OULU 2016
C 557

ACTA UNIVERSITATIS OULUENSIS

Jussi Kiljander

SEMANTIC INTEROPERABILITY FRAMEWORK FOR SMART SPACES

UNIVERSITY OF OULU GRADUATE SCHOOL;
UNIVERSITY OF OULU,
FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING;
VTT TECHNICAL RESEARCH CENTRE OF FINLAND

ACTA UNIVERSITATIS OULUENSIS
C Technica 557

JUSSI KILJANDER

SEMANTIC INTEROPERABILITY FRAMEWORK FOR SMART SPACES

Academic dissertation to be presented with the assent of the Doctoral Training Committee of Technology and Natural Sciences of the University of Oulu for public defence in Kuusamonsali (YB210), Linnanmaa, on 29 January 2016, at 12 noon

UNIVERSITY OF OULU, OULU 2016

Acta Univ. Oul. C 557, 2016

Supervised by
Professor Jukka Riekki

Reviewed by
Professor Johan Lilius
Assistant Professor Payam Barnaghi

Opponents
Associate Professor Paolo Bellavista
Professor Johan Lilius

ISBN 978-952-62-1080-3 (Paperback)
ISBN 978-952-62-1081-0 (PDF)

ISSN 0355-3213 (Printed)
ISSN 1796-2226 (Online)

Cover Design
Raimo Ahonen

JUVENES PRINT
TAMPERE 2016

Kiljander, Jussi, Semantic interoperability framework for smart spaces.
University of Oulu Graduate School; University of Oulu, Faculty of Information Technology and Electrical Engineering; VTT Technical Research Centre of Finland
Acta Univ. Oul. C 557, 2016
University of Oulu, P.O. Box 8000, FI-90014 University of Oulu, Finland

Abstract

At the heart of the smart space vision is the idea that devices interoperate with each other autonomously to assist people in their everyday activities. In order to make this vision a reality, it is important to achieve semantic-level interoperability between devices.

The goal of this dissertation is to enable Semantic Web technology-based interoperability in smart spaces. There are many challenges that need to be solved before this goal can be achieved. In this dissertation, the focus has been on the following four challenges: The first challenge is that the Semantic Web technologies have neither been designed for sharing real-time data nor large packets of data such as video and audio files. This makes it challenging to apply them in smart spaces, where it is typical that devices produce and consume this type of data. The second challenge is the verbose syntax and encoding formats of Semantic Web technologies that make it difficult to utilise them in resource-constrained devices and networks. The third challenge is the heterogeneity of smart space communication technologies that makes it difficult to achieve interoperability even at the connectivity level. The fourth challenge is to provide users with simple means to interact with and configure smart spaces where device interoperability is based on Semantic Web technologies. Even though autonomous operation of devices is a core idea in smart spaces, this is still important in order to achieve successful end-user adoption.

The main result of this dissertation is a semantic interoperability framework, which consists of following individual contributions: 1) a semantic-level interoperability architecture for smart spaces, 2) a knowledge sharing protocol for resource-constrained devices and networks, and 3) an approach to configuring Semantic Web-based smart spaces. The architecture, protocol and smart space configuration approach are evaluated with several reference implementations of the framework components and proof-of-concept smart spaces that are also key contributions of this dissertation.

Keywords: Internet of Things, interoperability, knowledge sharing protocol, resource-constrained devices and networks, Semantic Web, smart space, system architecture

Kiljander, Jussi, Viitekehys laitteiden semanttisen tason yhteentoimivuuteen älytiloissa.
Oulun yliopiston tutkijakoulu; Oulun yliopisto, Tieto- ja sähkötekniikan tiedekunta; Teknologian tutkimuskeskus VTT
Acta Univ. Oul. C 557, 2016
Oulun yliopisto, PL 8000, 90014 Oulun yliopisto

Tiivistelmä

Älytilavision ydinajatuksena on, että erilaiset laitteet tuottavat yhteistyössä ihmisten elämää helpottavia palveluita. Vision toteutumisen kannalta on tärkeää saavuttaa semanttisen tason yhteentoimivuus laitteiden välillä.

Tämän väitöskirjan tavoitteena on mahdollistaa semanttisen webin teknologioihin pohjautuva yhteentoimivuus älytilan laitteiden välillä. Monenlaisia haasteita täytyy ratkaista, ennen kuin tämä tavoite voidaan saavuttaa. Tässä työssä keskityttiin seuraaviin neljään haasteeseen: Ensimmäinen haaste on, että semanttisen webin teknologioita ei ole suunniteltu reaaliaikaiseen kommunikaatioon, eivätkä ne sovellu isojen tiedostojen jakamiseen. Tämän vuoksi on haasteellista hyödyntää niitä älytiloissa, joissa laitteet tyypillisesti jakavat tällaista tietoa. Toinen haaste on, että semanttisen webin teknologiat perustuvat syntakseihin ja koodausformaatteihin, jotka tuottavat laitteiden kannalta tarpeettoman pitkiä viestejä. Tämä tekee niiden hyödyntämisestä hankalaa resurssirajoittuneissa laitteissa ja verkoissa. Kolmas haaste on, että älytiloissa hyödynnetään hyvin erilaisia kommunikaatioteknologioita, minkä vuoksi jopa tiedonsiirto laitteiden välillä on haasteellista. Neljäs haaste on tarjota loppukäyttäjälle helppoja menetelmiä sekä vuorovaikutukseen semanttiseen webiin pohjautuvien älytilojen kanssa että tällaisen älytilan muokkaamiseen käyttäjän tarpeiden mukaiseksi. Vaikka laitteiden itsenäinen toiminta onkin älytilojen perusajatuksia, tämä on kuitenkin tärkeää teknologian hyväksymisen ja käyttöönoton kannalta.

Väitöskirjan päätulos on laitteiden semanttisen yhteentoimivuuden viitekehys, joka koostuu seuraavista itsenäisistä kontribuutioista: 1) semanttisen tason yhteentoimivuusarkkitehtuuri älytiloille, 2) tiedonjakoprotokolla resurssirajoittuneille laitteille ja verkoille sekä 3) menetelmä semanttiseen webiin pohjautuvien älytilojen konfigurointiin. Näiden kontribuutioiden evaluointi suoritettiin erilaisten järjestelmäkomponenttien referenssitoteutuksilla ja prototyyppiälytiloilla, jotka kuuluvat myös väitöskirjan keskeisiin kontribuutioihin.

Asiasanat: esineiden internet, järjestelmäarkkitehtuuri, resurssirajoittuneet laitteet ja verkot, semanttinen web, tiedonjakoprotokolla, yhteentoimivuus, älytila

Preface

The research for this dissertation was performed at VTT Technical Research Centre of Finland. The work was started in the DIEM (Device and Interoperability Ecosystem) project in 2010 and was continued in the OPUTE (Open Ubiquitous Technology), MIOTE (Merging IoT Technologies), IoT-A (Internet of Things – Architecture), OLDA (Open Local Data Applications) and IMPReSS (Intelligent System Development Platform for Intelligent and Sustainable Society) projects funded by VTT Technical Research Centre of Finland, Tekes (the Finnish Funding Agency for Technology and Innovation) and the European Commission. I would like to thank the abovementioned organisations for making this research possible.

Several people have significantly contributed to the research presented in this dissertation and also helped me in many ways during the research work. I am very grateful to Professor Juha-Pekka Soininen and Mr. Kari Tiensyrjä for the advice, encouragement and general help they have provided during the last six years. Moreover, I wish to thank Professor Soininen for the ideas, guidance and feedback he has provided during the research work. I would like to thank my colleagues and co-authors of the original publications, *i.e.*, Luciano Bononi, Alfredo D'Elia, Matti Eteläperä, Pasi Hyttinen, Kari Keinänen, Francesco Morandi, Luca Roffia, Tullio Salmon Cinotti, Juha-Pekka Soininen, Janne Takalo-Mattila, Esa Viljamaa, Fabio Viola and Arto Ylisaukko-oja. Your contributions to this research have been invaluable. I would also like to thank Professor Tullio Salmon Cinotti for making it possible for me to work with his research team by inviting me and my wife to Bologna and helping us in many ways during our stay.

I am grateful to Professor Jukka Riekki from the University of Oulu for supervising my work and assisting me with the completion of this dissertation. I wish to thank Professor Johan Lilius and Assistant Professor Payam Barnaghi for reviewing the manuscript of this dissertation and for providing helpful advice on how it can be improved.

I am deeply grateful to my parents and other family members for their care and support throughout the years. Last but not least, I wish to thank my wife Minttu for her love, support and patience. She made it possible for me to concentrate on my dissertation during numerous evenings and weekends.

Oulu, 20 October 2015 Jussi Kiljander

List of abbreviations

AI	Artificial Intelligence
ANSI C	American National Standard Institute Standards for C Programming Language
API	Application Programming Interface
ARCES	Advanced Research Center on Electronic Systems for Information and Communication Technologies E. De Castro
ARM	Architectural Reference Model for IoT
ASCII	American Standard Code for Information Interchange
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
CoBrA	Context Broker Architecture for Pervasive Computing
COCOA	Conversation-based Service Composition in Pervasive Computing Environments with QoS Support
CPU	Central Processing Unit
DAML-S	DARPA Agent Mark-up Language for Services
DIEM	Device and Interoperability Ecosystem
DL	Description Logics
DNS	Domain Name System
EA-ONT	Event and Action Ontology
ECSE	Event-based Configuration of Smart Environments
FIPA	Foundation for Intelligent Physical Agents
FS-ONT	File Sharing Ontology
HCI	Human-Computer Interaction
HTTP	Hyper Text Transfer Protocol
ICT	Information and Communication Technology
IoT	Internet of Things
IoT-A	Internet of Things – Architecture
IP	Internet Protocol
IRI	Internationalised Resource Identifier
KP	Knowledge Processor
KPI-low	Low-level Knowledge Processor Interface
KR	Knowledge Representation
KSP	Knowledge Sharing Protocol
LED	Light-Emitting Diode
M2M	Machine-to-Machine

mDNS	Multicast Domain Name System
MIOTE	Merging IoT Technologies
MQTT	Message Queuing Telemetry Transport
NFC	Near Field Communication
NoTA	Network on Terminal Architecture
OLDA	Open Local Data Applications
OPUTE	Open and Ubiquitous Technologies
OS	Operating System
OSI	Open Systems Interconnection
OWL	Web Ontology Language
PIR sensor	Passive Infrared Sensor
QoS	Quality of Service
QR code	Quick Response Code
RACS	Radio Applications Cloud Server
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	RDF Data Query Language
REST	Representational State Transfer
RFID	Radio-Frequency Identification
RIBS	RDF Information Base Solution
RPC	Remote Procedure Call
SeRQL	Sesame RDF Query Language
SI	Semantic Interface
SIB	Semantic Information Broker
SoaM	Smart Objects Awareness and Adaptation Model
SOAP	Simple Object Access Protocol
SoC	System on a Chip
SOFIA	Smart Objects for Intelligent Applications
SOUPA	Standard Ontology for Ubiquitous and Pervasive Applications
SQL	Structured Query Language
SSAP	Smart Space Access Protocol
SSN	Semantic Sensor Network
SSO	Stimulus-Sensor-Observation
SSP	SIB Service Profile
TCE	Task Computing Environment
TCP	Transmission Control Protocol
ucodeRP	Ucode Resolution Protocol

UDP	User Datagram Protocol
UI	User Interface
uID	Ubiquitous ID
ULCO	Upper-Level Context Ontology
UPnP	Universal Plug and Play
uRC	Ucode Resolution Client
URI	Uniform Resource Identifier
VE	Virtual Entity
VTT	VTT Technical Research Centre of Finland
W3C	World Wide Web Consortium
WAX	Word Aligned XML
WLAN	Wireless Local Area Network
WQL	Wilbur Query Language
WSDL	Web Service Description Language
WSN	Wireless Sensor Network
WWW	World Wide Web
XML	Extensible Mark-up Language

List of original articles

I Kiljander J, Eteläperä M, Takalo-Mattila J & Soininen J. (2010) Opening Information of Low Capacity Embedded Systems for Smart Spaces. Proc IEEE 8th Workshop on Intelligent Solutions in Embedded Systems (WISES). Heraklion, Greece: 23–28. doi: 10.1109/WISES.2010.5548438.

II Kiljander J, Eteläperä M, Takalo-Mattila J, Soininen J & Keinänen K (2011) Autonomous File Sharing for Smart Environments. Proc 1st International Conference on Pervasive and Embedded Computing and Communication Systems. Algarve, Portugal: 191–196. doi: 10.5220/0003362501910196.

III Takalo-Mattila J, Kiljander J, Eteläperä M & Soininen J. (2011) Ubiquitous Computing by Utilizing Semantic Interoperability with Item-Level Object Identification. In: Pentikousis K, Agüero R, Garcìa-Arranz M & Papavassiliou S (eds.) Mobile Networks and Management, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg: 198–209. doi: 10.1007/978-3-642-21444-8_18.

IV Kiljander J, Takalo-Mattila J, Eteläperä M, Soininen J & Keinänen K. (2011) Enabling End-Users to Configure Smart Environments. Proc IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT). Munich, Germany: 303–308. doi: 10.1109/SAINT.2011.58.

V Ylisaukko-oja A, Hyttinen P, Kiljander J, Soininen J & Viljamaa E. (2011) Semantic Interface for Resource Constrained Wireless Sensors. Proc SciTePress International Conference on Knowledge Engineering and Ontology Development (KEOD). Paris, France: 505–511. doi: 10.5220/0003698905050511.

VI Viljamaa E, Kiljander J, Soininen J & Ylisaukko-oja A. (2011) A Smart Control System Solution Based on Semantic Web and uID. Proc IARIA 5th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM). Lisbon, Portugal: 105–110. isbn: 978-1-61208-171-7.

VII Kiljander J, Ylisaukko-oja A, Takalo-Mattila J, Eteläperä M & Soininen J. (2012) Enabling Semantic Technology Empowered Smart Spaces. Journal of Computer Networks and Communications 2012: 1–14. doi: 10.1155/2012/845762.

VIII Kiljander J, Morandi F & Soininen J. (2012) Knowledge sharing protocol for smart spaces. International Journal of Advanced Computer Science and Applications 3(9): 100–110. doi: 10.14569/IJACSA.2012.030915.

IX Roffia L, Morandi F, Kiljander J, D'Elia A, Vergari F, Viola F, Bononi L & Salmon Cinotti T. A Semantic Publish-Subscribe Architecture for the Internet of Things. Manuscript.

X Kiljander J, D'Elia A, Morandi F, Hyttinen P, Takalo-Mattila J, Ylisaukko-oja A, Soininen J & Salmon Cinotti T. (2014) Semantic Interoperability Architecture for Pervasive Computing and Internet of Things. IEEE Access 2: 856–873. doi: 10.1109/ACCESS.2014.2347992.

Table of contents

1 Introduction

1.1 Background

The number of devices populating our everyday living environments is constantly increasing. These devices contain means both for collecting data about the physical world and for interacting with it. It is widely agreed (Berners-Lee *et al.* 2001, Ashton 2009, Helal 2011) that by enabling these devices to share their data and capabilities, we could develop autonomous and context-aware systems in various application domains such as home automation, transportation, healthcare and logistics. In this dissertation, this type of a system in which devices share their information and capabilities with each other in order to assist people in their everyday life is referred to as a smart space.

During the last 20 years, smart space-related research has been performed under computing paradigms such as ubiquitous computing (Weiser 1991), pervasive computing (Saha & Mukherjee 2003), ambient intelligence (Aarts *et al.* 2001) and the Internet of Things (IoT) (Gershenfeld 2004), each focusing on slightly different aspects of smart spaces. Smart spaces are also closely related to research fields such as machine-to-machine (M2M) communication (Boswarthick *et al.* 2012), artificial intelligence (AI) (Rich 1983), wireless sensor networks (WSN) (Sohraby *et al.* 2007) and human-computer interaction (HCI) (Card *et al.* 1980). As can be seen, smart spaces is a wide research area that includes topics from environmental monitoring to machine learning.

In this dissertation, the main focus of research is on smart space interoperability. Heiler (1995) defines interoperability in the domain of large-scale distributed systems as the ability of components to share services and data with each other. Since smart spaces are special types of distributed systems, this definition is also suitable for them. The components that interoperate with each other in a smart space are different kinds of devices that can be classified into three groups based on capabilities they provide for the system: 1) devices that monitor the environment (*i.e.*, sensors), 2) devices that enable interaction with the physical world (*i.e.*, actuators) and 3) devices that provide services for end-users by utilising capabilities provided by sensors and actuators. A device can also belong to more than one of these groups.

In order to realise the vision of smart spaces, a common solution for device interoperability needs to be provided. The interoperability issues that need to be

solved before devices can successfully communicate with each other can be roughly divided into two levels, referred to as *connectivity* and *semantic* in this dissertation. Devices interoperable at the connectivity level are able to share data with each other. When devices are interoperable at the semantic level, they also share a common understanding of the meaning of the data. It is natural that most of the research and developments in the field of information and communications technology (ICT) has focused on solving connectivity-level interoperability challenges. This is because typical communication systems such as the telephone, email and the Web have been designed to be used by people, who are responsible for solving semantic-level interoperability issues. The central idea with smart spaces, on the other hand, is that devices communicate and interoperate with each other without human assistance. Therefore, to enable devices to cooperatively serve people in different kinds of situations (that were not necessarily imagined at the design time of the system), there is a need for technologies that provide devices with means to represent and interpret the meaning of any data in a general way.

