) are examples of standards being developed for the semantic web. Software frameworks also exist that use these standards, e.g. Jena (http://jena.sourceforge.net/), OpenRDF (http://openrdf.org/) and the Redland RDF Application Framework (http://librdf.org/).

However, RDF and OWL are mainly focused on describing web content rather than on describing product information. Furthermore, the related software tools are not, as such, designed to be used for implementing distributed applications. Therefore agent frameworks could be more suitable for this purpose. Examples of such agent frameworks are ABLE (http://www.alphaworks.ibm.com/tech/able) and JADE (\http://jade.tilab.com/) that integrate inter-organizational communication. In a multi-agent framework, agent references correspond to links between nodes of a semantic network. Therefore agent frameworks could be used as building blocks for a distributed implementation of semantic networks for describing product information.

When using an agent framework with support for data analysis and decision support, the nodes themselves can also be "intelligent". Especially the data analysis methods included in the ABLE framework could be applicable as decision support systems. ABLE data analysis and decision support agents provide support for many different data analysis methods, e.g. naïve Bayes, decision trees and neural networks. In addition to these, ABLE agents exist for both crisp and fuzzy rules that are useful for explicitly expressing expert knowledge. This portfolio of methods is particularly interesting for MOL scenarios that include diagnostics, prognostics and condition-based maintenance. ABLE agents can be trained both on- and offline and included in different software components to perform filtering or decision-making on different levels.

The Design Pattern [8] concept developed in the context of Object-Oriented Programming is an example of how object relationships, which are conceptually quite identical to semantic relations, can be combined with processing in well-documented ways that are known to be "good" from a program design point of view. In the DIALOG software platform [9], semantic relations have been used as a way of storing product-related information structures, while agents implemented the needed information processing in order to apply some major Design Patterns to the domain of product lifecycle information management [10].

DIALOG has been used for real-life applications in many application domains, such as shipment tracking (inventory management, detecting delays, project management), building automation (intelligent refrigerator, remote monitoring, condition-based maintenance), automotive (online tracking and collection of sensor and other data, condition-based maintenance) and telecommunications (configuration management). There are also ongoing projects in the same and new domains. Semantic nets have proven their value for storing Thing-related information, while the agent concept is a flexible and efficient paradigm for the processing of that information. However, DIALOG is implemented using "traditional" database and networking technologies, which are not initially conceived for semantic net and agent-based information processing. Smart-M3 is a paradigm and platform developed to overcome those limitations.

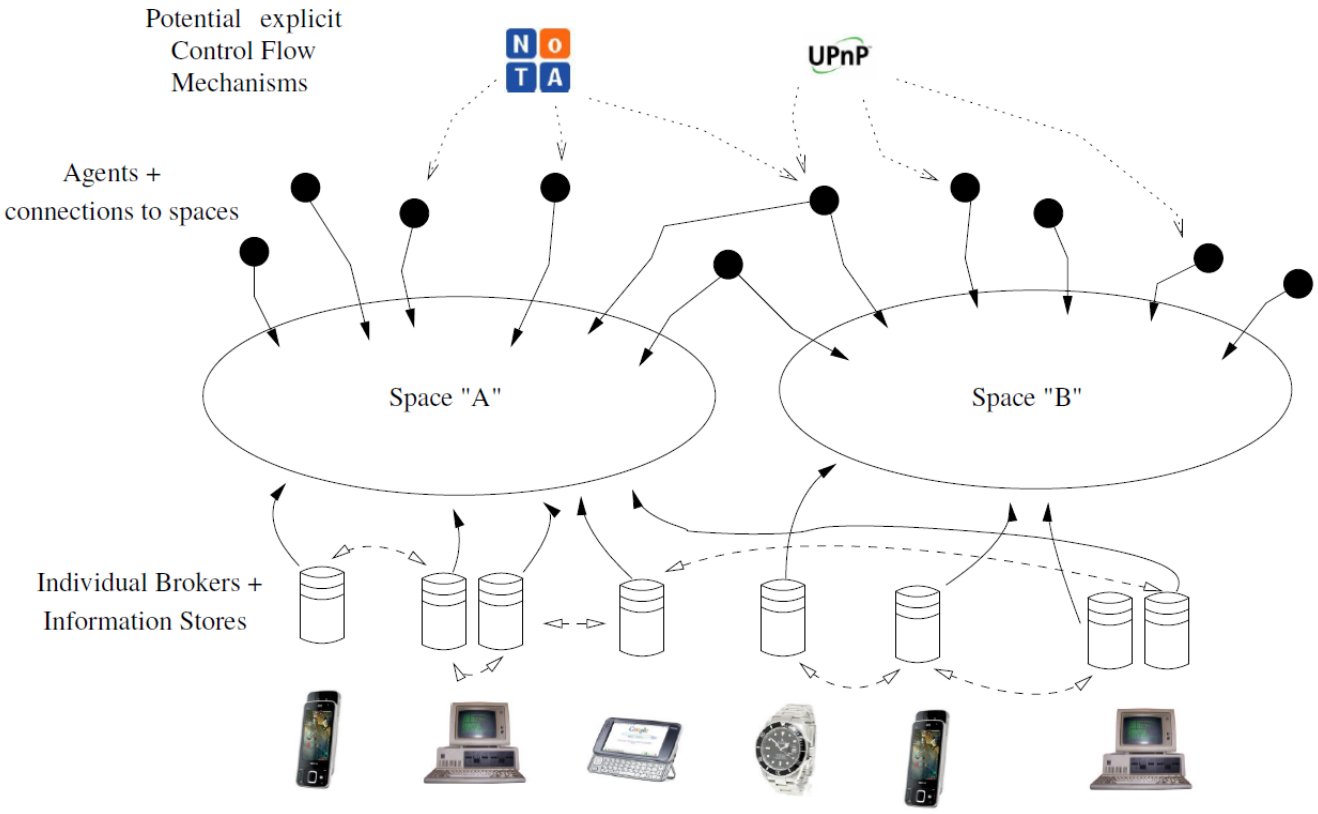

Figure 2. High-level system architecture and connectivity to external systems.