In this dissertation, the possibility to utilise Semantic Web (Berners-Lee *et al.* 2001) technologies for solving semantic-level interoperability challenges in smart spaces is investigated. The Semantic Web is an extension of the traditional World Wide Web (WWW). In contrast to the traditional Web, the meaning and context of data in the Semantic Web are represented and shared in a machine-interpretable format. This enables autonomous software programs, known as agents (Tamma 2008), to interpret the meaning of data and execute tasks on the user's behalf. Although Semantic Web technologies have been designed for representing information on the Web, many of their properties also make them suitable for enabling semantic-level interoperability in smart spaces. That is, the data model and vocabulary provided by knowledge representation (KR) technologies such as Resource Description Framework (RDF) (W3C RDF Working Group 2014a), RDF Schema (RDFS) (W3C RDF Working Group 2014b) and Web Ontology Language (OWL) (W3C OWL Working Group 2012) are well-fitted for representing relevant information (*e.g.* capabilities of devices and properties of non-ICT objects) about smart spaces. Additionally, the flexible RDF data model supports the natural evolution of smart spaces by allowing virtual representation of the environment to be altered and refined without a need to redo schemas. SPARQL (W3C SPARQL Working Group 2013a), the *de facto* access and management language for RDF data, in turn provides smart space devices with powerful means to perform data manipulation so that network traffic and computing performed in resource-constrained devices can be reduced. It should also be noted that by making it

possible to describe unknown concepts with the help of known ones (*e.g.* a temperature sensor is a device that can measure temperature), Semantic Web technologies provide a good basis for machine learning. In principle, this ability to learn new concepts enables the development of smart space agents that can communicate with other agents using terminology they were unfamiliar with at the design time.

It is not a new idea to apply Semantic Web technologies in smart spaces. Soon after the emergence of the Semantic Web vision, Lassila (2002) envisioned how Semantic Web technologies could be utilised for enhancing service discovery in the ubiquitous computing domain. Other approaches where Semantic Web technologies are used for improving service discovery and orchestration in service-based interoperability solutions include Task Computing Environment (Masuoka *et al.* 2003), COCOA (Mokhtar *et al.* 2007), Semantic Middleware for IoT (Song *et al.* 2010) and Amigo (Thomson *et al.* 2008), to name a few. To take full advantage of Semantic Web technologies, it is not enough to use them only for service discovery and orchestration, however. That is, to fully exploit the potential of Semantic Web technologies, M2M communication in smart spaces should be based solely on Semantic Web technologies. A typical way to realise this is to utilise the blackboard architectural style (Erman *et al.* 1980), where agents interoperate with each other by sharing information via a common knowledge broker. To the author's knowledge, the first examples of blackboard-based semantic interoperability solutions for smart spaces are Context Broker Architecture for Pervasive Computing (CoBrA) (Chen *et al.* 2004a) and Semantic Space (Wang *et al.* 2004). More recent examples include M3 (Lappeteläinen *et al.* 2008) and INSTANS (Rinne *et al.* 2012).

1.2 Definition of the research problems

Although several Semantic Web-based solutions for enabling interoperability in smart spaces have been proposed, there are still many problems that need to be solved before the full potential of these technologies can be realised in practice. The research work presented in this dissertation has focused on the following five problems.

The first problem is the performance of semantic information processing. Poor performance is a typical challenge with Semantic Web technologies and especially visible with large amounts of data and agents. The roots of this problem lie in the fact that Semantic Web technologies have been designed to provide general and

flexible methods for describing resources on the Web, where this metadata is normally updated at a much lower rate compared to typical data exchanged between devices in smart spaces.

The second problem addressed in this dissertation is also related to using Semantic Web technologies for all kinds of communications between devices. In addition to the scenarios where data needs to be processed in a near real-time fashion, scenarios where non-semantic data such as audio and video is exchanged between devices are problematic for Semantic Web technologies. This is because Semantic Web technologies have been designed for representing and accessing metadata about Web resources and are thus not feasible for accessing video and audio files or streams directly (*i.e.*, neither RDF databases nor SPARQL are suitable for handling very large literals).

The third notable problem in applying Semantic Web technologies to enable device interoperability in smart spaces is that it is difficult to exploit them with resource-constrained devices and networks that are common in smart spaces. The main challenge here is the verbose human-readable serialisation formats[1] that introduce significant overheads compared to optimised binary formats typically used with resource-constrained devices and networks.

The fourth problem addressed in this dissertation is related to the heterogeneity of connectivity-level communication technologies used in smart spaces. This is a common problem in smart spaces as the lack of interoperability between technologies such as the Wireless Local Area Network (WLAN), ZigBee and Bluetooth low energy (BLE), for example, makes it difficult to achieve interoperability even at the connectivity level. Additionally, the heterogeneity in connectivity-level technologies requires that the semantic-level interoperability solution is designed so that it is suitable for different kinds of connectivity technologies.

The fifth and last problem addressed in this dissertation is related to the interaction between smart spaces and end-users. In order to gain wider acceptance for smart spaces, the interaction between devices and users should be as easy and natural as possible. Additionally, since people have individual needs and preferences, users should be able to define the behaviour of devices in different situations so that the services provided by the smart space are relevant for the given user.

[1] For example, the Extensible Markup Language (XML) serialisation for RDF and the Turtle-like notation used for representing SPARQL operations.

1.3 Objectives and scope of the research

The goal of the research presented in this dissertation is to make Semantic Web technologies a more feasible solution for enabling device interoperability in smart spaces by investigating solutions to the problems elaborated in Section 1.2. In particular, the research objectives of this dissertation are as follows:

1. To enable smart space devices to execute all semantic-level communication with Semantic Web technologies.
2. To facilitate exploitation of Semantic Web technologies with resource-constrained devices and networks.
3. To make it possible for smart space devices to interoperate over heterogeneous connectivity-level technologies.
4. To provide simple ways for users to interact with and configure smart spaces where semantic-level interoperability is based on Semantic Web technologies.

1.4 Research methods and history

The constructive research approach (Järvinen 2004) was used as the main research method in this work. That is, the research presented in this dissertation was executed by designing and implementing a semantic interoperability framework for smart spaces and evaluating how well it meets the objectives presented in Section 1.3. The framework consists of several individual contributions, including a semantic interoperability architecture, a knowledge access and management protocol, an approach to configuring smart spaces, and various reference implementations and demonstrations. These contributions are introduced in more detail in Chapter 4.

The design and implementation of the interoperability framework has been an iterative process executed during several research projects. The research started in the Device and Interoperability Ecosystem (DIEM) project, which aimed to develop an interoperability platform and ecosystem for smart environments. M3, as the state-of-the-art solution for semantic interoperability in smart spaces, was taken as a starting point for the research, and its feasibility for enabling semantic interoperability in smart spaces was studied. To this end, two proof-of-concept smart space systems, called Open-M3 and Smart Greenhouse, were implemented. Open-M3 was our first semantic-interoperability demonstration and it was developed to experiment with simple interaction between semantic sensors and user

interface (UI) devices (Eteläperä, *et al.* 2010). The scope of the Smart Greenhouse case study, on the other hand, was broader and covered four of the five research problems addressed in this dissertation. In particular, this work contributed towards Objectives 1 and 3 with the following results: 1) an approach to interacting with smart space sensors and actuators by monitoring and modifying their RDF-based virtual representations and 2) a reference implementation of a smart space agent in a resource-constrained computing platform. The Smart Greenhouse case study is presented in more detail in Publication I. To the author's knowledge, this is the first paper that describes how to control actuators with Semantic Web technologies.

As can be seen from Objective 1, a central aim of the semantic interoperability framework designed and developed in this thesis work is to make it possible to represent all semantic-level M2M communication with Semantic Web technologies. In the Smart Greenhouse case study, we developed an interaction model for reading sensor data and controlling actuators but did not study means for sharing non-semantic data such as files and streams that cannot be directly shared through RDF databases. To address this limitation, the work in the DIEM project was continued by designing a novel approach to non-semantic content sharing with Semantic Web technologies. By following this approach, smart space agents can utilise Semantic Web technologies both for advertising their files and for negotiating with each other how to transfer the content between devices. To evaluate the approach in practice, a case study called Smart Meeting was implemented. The approach and the case study are presented in Publication II.

In parallel with the DIEM project, research work was executed in the Open and Ubiquitous Technologies (OPUTE) project. The author's work in the OPUTE project focused on solutions for seamless smart space interaction by combining item-level object identification and Semantic Web technologies. In particular, this work contributes towards Objective 4 with a novel approach to touch-based interaction with semantic technology-enhanced smart spaces. The approach was evaluated by extending the Smart Greenhouse case study with radio-frequency identification (RFID)-based object identification methods. This approach and the case study are presented in Publication III.

At the time, Objective 4 was also the author's main focus in the DIEM project, in which a novel approach to configuring the behaviour of Semantic Web-empowered smart spaces was designed. This approach was evaluated by implementing a tool for Event-based Configuration of Smart Environments (ECSE) and testing the ECSE tool in the Smart Greenhouse environment. This work is presented in Publication IV.

During the DIEM and OPUTE projects, a new M3 knowledge broker reference implementation, called RDF Information Base Solution (RIBS), was implemented in the Smart Objects for Intelligent Applications (SOFIA) project. The new RIBS implementation (equipped with a more compact agent communication protocol) inspired us to continue with experiments of agents in resource-constrained devices. In particular, this work contributed towards Objectives 2 and 3 with a proof-of-concept implementation of a smart space agent in an ARM7-based system on a chip (SoC) that utilises a resource-constrained IEEE 802.15.4 radio at the connectivity level. In addition to Objectives 2 and 3, the author's work in the OPUTE project contributed towards Objective 1 by making it possible to improve the performance of semantic information processing in larger-scale systems such as the Interned of Things. The main result of this work is a novel Ubiquitous ID (uID) architecture (Koshizuka & Sakamura 2010) based discovery approach, which makes it possible to resolve the address of a knowledge broker containing a virtual representation of any physical-world object identified with a ucode. With this approach, it is possible to improve the latency of query, subscription and update operations in larger-scale systems by dividing the system data into several knowledge brokers that can be discovered by using the uID architecture. To evaluate these approaches in practice, a new demonstration called Smart Home Garden was implemented. This research is presented in Publications V and VI.

After the OPUTE project, the research with resource-constrained devices and the uID architecture continued in the Merging IoT Technologies (MIOTE) project. This work was started by analysing the results obtained in the OPUTE and DIEM projects. Publication VII is a synthesis of this work and presents the significance of these individual contributions in a broader context. After analysing the current status of the research, the work continued on two parallel tracks. The first track contributed towards Objective 2 with a novel knowledge sharing protocol (KSP) which had on average 70% shorter messages than the most compact M3 communication protocol in an evaluation executed in the Smart Greenhouse case study. The KSP is presented in more detail in Publication VIII. The second track contributed towards Objective 1 by continuing the work on a distributed interoperability architecture for large-scale smart space systems. This architecture work was also partly executed in the Internet of Things – Architecture (IoT-A) project, where the author's work focused on the design and implementation of a resolution infrastructure for the Internet of Things.

During the MIOTE and IoT-A projects, the author visited the Advanced Research Center on Electronic Systems for Information and Communication

Technologies E. De Castro (ARCES) of the University of Bologna for three and a half months. The aim of this visit was to improve cooperation between our research team and the ARCES semantic interoperability team that had worked with semantic interoperability platforms in the SOFIA project. During this time, the author worked with ARCES researchers on two topics. The first topic concentrated on improving the performance of SPARQL subscription processing. The main result of this work is a novel semantic event processing architecture and engine which makes it possible to improve the performance of SPARQL subscription processing when compared to the original M3 platform (Honkola *et al.* 2010). This work is described in Publication IX. The objective of the second topic was to continue the work on the distributed interoperability architecture for large-scale systems. The result of this work, presented in Publication X, is a novel semantic interoperability architecture for the IoT, which is a synthesis of the work done by the author and others in the DIEM, SOFIA, OPUTE, MIOTE and IoT-A projects.

1.5 Organisation of the dissertation

This dissertation consists of an introductory part and ten original publications written between 2010 and 2015. The introductory part of this dissertation is organised as follows: Chapter 2 presents the background for the dissertation. Chapter 3 provides a review of the state-of-the-art solutions for Semantic Web-based interoperability frameworks for smart spaces. Chapter 4 gives a summary of the main scientific contributions. Introduction to the original papers is presented in Chapter 5. In Chapter 6, the results of the work are analysed and compared to the related work. Finally, Chapter 7 summarises the introductory part of this dissertation.

2 Main concepts

2.1 Smart spaces

Smart spaces have been researched in numerous projects for over 20 years. Well-known smart space research projects include the Boulder's Adaptive House (Mozer 2005), Buxton's Reactive Environment (Cooperstock *et al.* 1995), Massachusetts Institute of Technology's Oxygen (Dertouzos 1999), the Microsofts EasyLiving (Brumitt *et al.* 2000), Hewlett Packard's Cooltown (Kindberg *et al.* 2000), and Stanford University's iRoom (Johanson *et al.* 2002), to name a few.

During its 20-year history, smart spaces have been studied under different computing paradigms, including ubiquitous and pervasive computing, ambient intelligence and the Internet of Things. The main idea behind the ubiquitous and pervasive computing is to make the human-computer interaction as seamless and transparent as possible by embedding computing everywhere in our environment. The ambient intelligence vision builds on top of the ubiquitous and pervasive computing and aims to realise an environment that makes people's lives easier by carrying out activities on their behalf. To this end, ambient intelligence research focuses on technologies that can provide people with context-aware, personalised, adaptive, and anticipatory services. Whereas ubiquitous computing and ambient intelligence mainly focus on local environments, the vision of the Internet of Things is to provide a worldwide interoperability and information sharing infrastructure for devices (*i.e.,* a global smart space infrastructure). The original IoT vision introduced by Kevin Ashton in 1999 (Ashton 2009) focused on creating a machine-readable representation of the physical world with the help of RFID and sensor technologies. Since then, the IoT vision has evolved so that it now (Vermesan & Friess 2013) covers many aspects of ubiquitous computing and ambient intelligence and can be thus seen as a global-scale extension of these paradigms.

Mainly due to the different area of emphasis in the abovementioned computing paradigms and the variety in smart space application domains, many different definitions for the term "smart space" have been introduced in the literature. Nixon *et al.* (2003) define smart space as follows:

> *"A smart space is a region of the real world that is extensively equipped with sensors, actuators and computing components".*

This definition emphasises that a smart space is a physical place that is inhabited by a large number of electronic devices but it does not say anything about the role

of the people or about the capabilities of these devices. In the definition presented by Singh *et al.* (2006), the users and the capabilities of devices are incorporated into the definition that is formulated as follows:

> *"Smart spaces are ordinary environments equipped with visual and audio sensing systems, pervasive devices, sensors, and networks that can perceive and react to people, sense ongoing human activities and respond to them."*

A similar definition is proposed by Lupiana *et al.* (2009), who argue that a smart space should contain ubiquitous devices, wireless networks, sensors, and reasoning mechanisms. They define a smart space as:

> *"a highly integrated computing and sensory environment that effectively reasons about the physical and user context of the space to transparently act on human desires".*

Sathish *et al.* (2007), in turn, focus on the heterogeneous and dynamic nature of smart spaces and present the following definition:

> *"A smart space is a multi-user multi-device dynamic interaction environment that is aware of its physical environment and that works on top of heterogeneous networking technologies and software platforms".*

Despite the variety in definitions, there are few features that can be identified to be typical for smart spaces. A good summary of common smart space properties is presented by Jianhua *et al.* (2005). First, they emphasise that a smart space is a physical space that is inhabited by a heterogeneous group of electronic devices which utilise different kinds of wired and wireless communication technologies for sharing data with each other. Second, they state that a smart space must possess capabilities for perceiving, analysing and reasoning the needs of users and the context of the environment so that relevant actions can be taken in proper situations. They emphasise that the ability to perform the right actions in the right situations is what justifies a physical environment to be called smart. Third, they define that a smart space should aim at providing better services for people in their everyday activities by adapting to human behaviour.

2.2 Interoperability in smart spaces

In the domain of smart spaces, the term "interoperability" refers to the capability of devices to successfully communicate and interact with each other. Achieving

interoperability in smart spaces is a challenging task due to the following characteristics of smart spaces.

First, smart spaces are heterogeneous in terms of communication technologies, devices, physical environments, and application domains. This means that interoperability technologies need to provide interoperability between a heterogeneous group of devices and connectivity-level technologies used in smart spaces. Interoperability solutions also need to be application-domain agnostic and they should facilitate sharing of information and services across applications even from different domains. Additionally, the technologies need to be suitable for a wide range of computing platforms and communication technologies from high-end servers and Internet communication to the most resource-restricted networks and low-power embedded systems.

Second, smart spaces are dynamic environments but on the other hand also need to last a long time. For example, physical environments such as buildings can last hundreds of years but the contents of the buildings (people, devices, items, etc.) may change very rapidly. This means that smart space interoperability solutions need to support future extensions so that new applications and devices can be deployed into the smart space without a need to do any modifications to the existing system.

Third, a key idea in smart spaces is that devices interact with each other autonomously without human assistance. This means that it is not enough to solve interoperability at the physical interface and connectivity level but the meaning and context of the data also need to be exchanged in an interoperable way.

The level at which devices are interoperable with each other can be described with different kinds of layered models. Next, three interoperability models presented in the literature are introduced.

The first model, proposed by Tolk (2004), presents six levels for interoperability: *no connection*, *technical*, *syntactical*, *semantic*, *pragmatic/dynamic*, and *conceptual*. At the *no connection* level, there is no interoperability between systems. At the *technical* level, there is a physical connection between devices allowing raw data to be exchanged. At the *syntactic* level, the data shared between systems is serialised with standardised formats and protocols. At the *semantic* level, the meaning and context of the data is represented. At the *pragmatic/dynamic* level, the applicability of the information is also defined. Finally, at the *conceptual* level, a common view of the world is established. It should be noted that Tolk's model for interoperability has been designed for a

simulation theory instead of smart spaces. As can be seen, however, the model is also suitable for representing different levels of interoperability in smart spaces.

The second model proposed by Pantsar-Syväniemi *et al.* (2012) contains the following six levels: *connection*, *communication*, *semantic*, *dynamic*, *behavioural,* and *conceptual*. The *connection*, *communication*, *semantic*, *dynamic* and *conceptual* levels are more or less equivalent to the *technical*, *syntactical*, *semantic*, *pragmatic/dynamic* and *conceptual* levels in Tolk's model, respectively. The main differences between the two models are that the model proposed by Pantsar-Syväniemi *et al.* focuses more on the technologies that can be used to solve interoperability challenges at different levels. Because of this, their model does not include the *no connection* level, where no interoperability between two systems is achieved. Additionally, the *behavioural* level that covers interoperability challenges related to matching actions together so that the correct behaviour is achieved is not presented in Tolk's model.