C. *Semantic Net and Agent-based information processing with Smart-M3*

The Smart-M3 system [2][11] consists of a space based communication mechanism [12][13] for independent agents. The agents communicate implicitly by inserting information to the space and querying the information in the space. The space is represented by one or more Semantic Information Brokers (SIBs), which store the information as an RDF graph. The agents can access the space by connecting to any of the SIBs making up the space by whatever connectivity mechanisms the SIBs offer. Usually, the connection will be over some network, and the agents will be running on various devices. The information in the space is the union of the information contained in the participating SIBs. Thus, the agent sees the same information content regardless of the SIB to which it is connected. The high-level system architecture is shown in Figure 2, which includes the distribution routing between SIBs and external interfaces to protocols such as NoTA and UPnP from the agents.

The agents may use five different operations to access the information stored in the space:

Insert: Insert information in the space
Remove: Remove information from the space
Update: Atomically update the information, i.e. a combination of insert and remove executed atomically
Query: Query for information in the space
Subscribe: Set up a persistent query in the space; changes to the query results are reported to the subscriber

In addition to these access operations there are *Join* and *Leave* operations. An agent must have *joined* the space in order to access the information in the space. The join and leave operations can thus be used to provide access control and encrypted sessions, though the exact mechanisms for these are still undefined.

In its basic form the M3 space does not restrict the structure or semantics of the information in any way. Thus, we do not enforce nor guarantee adherence to any specific ontologies, neither do we provide any complex reasoning[1] [14][15]. Furthermore, information consistency is not guaranteed. The agents accessing the space are free to interpret the information in whatever way they want.

We are planning to provide, though, a mechanism to attach agents directly to the SIBs. These agents have a more powerful interface to access the information and can be e.g. guaranteed exclusive access to the information for series of operations. Such agents may perform more complex reasoning, for example ontology repair or translation between different ontologies. However, they may not join any other spaces but are fixed to a single SIB and thus a single space.

[1] The current implementation of the concept understands the owl:sameAs concept

The M3 spaces are of local and dynamic nature, in contrast to semantic web which embodies Tim Berners-Lee's idea of semantic web [16] as a "giant global graph". We envision that the spaces will store very dynamic context information, which poses different challenges than the internet-wide semantic web. For example, in order to provide a true interoperability for local ubiquitous agents, the space (i.e. SIBs) will have to provide a multitude of connectivity options in addition to http: plain TCP/IP, NoTA [17], Bluetooth, RFID [18]. Furthermore, the space should be fairly responsive. While we do not aim for real-time or near real-time systems, we think response times need to remain within seconds in order to be acceptable.

The responsiveness is one of the factors behind the fundamental decision to not enforce any specific ontologies and allowing the agents to interpret the information freely, as it lessens the computational burden of the infrastructure. Another, and more important reason is that we explicitly want to allow mashing up information from different domains in whatever way the agents see best. Strict ontology enforcement would make this kind of activity extremely difficult as all new ways of mashing up the information would require approval from some ontology governance committee. However, we still plan to provide means for ontology enforcement for cases where the space provider explicitly wishes to restrict the ways the information is. Such situations will occur in reality where such enforcement is the best approach.

The information content in a M3 space may be distributed over several SIBs. The distribution mechanism assumes that the set of SIBs forming a M3 space are totally routable but not necessarily totally connected. The information content that the agents see is the same regardless of the SIB where they are connected [19]. Distribution may also occur between first order space interaction as described in [20].

Security is provided firstly as an effect of the localised nature of spaces coupled with the agent-join mechanisms. Within the space there is need for a more sophisticated policy mechanism to regulate access, update and the trust of the information at both individual triple and larger RDF graph structure levels [21].

D. Applications in M3 Spaces

The notion of application in M3 space differs radically from the traditional notion of a monolithic application. Rather, as a long term vision, we see the applications as possible scenarios which are enabled by certain sets of agents [22][23][24]. Thus, we do not see an email application running in M3 space, but we could have a collection of distributed agents present which allow for sending, receiving, composing and reading email. Figure 3 pictorially depicts the relationship between the user, her agents and, in this case, one space, while Figure 4 shows the user (via agents) interacting with many spaces.

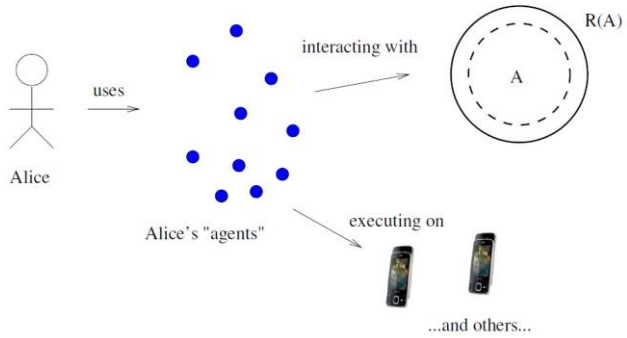

Figure 3. A User's Agents, Devices, Spaces and Information.

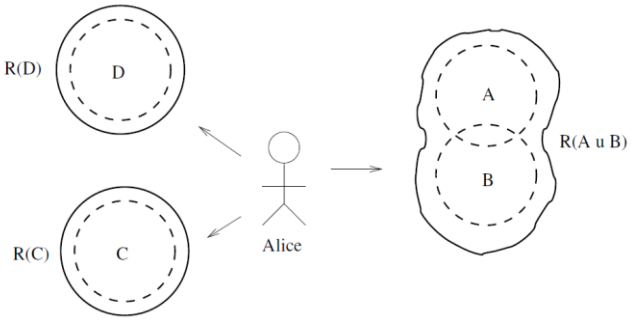

Figure 4. A User and Multiple Spaces

For this kind of scenario based notion of application, we also would like to know whether the available agents can successfully execute the scenario. The envisioned model of using this system is that the user has a set of agents which are capable of executing certain scenarios. If a user needs to perform a new scenario that the current set of agents are not capable of executing, she could go and find a suitable agent from some directory by describing the desired scenario and the agents she already has.

Thus, we need some formal or semi-formal way of describing agent behavior both with respect to the M3 space and to the environment. While there exists research addressing behavior in multi-agent systems, for example by Herlea, Jonker, Treur and Wijngaards [25], this kind of ad-hoc assembly of agents in order to execute a certain scenario seems to be quite unaddressed in current research. However, slightly similar problems have been addressed in e.g. web service orchestration research [26], but these still seem to concentrate on design-time analysis rather than run-time analysis. As for shorter term, our vision is that sets of existing applications would be enhanced by being able to interoperate and thus allow execution of (automatic) scenarios that would have been impossible or required extensive work to implement without the M3 approach.

III. BULDING SPACE AND SERVICES IT CAN PROVIDE

Despite the lack of universally accepted ontologies for representing information related to buildings and the systems found in them, the application domain still presents some advantages [27]. It is possible to identify a common name for some key concepts, such as "temperature" and "humidity".

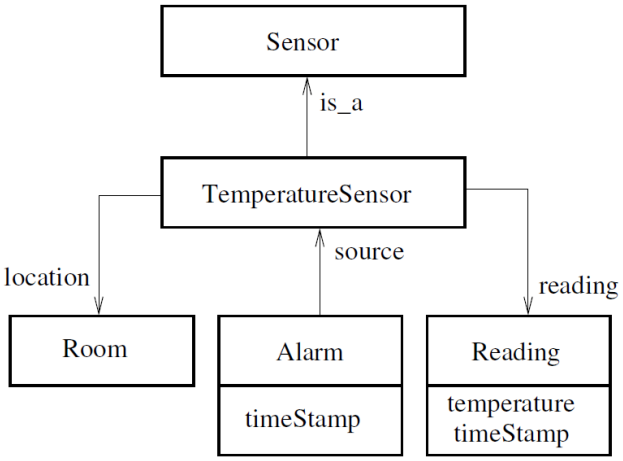

Figure 5. Example of an Ontology (Schema).

When it comes to the CO_2 level it already becomes more difficult to agree on a common name. There may also be several different sensors of the same type. For instance, a ventilation machine with heat recovery would normally have at least four temperature sensors that need to be named. Still, the number of manufacturers of such machines is typically not too big (less than ten in Finland) so even though all manufacturers would choose their own names for those sensors, it could still remain manageable. We also believe that as equipment manufacturers start publishing information in the smart space and there are services built upon that information, there may also be more incentive for the manufacturers to start using common ontologies such as the simple example in Figure 5 written in a UML or Entity-Relationship style notation. Another approach could be to use "forced semantics" [28] implemented by "translation agents" that would automatically translate information from one ontology to another [29].

Figure 6 shows an example of a small semantic net [30] expressed as an RDF graph for representing sensor values from three different temperature sensors, of which two are located in the same room. This graph satisfies the ontology given in Figure 5, where the reader should be able to figure out the names of relationships (other than object typing) which are not shown for clarity.

This limited net also illustrates some basic processing needs, implemented by agents. In Figure 6, sensor *ts3* has produced three temperature readings, which is the beginning of a reading history. With an increasing number of sensors that may store historical information in the space, it becomes necessary to at least implement *cleaning agents* who take care of removing expired information or removing the "least useful" information if memory is filling up. To avoid losing too much information when cleaning, *summarization agents* become essential. Summarization agents will keep track of minimum, maximum, running average etc. values even after the cleaning agents have removed the original values.

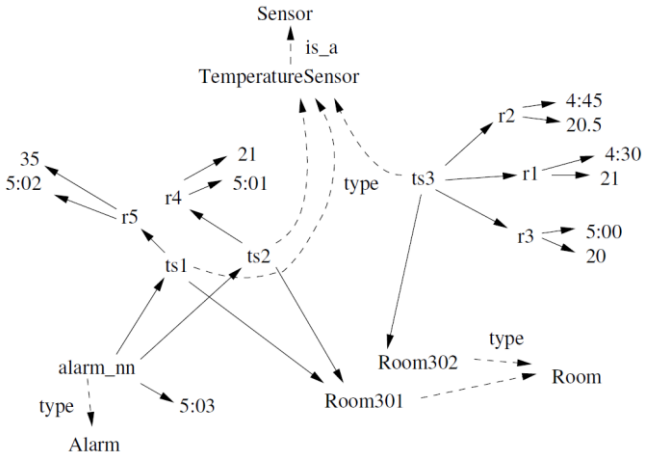

Figure 6. Example of partial semantic net for a home with several temperature sensors.