When compared to the models proposed by Tolk and Pantsar-Syväniemi *et al.*, the third model, introduced by Lappeteläinen *et al.* (2008), describes a more simplified view of interoperability in smart spaces. This model consists of three levels: *device*, *service* and *information*. Systems interoperable at the *device* level are capable of forming networks for data exchange. At the *service* level, the interoperability challenge is related to discovering and utilising services. The challenge in the *information*-level interoperability, in turn, is related to accessing the semantics of the data. These levels are more or less equivalent to the *technical*/*connection*, *syntactical/communication* and *semantic* levels proposed by Tolk and Pantsar-Syväniemi *et al*.

In this dissertation, interoperability in smart spaces is depicted with a simple two-level model. The first level of interoperability is referred to as *connectivity* and the second level as *semantic*. At the connectivity level, the interoperability challenge is related to enabling devices (or more specifically agents) to transmit data with each other. The challenge in the semantic-level interoperability, on the other hand, is to enable the communicating parties to interpret the meaning (*i.e.,* semantics) of the data in the same way. This model differs from the existing interoperability models in that it focuses only on the two aspects of communication that need to be solved before devices can interoperate with each other instead of trying to classify in detail all the identifiable levels at which device interoperability can be achieved. For this reason, the two-level interoperability model does not contain the service/syntactic level, for example, since that level can be seen as a

sublevel of the semantic level (*i.e.*, part of the semantic level is the syntax that defines the allowed combination and order of symbols in messages).

From the perspective of the traditional Open System Interconnection (OSI) model, the connectivity level covers all the layers from the physical to the session layer. The research in the field of ICT has been very successful and it has produced many technologies for solving connectivity-level interoperability challenges in smart spaces. These solutions range from mobile telecommunication technologies to the wireless local area network (WLAN) and include technologies such as Transmission Control Protocol (TCP), Internet Protocol (IP) and User Datagram Protocol (UDP) that are the basic building blocks of the Internet. From the perspective of resource-constrained devices, the most interesting technologies at the connectivity level are 6LowPAN (Shelby & Bormann 2010) and Bluetooth low energy (BLE). In the future, a big technological step at the connectivity level will be 5G, which is predicted to emerge as the universal solution for the next generation wireless systems (Andrews *et al.* 2014).

In his famous information theory, Claude E. Shannon (1948) stated that the semantic aspects of communication are irrelevant for the engineering problem. This statement was, of course, true for the traditional communication systems of that time that were designed to be used by humans. Nowadays, communication systems are not only used by people but also by devices that interoperate with each other autonomously. In these modern M2M communication systems such as smart spaces, the semantic aspects of communication are in fact an important engineering problem to be solved. In early smart space systems, semantic-level interoperability was realised with proprietary mechanisms, and robust behaviour of the system was achieved by defining the data exchange for each scenario at the design time. This approach is not suitable for the modern vision of smart spaces because smart spaces should be extendable for future needs and the future functionality of the system is not known at the design time. That is, to be suitable for modern smart spaces, interoperability solutions should enable reusability of services and information within a smart space.

The need for reusable interoperability technologies that provide means for accessing services and information hosted in remote computers has been present since the popularisation of personal computer networks in the 1980s. The Remote Procedure Call (RPC) (Birrell 1984) (since circa 1981) and the Hyper Text Transfer Protocol (HTTP) (Fielding *et al.* 1999) (since circa 1989) are the most notable solutions that have been developed to match this need. In the field of M2M communication in turn, the need for reusable interoperability solutions has led to

the design of technologies such as Devices Profile for Web Services (DPWS) (Driscoll & Mensch 2009), Simple Object Access Protocol (SOAP) (Gudgin *et al.* 2003), Universal Plug and Play (UPnP) (Contributing members of UPnP forum 2008), OSGi (Hall *et al.* 2011), Constrained Application Protocol (CoAP) (Shelby *et al*. 2013), and Message Queuing Telemetry Transport (MQTT) (IBM & Eurotech 2010).

In the models presented by Tolk, Pantsar-Syväniemi *et al.* and Lappeteläinen *et al.*, these M2M interoperability technologies belong mainly to the *syntactic*, *communication* and *service* levels. When compared with the proprietary methods used in early home automation systems, these technologies represent progress towards the right direction as they provide more reusable solutions for accessing device capabilities. However, these technologies are still not feasible for providing semantic-level interoperability in smart spaces because they lack reusable mechanisms for representing the semantics of data (*i.e.*, these technologies do not provide means to define concepts in terms of other concepts). In addition to providing a reusable format for information representation, these types of technologies that would enable concepts to be defined in terms of other concepts, would enable devices to interpret the meaning of concepts unknown for them at the design time much in the same way as we humans use encyclopaedias to learn new terms. As a result, we could have agents capable for "serendipitous interoperability" (Lassila 2007), which basically means that agents are able to utilise data and capabilities that were not considered when these agents were designed.

Unlike Shannon's information theory focusing on connectivity-level communication, there is no universal and commonly accepted theory for semantic-level communication (Checkland 1999). Despite the lack of a uniform semantic information theory, the research especially in the field of knowledge representation (KR) has produced many solutions that can be used to solve semantic-level interoperability challenges in smart spaces. KR is a subfield of artificial intelligence that focuses on representing information about the world in a way that allows devices to easily interpret it. In its early days, the focus in KR research was on technologies that enable computers to translate speech or text from one human language to another. This research led to the advent of structures called "semantic networks" (Ross Quillian 1967, Woods 1975). Semantic networks (also called concept networks) are directed or undirected graphs consisting of vertices and edges. Vertices represent concepts and edges represent the relations between concepts. Frames (Minsky 1974, Fikes & Kehler 1985) are another early initiative for KR. In frame-based systems, a frame represents a concept or an object. The

attributes of the object are modelled as slots that can be altered to make the frame match the situation at hand. The possible values of slots can be restricted with facets, and it is possible to use inheritance mechanisms to inherit slots from one frame to another.

The attempt to formalise semantic networks and frame-based systems led to the work on description logics (DL) (Baader *et al.* 2003). DL is a family of knowledge representation languages that provide formal logic-based semantics for describing concepts and roles (*i.e.*, relations). The semantics of DL are based on the first-order predicate logic, but it is designed to be more practical for modelling purposes. A DL-based knowledge base consists of two components: TBox and ABox. TBox specifies the terminology for the application domain in the form of concept and role definitions. ABox contains TBox-compliant assertions about individuals.

In addition to the abovementioned technologies, many other methods for KR have been proposed during its history. Some of these technologies such as neural networks (McCulloch & Pitts 1943), automated theorem proving (Duffy 1991), fuzzy logics (Zadeh 1965), and expert systems (Jackson 1998) are also still widely studied. Semantic Web technologies are a more recent advancement in the field of KR and build on top of the semantic networks, frames and DL. The fundamental ideas behind the Semantic Web and Semantic Web technologies utilised in the work are discussed in more detail in the next section.

2.3 Semantic Web

To better understand the idea behind the Semantic Web, a comparison with the traditional Web needs to be made. As stated by Tim Berners-Lee (1996), the fundamental idea of the Web is to provide a universal solution for information distribution. To date, this idea has concretised very well and the Web is used as a medium for all kind of information sharing in numerous human languages. The current Web has its shortcomings, however. One of the problems with the Web is that it is mainly used for distributing information in a human-interpretable format. Because of this, it is difficult to develop autonomous software agents that are able to provide advanced services that utilise the heterogeneous information on the Web. For example, agents could search the cheapest suitable parts for your antique motorcycle, or answer trivia questions such as “What is the total population of Finland’s neighbouring countries?” To perform this kind of a functionality, agents need to be able both to mash-up information from various sources and to interpret

the meaning of this information. This is the role of the Semantic Web. The Semantic Web concept is summarised by Tim Berners-Lee *et al.* as follows:

> *"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."*

The Semantic Web consists of various design principles and technologies developed by collaborative working groups of the World Wide Web Consortium's (W3C) Semantic Web activity. The most relevant technologies for this dissertation (illustrated in Fig. 1) include the following standards: RDF, RDFS, OWL, and SPARQL. RDFS and OWL provide vocabularies and rules for describing concepts and relationships between these concepts (*i.e.*, ontologies). RDF provides a data model for representing the RDFS and OWL ontologies in the form of subject-predicate-object triples. SPARQL is a query language that provides means to query and manipulate data presented in the RDF format. These key technologies are described in more detail in the following sections.

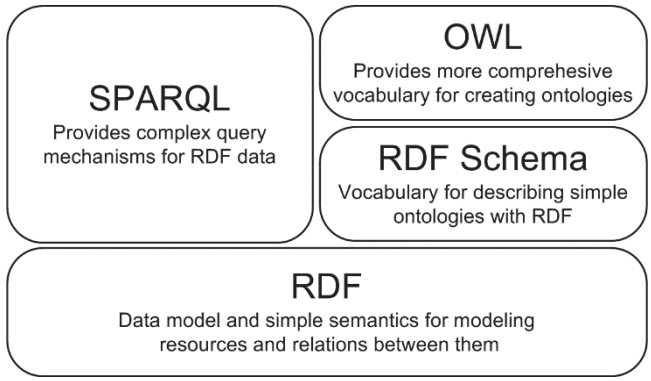

Fig. 1. Core technologies of the Semantic Web.

2.3.1 Resource Description Framework

RDF is a family of W3C specifications designed for describing Web resources in a structured manner. The first release of RDF was published as a W3C recommendation in 1999 (W3C 1999). The latest version called RDF 1.1 was released in February 2014 (W3C RDF Working Group 2014a).

RDF represents resources and their relationships with subject-predicate-object triples, also called statements. Since the subject-predicate-object format of RDF is much like a basic sentence of a human language, it is a natural way for us to make statements about information. Due to its simple and flexible structure, RDF is also an attractive solution for presenting facts about data in a machine-interpretable format. Because of these properties, the RDF data model has gained wide acceptance and forms the core building block of the Semantic Web.

A set of RDF triples is called a graph. RDF graphs are typically illustrated with a directed node-arc-node diagrams as presented in Fig. 2. The subject and object of the triple are called nodes and the predicate is called an arch. The arch, pointing always from subject to object, presents the relationship between the two nodes.

Fig. 2. Node-arch-node presentation of a single RDF triple.

RDF specifies that each component of a triple (*i.e.*, subject, predicate and object) must have a type: Uniform Resource Identifier (URI), blank node (bnode) or literal. Additionally, RDF states that the subject of a triple can only be a URI or bnode, the predicate can only be a URI, and the object can be any of the tree types. URI is a unique American Standard Code for Information Interchange (ASCII) character sequence that identifies an abstract or physical resource. In RDF 1.1, URIs are replaced with Internationalised Resource Identifiers (IRIs). IRI is a complement to the traditional URI and supports all Unicode (ISO 10646) characters allowing resource identifiers to be presented with a wider range of languages. Bnodes are used to represent resources for which real URIs do not exist. Literals in turn are used for values such as names and numbers. RDF states that literals can be either plain or typed. XML Schema data types are used for typed RDF literals.

RDF data can be presented in various ways. The official serialisation format for RDF is XML (W3C RDF Working Group 2014c). Other syntaxes proposed for presenting RDF include N-triples (W3C RDF Working Group 2014d), Turtle (W3C RDF Working Group 2014e) and Notation 3 (Berners-Lee & Connolly 2011).

2.3.2 Ontologies in the Semantic Web

Knowledge presentation in the Semantic Web is based on ontologies. Ontology is not a new concept and the term has been used in the field of philosophy for centuries[2]. The traditional definition for the term is typically related to the nature of being and existence in general. In the domain of information theory, the term "ontology" has a narrower meaning; a popular definition is formulated by Gruber (1993) as follows:

> *"Ontology is an explicit specification of a conceptualization."*

In other words, an ontology is a specification of a simplified view of real-world concepts. Ontologies consist of a taxonomy and a set of inference rules. The taxonomy part of an ontology includes definitions of relevant concepts and their relationship with each other. Inference rules define what kind of deductive reasoning can be executed to the knowledge defined in terms of the ontology.

Plain RDF provides a basic model for data presentation, but lacks methods for describing taxonomies and expressive inference rules. RDF Schema adds some of these features to RDF by providing a simple vocabulary for building ontologies. RDF and RDFS are closely associated and the term "RDF" is often used to refer to the combination of the RDF data model and RDFS vocabulary. RDFS can be seen both as an extension and as a restriction to RDF. RDFS extends the RDF data model by providing concepts such as class, subclass, domain and range, which are useful for modelling simple taxonomies. On the other hand, RDFS properties such as *rdfs:domain* and *rdfs:range* can be used to restrict the open data model of RDF by defining the allowed[3] values for a domain and range of a property.

OWL is the *de facto* ontology language of the Semantic Web. It builds on top of RDFS by providing a more comprehensive vocabulary for defining ontologies. There are two versions of OWL available. The newest version, OWL 2, was completed in 2009 and it is backward compatible with the OWL 1 specification published in 2004 (W3C OWL Working Group 2004). The semantic specification of OWL 2 consists of two separate specifications: Direct Semantics and RDF-Based Semantics. These specifications provide two alternative approaches to attaching meaning to OWL 2 ontologies. Direct Semantics is compatible with the

[2] According to a Merriam-Webster dictionary, the term "ontology" was first used circa 1721.

[3] Philosophically, RDFS does not restrict what is allowed to be used as a domain or range of a property but instead what will be interfered based on these values. In practice this is the same thing, however, because with invalid values, the interference will produce untrue results.

model-theoretic semantics of the SROIQ descriptions logic (Horrocks *et al.* 2006). The term "OWL 2 DL ontologies" (from Description Logic) is used to denote ontologies that satisfy the syntactic conditions of OWL Direct Semantics. RDF-Based Semantics extend the semantic conditions of RDF by defining a precise formal meaning for every RDF graph. The term "OWL 2 Full ontologies" is used for RDF graphs interpreted using OWL 2 RDF-Based Semantics.

OWL 2 also provides three profiles (*i.e.,* sublanguages): OWL 2 EL, OWL 2 QL and OWL 2 RL. These profiles are syntactic subsets of OWL 2, designed both to simplify the implementation and to offer performance advantages in specific application domains. OWL 2 EL is meant for applications that utilise large and complex ontologies. Its semantics are very close to OWL 2 Direct Semantics with just a few simplifications in the data model. OWL 2 QL is targeted for applications where query answering is the highest priority. It is designed so that it can be implemented using conventional relational database systems such as Structured Query Language (SQL). OWL 2 RL is designed to provide scalable reasoning without significantly limiting the expressive power of the language.

In addition to these ontology languages, W3C working, interest and incubator groups have designed various domain-specific ontologies for the Semantic Web such as Organization Ontology (W3C Government Linked Data Working Group 2014a), Data Catalog Vocabulary (W3C Government Linked Data Working Group 2014b) and vCard Ontology (Iannella R & McKinney 2014), to name a few. For this dissertation, the most relevant domain-specific ontology is the Semantic Sensor Network (SSN) ontology (W3C Semantic Sensor Network Incubator Group 2011).

The SSN ontology is designed for describing sensors, sensing methods and observations. For modularity and reusability reasons, concepts such as units of measurements, sensor type hierarchies and locations that are related but not only sensor-specific are out of the scope of the ontology. The SSN ontology is based on the Stimulus-Sensor-Observation (SSO) pattern (Janowicz & Compton 2010), which is further extended with the following perspectives: sensor, observation, feature, and system. As the name implies, the SSO pattern is based on the stimuli, sensor and observation concepts. "Stimuli" represent a change or state in a physical environment that can be detected by a sensor. "Sensor" is a physical object that observes the environment by detecting a stimulus and transforming it into a sensor output. "Observation" provides a context for interpreting the sensor stimuli by linking the observed feature, act of sensing, stimulus, sensor, sensing method, and result. The sensor perspective extends the SSO pattern by providing a vocabulary to represent sensor capabilities. The observation and feature perspectives in turn

extend the observation concept provided by the SSO pattern. Finally, the system perspective enhances the SSO pattern by providing means to represent deployments and operating conditions of sensors.

2.3.3 Accessing and manipulating Semantic Web data

Several languages for querying RDF data have been proposed. A good comparison of early RDF query languages, including RDF Data Query Language (RDQL), Sesame RDF Query Language (SeRQL), TRIPLE and Versa, is presented by Haase *et al.* (2004). More recent examples of RDF query languages include Wilbur Query Langue (WQL) (Lassila 2006) and SPARQL. RDF query languages are typically based either on relational or on path queries. The main difference between them is that relational query languages are designed for finding patterns of (possibly unconnected) RDF triples from an RDF graph whereas path queries are designed for finding paths between two nodes in an RDF graph. The main advantage of path query languages is that they can be used to present infinite-length paths and are thus capable of computing a transitive closure. That is, path query languages support natively simple RDFS/OWL reasoning (*e.g.* subclass/sub-property and *owl:TransitiveProperty*). With relational query languages, on the other hand, this type of reasoning needs to be provided by an external reasoning engine. RDQL and SPARQL version 1.0 (W3C SPARQL Working Group 2008) are typical examples of SQL-like languages based on relational algebra. WQL and Versa in turn are examples of RDF path query languages.

SPARQL 1.1 has gained the status as the official query and update language for RDF data. The SPARQL 1.1 specifications consist of eleven W3C Recommendations listed in Table 1. Of these specifications, SPARQL 1.1 Query Language (W3C SPARQL Working Group 2013a) and SPARQL 1.1 Update (W3C SPARQL Working Group 2013b) are the most relevant for this dissertation and they are thus discussed in more detail.

Table 1. SPARQL 1.1 Recommendations.