Using Figure 6 we can write queries across this graph to obtain readings such as those described above. We nominally use here a graph traversal language such as WQL [31] or XPath - M3 specifically supports WQL at this time and a SPARQL parser is being developed.

Given a specific temperature sensor (*ts2*), the query to obtain the current temperature would take the pseudo-code form:

```
ts2 | readings.filter(
      latest(timestamp).
      temperature
```

Given a specfic room, the average temperature would take the form:

```
Room | ( location-1.readings.
       temperature.asBag() )
       / size(location-1.readings)
```

where the suffix -1 denotes inverse traversal of a link and the functions `latest()`, `asBag()`, and `size()` take their common sense meanings when working with sets (or bags) of values.

Figure 6 also shows the alarm event *alarm_nn* in the space as an example of how to handle anomaly detection. A sensor consistency check agent notices an abnormally great value difference for sensors *ts1* and *ts2* that are in the same room. The presence of a new alarm event can be detected by a user notification agent that takes care of notifying the user about the situation. The user can then take the appropriate action, after which the alarm event is removed manually or automatically when the anomaly is no longer present.

It is quite easy to imagine a great number of other functionality that agents could implement based on the information in the space. Such functionality could be deciding on the target temperature in a room based on some "voting" rule, automatically telling the coffee and tea making machines how many people prefer which one, automatic agreement of the next appointment based on the calendar information available from participants in a meeting etc. However, the implementation of such functionality is more

complex than the anomaly detection described previously. It is also much less certain to what degree users want to have or even accept such functionality. However, the purpose of this paper is not to claim that some specific service or functionality is useful as such, the objective is rather to show that Smart-M3 significantly simplify the creation of such services.

Finally, Smart Spaces as described here are not constrained to buildings. They can also cover greater geographical areas and be dedicated for other application domains. One example of such a domain would be publishing weather information collected from private weather stations, ventilation machines etc. for improved local weather forecasts, thereby improved control of heating and cooling in buildings and, as a consequence, improved energy-efficiency as a whole.

IV. IMPLEMENTATION

The Electrical Building Services Centre of Posintra Oy in Porvoo, Finland, develops demonstration solutions as well as commercially applicable components for integrated building automation. The developed systems should be low-cost, easy to use and easy to integrate with existing and new building automation systems. Current systems installed both in the Centre's demonstration facility and in real buildings make it possible to bridge between the Internet, and a number of building automation protocols and proprietary device protocols. The software platform is based upon OpenWRT (http://www.openwrt.org), a Linux software distribution for embedded systems. The platform makes it possible to communicate with proprietary building automation protocols, and translate the messages to a common format, namely oBIX. This makes it possible to network the devices together, which previously were isolated from each other because of the lack of an universal protocol. The platform itself is running on inexpensive consumer grade hardware (a wireless router with Universal Serial Bus). Currently supported systems include real-time energy metering, data collection and control of air handling units, a control unit for electrical systems of small buildings, wireless power outlets, a consumer-end weather station etc.

The difference of this approach compared to earlier attempts to create a common building automation protocol, is that we have a protocol-agnostic approach. Any building automation protocol can be integrated to the platform by means of adapter software and hardware, and thus we can enable any protocol for Internet connectivity. By connecting together various building automation protocols, we make it possible to combine the functionalities of various subsystems, and create new services that would not be possible without seamless integration of the subsystems. Currently, the subsystems are combined together by the oBIX protocol, which makes it possible to build hierarchal systems by interconnecting the devices on a local level and export the combined information to upper-level systems via oBIX as illustrated in Figure 7.

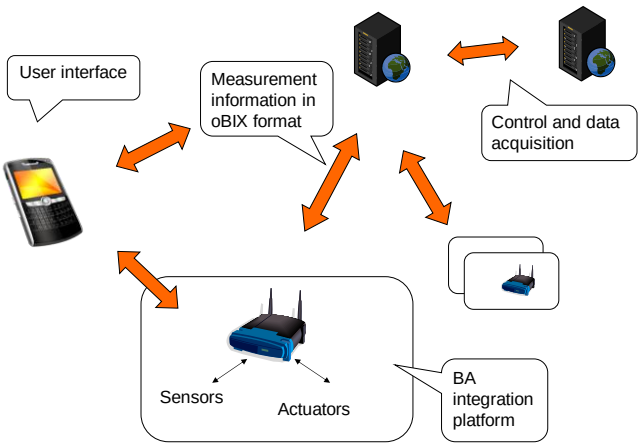

Figure 7. Integration platform makes it possible to control and acquire data from building automation systems, and export the data to back-end systems.

Converting data from proprietary protocols to oBIX simplifies the creation of traditional Supervisory Control and Data Acquistion (SCADA) applications, because the SCADA application now needs to understand only one protocol, instead of a myriad of protocols.