Document	Description
SPARQL 1.1 Overview	The document provides an overview of W3C specifications that facilitate querying and manipulating RDF data.
SPARQL 1.1 Query Language	The document describes the syntax and semantics of the SPARQL query language.
SPARQL 1.1 Update	The document presents the official update language for RDF data.
SPARQL 1.1 Protocol	The document describes how SPARQL 1.1 query/update requests and results are transferred between a client and a SPARQL processor service on top of HTTP.
SPARQL 1.1 Service Description	The document provides a vocabulary to describe and a method to discover SPARQL processor services.
SPARQL 1.1 Federated Query	The document describes a SPARQL query extension for expressing queries across different RDF data sources.
SPARQL 1.1 Entailment Regimes	The document describes several Semantic Web entailment regimes that can be used in SPARQL 1.1 operations to redefine the evaluation of basic graph pattern matching.
SPARQL 1.1 Graph Store HTTP Protocol	The document describes an alternative HTTP-based interface for managing a collection of RDF graphs.
SPARQL 1.1 Query Results CSV and TSV Formats	The document describes how SPARQL SELECT query results can be expressed with CSV and TSV formats.
SPARQL 1.1 Query Results JSON Format	The document describes a JSON serialised format for representing SPARQL SELECT and ASK query results.
SPARQL Query Results XML Format (Second Edition)	The document describes an XML format for representing SPARQL query results.

SPARQL 1.1 Query Language defines four types of query forms: SELECT, CONSTRUCT, ASK and DESCRIBE. All these query forms obtain results by matching a group graph pattern against an RDF dataset. The SELECT query returns the bound variables as such. The CONSTRUCT query returns an RDF graph constructed by substituting variables in a set of triple patterns with bound variables obtained from the pattern matching. The ASK query returns a Boolean value

indicating whether the query has a solution or not. The DESCRIBE query returns a single RDF graph that contains data about the requested resource. Compared to SPARQL 1.0, the new features in SPARQL 1.1 Query Language include path queries, aggregates, subqueries, and value assignment.

The SPARQL 1.1 Update language defines the means to manipulate RDF data. These operations are classified into two groups: graph management and graph update. Management-type operations include CREATE, DROP, COPY, MOVE and ADD. As the name of these operations imply, the graph management operations provide mechanisms for creating, destroying, moving and copying named graphs, or adding the contents of one graph to another. In contrast to management operations used for operating at the graph level, the update operations provide means to modify RDF triples inside existing graphs. These operations include the INSERT DATA, DELETE DATA, DELETE/INSERT, LOAD and CLEAR operations. The INSERT DATA and DELETE DATA operations operate on concrete RDF triples, meaning that the use of variables is prohibited. Blank nodes are also not permitted in the DELETE DATA operation. The advantage of having specific operations for concrete data is that large, pure-data updates can be done without the need to first query bindings to variables. The DELETE/INSERT operation is used to modify triples in the graph store based on bindings obtained via query pattern matching. It is possible to omit either the DELETE or INSERT part of the operation, in which case only the remaining part of the operation is executed. The LOAD operation is used to insert triples from an RDF document specified by a URI into a graph store. The CLEAR operation is used to remove all triples from specified graphs.

3 Semantic Web-based interoperability frameworks for smart spaces

This chapter presents existing frameworks that utilise Semantic Web technologies for enabling interoperability in smart spaces. These semantic interoperability solutions can be classified into two categories. The first category contains approaches that use Semantic Web technologies to improve interoperability solutions, in which device interaction is based on *service/syntactic*-level interoperability technologies. These types of solutions are presented in Sections 3.1–3.5. The second category contains approaches, in which semantic-level communication is mainly based on Semantic Web technologies. These approaches are presented in more detail in Sections 3.6–3.10. A comparison between the state-of-the-art solutions described in this chapter and the semantic interoperability framework proposed in this dissertation is presented in Chapter 6.

3.1 Task Computing Environment

Task Computing Environment (TCE) (Masuoka *et al.* 2003) is one of the first initiatives towards Semantic Web-based interoperability in smart spaces. The aim of TCE is to enable users to focus on tasks they are interested in by taking care of technology-specific issues related to accomplishing the tasks. To this end, TCE provides means for discovery, composition and execution of services as well as for execution and reuse of tasks.

The TCE architecture consists of the Semantically Described Service, Semantic Service Discovery Mechanism, Task Computing Client and Service Control components. The service descriptions are based on Web Service and Semantic Web technologies. Web Service Description Language (WSDL) (Christensen *et al.* 2001) is utilised for low-level service descriptions. RDF, OWL and DAML-S (Ankolekar *et al.* 2002) in turn are exploited for high-level service descriptions. Semantically Described Services available in a system are discovered with UPnP Service Discovery Mechanism. During this process, the DAML-S description of the Semantically Described Service is transferred into a Task Computing Client that provides users with means to create tasks by combining functionalities of services with suitable descriptions. The actual client-service interaction is based on the SOAP technology. The optional Service Control component is used for managing and configuring services.

3.2 COCOA

The COnversation-based service COmposition in pervAsive computing environments with Quality of Service (QoS) support (COCOA) (Mokhtar *et al.* 2007) is an integral part of the Amigo architecture (Thomson *et al.* 2008) targeted to provide interoperability between heterogeneous services and devices in a smart home domain. Similarly to TCE, COCOA utilises Semantic Web and Web Service technologies to enable discovery and composition of services.

The COCOA solution consists of three distinct contributions called COCOA-L, COCOA-SD and COCOA-CI. COCOA-L is OWL-S (Martin *et al.* 2004)-based language for modelling capabilities, conversations and QoS properties of services. Service conversations are represented as finite-state machines modelled with the OWL-S process model. COCOA-SD defines mechanism for QoS-aware service discovery. The basic idea in COCOA-SD is to find a service whose capabilities are closest to the capabilities required by the user. This closeness is measured in terms of a semantic distance that specifies how many levels of classes separate two classes in the given ontology. If neither of the classes is a subclass of another, the semantic distance cannot be defined. COCOA-CI provides a mechanism for a dynamic composition of services selected with COCOA-SD. This is done by integrating the conversations of the selected services in a way that the conversation of the task required by the user can be realised.

3.3 Semantic middleware for the IoT

Song *et al.* (2010) propose a semantic middleware for enabling heterogeneous device interoperability in the IoT domain. The aims of the proposed middleware are similar to the ones presented for TCE and COCOA (*i.e.*, assists a user in executing tasks that combine capabilities of different devices by abstracting the device-specific technologies from the user). The main difference between the approaches is that the middleware proposed by Song *et al.* takes the heterogeneity of devices into account and describes how interoperability in the IoT domain can be enhanced by mapping existing technologies such as Bluetooth and UPnP into OWL-S services.

There are three phases in the approach. In the first phase, called device semanticisation, devices and services are discovered with technology-specific discovery mechanisms and necessary data is extracted from service and device descriptions in order to create OWL-S descriptions. In the second phase, called task

building, the user is assisted in the creation of tasks by proposing suitable services and the validity of the service composition is evaluated. In the third phase, called device grounding, the task built by the user is executed by invoking the technology-specific service implementations.

3.4 SPITFIRE architecture

The SPITFIRE architecture for the Semantic Web of Things (Pfisterer *et al.* 2011) builds upon the ideas of the Semantic Sensor Web (Sheth 2008) and Linked Data (Bizer 2009). The SPITFIRE architecture defines a RESTful service infrastructure that provides developers with means to create IoT applications utilising data produced by Internet-connected sensors. The key idea in the SPITFIRE architecture is that higher-level states of physical entities are provided by CoAP (Shelby *et al*. 2013) services called virtual sensors, which calculate the state from raw sensor data.

The core contributions of the SPITFIRE architecture are threefold. The first contribution is an ontology for modelling physical and virtual sensors as well as associated physical-world objects. The ontology is aligned with several higher-level ontologies such as Dolce Ultralite (Gangemi 2007), the W3C Semantic Sensor Network (SSN) ontology (Compton *et al.* 2012) and Event-Model-F[4]. The second contribution is a mechanism for semi-automated creation of semantic sensor representations. The basic idea in the mechanism is that descriptions of existing sensors can be used to represent newly deployed sensors if the new sensors produce similar data as the old sensors. The approach clusters sensors into different groups, calculates the correlation between a new sensor and the groups, and computes the probability in which the sensor belongs to a specific group. The third contribution is a scalable approach to searching semantic entities based on their higher-level state. The idea in the approach is to only interact with virtual sensors for which it is probable that the state matches with the searched value. The probabilities are computed based on prediction models (*e.g.* periodic patterns and correlation of sensors) created by virtual sensors.

3.5 Architectural framework for pervasive computing services

Although Semantic Web technologies are typically used for service discovery and integration, they can also be used in other ways to enhance smart space

[4] http://www.uni-koblenz-landau.de/koblenz/fb4/AGStaab/Research/ontologies/events

interoperability platforms. Soldatos *et al.* (2007) propose an interoperability framework for pervasive computing. In the framework, an RDF database is used as a persistent storage for data produced by sensors, perceptual components (*i.e.*, data processing components) and agents.

The architecture supports the deployment of pervasive applications that consist of systems manufactured by different technology providers. The main focus in the architecture is on the integration of sensors and perceptual components. The overall architecture framework consists of three tiers: sensors, perceptual components and services. Sensors monitor the environment and provide data into the system. The data produced by the sensors is either context data (temperature, humidity, etc.) or raw data (*e.g.* audio and video streams). The role of the perceptual component tier is to process audio and video streams in order to produce context data for the system. The communication between the sensors and perceptual components is based on the NIST Smart Flow middleware (Rosenthal & Stanford 2000). The service tier consists of agents that track higher-level contextual situations and provide services for the end-user. The agent communication is based on the Foundation for Intelligent Physical Agents (FIPA) (Charlton *et al.* 2000) standards.

3.6 Context Broker Architecture for Pervasive Computing

Context Broker Architecture for Pervasive Computing (CoBrA) presented by Chen *et al.* (2004a, 2004b) is a broker-centric solution for enabling context-aware systems in smart spaces. The CoBrA framework consists of the CoBrA ontology and a context broker. CoBrA ontology (Chen *et al.* 2003) is a smart meeting ontology developed for prototyping CoBrA applications. It imports various upper-level ontologies from the Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) (Chen *et al.* 2004c) ontology suite.

The context broker provides a centralised access to the context of a smart space. It consists of four functional components: context knowledge base, context-reasoning engine, context-acquisition module, and privacy-management module. The knowledge base module, implemented on top of the Jena 2 Semantic Web framework, is responsible for providing a persistent storage for context data. The context-reasoning engine is a rule-based inference engine that refines context data by deducing new facts using OWL and domain-heuristic inference rules. The context-acquisition module provides a set of library procedures for fetching raw context data from sensors, agents and the Web. The idea is that for each new type of data source, a new source-specific context-acquisition module is implemented

into the context broker. The users' privacy policies and access rights are managed with the privacy-management module (also called the policy-management module). The agent communication is based on the FIPA standards.

3.7 Semantic Space

Semantic Space presented by Wang *et al.* (2004) is a pervasive computing infrastructure that utilises Semantic Web technologies for achieving interoperability in smart spaces. In particular, Semantic Space focuses on explicit representation of contexts, context querying and context reasoning. The Semantic Space approach consists of an ontology model for representing contexts in smart spaces and a context infrastructure that defines how the context information is shared in smart spaces.

The Semantic Space context model provides an upper-level context ontology (ULCO) modelled with the OWL vocabulary. ULCO provides a vocabulary for representing concepts that are common for all kinds of smart spaces, and the idea is that it can be extended with a domain-specific ontology when necessary. The base class of ULCO is *ContextEntity*, which has four direct subclasses: *Activity*, *ComputingEntity*, *Location* and *User*. *Activity* is the base class for all kinds of activities that may take place in a smart space. The other three classes provide the base classes for typical real-world objects that exist in smart spaces.

The context infrastructure defines how context information is created and accessed within a smart space. It consists of context wrappers, a context aggregator, a knowledge base, a query engine, and context reasoner components. Context wrapper components transform raw data provided by sensors and other devices into a semantic format. They are implemented as UPnP services that are capable of dynamically joining the smart space and multicasting their presence to other devices in the network. The context aggregator is implemented as a UPnP control point. Its role is to subscribe to the context information provided by context wrappers and update the information into the context knowledge base whenever a new event occurs. The context knowledge base provides a persistent storage for the semantic data and interfaces for the context reasoner and query engine. The context query engine in turn provides RDQL (Seaborne 2004)-based queries for applications to access the context information.

3.8 M3

M3, also known as Smart-M3 (Honkola *et al.* 2010), is a blackboard (Erman *et al.* 1980)-based semantic interoperability solution for smart spaces. It aims to provide multi-device, multi-domain and multi-vendor interoperability by combining Semantic Web technologies with a publish/subscribe-based interaction model. The basic principle of M3 is that semantic-level interoperability between devices is achieved by agreeing on a common ontology. At the connectivity level, M3 relies on existing technologies.

When compared to CoBrA and Semantic Space, the main difference in M3 is that it enables agents to subscribe to the context information. Additionally, M3 differs from CoBrA and Semantic Space solutions in its simplicity as it defines only two types of processes: Semantic Information Broker (SIB) and Knowledge Processor (KP). SIB is a passive blackboard that enables KPs to share semantic data with each other. This differs from the CoBrA and Semantic Space approaches, where the brokering entity actively subscribes to or fetches information from devices. KPs are software agents that provide the end-user with services by sharing information with each other via the SIB. The Smart Space Access Protocol (SSAP) specifies how the KP and SIB communicate with each other. Originally, the SSAP defined only non-standard query and update formats. Currently, SPARQL 1.1 query and update operations are also supported.

Several SIB reference implementations have been developed. The first SIB was implemented with Python on top of the Piglet RDF database (Lassila 2006) that supports only non-standard WQL and Triple format query and update operations. Smart-M3 (Honkola *et al.* 2010) in turn is the first official SIB reference implementation. It is implemented with the C programming language and utilises also the Piglet database for storing RDF data. RDF Information Base Solution (RIBS) (Suomalainen *et al.* 2010) is an ANSI C SIB reference implementation targeted for resource-limited devices. It is based on a native triplestore called bitcupe and provides support for non-standard query and update operations and for a limited set of SPARQL 1.0 query functionalities. In addition to the C implementations, there is a Java implementation of the SIB based on the OSGi technology (Manzaroli *et al.* 2011). The RDF storage, querying and reasoning in OSGi-SIB is based on the Jena framework[5] and Pellet OWL-DL reasoner (Sirin *et al.* 2007). Red-SIB (Morandi *et al.* 2012) is the latest SIB implementation. It has

[5] https://jena.apache.org/

been developed from Smart-M3 by substituting Piglet with the Redland RDF database. The main improvement in Red-SIB over other SIB implementations is the fast subscription processing engine, which is based on the approach presented in Publication IX.

3.9 INSTANS

INSTANS (Abdullah *et al.* 2012, Rinne *et al.* 2012) is a blackboard-style semantic interoperability solution similar to M3. Similarly to Red-SIB, INSTANS focuses on the near real-time evaluation of complex and heterogeneous events in smart spaces. The main difference between Red-SIB and INSTANS is how the processing of events is executed in practice. INSTANS proposes an event-processing engine where the evaluation of persistent SPARQL queries is based on the Rete algorithm. The Rete algorithm (Forgy 1982) is designed to provide a performance-efficient (at the expense of memory) way to match rules against facts. In INSTANS, the rules are presented with SPARQL queries and the facts with RDF triples.

A Scala programming language-based reference implementation of the INSTANS platform consist of the following modules: Control, SPARQL parser, Rete network, RDF triple store, and Garbage collector. The Control module receives SPARQL subscriptions and forwards them to the parser module for parsing. The SPARQL parsing module in turn transforms the SPARQL queries into a list of string lists where each string list corresponds to a RDF triple in the SPARQL query. The Rete module takes the string lists as input and creates a Rete network tree. Whenever new RDF triples are inserted into the system, the triples are matched against the Rete network and stored into the RDF triplestore. These triples contribute to the current state of the system and will be used to provide initial results for new subscriptions created into the system. If new facts are produced in the Rete network matching process the clients are notified about the results. The role of the Garbage collection module is to clean the RDF triplestore when necessary.

3.10 Smart Objects Awareness and Adaptation Model

In contrast to the blackboard-based semantic interoperability solutions, Vazquez et al. (2006) propose a peer-to-peer (P2P) semantic interoperability architecture for pervasive computing. This architecture, called Smart Objects Awareness and Adaptation Model (SoaM), categorises the information shared by devices into the following groups: Context Information, Capabilities, Constraints, and

Adaptation/Behavioural Profiles. Context Information refers to all information that can be used to represents the state of the physical world. Capabilities describe the perception and operation capabilities of devices. Constraints define a desired state for the physical environment. Finally, Adaptation/Behavioural profiles provide a rule-like format for representing a desired context-aware reactivity in the smart space.

A central entity in the SoaM architecture is a smart object (smobject). Smobjects are context-aware devices (or agents) that interoperate with each other in P2P manner. The smobject internal architecture consists of SmobjectBase and SmobjectAware components. The SmobjectBase components provide means for collecting raw data from the environment, transferring the raw data into a semantic format (*i.e.* Context Information), and sharing Context Information with other entities. Additionally, the SmobjectBase components provide means both for receiving Constraints from other entities and for controlling actuators attached to the smobject so that the state specified by the Constraints can be met. The SmobjectAware components enable autonomous context-aware reactivity by providing means for interpreting and executing Adaptation/Behavioural profiles.

In addition to smobjects, the SoaM architecture contains Orchestrator and SoaM client entities. The Orchestrator is an optional entity of the SoaM architecture that provides more advanced means for information processing and reasoning. Its role and functionality are similar to that of the knowledge broker in blackboard-based interoperability architectures. If the Orchestrator component is used, its main role is to process Adaptation/Behavioural profiles and transfer them into lower-level Constraints that are easier to process for smobjects. A SoaM client is a special type of an agent that provides users with means to interact with smobjects. In contrast to the other agents, SoaM does not provide any services for other entities.

4 Main results of the research

In this dissertation, the feasibility of Semantic Web technologies for achieving device interoperability in pervasive computing and Internet of Things systems is studied. The aim is to enable semantic-level interoperability in smart spaces by providing approaches and technologies that make it possible to base semantic-level communication between devices solely on Semantic Web technologies. The main results of the research work described in this dissertation include a semantic interoperability architecture, a knowledge sharing protocol for smart spaces, an approach to configuring smart spaces, and several reference implementations and demonstrations. These contributions are described in more detail in the following sections. The author's role in the research that led to these contributions is described in Chapter 5.