However, optimal control of a building's automation system also needs information from other sources than the various systems located in the building. A simple example is to use weather forecast information from a meteorological website so that the control system can decide to start heating the house during the night (when electricity is cheaper) if the weather forecast says the next day will be colder. Including this type of information from outside the domain of building automation is difficult, if we have to use a building automation specific data format like oBIX.

The Smart-M3 architecture is being integrated to the demonstration platform to make it possible to combine information from various data sources, and to do automated reasoning over it as illustrated in Figure 8. Reasoning agents might change over time, or be only temporarily available, e.g. if they are located in a visitor's mobile phone, PDA or similar.

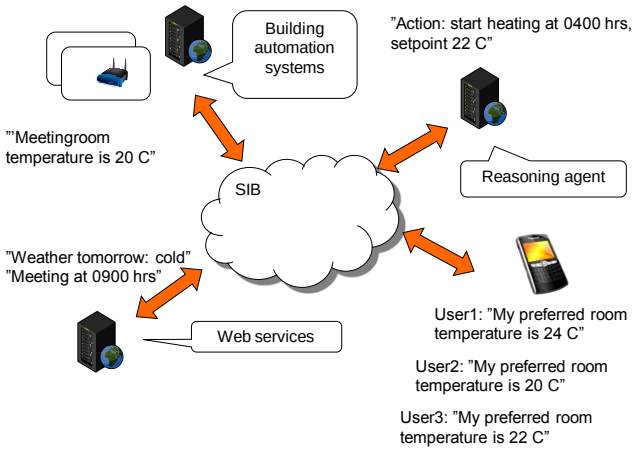

Figure 8. The SIB is an implementation of a data store supporting reasoning over cross-domain information.

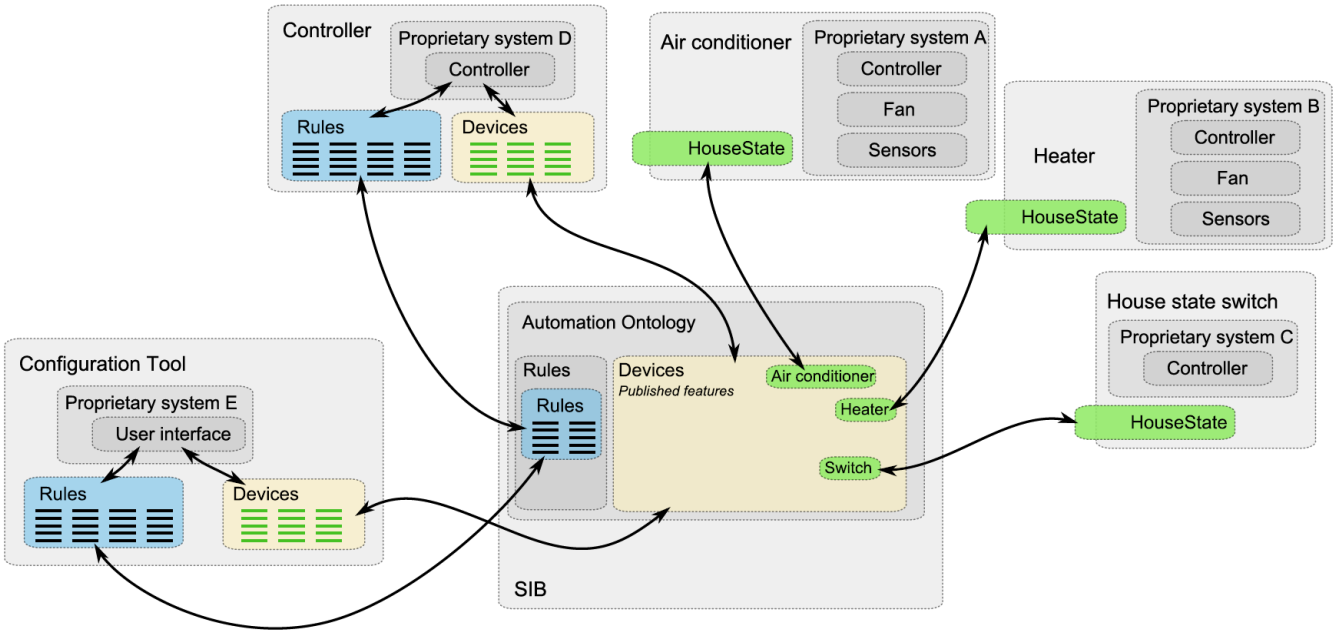

Figure 9. Case study overview showing an interoperability solution for a simple building automation problem.

The integration to the SIB is not intended to replace the control logic embedded in building automation systems, but rather to facilitate using the information from the building automation systems together with other information.

A. *Smart-M3 Implemented for Building Automation*

In building automation there are several different automated devices which benefit from the knowledge of three simple states. These states are "Home", "Away" and "Vacation". This state data can be modified by status changes in several different system, and this data could be requested by any device compatible with the Smart-M3 stack, running an agent built for this purpose. This simple case study was expanded to be configurable to demonstrate an additional benefit which could be added by the Smart Space approach. Our demonstration scenario requires a home state switch, reflecting the global state (i.e. "Home", "Away" and "Vacation"), and a heating system. In addition to these devices there are two additional parts for enabling interoperability: a controller and a configuration tool. In addition to the required interoperating components there is also a temperature display, and a temperature slider which can be configured to correspond to or set the different temperatures available from the heater appliance. The demonstration application consists of several agents [2] representing the functionality of the devices and a user interface. A conceptual model is shown in Figure 9.