4.1 Semantic interoperability architecture for smart spaces

The semantic interoperability framework provides a distributed architecture (depicted in Fig. 3) that consists of several parallel smart spaces. Each smart space is typically tied to a specific physical location, but mobile smart spaces such as people and vehicles are also possible. For example, a large office building can be distributed so that each room is an individual smart space; the workers can also carry personal smart spaces in their mobile phones. The distribution of the system into smart spaces is typically done during the initial setup. However, it is also possible to launch new smart spaces at runtime when necessary.

At the heart of each smart space is a SIB[6] that provides Agents[7] with means to share information about and to interact with the physical world. Physical objects relevant for a given smart space are modelled as Virtual Entities (VEs). Each VE is identified with a URI created from a ucode (Ishikawa 2012) and represented with a set of RDF triples. The idea is that the same ucode used to identify the VE inside the RDF graph is stored into a tag attached to the corresponding physical object. By linking the physical object directly to its virtual counterpart, this approach provides a natural way to identify the object with which the user wants to interact. Since VEs are represented with RDF triples, it is easy to create VE graphs by linking VEs together with RDF predicates. The ontology used to represent VEs

[6] Here the term "SIB" means an RDF database that provides the operations defined in the KSP protocol.
[7] A capital letter is used for Agents that are part of the framework presented in this dissertation.

depends on the VE domain and the proposed architecture does not enforce any particular ontologies; the only requirement is that the ontology makes it possible to monitor and interact with the corresponding physical world objects. A simplified example of sensor and actuator VEs identified with the URIs[8] "*<urn:ucode:_1>*" and "*<urn:ucode:_2>*" is presented in Fig. 3.

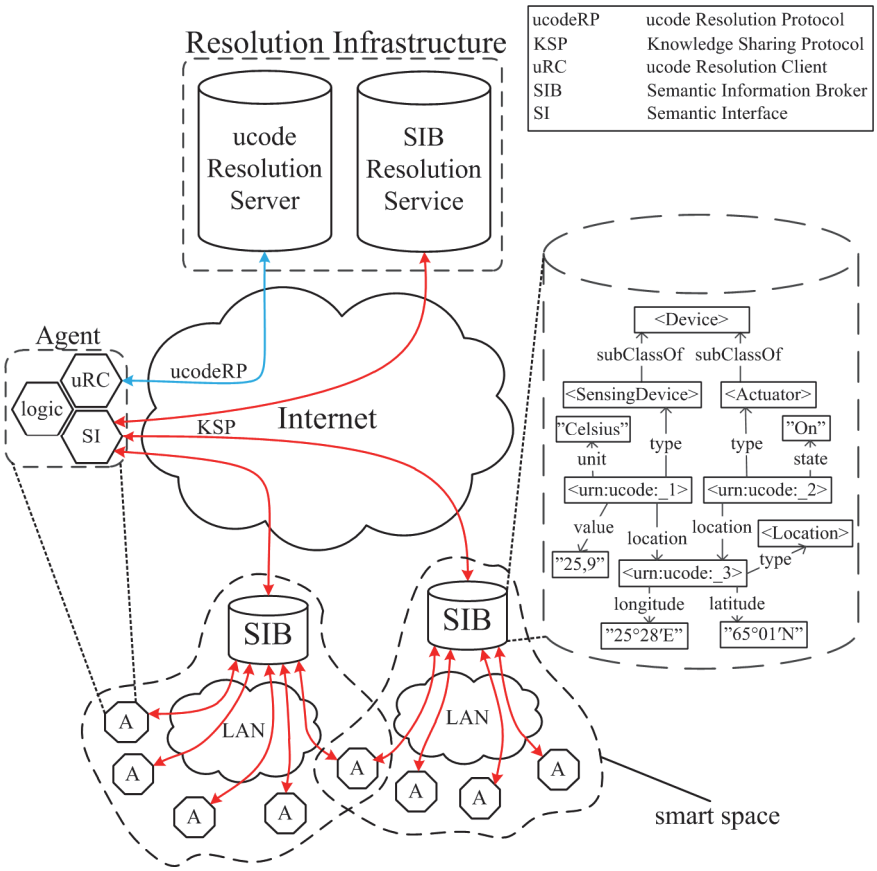

Fig. 3. Semantic interoperability architecture for smart spaces.

The granularity level, *i.e.*, whether one SIB (and smart space) is assigned for a whole city, every building in a city, every room in a building, *etc.* is a trade-off between performance and data fragmentation. To enable Agent interaction in a near

[8] The ucode-based URIs are actually 43 bytes long in hexadecimal format and they are shortened here for illustration purposes.

real-time fashion, the architecture supports fine granularity-level of smart spaces with a Resolution Infrastructure, which provides Agents with two ways to discover VEs distributed across different SIBs: *lookup* and *discovery*. The lookup operation, provided by the ucode Resolution Server, can be used to map the ucode of a VE to a SIB address. The discovery operation, provided by a SIB Resolution Service, allows more advanced means for VE and SIB resolution. In practice, the SIB Resolution Service is a special-purpose SIB, which contains a compact description of each smart space represented with the SIB service profile (SSP)[9] ontology. Agents can discover smart spaces and associated VEs by querying the SIB Resolution Service for SIB(s) that match a specification represented with a SPARQL query. The SIB descriptions are inserted into the SIB Resolution Service by a special-purpose Agent called SIB Advertiser. SIB Advertiser is also responsible for updating the description whenever it changes in any way (*e.g.* new VEs are added or the SIB location changes).

Agents that interact with each other to provide services for end-users can be divided roughly into five categories[10] based on their role in the system: sensors, actuators, aggregators, user interface, and management. The sensor category includes all kinds of Agents that are capable of monitoring the physical world and publishing their observations (represented as a VE) into the SIB. Actuators are Agents that publish their representations into the SIB, monitor the virtual world (*e.g.* state of their VE), and actuate changes into the real world when necessary. The role of aggregators is to provide end-users with applications that are relevant in the given situation by connecting the functionality provided by the sensors and actuators (*e.g.* an Agent that monitors room temperature and modifies the virtual representation of a fan actuator based on this data). The user interface category contains all Agents that provide users with means to control and obtain information about physical-world objects. The management category includes Agents that perform any kind of maintenance activities (*e.g.* garbage collection, security, network management) in a smart space. The only management Agent in the current architecture is SIB Advertiser, which has an important role in the SIB deployment and discovery.

As can be seen, the main idea in the interaction model proposed by the architecture is that all communication related to the physical world is executed by

[9] Introduced in Publication X.

[10] These are the most typical Agents but Agents that do not belong to any of these categories are also possible.

modifying and monitoring the VE graph. It should be noted, however, that some Agents (*e.g.* cameras and screens) can produce and consume data in non-semantic formats (*e.g.* raw video files or streams) that are not suitable to be stored into the SIB. To enable these types of Agents to interoperate at the semantic level, the following interaction model is proposed. Agents that provide non-semantic data into the smart space publish metadata about the content into the SIB, describe how the file (or stream) can be accessed, and subscribe to all requests made to the content. Consumer Agents can discover the non-semantic data by performing queries or subscriptions and then request the content from the Agent by publishing a request to the SIB. The vocabulary for this Agent interaction is provided by the File Sharing Ontology (FS-ONT)[11].

4.2 Knowledge sharing protocol for smart spaces

KSP is a SPARQL-like protocol for accessing and manipulating RDF data. It is designed especially for resource-constrained devices and networks. The fundamental idea with the KSP is to provide a compact alternative for the SPARQL 1.1 Query and Update protocols in the same way as CoAP (Shelby *et al.* 2013) provides an alternative for HTTP. There are three features in particular that make the KSP more suitable for resource-limited devices and networks than SPARQL 1.1 Protocol or the SSAP used in the M3 interoperability platform. First, the KSP is based on a binary format[12] that enables more compact messages to be created. Second, the KSP provides Agents with means to define the maximum size (in bytes) for SIB responses, which makes it easier to implement Agents into devices with limited memory capacities. Third, the KPS defines a persistent format for query and update operations that allows Agents both to move computational load from resource-constrained device into the SIB and to reduce traffic in resource-constrained networks.

The KSP is defined in terms of messaging models and message formats. The messaging model defines three types of messages: requests, responses and indications. Request messages can be either confirmable or non-confirmable. The KSP has been designed to be used on top of any connectivity-level protocol. The differences in the connectivity technologies are managed in the *Header* field. The messaging model also depends on the used connectivity protocol. At the moment,

[11] Presented in Publications II and VII.
[12] The customised binary format of the KSP is presented in more detail in Publication VIII.

Header fields and messaging models have been defined for the TCP, UDP and BLE protocols. In addition to protocol-specific fields such as *Length* and *Sequence number*, the KPS header contains fields such as 8-bit *Version*, 8-bit *Transaction type*, 2-bit *Request type*, and 16-bit *Transaction identifier*, which are common for different connectivity technologies.

In addition to the *Header* field, KSP messages consist of the *Data* and *Options* fields. The *Data* field contains the parameters that are specific for each KSP operation (*e.g.* SELECT, CONSTRUCT, PERSISTENT UPDATE). The operations provided by the KSP differ from the SSAP in three ways. First, all KSP operations are based on SPARQL 1.1 whereas the SSAP supports also non-standard query and update formats. Second, the SSAP provides separate *join* and *leave* operations whereas the KSP handles the security aspects for each operation separately. Third, the KSP provides a persistent update operation where the DELETE/INSERT part of SPARQL 1.1 Update is executed only if the WHERE pattern produces a solution. A central component for all KSP operations, except for the TERMINATE and the graph-level management operations, are various graph fields (*i.e.*, the Basic graph, Optional graph, Delete graph and Insert graph). Each graph field starts with the 8-bit *Triple count* field that defines the amount of RDF triples in the graph. Each individual *Triple* field in turn begins with the 3-bit *Stype*, 2-bit *Ptype* and 3-bit *Otype* fields that define the types for the following *Subject*, *Predicate* and *Object* fields. The length and structure of these fields depends on the type, and the possible types include Empty, URI, Reserved Word, Variable and Literal. For Literals a more specific subtype needs to be also specified. This is done with the first eight bits of the Literal field; possible types include xsd:string, xsd:integer, xsd:float, xsd:dateTime and xsd:Boolean.

As the name implies, the *Options* field contains parameters that can be added to request messages when needed. It starts with the 8-bit *Options count* field that specifies the number of options in the message. The first eight bits in each option are reserved for the option type and the following content varies depending on the type. Possible option types include prefix, named graph, filter, bind, maximum response size, credentials and solution modifiers. The advantages of the *Options* field are twofold. First, it provides a certain level of extendibility by allowing new features to be added as options without the need to modify other parts of the protocol. Second, options make the messages more compact because parts that are not necessary for an operation can be left out. To demonstrate the KSP message format in practice, a KSP SELECT operation with the UDP header and Filter option

is presented in Fig. 4. For the sake of simplicity, the detailed structures of the *Subject*, *Predicate*, *Object* and *Filter* fields are not depicted in the figure.

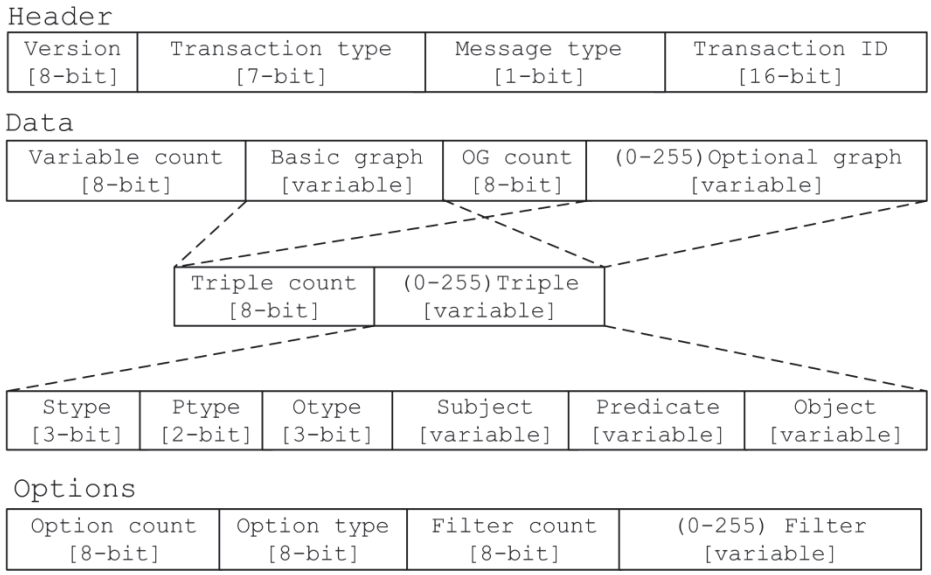

Fig. 4. Message format for the SELECT query with the UDP header and Filter option.

More information about the KSP and how it has been evaluated is available in Publication VIII.

4.3 Approach for configuring smart spaces

The main idea behind the smart space configuration approach is to present the capabilities of Agents as events and actions. These events and actions have two representation formats. The machine-readable RDFS ontology is used for devices and human language for people. In order to modify the behaviour of a smart space, users first browse events and actions available in the given smart space and then create simple rules that specify how the smart space should behave in different situations.

The approach consists of the Event and Action ontology (EA-ONT) and functional architecture. The ontology provides classes and properties to describe relevant information about events and actions (*e.g.* a human-readable description, query/update language format, possible range for values). An example of a

humidity event and a fan action modelled with the vocabulary provided by the EA-ONT ontology is presented inside the SIB in Fig. 5.

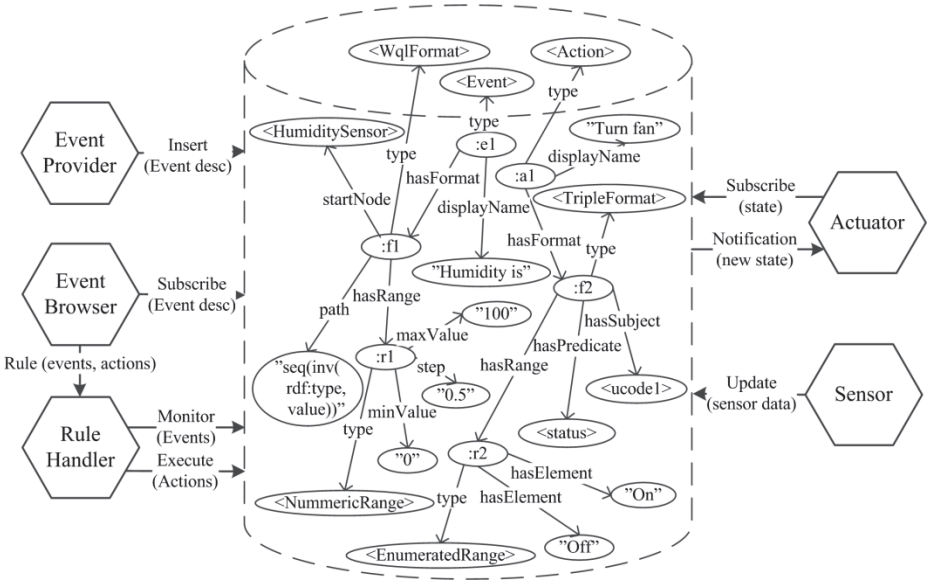

Fig. 5. Functional architecture for modifying behaviour in smart spaces.

The functional architecture (depicted in Fig. 5) for configuring Semantic Web technology-based smart spaces consist of three types of Agents: Event Provider, Event Browser and Rule Handler.

Event Providers are responsible for publishing information about events and actions into the SIB. In principle, the idea is that the Event Providers are hosted on the devices that provide sensing and actuating capabilities for a smart space, but it is also possible to create events and actions about capabilities of devices that do not include an Event Provider. This makes it possible to incorporate Agents that are not familiar with the configuration approach presented here. In practice, this can be done by 1) viewing VEs published into a SIB, 2) selecting a VE to be incorporated, and then 3) modelling possible changes in a VE attribute as an event and the ability to interact with the physical world by modifying a VE attribute as an action.

Event Browser is an Agent that displays the events and actions available in a smart space for users and enables them to define simple rules by combining these capabilities so that relevant actions are taken in right situations. When a new rule is created, it is passed to the Rule Handler that subscribes to the triggering condition

(combination of events and algebra operators) of the rule and performs actions (by executing the updates) whenever the triggering condition is met. It is noteworthy that since the events and actions are expressed with the RDF query and update languages, the Rule Handler does not need to be familiar with the domain-specific ontologies used to represent the sensing and actuating capabilities of devices. This makes the approach more generic and usable with different kinds of devices. The smart space configuration approach is described in more detail in Publications IV and VII.

4.4 Reference implementations and case studies

The research work presented in this dissertation has produced several proof-of-concept implementations and demonstrations that have been used to evaluate the proposed framework in practice. These demonstrations include Open-M3, Smart Greenhouse, Smart Meeting, Home Garden, and Smart Lighting.

Open-M3 (Eteläperä *et al.* 2010) is the first demonstration that we implemented in order to experiment with the M3 semantic interoperability solution. The demonstration contains only two types of Agents: sensors and user interface devices. The Sensor Agent, deployed on Crossbow Stargate Netbrige, receives humidity, acceleration and temperature measurements from two Crossbow sensor motes via a ZigBee radio (Farahani 2008) and publishes the measurement data into the SIB. The information published by the Sensor Agent is modelled with a simple sensor ontology developed for the demonstration. The User Interface Agents deployed on several mobile phones query the measurement data from the SIB and display it for the end-user. The interaction between the Agents and the SIB is based on non-standard RDF data access and manipulation operations provided by the SSAP. TCP/IP on top of WLAN and Ethernet are used as the connectivity-level solutions in the demonstration.