All these components contain a proprietary solution, and provide only a limited set of services through their agents. Additional services could be added to the model, and published to the SIB in order for the configuration tool to make new rules of interaction. The demonstration implementation contains a temperature service concept in addition to the house state concept shown in Figure 9. The temperature data service is contained in the Heater, Temperature Slider and the display. An example configuration is to set the display to show the active temperature setting in the heater, but if there was an independent temperature sensor it could be configured to display its value as well.

The configuration tool can add and remove dependencies, rules and connections by querying the SIB. In our scenario the heater and air-conditioning agents can be configured to request changes in the State switch value, or remove their dependency. The state is indicated by an integer value. When configuration has been set up, the configuration tool agent can leave the Smart Space, as all the necessary data is contained within the ontology in the SIB.

1) *Example Scenario*

All devices in the home connect to the SIB, through their respective SIB interfaces, and insert information about themselves. This information consists of a user friendly name, a list of services it provides and the data which describes its state. No automatic configuration about how they interact exists at this time. When the configuration tool is run, the user is presented with devices registered in the SIB, and can then configure rules.

Rules are interpreted by a controller agent. The controller subscribes to changes in the data of the devices. In order to catch changes in state of the switch the controller listens to new instances of the `Event` class. This instance contains information about what has occurred. When the controller receives a new instance of an `Event` it parses through the list of rules, and if there is a matching rule, it will execute the rule. In this simple implementation a matching rule will create a new instance of the class `Invoke` and add properties to it according to the configured rules.

[2] Smart-M3 agents are also called *Knowledge Processors*.

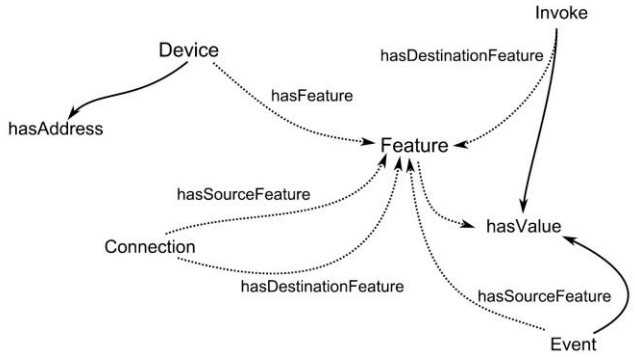

Figure 10. Overview of the ontology used in the implementation of the case study.

The new `Invoke` instance is subscribed to by the agent representing the service invoked, and can then be used to alter the internal state accordingly.

2) Ontology

In this use-case we created an ontology containing rules for automation, and concepts for expressing house state, and temperature. The house state is mapped to an integer value, but should more correctly be defined as an enumerated class consisting of the three states as individuals 'Home", "Away", "Vacation". An overview of the ontology is shown in Figure 10. In this way the configuration tool could suggest potentially interesting counterparts for creating connections or rules. The ontology does not contain a complete set of concepts for building automation, but merely the concepts needed for this specific automation task and demonstration. The ontology is expressed in OWL-DL and contains classes, data properties and object properties.

3) Code Generation

The Python Code Generator was used to generate the agent ontology API. We recognized several features which would be required in order to create a practical implementation of the initial plan of the building automation ontology. Updating data in the SIB requires a delete and insert query. There is no support for subscribing to changes in properties, and thus the implementation uses classes of `Event` and `Invoke` to register changes. This approach adds significant overhead to network utilization. The generated API also populates all instance properties depending on which class it is loaded from, which could also be changed to a populate on demand approach in order to reduce unnecessary queries to the SIB, or alternatively it could make use of a better query receiving all properties in one message. At the moment all properties are queried separately.

4) Building Automation Configuration Tool

The tool for creating rules for the controller is a command line tool. The configuration tool inserts or removes instances of `Connection`. Typing "?" or "help" provides the list of commands and a brief explanation. There are three main commands; `list`, `connect` and `disconnect`.

With the list command a list of registered devices are shown. The list also shows rules for the controller, as shown in the following listing of configuration tool output:

```
HomeStateSwitch (addr=17)
   0 State
      This     three     state     switch
      corresponds   to   the   Home,   Away,
      Vacation     modes     used     in
      heaters etc.
      Connected to:
         Heater => State
```

5) Running the demonstration

The source code is available from SourceForge (www.sourceforge.net), under the name "smart-m3". The demonstration is tested only for specific versions of the components, but may well work with other versions as well. If you are having trouble running the demonstration, please consider installing the following versions; Python 2.6.x, PyQt v.4.5.4 for Python 2.6 and Nokia SIB revision 98. Python is needed because this generated API and the Smart-M3 Mediator[3] are written in Python. The Qt library is used for the message pump of the persistent agent. The Nokia SIB provides the database back-end and connection library for the Smart-M3 Mediator. The SIB is also available from SourceForge, with installation instructions. To the best of our knowledge it does not compile on Windows.

The agents in the building automation demonstration try to connect to a smart space named "x" on 127.0.0.1 at port 10010 by default, but this can be changed for all agents in the file `SmartSpaceConf.py`. Port 10010 is the default for the SIB, but the smart space name must be provided. Running `python SIB.py x` starts a SIB running locally at port 10010 with the smart space name `x`.

The simplest use-case to run is the HomeStateSwitch and the Heater. They have pre-configured addresses of 17 and 7 respectively. These can be connected by the configuration tool using the following commands:

```
Command] list
  HomeStateSwitch (addr=17)
    0 State
        Connected to:
          -
Command] connect
Source address: 17
Source feature: 0
Destination address: 7
Destination  feature:  0  (State  might  have
another number)
Rule name: TestRule
```

If these commands have executed correctly the heater will output its changing state following the home state switch.

The agents can be started in any order, but the configuration tool agent does not find any devices until they are started. The suggested order is to run the controller and the configuration tool, and then any of the service providing devices/agents. Observe that subscriptions to the SIB result

[3] A caching middleware for accessing the SIB.

in a TCP timeout if no subscribed data is sent by the SIB within a quite short period of time depending on network configuration. The Python platform cannot independently set TCP keep-alive messages. It is therefore recommended to run the demonstration locally. There are some subscriptions which are not required after running the configuration tool, thus the example might work even after this timeout error.

The SIB does not clear all data when running the `clear` command, and thus it is recommended to remove the smart space file if problems occur. This is done by running `quit` in the SIB command line interface, and then `rm x`, where `x` is the smart space name, and then running the SIB again `python SIB.py x`.

6) Lessons learned

Tool-chain: We used the Application Development Kit, ADK, available from SourceForge. The ADK was able to generate the Python API from the use-case's OWL ontology without problems. It was found that there are several features to be implemented into the ADK which would improve this building automation use-case. The ADK does not support subscriptions to changes in properties. This lead to modifications in the ontology in order to use subscription to instances instead. An improved ontology API would reduce overhead of creating new instances of `Event` for changes in values of the devices and sensors, instead of updating one property.

Smart-M3: The architecture of the interoperability package as described is not, in its current form, very well suited for building automation. However, it raises interesting points about potential cross domain interoperability scenarios, which are much easier to implement as a result of access to data in a well structured form. By revealing an interface to the smart space for programmatically accessing features in devices, normally accessed by infrared remote controls and proprietary systems, the possibility of interoperating between the virtual and the physical world is realized. Cross domain implementations can be aided by the ADK, by loading several ontologies for a single agent and using input data from one domain and translate it into another.

There is significant overhead in communication and application size. To the best of our knowledge the current SIB implementation does not scale beyond a very modest number of devices in a building automation scenario, and cannot automatically be distributed over multiple SIBs. The workaround to a distributed environment would be to implement an agent which moves data between two SIBs. We recognize that the overhead is partially due to the ADK generated ontology API and this specific implementation.

Improvement proposals: The implementation could be further improved by removing the addressing scheme used here. It should not be needed as long as the instances are uniquely identifiable. The solution used here is for convenience when querying for a specific device, we need only an integer value instead of the UUID. The current implementation is still inadequate for real building automation applications, but demonstrates a working concept of connecting features together via the SIB.

V. CONCLUSIONS

Buildings are a major context for creating smart environments and achieving the goals of Ubiquitous Computing, where people could interact seamlessly with their everyday environment and where the various devices in the environment co-operate in order to achieve some "smart" behavior. However, there are still great interoperability challenges between systems in the domain of building automation due to several competing bus standards and proprietary solutions. The paper shows how such issues can be solved using device adapters and protocol conversion in so called home gateways as illustrated in Fig. 1.

However, the question of semantic interoperability is not solved by home gateways. Semantic interoperability can be partially achieved by standards based on e.g. XML Schema such as oBIX and PMI but it does not seem probable that such standards will achieve a similar global acceptance as HTTP and HTML in any near future. Furthermore, those standards do not define device-specific semantics, such as the names of devices, sensors, alarms etc. In order to overcome such limitations, this paper describes an information publishing mechanism that does not require any pre-defined standard for making the information visible, discoverable and usable to others. That also signifies that it becomes possible for third-party solution providers, who are not themselves manufacturers of building material or building automation, to create Smart Space applications. Such solution providers can provide agents or agent frameworks that implement new functionality. Therefore, our goal is to provide an easy to use basic mechanism that makes it possible to create an open ecosystem where the set of potential applications is open and impossible to predict in advance.

The home gateway solutions described in the paper are currently in use in many real buildings and are expected to become commercial-level volume products within a year. The Smart-M3 implementation described in the paper is implemented on a demonstration laboratory level and will eventually be tested in real pilot targets. Earlier experiences using semantic nets and agents with "classical" tools such as the DIALOG platform have shown their power in several domains such as shipment tracking, product lifecycle management etc. The technical, conceptual and business feasibility of Smart-M3 as an enabler of semantic nets and agents in the building automation domain still remains to be proven. However, the ad hoc data and processing distribution mechanisms of Smart-M3 that are conceived also for embedded devices, and notably mobile phones, are expected to be key enablers of future smart environments where buildings, vehicles, public spaces etc. can be accessed and used in a uniform way.

ACKNOWLEDGMENT

This work has been carried out in the Devices and Interoperability Ecosystem (DIEM) project (www.diem.fi), financed by the Technology Development center of Finland (TEKES) and by the partner organizations of DIEM, and in

the AsEMo project financed by the European Regional Development Fund.

REFERENCES

[1] K. Främling, I. Oliver, J. Nyman, and J. Honkola, "Smart spaces for ubiquitously smart buildings," Proc. Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), Sliema, Malta, October 11-16 2009, pp. 295–300.