Smart Greenhouse demonstrates the interoperability between Agents in a miniature greenhouse environment. The demonstration was developed incrementally in three phases. In the first phase, three Agents, namely the Sensor Agent, Actuator Agent and Gardener Interface Agent, were implemented to study means for controlling actuators with Semantic Web technologies. The Smart Greenhouse Sensor Agent is almost identical to the one used in the Open-M3 demonstration, and its role is to provide information about humidity and temperature of the greenhouse into the Smart Greenhouse SIB. The Actuator Agent (deployed on T-Engine Teaboard2 with real-time T-Kernel OS) is responsible for

controlling LEDs, fans and a water pump attached to the miniature greenhouse. To this end, it publishes its description into the SIB and subscribes to its virtual state to be aware when other Agents want to control the physical actuators. The Gardener Interface Agent deployed on an N810 Internet tablet provides the gardener with means to monitor the wellbeing of plants and to control the state of the actuators. In the second phase, the idea was to add more autonomous behaviour into the Smart Greenhouse and the demonstration was expanded with the Autocontrol Agent (deployed on the Gumstix Verdex board) that controls the actuators on the gardener's behalf. In the third phase, the objective was to study how the NFC technology can be utilised in interaction with semantic technology-enhanced smart spaces. To this end, the different sensors, actuators and plant locations were identified with ucodes (stored into NFC tags) and the gardener was provided with an NFC reader that enabled him to interact with these objects by touching them. The different versions of the Smart Greenhouse demonstration are introduced in Publications I, III, IV, VII and X.

The Smart Meeting demonstration was developed with three goals in mind. The first goal was to demonstrate how Agents can enhance a meeting experience by taking care of the technology-specific issues on the user's behalf. The second goal was to evaluate the approach designed for sharing non-semantic content in semantic technology-based smart spaces. The third goal was to study and develop means for creating and discovering smart spaces at runtime. The demonstration consists of Meeting Agents deployed on Nokia N97, Nokia N900, Apple iPhone and Google Nexus One smart phones. These Meeting Agents provide users with means to setup and participate in meetings where they can share their contact information and files with the other participants of the meeting. Additionally, the Meeting Agent assists the user by searching additional data related to the files requested by the user (*e.g.* a contact card and other papers of an author whose paper the user has just downloaded). The Meeting application on N900 provides users also with means to create new smart spaces. The discovery of the smart spaces in the local network is based on the multicast Domain Name System (mDNS) technology. The Smart Meeting demonstration is described in more detail in Publications II and VII.

The Home Garden demonstration was developed to evaluate how resource-constrained Agents can be implemented using the WAX serialised SSAP protocol. The demonstration contains two Agents, the Active Tag and Home Garden Agent, and it continues the gardening theme introduced in the Smart Greenhouse demonstration. The Active Tag is a smart moisture sensor deployed in a low-

capacity battery-powered device (*i.e.* redwire LLC Econotag board with 32-bit ARM7 processor). The idea is that the Active Tag is attached to a potted plant where it measures soil moisture and publishes this information into the SIB. Additionally, the Active Tag will notify the user by blinking a light-emitting diode (LED) if the user is present and the soil is too dry for the given plant. Information about user presence and plant preference values for soil moisture is obtained from the SIB. The Home Garden Agent (deployed on a Google Nexus S Android smart phone) provides the user with means to monitor the wellbeing of the potted plants and notifies the user if a plant needs more water. Additionally, the Home Garden Agent is used for deploying new plants and Active Tags into the user's home. The deployment process works so that the Active Tags and potted plants are identified with ucodes stored in NCF tags and Quick Response codes (QR codes), respectively. When a new plant and Active Tag are added into the system, they are paired by reading their ucodes with a smart phone. During this process, the Home Garden Agent publishes information about the Active Tag location into the SIB and fetches information about the plant from an external database whose address is resolved using the ucode Resolution Server. The Smart Home Garden case study is described in more detail in Publications VI and VII.

The Smart Lighting system was developed both to evaluate the KSP protocol and to demonstrate how Semantic Web technologies can be applied in extremely resource-restricted devices and networks. The KSP is used as a semantic-level communication protocol between Agents and the SIB. At the connectivity level, the Agent-SIB communication is based on a BLE radio. The demonstration contains a Light Actuator and Motion Detector Agents that are deployed on an 8-bit 8051 microcontroller with 128 kB of flash and 8 kB of RAM memory. The Light Actuator provides two modes: manual and autonomous. The configuration in which the Light Actuator operates can be defined by modifying its representation inside the SIB. In the manual mode, other Agents can control the Light Actuator by updating its state inside the SIB. In the autonomous mode, the Agent monitors motion detection events and controls the lights based on this information. The motion detection events are created by the Motion Detector Agent equipped with a passive infrared sensor (PIR sensor). The Smart Lighting system demonstration is introduced in Publication X.

In addition to the abovementioned demonstrations, the research work has produced several proof-of-concept implementations, including the ECSE tool, Application Programming Interfaces (API) for Agent development, a SIB reference implementation, performance evaluation Agents, and SIB Advertiser Agent. The

ECSE tool is a reference implementation of a tool that enables end-users to configure semantic technology-based smart spaces as described in the approach presented in Section 4.3. It is implemented with the Qt programming language and contains all the three Agents required for a smart space configuration. The ECSE tool is described in more detail in Publications IV and VII. The APIs for Agent development include ANSI C libraries for the SSAP/XML and KSP protocols. The SIB reference implementation was developed to tackle the limitations of the RIBS and Smart-M3 implementations. It is implemented with C on top of an SQLite[13] database and it is the first SIB reference implementation that supports persistent SPARQL Update operation and the KSP protocol. The performance evaluation Agents have been developed for four different performance studies. In the first study (presented in Publication I), the aim was to measure latencies for non-standard RDF access and management operations provided by the SSAP. In the second and third studies (presented in Publications IX and X), the goal was to evaluate the performance for SPARQL subscription and update operations with Red-SIB. The goal of the fourth study (presented in Publication X) was to measure the performance and scalability of the SIB *discovery* and *lookup* operations.

[13] http://www.sqlite.org/

5 Summary of original papers

The research contributions of this dissertation have been presented in ten original publications. These publications include three open access journal papers, six conference and workshop papers, and one unpublished paper submitted to a journal. The author of this dissertation is the first author in six of the publications. In the other four publications, the author has been one of the main contributors to the research presented in the papers. Table 2 shows how the four research objectives introduced in Section 1.3 relate to these ten publications. In the following sections, these publications and their contributions are shortly summarised.

Table 2. Relations between the publications and research objectives.

Publication no.	Section	Research objective no.			
		1	2	3	4
I	5.1	X	X	X	X
II	5.2	X		X	X
III	5.3	X			X
IV	5.4	X			X
V	5.5	X	X		
VI	5.6	X		X	X
VII	5.7	X	X	X	X
VIII	5.8	X	X	X	
IX	5.9	X			
X	5.10	X	X	X	X

5.1 Opening information of low capacity embedded systems for smart spaces

Publication I describes a Semantic Web-inspired approach to designing embedded systems that provide a high abstraction-level interface to their sensing and actuating capabilities. The idea in the approach is that a common ontology is used for modelling relevant information about embedded systems and the physical world they interact with. All semantic-level communications with these embedded systems are executed via a SIB using the vocabulary defined in a domain-specific ontology. The connectivity-level interoperability in the proposed approach is based on the Network on Terminal Architecture (NoTA) (Lappeteläinen *et al.* 2008). A central contribution of the paper is also the KPI-low library, which is a compact ANSI C implementation of the SSAP API. It required 198 kB of flash memory in

a Gumstix Verdex pro XM4 board. In addition to the KPI-low library, the suitability of Semantic Web technologies for resource-constrained devices is improved with compact ontologies where long human-readable URIs are replaced with a shorter numeric format. The approach was evaluated by implementing the Smart Greenhouse demonstration and executing a performance study for non-standard RDF data manipulation operations provide by the SSAP.

The author of this dissertation is the first author of the paper and the scientific contributions of the paper are mostly based on his work. Janne Takalo-Mattila and Matti Eteläperä participated in the implementation of the Smart Greenhouse demonstration. Prof. Juha-Pekka Soininen guided the research and documentation of the results.

5.2 Autonomous file sharing for smart environments

Publication II describes how non-semantic content such as image, audio and video files can be shared between Agents in smart spaces. The idea in the approach is that the files are transferred directly between devices but the actual communication related to the file transfer (*i.e.*, selection of the file, the file transfer method and to which address the file is sent) is negotiated with semantic technologies via a common knowledge broker.

The approach consists of the FS-ONT ontology and an interaction model for non-semantic content sharing. FS-ONT provides a vocabulary for describing files, file sharing methods and requests to files. The interaction model for non-semantic content sharing describes how Agents can publish information about files into a smart space, subscribe to the files available in a given smart space, and perform request to files of interest. The approach was evaluated with the Smart Meeting demonstration that consisted of Meeting Agents deployed on several mobile phones.

The author of this dissertation is the first author of the publication and the scientific contributions are mostly based on his work. Kari Keinänen contributed with his knowledge on the Bluetooth communication technology. Janne Takalo-Mattila and Matti Eteläperä provided valuable feedback during the research. Prof. Juha-Pekka Soininen guided the research and provided feedback during the paper writing.

5.3 Ubiquitous computing by utilizing semantic interoperability with item-level object identification

Publication III proposes an approach for combining item-level object identification technologies with Semantic Web-inspired smart spaces. The main idea in the approach is to use ucodes for identifying physical objects and their virtual counterparts. On the physical-world side, the ucode is stored into a tag attached to the physical object. In the virtual world, the ucode is used as an identifier for the virtual object represented with an RDF graph. This approach enhances semantic technology-empowered smart spaces in two ways. First, it enables more natural user interaction by providing users with means to control devices and obtain information about objects by reading their identifiers from tags or QR codes. Second, the approach makes it easier to deploy and configure smart spaces by providing simple ways to pair physical and virtual objects with each other. The approach was evaluated in practice by implementing RFID-based object identification mechanisms into the Smart Greenhouse demonstration.

The approach presented in the paper was designed in cooperation by Janne Takalo-Mattila and the author. Takalo-Mattila is the first author of the publication and main writer of the paper and was responsible for the implementation of the Smart Greenhouse extension. The author is the second author of the publication. In addition to the contributions to the approach, he reviewed and edited the paper. Matti Eteläperä and Prof. Juha-Pekka Soininen supported the research and provided feedback during the paper writing process.

5.4 Enabling end-users to configure smart environments

The approach to enabling end-users to modify the behaviour of smart spaces is presented in Publication IV. The idea is to present the capabilities of devices as events and actions and to provide users with means for creating simple rules that use events as inputs and actions as outputs. The approach consists of the EA-ONT ontology and a system architecture. EA-ONT defines a vocabulary for representing events and actions. The system architecture in turn defines the Agents that provide end-users with means to configure smart spaces. In addition to the approach, a key contribution of the paper is the ECSE tool that implements the Agents responsible for smart space configuration. The evaluation of the approach and the ECSE tool was executed in the Smart Greenhouse environment.

The author of this dissertation is the first author of the publication and the scientific contributions are based his work. Janne Takalo-Mattila, Matti Eteläperä and Kari Keinänen provided feedback during the research. Prof. Juha-Pekka Soininen guided the research and reviewed the paper internally.

5.5 Semantic interface for resource constrained wireless sensors

In Publication V, the implementation of a resource-constrained Active Tag is presented. A central idea in the work is to demonstrate how the functionality of resource-constrained devices can be improved by enabling them to utilise context information available in smart spaces. This idea is demonstrated with the Active Tag, which receives information about plant moisture preferences and is thus able to inform the user by blinking an LED whenever the given plant needs watering. Additionally, the Active Tag monitors a graph that represents information about the user presence so that it does not blink the LED unnecessarily. The proof-of-concept implementation of the Active Tag was deployed into an ARM7-based MC13224 SoC that provides an IEEE 802.15.4-compatible radio. The implementation memory footprint is 39.7 kB and the estimated battery duration 1.3 years with two 1.5V, 2700 mAh alkaline batteries.

Arto Ylisaukko-oja is the first author of the publication. He ported the Active Tag into the resource-constrained platform and performed the power consumption and battery duration analysis. Pasi Hyttinen is the second author. He designed the WAX serialisation for the SSAP that was used in the communication between Agents and the RIBS. Additionally, he implemented the RIBS used as a knowledge broker in the demonstration. The author of this dissertation is the third author of the paper. It was his idea to improve the capabilities of resource-constrained devices by enabling them to access information in semantic knowledge bases. The author also developed the ontology used in the demonstration and implemented the Active Tag in a Linux environment to test the overall functionality of the system. Additionally, the author wrote the background section and edited other sections of the paper. Esa Viljamaa is the fourth author. He implemented the IEEE 802.15.4 communication gateway into the SIB. Prof. Juha-Pekka Soininen guided the research and provided valuable feedback during the paper writing.

5.6 A smart control system solution based on Semantic Web and uID

Publication VI proposes a novel IoT system architecture where physical object identification and resolution is based on a uID architecture and where Semantic Web technologies provide means to represent and access information in local smart spaces. The fundamental idea in the architecture is to extend the ucode-based virtual object identification approach (originally introduced in Publication III) with the functionality provided by the ucode Resolution Server so that information about a physical object can be discovered in any semantic knowledge base. A central contribution of the paper is also a proof-of-concept demonstration called Home Garden, which was used to evaluate the approach in practice. The Active Tags presented in Publication V have a central role in the Home Garden demonstration. From the architectural point of view, the main Agent in the demonstration is the Home Garden Agent deployed on an Android smart phone. It provides users with means for deploying Active Tags and monitoring the wellbeing of plants in their home garden.

Esa Viljamaa is the first author of the publication. He developed the Home Garden Agent for a Google Nexus S Android smart phone and implemented the interaction between the Agent and the ucode Resolution Server. The author is the second author. His contributions are the IoT system architecture and the ontology used to represent the physical-world objects. The author also designed the overall proof-of-concept demonstration and wrote several sections of the paper. Arto Ylisaukko-oja implemented the Active Tag used in the demonstration. Prof. Juha-Pekka Soininen gave ideas and valuable feedback during the research.

5.7 Enabling semantic technology empowered smart spaces

Publication VII is the first journal paper of the dissertation. It provides a synthesis of the research results originally described in Publications I, II, IV, V and VI. The Agent interaction in the demonstrations described in these previous conference and workshop papers is solely based on Semantic Web technologies. However, neither the overall vision behind this approach nor its benefits were analysed in these papers. In Publication VII, the approach to utilise Semantic Web technologies in all semantic-level communication is described and the advantages of the approach compared to typical interoperability platforms which utilise Semantic Web technologies only for enhancing service discovery and composition are elaborated.

As stated above, the technical contents of the paper are mainly based on the original workshop and conference papers. The fundamental challenges addressed in these previous papers are first analysed and their relevance in achieving semantic-level interoperability in smart spaces is discussed in more detail. Then the main results of these papers are presented in a broader context. Some of the approaches are also extended and improved. For instance, new ideas that support implementation of Agents into resource-constrained devices are presented. The idea to utilise a uID architecture in smart space discovery is also elaborated. Additionally, the file sharing and event ontologies utilised in the autonomous file sharing and smart space configuration approaches are presented in a formal manner using the Turtle syntax.

The author is the first author of the publication and the approaches presented in the paper are mostly based on his ideas. The only exception is the approach to use ucodes as physical object identifiers that was designed in cooperation with Janne Takalo-Mattila. Takalo-Mattila also contributed to the proof-of-concept implementations with Arto Ylisaukko-oja and Matti Eteläperä. Prof. Juha-Pekka Soininen guided the research and performed an internal review of the paper.

5.8 Knowledge sharing protocol for smart spaces

Publication VIII introduces the KSP protocol designed for communication between resource-constrained Agents and the SIB. The KSP provides SPARQL 1.1-like mechanisms to access and manipulate RDF data in a compact binary format. Additionally, the protocol makes it possible to simplify the Agent logic and reduce network traffic by providing a persistent format for update and query operations.

The KSP was evaluated by comparing its average message size to the SSAP/XML and the SSAP/WAX protocols in the Smart Greenhouse case study. The results of this evaluation show that the size of KSP request messages was on average 17% of the SSAP/XML and 36% of the SSAP/WAX message sizes in the chosen case study. The sizes of response and indication messages in turn were on average 6.89% and 19.06% of the sizes of corresponding SSAP/XML and SSAP/WAX messages, respectively. Additionally, with persistent update it was possible to implement the autocontrol Agent completely with two persistent update operations (*i.e.* it was possible to limit the network traffic to two messages).

The author is the main contributor of this publication. He designed the KSP and implemented a C API for the protocol and a new SIB reference implementation capable of handling persistent SPARQL updates. Additionally, the author

implemented the autocontrol KP with different protocols and performed the message size evaluation. Francesco Morandi and Prof. Juha-Pekka Soininen gave valuable feedback and comments during the paper writing.

5.9 A semantic publish-subscribe architecture for the Internet of Things

Publication IX describes an architecture for event processing with SPARQL 1.1. The proposed architecture is divided into three parts: clients, a system memory and a processing infrastructure. Clients monitor events by specifying subscriptions in the SELECT query format. Events can be triggered by modifying the system state with update operations. A standard RDF triple store is used as the system memory. The main scientific contribution of the paper is the processing infrastructure that provides means for near real-time processing of SPARQL subscriptions. The proposed processing infrastructure optimises (compared to the Smart-M3 SIB reference implementation) the performance of SPARQL subscription processing in three ways. First, it filters out RDF triples that cannot modify the result of the subscription. This improves the performance by reducing the amount of triples to be processed by the subscription engine. Second, it provides a separate context store for each SPARQL subscription. This improves SPARQL subscription processing by 1) restricting the amount of triples to be processed and by 2) making it possible to execute SPARQL query evaluations in parallel. Third, it provides an event processing algorithm that evaluates only how each RDF triple in the SPARQL update modifies the query result (instead of performing the whole query to obtain the new result set that is then compared to the previous set to obtain the new and obsolete results). The event processing algorithm is based on the idea that new triples can only add results and deleted triples can only remove results (with the NOT EXISTS filter pattern the behaviour is the opposite). In addition to the event processing architecture, the main contributions of the paper include a reference implementation of the architecture and a performance evaluation method consisting of a performance model, key performance indicators, a benchmark scenario and a set of experiments.

Luca Roffia is the first author and main writer of the paper. Additionally, he designed the benchmarks for the performance evaluation method and performed the performance evaluation in practice. Francesco Morandi is the second author of the paper, and the SPARQL event processing optimisations are mainly based on his ideas. He also implemented the new SPARQL processing engine into Red-SIB. The

author is the third contributor to this publication. He contributed to the design of the SPARQL event processing engine by proposing improvements to Morandi's original approach and architecture (*e.g.* ways to handle the NOT EXISTS filter pattern and ideas for event synchronisation). He also contributed to the design of the overall event processing architecture and the performance evaluation method. Additionally, he performed the state-of-the-art review, wrote several sections and edited the paper. Alfredo D'Elia, Fabio Vergari, Fabio Viola and Luciano Bononi contributed to the paper writing. Prof. Tullio Salmon Cinotti guided the research and is the main contributor for the performance evaluation approach. Additionally, he reviewed and edited the paper.

5.10 Semantic interoperability architecture for pervasive computing and Internet of Things

Publication X proposes a semantic-level interoperability architecture for pervasive computing and the IoT. The architecture is a synthesis of the results obtained in several projects over the period of five years. A key idea in the architecture is that the global IoT system is divided into numerous local smart spaces, where the interaction between Agents is based on the operations provided by the KSP. This makes it possible to achieve responsive interaction even in larger scale systems. To enable lookup and discovery of smart spaces, the architecture proposes a Resolution Infrastructure consisting of a ucode Resolution Server and SIB Resolution Service.

To demonstrate that the proposed architecture conforms to the common model for IoT architectures, we mapped central components of the architecture to the IoT-A Architectural Reference Model (ARM) (Bassi *et al.* 2013). We also present several reference implementations and applications that we have developed to test different parts of the architecture in practice. The evaluation of the architecture also includes two performance tests. In the first test, we measured latencies for update and subscription operations in an example IoT use case with different amounts of Agents and data. It was possible to execute the all operations with relatively short latencies (*i.e.*, the latency was between 0.53 ms to 46.33 ms). We also noticed that increasing the RDF triple count from 1,000 to 1,000,000 does not affect the latency of the operation if the triples are not relevant for the operation. This means that the SIB scales well with respect to different applications running on the system (assuming that the applications use a different ontology). The second test focused on the evaluation of the discovery and lookup operations with different amounts of

smart spaces. The Resolution Infrastructure reference implementation scales quite well up to 1,000 smart spaces. With 10,000 smart spaces, the latency increases dramatically, however.

The author of this dissertation is the first author of the paper. He extended the centralised interoperability architecture to the IoT domain and developed the Resolution Infrastructure, which supports the distributed architecture by enabling discovery of smart spaces and virtual entities. Additionally, he aligned the architectural components to the IoT-A ARM, executed the performance studies and contributed to several reference implementations presented in the paper. Alfredo De'Elia and Francesco Morandi are the authors of the OSGi-SIB and Red-SIB reference implementations presented in the paper. They also participated in the implementation of various applications and Agents used to evaluate the architecture. Pasi Hyttinen is the main developer of the RIBS. Arto Ylisaukko-oja and Janne Takalo-Mattila participated in the implementation of several smart space reference implementations described in the paper. Professors Juha-Pekka Soininen and Tullio Salmon Cinotti have guided the research and provided feedback in the paper writing phase.

6 Discussion

6.1 Analysis of the results

The aim of the work presented in this dissertation was to study the feasibility of Semantic Web technologies and the blackboard architectural style for achieving semantic-level device interoperability in smart spaces. In particular, four objectives were defined for the research. The first objective was to design an interoperability framework that allows devices to utilise Semantic Web technologies in all communication – even in situations that require responsive interaction or exchange of non-semantic content between devices. The second objective was to develop solutions that make it possible to exploit Semantic Web technologies in resource-constrained devices and networks. The third objective was to design a semantic interoperability framework so that it can support and provide interoperability across heterogeneous connectivity technologies utilised in smart spaces. The fourth objective was to develop natural ways for users to interact with smart spaces and to enable users to configure smart spaces so that it meets their needs. In this section, it is evaluated how the proposed framework meets these objectives.

6.1.1 Semantic-level communication based on Semantic Web technologies

The semantic-level interoperability architecture presented in this dissertation proposes an interaction model where all semantic-level communication is based on Semantic Web technologies. For sensors, this interaction model is very simple as they just publish the concrete measurement data along with the sensor descriptions into the SIB. This enables Agents subscribed to the data to monitor it in a near real-time fashion. Actuators in turn publish their virtual representation into the SIB and subscribe to events that inform when the state of the physical actuator needs to be modified. In addition to sensors and actuators, smart spaces can contain devices such as cameras and screens that produce and consume large amounts of non-semantic data (*e.g.* audio and video files). To enable these types of Agents to interoperate with each other, the framework proposes an interaction model where communication related to non-semantic content transfer is executed via the SIB, but the actual transfer of the physical file happens directly between devices.

One of the biggest challenges in the proposed interaction model is the performance of semantic information processing. Responsive interaction between different kinds of Agents in relatively small-scale smart spaces (fewer than 10 Agents) is demonstrated in the case studies presented in Section 4.4. In addition to these approximate observations, the performance of semantic information processing was evaluated with several performance tests. The performance measurements presented in Publication I show that reasonably short (under 20 ms) latencies are achieved with simple RDF data manipulation operations based on non-standard update and query formats. However, because these query and update formats do not provide means for any kind of data processing or reasoning, they are not feasible for Agents that need to process large amounts of data (*i.e.*, all the data relevant for the application needs to be transferred to the Agent). SPARQL 1.1, on the other hand, provides advanced processing mechanisms, but the latency of operations increases significantly when the query complexity and the amount of data relevant for the operation is increased. To tackle this scalability problem, the framework proposes a way to reduce the amount of data in an SIB by distributing the global graph into numerous smaller graphs that can be discovered by using the Resolution Infrastructure.

It should also be noted that the interaction model based on subscriptions and persistent updates provides a good possibility to optimise SPARQL processing. This type of an optimisation approach is presented in Publication IX. Using this approach, it was possible to achieve relatively short latencies (0.53–46.33 ms depending on the complexity of the SPARQL operation) even in larger-scale smart spaces (1 million triples and over 200 Agents) as presented in Publication X. However, it should be noted that although it is possible to achieve responsive interaction with Semantic Web technologies, the exact latency of an operation depends on many factors and it is not thus possible to guarantee fixed deadlines required in hard real-time systems.

6.1.2 Suitability for resource-constrained devices and networks

Publications I, V, VII, VIII and X present proof-of-concept implementations demonstrating that it is possible to exploit Semantic Web technologies in resource-constrained devices and networks. The best example is the Smart Lighting system where the Agents (deployed into an 8-bit 8051 microcontroller with 128 kB of flash and 8 kB of RAM memory) utilise KSP over resource-constrained BLE radio to interact with each other via the SIB. To be precise, the Motion Detector Agent

consumed 122.9 kB of flash memory (11% increase to the implementation without the KSP protocol); the RAM memory-footprint in turn was 6.3 kB (3% increase to the implementation without the KSP protocol). In addition to these reference implementations, the KSP was evaluated by comparing its message sizes to existing M3 communication protocols in the Smart Greenhouse case study. In this evaluation, the KSP messages were on average 70.09% and 87.08% shorter than the messages represented with the SSAP/WAX and the SSAP/XML, respectively.

It is also noteworthy that the blackboard architectural style with publish/subscribe-based interaction model turned out to be ideal for resource-constrained devices and networks. There are two reasons for this. First, because devices do not interact with each other directly, the resource-restricted devices are not required to serve request at any given time and can therefore save power by entering into a sleep mode when necessary. Second, because all the information relevant for a system is available in a SIB, it is possible to simplify Agent implementations by performing data processing inside the SIB. The centralised knowledge store also reduces network traffic since devices do not need to request data separately from different sources.

6.1.3 Support for heterogeneous connectivity-level solutions

The blackboard architectural style is ideal for devices that interact on top of heterogeneous connectivity solutions. This is because all communication goes through a knowledge broker, which means that it is the only device that needs to support all connectivity technologies used in a smart space. Because devices do not communicate with each other directly, it is also possible to access data and capabilities of devices that do not support the IP protocol over the Internet. It is also noteworthy, that the blackboard architecture makes it easier to secure a smart space against Internet-based attacks because the knowledge broker can act as a "firewall" and heavy security mechanisms do not need to be implemented into the resource-constrained devices.

In addition to the architecture style, the support for different connectivity-level technologies has been taken into account in the design of the KSP. The requirements of different connectivity technologies are handled by defining separate HEADER fields and messaging models for each connectivity solution. This way no overhead is caused to individual protocols and only small part of the protocol needs to be modified when the connectivity technology is changed.

Currently, header fields and messaging models have been defined for the TCP, UDP and BLE connectivity technologies.

The use of different connectivity-level technologies has also been demonstrated in the proof-of-concept implementations. The first version of the Smart Greenhouse demonstration used the NoTA as the connectivity-level solution. Later versions of the Smart Greenhouse also utilised plain TCP/IP on top of Wi-Fi and Ethernet. Bluetooth and TCP/IP on top of WLAN connectivity technologies were used in the Smart Meeting demonstration. In the Smart Home Garden, the connectivity technologies included the RIME protocol on top of an IEEE 802.15.4 radio and 3G and TCP/IP on top of WLAN. The connectivity-level interoperability in the Smart Lighting demonstration was based on a resource-constrained BLE radio.

6.1.4 Means to interact with and configure smart spaces

Various methods for user interaction with Agents have been presented in the demonstrations. The first version of the Smart Greenhouse provides the gardener with a UI on a Nokia N810 that enabled him to view the environmental status of the greenhouse, control the actuators and insert information about the plants into the smart space. The Smart Greenhouse demonstration was later expanded with object identification methods that enabled the gardener to view sensor measurements and control the actuators by touching them with an RFID-capable Ubiquitous communicator. In the Smart Home Garden, the UI provided users with means to deploy Active Tags and monitor the wellbeing of plants. In the Smart Meeting in turn, the idea in the whole demonstration was to enhance user interaction in meetings with semantic technology-empowered Agents.

In addition to the means for direct interaction with smart space devices, the objective of the research was to provide users with means to configure smart spaces. To this end, the framework provides the ECSE tool that models information and functionality of devices as events and actions and enables users to create simple rules to define how the environment should behave in different situations. According to the studies performed by Pane *et al.* (2001), this type of rule and event-based approaches are most natural for non-programmers. It should also be noted that in contrast to the above-described UIs tailored for specific ontologies, the ECSE tool is agnostic to domain-specific ontologies and can be thus seen as a more general-purpose UI for Semantic Web-based smart spaces.

6.2 Comparison to related work

Many smart space interoperability platforms (*e.g.* TCE, Amigo and SPITFIRE) exploit Semantic Web technologies for enhancing service discovery and composition. The main difference between the interoperability framework presented in this dissertation and these solutions is that all M2M communication in the proposed interoperability architecture is executed with Semantic Web technologies. This improves interoperability and makes M2M communication more flexible and easily extendable for future needs.

As presented in Sections 3.6–3.10, there also exist interoperability solutions such as CoBrA, Semantic Space, M3, INSTANS and SoaM that use Semantic Web technologies in almost [14] all M2M communications. The proposed semantic interoperability architecture and communication protocol differ from and improve these existing solutions in several ways. The first notable difference is the target domain and scalability of the architectures; that is, CoBrA, Semantic Space, M3, INSTANS and SoaM are targeted for local smart spaces whereas the distributed architecture proposed in this dissertation provides scalability through several parallel smart spaces. The second improvement is the design choice to use ucode-based identifiers for physical objects and their virtual representations. By creating a link between the physical and virtual worlds, this design choice provides users with a natural way to identify the virtual object they want their Agent to interact with. The third difference and improvement is the KSP (and related APIs) that provide means for exploiting Semantic Web technologies in resource-constrained devices and networks. The SoaM smobject has also been deployed into a relatively resource-constrained device (*i.e.*, 55 MHz ARM7 microprocessor with 8 MB of flash and 16 MB of RAM memory), but this device is still significantly more powerful when compared to the resource-constrained Agents presented in this dissertation. Additionally, in contrast to the communication protocols utilised in these existing solutions, the KSP is designed to be used on top of any connectivity-technology and it provides compact packets, as well as, mechanisms to reduce network traffic. These features make it more suitable for heterogeneous and resource-constrained networks.

[14] Device communication in M3, Semantic Space and CoBrA is not based solely on Semantic Web technologies. In M3 it is possible to use service-level solutions when feasible. Semantic Space and CoBrA in turn equip the broker with means to fetch information form sensors with non-semantic protocols.

Although the aforementioned interoperability solutions do not focus on resource-constrained devices and networks, specific approaches for this purpose have been proposed by Preuveneers and Berbers (2008) and Su *et al.* (2011). Preuveneers and Berbers propose a novel encoding scheme that provides a compact binary representation for Semantic Web ontologies. Additionally, the contributions of Preuveneers and Berbers include a fast algorithm for simple ontological reasoning (*i.e.*, sub-classes, sub-properties, and domains and ranges of properties). In principle, this encoding idea is similar (just taken a lot further) to the idea (introduced in Publication I) to replace long human-readable URIs used in classes, properties and instances with compact numeric identifiers. The Reserved Word option of the KSP also provides this type of a functionality (*i.e.*, compact binary representations for the RDFS/OWL class and property URIs). Despite these similarities, there are also fundamental differences between the two approaches. The aim of the approach proposed by Preuveneers and Berbers is to enable resource-constrained devices to store semantic data and perform simple reasoning. The idea in the framework proposed in this dissertation, on the other hand, is that Agents deployed on resource-constrained devices can exploit all advantages of Semantic Web technologies by transferring heavy computing related to semantic information processing into the SIB, which is hosted on a significantly more powerful computing platform when compared to the Agents.

Su *et al.* (2011) propose a more similar approach to the KSP, called Entity Notation (EN). EN is an RDF-like data model and lightweight serialisation format for resource-constrained devices. It provides two packet formats called complete packets and short packets, encoded as a sequence of UTF-8 characters. The complete packet consists of the type, identifier and an arbitrary number of property-value pair fields. In the short packet format, the packets transferred over a communication channel are made more compact by leaving out the parts that are constant between messages. The main difference between the KSP and EN is that the KSP is a SPARQL-like access and management protocol for RDF data whereas the EN is an RDF-like data representation format. Another notable difference is that the EN provides human-readable notations whereas the KSP is based on binary encoding.

In addition to the architecture and the KSP protocol, an important part of the interoperability framework is the approach to and tool for configuring semantic technology-based smart spaces. Semantic Space, CoBrA, M3 and INSTANS do not natively provide this type of a functionality. In the SoaM architecture, however, the Adaptation/Behavioural profiles can be used to configure smobject behaviour.

Several Semantic Web-based approaches focusing especially on configuring Semantic Web-enhanced smart spaces have also been introduced in the literature.

A similar approach to the one utilised in SoaM is proposed by Kosek *et al.* (2010). In their approach, each device is a general purpose entity whose behaviour can be configured with RDF recipe(s). The main difference between the ECSE tool and these two approaches is that with RDF recipes and SoaM Adaptation/Behavioural profiles all devices are configured separately whereas with the ECSE tool the configuration is executed by a single Agent (*i.e.*, the Rule Handler). Consequently, the RDF recipes and SoaM Adaptation/Behavioural profiles are more suitable when the behaviour of each device needs to be defined in high detail. The main drawback in these approaches is that all devices need to both be familiar with the corresponding ontologies (*i.e.*, the Recipe ontology and Adaptation/Behavioural profile ontology) and have enough resources for processing these ontologies. With the ECSE tool, on the other hand, it is possible to also utilise resource-constrained devices and functionalities of devices that are not familiar with the configuration approach.

In addition to RDF recipes and SoaM Adaptation/Behavioural profiles, there are smart space configuration approaches such as the Smart Modeller (Katasonov & Palviainen 2010a), Framework for End-user Programming of Cross-Smart Space Applications (Palviainen *et al.* 2012) and *semantic connections* (Niezen *et al.* 2010, and van der Vlist *et al.* 2010) designed especially for blackboard-based semantic interoperability systems. The Smart Modeller enables application developers (and end-users (Katasonov 2010b)) to modify the behaviour of smart spaces by providing a graphical UI for ontology-driven development of applications. The focus in the Smart Modeller is a bit different from the ECSE tool. That is, ECSE is a runtime tool that enables users to modify the behaviour of smart spaces by mashing up events and actions provided by existing agents. The Smart Modeller, on the other hand, is mainly a design-time tool that provides means to create executable programming code for new agents.

To address the limitations of the Smart Modeller as an end-user tool, Palviainen *et al.* (2012) propose a novel script language, called RDFScript, and a framework for end-user programming. Although the approach is simpler to use than the native Smart Modeller, it is still more complex (and expressive) than the approach proposed in this dissertation. Another notable difference between the approaches is that in our approach the interaction with devices is executed directly by modifying their representations inside the SIB whereas the framework proposed by Palviainen

et al. uses the SIB only for transferring XML format messages between Driver and Application Executor components.

The approach proposed by Niezen *et al.* (2010) and van der Vlist *et al.* (2010) enables smart space configuration by providing means to manipulate *semantic connections* (both physical and conceptual) that exist between devices. In their early work, the RFID reader-equipped *interaction tile* component was used for representing, testing and creating semantic connections. In their later work, (van der Vlist *et al.* 2013), they propose two alternative means (based on augmented reality and tangible interaction technologies) for the same task. The semantic connections approach has many similarities to the interaction and configuration methods proposed in this dissertation (*e.g.* both utilise item-level object identification methods and model capabilities of devices as events). The main difference between the two approaches is how the events are represented in the ontology. In the ECSE tool, events are represented with query and update language patterns. The event ontology utilised in the semantic connections approach, on the other hand, models each event type as a separate class. The advantage of the event-modelling approach used in the ECSE tool is that the event-processing engine can use a query engine for event processing and no modifications are thus needed when new types of events are created.

6.3 Limitations and recommendations for future work

Although the scientific contributions presented in this dissertation provide key building blocks for enabling semantic-level interoperability in smart spaces, there are still several areas in the framework that can be improved and developed further.

The first area for improvement is the KSP, which has some limitations when compared to the SPARQL 1.1 Query and Update protocols. For instance, the current version of the KSP does not support SPARQL 1.1 features such as path queries, aggregates and sub-queries. In addition to adding support for the missing SPARQL 1.1 features, an interesting topic for future work would be developing completely new types of operations that would improve the capability and performance of event processing. One idea to this end is to use persistent query and update operations that are evaluated only when a certain part of the WHERE pattern is modified. The main advantages of these fine-grained operations are twofold. First, they will allow monitoring new types of events (*e.g.* send me a description of a device only at the moment when a certain malfunction happens). Second, the new operations will make it possible to optimise SPARQL processing as only a subset

of the WHERE pattern needs to be monitored. In addition to the fine-grained subscription and persistent update operations, transactions that will provide means to monitor events in a specific temporal order would be useful additions to the KSP.