[2] I. Oliver and J. Honkola, "Sedvice: A triple space computing exploration environment," Proc. Tripcom workshop, April, 2008.

[3] D. Lewis, Convention: a philosophical study. Blackwell Publishing, 2002, 0-631-23257-5.

[4] PROMISE, "Volume 3: Architecture reference: Promise messaging interface (pmi),". [Online; accessed 25 August 2010] http://cl2m.com/system/files/private/PROMISE AS Volume 3 Architecture Reference PMI.pdf, 2008,.

[5] R. T. Fielding, "Architectural styles and the design of networkbased software architectures," Ph.D. dissertation, University of California, Irvine, 2000. [Online; accessed 21 January 2011] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[6] K. Främling, T. Ala-Risku, M. Kärkkäinen, and J. Holmström, "Agent-based model for managing composite product information," Computers in Industry, vol. 57, no. 1, 2006, pp. 72-81.

[7] K. Främling and L. Rabe, "Enriching product information during the product lifecycle," Proc. 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM), 17-19 May 2006, Saint-Etienne, France, 2006, pp. 861–866.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns – Elements of Reusable Object-Oriented Software. Addison Wesley, Reading, MA, 1995.

[9] DIALOG, "Distributed information architectures for collaborative logistics," 2001. [Online; accessed 21 January 2011] http://dialog.hut.fi/.

[10] K. Främling, T. Ala-Risku, M. Kärkkäinen, and J. Holmström, "Design patterns for managing product life cycle information," Communications of the ACM, vol. 50, no. 6, 2007, pp. 75–79.

[11] I. Oliver, "Information spaces as a basis for personalising the semantic web," Proc. 11th International Conference on Enterprise Information Systems, May 2009.

[12] L. J. B. Nixon, E. Simperl, R. Krummenacher, and F. Martin-Recuerda, "Tuplespace-based computing for the semantic web: a survey of the state-of-the-art," The Knowledge Engineering Review, vol. 23, no. 2, June 2008, pp. 181–212.

[13] B. Hayes-Roth, "A blackboard architecture for control," Artif. Intell., vol. 26, no. 3, 1985, pp. 251–321.

[14] A. Passant, ":me owl:sameas flickr:33669349@n00," Proc. Linked Data on the Web (LDOW 2008), Beijing, China, April 2008.

[15] K. Idehen and O. Erling, "Linked data spaces and data portability," Proc. Linked Data on the Web (LDOW 2008), Beijing, China, April 2008.

[16] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," Scientific American, May 2001.

[17] "Network on terminal architecture," http://www.notaworld.org, 11 2008.

[18] J. Jantunen, I. Oliver, S. Boldyrev, and J. Honkola, "Agent/spacebased computing and rf memory tag interaction," Proc. 3rd International Workshop on RFID Technology - Concepts, Applications, Challenges (IWRT 2009), May 2009.

[19] S. Boldyrev, I. Oliver, and J. Honkola, "A mechanism for managing and distributing information and queries in a smart space environment," Proc. 1st International Workshop on Managing Data with Mobile Devices (MDMD 2009) 6-7 May, 2009 - Milan, Italy, 2009.

[20] I. Oliver and S. Boldyrev, "Operations on spaces of information," in Proc. IEEE Conference on Semantic Computation. Berkeley, CA., September 2009.

[21] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila, "Proteus: A semantic context-aware adaptive policy model," in POLICY. IEEE Computer Society, 2007, pp. 129–140.

[22] I. Oliver, E. Nuutila, and S. Törmä, "Context gathering in meetings: Business processes meet the agents and the semantic web," Proc. 4th International Workshop on Technologies for Context-Aware Business Process Management (TCoB 2009), May 2009.

[23] J. Honkola, H. Laine, R. Brown, and I. Oliver, "Cross-domain interoperability: a case study," Lecture Notes in Computer Science, vol. 5764, Springer Berlin / Heidelberg, September 2009, pp. 22-31.

[24] S. Balandin, I. Oliver, and S. Boldyrev, "Distributed architecture of a professional social network on top of m3 smart space solution made in pcs and mobile devices friendly manner," Proc. Ubicomm 2009. Malta, 2009.

[25] D. E. Herlea, C. M. Jonker, J. Treur, and N. J. E. Wijngaards, Multi-Agent System Engineering, ser. LNCS. Springer, 1999, vol. 1647, ch. Specification of Behavioural Requirements within Compositional Multi-agent System Design, pp. 8–27.

[26] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Compatibility verification for web service choreography," Proc. IEEE International Conference on Web Services, July 6–9 2004. IEEE, 2004, pp. 738–741.

[27] H.-G. Kim and Anseo-Dong, "Pragmatics of the semantic web," Proc. Semantic Web Workshop. Hawaii, USA, 2002.

[28] S. Staab, "Emergent semantics," IEEE Intelligent Systems, vol. 17, no. 1, 2002, pp. 78–86.

[29] B. Hu, S. Dasmahapatra, P. H. Lewis, and N. Shadbolt, "On capturing semantics in ontology mapping," Proc. AAAI. AAAI Press, 2007, pp. 311–316.

[30] M. A. Rodriguez and J. Bollen, "Modeling computations in a semantic network," CoRR, vol. abs/0706.0022, 2007.

[31] O. Lassila, "Programming Semantic Web Applications: A Synthesis of Knowledge Representation and Semi-Structured Data," Ph.D. dissertation, Helsinki University of Technology, November 2007.