At the architectural level, important future development areas are mainly related to two components. The first area for improvement is the performance and scalability of the SIB Resolution Service. This is essential because with more than 1,000 smart spaces, the latencies of the resolution operations provided by the current SIB Resolution Service implementation are not feasible (as presented in Publication X). There are many ways to improve the performance. A simple option is to investigate the feasibility of a SPARQL Service that supports geographical indexing of GeoSPARQL (Battle & Kolas 2011) as the SIB Resolution Service. Another option is to improve the performance by implementing a distributed version of the SIB Resolution Service. This option requires more work and could be done, for example, by adopting a similar approach to the Domain Name System (DNS) servers where geographical distribution of the servers is used. The second future development area in the architectural components is the performance of subscription and persistent update processing. This is true despite the fact that several high-performance SPARQL subscription processing solutions, such as Red-SIB (Publication X), INSTANS (Rinne *et al.* 2012), C-SPARQL (Barbieri *et al.* 2010), EP-SPARQL (Anicic *et al.* 2011), CQELS (Le-Phuoc *et al.* 2011), Sparkwave (Komazec *et al.* 2012), SENS (Murth M & Kühn 2009a) (Murth M & Kühn 2009b) and EventCloud (Pellegrino *et al.* 2013), have already been proposed in the literature. The importance of this topic, however, makes it still one of the main areas for future work. One possibility for improving the performance of SPARQL processing are the fine-grained operations discussed above. Additionally, since subscription processing in Red-SIB is executed in parallel, the performance could be improved by deploying the SIB into a cloud where a separate central processing unit (CPU) core is assigned for each SPARQL subscription evaluation process. Edge computing platforms such as the Nokia Radio Applications Cloud Server (RACS) [15] would also be an interesting environment for hosting the knowledge broker as it would provide smart space devices with lower communication latencies when compared to traditional cloud infrastructures.

In addition to the architectural components and the KSP protocol, the approach for modifying the behaviour of smart spaces also contain some limitations and areas that can be improved. For instance, the ECSE tool was designed when the SIB

[15]http://networks.nokia.com/portfolio/liquid-net/intelligent-broadband-management/liquid-applications

supported only non-standard query and update languages. A simple improvement to the ECSE tool is thus to extend it with support for SPARQL 1.1 format events and actions. Another interesting future work area is to design completely new ways for users to modify and extend the functionality of their smart spaces. The main limitation in the current approach (and in the ECSE tool) is that it might still be too complex for typical non-expert users. Consequently, it would be useful to provide users with simpler ways to modify the functionality of their smart spaces. There are, of course, many ways to do this. An interesting idea is to enable users to configure their personal smart spaces by downloading and installing simple applications (implemented with persistent SPARQL update rules) from application stores in the same way they can configure and extend the capabilities of their smart phones, tablets and computers at the moment.

This idea is actually quite interesting as it would open a totally new market for third-party application developers. In order to realise this idea, there is a need for technologies that make it possible to represent capabilities distributed over a heterogeneous group of devices as a uniform virtual computing platform. The semantic interoperability framework presented in this dissertation is an important part of this platform as it provides developers with a high abstraction-level interface to the capabilities of devices. What is further needed is an approach to managing access to sensor, actuator and other type of resources available in a smart space so that applications do not interfere with each other and the functionality of the whole smart space is optimal. This type of an approach will extend the mixed criticality system (MCS) ideas (elaborated, for example, by Hill and Lake (2000) and Vestal (2007)) from closed systems to the IoT domain where resources and applications are not necessarily known at the design time. In addition to the approach for mixed criticality management in the IoT domain, the virtual computing platform requires flexible solutions for security and data management as well as tools to support the development, deployment and management of the devices and applications running on the platform.

7 Summary

Smart spaces are physical environments (*e.g.* smart homes, cities, offices, vehicles) equipped with networked devices which provide people with relevant services in their everyday life. A key characteristic of smart spaces is that devices interact with each other autonomously in different kind of situations without human assistance. To make this possible, there is a need for generic technologies that provide devices with means to interoperate at the semantic level.

In this dissertation, the suitability of Semantic Web technologies for device interoperability in smart spaces has been studied. The M3 semantic interoperability solution was taken as a starting point for the research, and the aim was to develop an interoperability framework that addresses the following four requirements: First, to ensure that information sharing between agents is executed at the semantic level, it was required that the communication between devices is based solely on Semantic Web technologies. Second, to enable the deployment of the framework into typical pervasive computing and IoT environments, it was required that the solution should be suitable for resource-constrained networks and devices. Third, because devices utilise several connectivity technologies, the interoperability framework should be agnostic to the connectivity technologies used by different devices. Fourth, to enable end-users to easily adopt the technologies and solutions, the framework should provide means for users to both seamlessly interact with and to modify the behaviour of smart spaces.

The main contribution of this dissertation is the semantic interoperability framework that consists of three individual contributions. The first contribution is the semantic-level interoperability architecture for smart spaces. The interoperability architecture is built on of the M3 concept, and the main additions and improvements to the original M3 functional architecture are twofold. First, a new interaction model in which all communication is based on semantic information sharing; the original M3 concept is more open and does not restrict the use of service-level solutions. The interaction model describes how to interact with different types of devices, including devices that produce data (*e.g.* large files) not feasible to be shared via the SIB. Second, a new distributed architecture that improves the performance and scalability of Agent interaction by dividing the worldwide smart space into numerous local smart spaces. A key component of the distributed architecture is the Resolution Infrastructure that enables the discovery of smart spaces and virtual entities all over the world. The second main contribution of the interoperability framework is the KSP that provides SPARQL-like

mechanisms to access and manipulate RDF data in a compact binary format. In addition to the standard SPARQL 1.1 operations, the KSP defines persistent query and update operations that make it possible to simplify Agent logic and reduce traffic in the network. The third component of the interoperability framework is the approach to (and tool for) modifying the behaviour of semantic technology-based smart spaces. The tool enables users to view device capabilities in their environment and provides them with means to create simple rules that defined how the smart space should behave in different situations. In addition to the interoperability framework, the dissertation contributions include five demonstrations, several reference implementations of the architectural components, and performance studies that have been used to evaluate the contributions in practice.

References

Aarts E, Harwig R & Schuurmans M (2002) Ambient Intelligence. In: Denning PJ (ed) The Invisible Future. New York, NY, USA: McGraw-Hill, Inc: 235–250.

Abdullah H, Rinne M, Törmä Seppo & Nuutila E. (2012) Efficient Matching of SPARQL Subscriptions using Rete. Proc 27th Annual ACM Symposium on Applied Computing. New York, NY, USA: ACM: 372–377.

Andrews JG, Buzzi S, Wan Choi, Hanly SV, Lozano A, Soong ACK & Zhang JC. (2014) What Will 5G Be? IEEE Journal on Selected Areas in Communications 32(6): 1065-1082.

Anicic D, Fodor P, Rudolph S & Stojanovic N. (2011) EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. Proc 20th International Conference on World Wide Web. New York, NY, USA: ACM: 635–644.

Ankolekar A, Burstein M, Hobbs JR, Lassila O, Martin D, Mcdermott D, Narayanan S, Mcilraith SA, Paolucci M, Payne T & Sycara K. (2002) DAML-S: Web Service Description for the Semantic Web. In: Horrocks I & Hendler J (eds) The Semantic Web — ISWC 2002. : Springer Berlin Heidelberg: 348–363.

Ashton K. (2009) That 'Internet of Things' Thing. RFID journal. URI: http://www.rfidjournal.com/articles/view?4986. 2014/3/18. Cited 2014/6/20.

Baader F, Calvanese D, McGuinness DL, Nardi D & Patel-Schneider PF. (2003) The Description Logic Handbook: Theory, Implementation, and Applications. New York, NY, USA: Cambridge University Press.

Barbieri DF, Braga D, Ceri S & Grossniklaus M. (2010) An Execution Environment for C-SPARQL Queries. Proc 13th International Conference on Extending Database Technology. New York, NY, USA: ACM: 441–452.

Bassi A, Bauer M, Fiedler M, Kramp T, van Kranenburg R, Lange S & Meissner S. (2013) Enabling Things to Talk - Designing IoT Solutions with the IoT Architectural Reference Model. Heidelberg, New York, Dordrecht, London: Springer.

Battle R & Kolas D. (2012) Enabling the geospatial Semantic Web with Parliament and GeoSPARQL. Semantic Web 3(4): 355–370.

Berners-Lee T & Connolly D. (2011) Notation3 (N3): A readable RDF syntax, W3C Team Submission. URI: http://www.w3.org/TeamSubmission/n3/. Cited 2014/9/9.

Berners-Lee T, Hendler J & Lassila O. (2001) The Semantic Web. Scientific American. 284(5):34-43: 29–37.

Berners-Lee T. (1996) WWW: Past, Present, and Future. IEEE Computer 29(10): 69–77.

Birrell AD & Nelson BJ. (1984) Implementing Remote Procedure Calls. ACM Trans.Comput.Syst. 2(1): 39–59.

Bizer C, Heath T & Berners-Lee T. (2009) Linked data - the story so far. International journal on semantic web and information systems 5(3): 1–22.

Boswarthick D, Hersent O & Elloumi O. (2012) M2M Communications: A Systems Approach. Oxford: Wiley-Blackwell.

Brumitt B, Meyers B, Krumm J, Kern A & Shafer SA. (2000) EasyLiving: Technologies for Intelligent Environments. Proc Second International Symposium on Handheld and Ubiquitous Computing (HUC). Bristol, UK: 12–29.

Card SK, Moran TP & Newell A. (1980) The Keystroke-level Model for User Performance Time with Interactive Systems. Communications of the ACM 23(7): 396–410.

Charlton P, Cattoni R, Potrich A & Mamdani E. (2000) Evaluating the FIPA Standards and their Role in Achieving Cooperation in Multi-Agent Systems. Proc IEEE 33rd Annual Hawaii International Conference on System Sciences: Hawaii, USA: 1–10.

Checkland P. (1999) Systems Thinking, Systems Practice: Includes a 30-Year Retrospective. John Wiley and Sons Ltd.

Chen H, Finin T & Joshi A. (2004b) Semantic Web in the Context Broker Architecture. Proc 2nd IEEE Annual Conference on Pervasive Computing and Communications (PerCom). 277–286.

Chen H, Finin T, Joshi A, Kagal L, Perich F & Chakraborty D. (2004a) Intelligent agents meet the semantic Web in smart spaces. Internet Computing, IEEE 8(6): 69–79.

Chen H, Finin T & Joshi A. (2003) An Ontology for Context-aware Pervasive Computing Environments. Knowl Eng Rev 18(3): 197-207.

Chen H, Perich F, Finin T & Joshi A. (2004c) SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. Proc IEEE 1st International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS). Boston, USA: 258–267.

Christensen E, Curbera F, Meredith G & Weerawarana S. (2001) Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001. URL: http://www.w3.org/TR/wsdl. 2015(10/10).

Compton M, Barnaghi P, Bermudez L, García-Castro R, Corcho O, Cox S, Graybeal J, Hauswirth M, Henson C, Herzog A, Huang V, Janowicz K, Kelsey WD, Le Phuoc D, Lefort L, Leggieri M, Neuhaus H, Nikolov A, Page K, Passant A, Sheth A & Taylor K. (2012) The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web 17: 25–32.

Contributing members of UPnP forum. (2008) UPnP Device Architecture 1.1. URI: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf. Cited 2014/3/19.

Cooperstock JR, Tanikoshi K, Beirne G, Narine T & Buxton WAS. (1995) Evolution of a Reactive Environment. Proc SIGCHI Conference on Human Factors in Computing Systems. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co: 170–177.

Dertouzos M. (1999) The Future of Computing. Sci Am 281(2): 52-55.

Driscoll D & Mensch A. (2009) Device Profile for Web Services Version 1.1. OASIS Standard. URI: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf. Cited 2014/4/10.

Duffy DA. (1991) Principles of Automated Theorem Proving. New York, NY, USA: John Wiley \& Sons, Inc.

Erman LD, Hayes-Roth F, Lesser VR & Reddy DR. (1980) The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. ACM Comput.Surv. 12(2): 213–253.

Eteläperä M, Keinänen K, Kiljander J, Hyttinen P, Pehkonen V, Väre J & Soininen J. (2010) Open-M3: Smart Space with COTS Devices. Proc 12th ACM international conference adjunct papers on Ubiquitous computing - Adjunct. Copenhagen, Denmark: ACM: 363–364.

Farahani S. (2008) ZigBee Wireless Networks and Transceivers. Newton, MA, USA: Newnes.

Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P & Berners-Lee T. (1999) Hypertext Transfer Protocol -- HTTP/1.1, RFC 2616, URI: https://www.ietf.org/rfc/rfc2616.txt. Cited 2014/10/23.

Fikes R & Kehler T. (1985) The Role of Frame-based Representation in Reasoning. Commun ACM 28(9): 904–920.

Forgy C. (1982) Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. Artificial Intelligence 19(1): 17–37.

Gangemi A. (2007) DOLCE UltraLite OWL Ontology. URI: http://www.loa.istc.cnr.it/ontologies/DUL.owl. Cited 2014/10/11.

Gershenfeld N, Krikorian R & Cohen D. (2004) Scientific American 291: 76–81.

Gruber TR. (1993) A Translation Approach to Portable Ontology Specifications. Knowl.Acquis. 5(2): 199–220.

Gudgin M, Hadley M, Mendelsohn N, Moreau J & Frystyk Nielsen Henrik. (2003) SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation 27 April 2007. URI: http://www.w3.org/TR/2007/REC-soap12-part1-20070427. Cited 2014/10/6.

Haase P, Broekstra J, Eberhart A & Volz R. (2004) A Comparison of RDF Query Languages. The Semantic Web – ISWC 2004, Lecture Notes in Computer Science 3298: 502–517.

Hall RS, Pauls K & McCulloch S. (2011) OSGi in Action: Creating Modular Applications in Java. Stamford: Manning Publications co.

Heiler S. (1995) Semantic Interoperability. ACM Comput.Surv. 27(2): 271–273.

Helal S. (2011) IT Footprinting - Groundwork for Future Smart Cities. Computer 44(6): 30–31.

Hill MG & Lake TW. (2000) Non-Interference Analysis for Mixed Criticality Code in Avionics Systems. Proc 15th IEEE International Conference on Automated Software Engineering (ASE). Grenoble, France: 257–260.

Honkola J, Laine H, Brown R & Tyrkko O. (2010) Smart-M3 Information Sharing Platform. Proc IEEE Symposium on Computers and Communications (ISCC). Riccione, Italy: 1041–1046.

IBM & Eurotech. (2010) MQTT V3.1 Protocol Specification. URI: http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/MQTT_V3.1_Protocol_Specific.pdf. Cited 2014/9/3.

Iannella R & McKinney J. (2014) vCard Ontology - for describing People and Organizations. W3C Interest Group Note 22 May 2014. URL: http://www.w3.org/TR/vcard-rdf. 2015(10/25).

Ishikawa C. (2012) A URN Namespace for ucode. RFC 6588. URI: https://tools.ietf.org/html/rfc6588. Cited 2014/4/9.

Jackson P. (1998) Introduction to Expert Systems. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Janowicz K & Compton M. (2010) The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. 3rd International workshop on Semantic Sensor Networks. : 1-15.

Jianhua Ma, Yang LT, Apduhan BO, Runhe Huang, Barolli L, Takizawa M & Shih TK. (2005) A Walkthrough from Smart Spaces to Smart Hyperspaces Towards a Smart World with Ubiquitous Intelligence. Proc IEEE 11th International Conference on Parallel and Distributed Systems. Fukuoka, Japan 1: 370–376.

Johanson B, Fox A & Winograd T. (2002) The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Pervasive Computing 1(2): 67–74.

Järvinen P. (2004) On Research Methods. Tampere, Finland: Opinpajan kirja, Tampereen yliopistopaino Oy.

Katasonov A. (2010) Enabling Non-Programmers to Develop Smart Environment Applications. Proc IEEE Symposium on Computers and Communications (ISCC). Riccione, Italy: 1059–1064.

Katasonov A & Palviainen M. (2010) Towards Ontology-Driven Development of Applications for Smart Environments. Proc 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom). Mannheim, Germany: 696–701.

Kindberg T, Barton JJ, Morgan J, Becker G, Caswell D, Debaty P, Gopal G, Frid M, Krishnan V, Morris H, Schettino J, Serra B & Spasojevic M. (2000) People, Places, Things: Web Presence for the Real World. Proc 3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA). Los Alamitos, CA: 19–28.

Komazec S, Cerri D & Fensel D. (2012) Sparkwave: Continuous Schema-Enhanced Pattern Matching Over RDF Data Streams. Proc 6th ACM International Conference on Distributed Event-Based Systems. New York, USA: 58–68.

Koshizuka N & Sakamura K. (2010) Ubiquitous ID: Standards for Ubiquitous Computing and the Internet of Things. Pervasive Computing, IEEE 9(4): 98–101.

Lappeteläinen A, Tuupola J, Palin A & Eriksson T. (2008) Networked Systems, Services and Information – the Ultimate Digital Convergence. Proc 1st International NoTA conference. Helsinki, Finland: 1–7.

Lassila O. (2007) Programming Semantic Web Applications: A Synthesis of Knowledge Representation and Semi-Structured Data. Doctor of Science in Technology thesis. Helsinki University of Technology, Espoo, Finland.

Lassila O. (2006) Generating Rewrite Rules by Browsing RDF Data. Proc 2nd IEEE International Conference on Rules and Rule Markup Languages for the Semantic Web. Athens, Georgia, USA: 51–57.

Lassila O. (2002) Serendipitous Interoperability. The Semantic Web Kick-off in Finland – Vision, Technologies, Research, and Applications. University of Helsinki: HIIT Publications.

Le-Phuoc D, Dao-Tran M, Parreira JX & Hauswirth M. (2011) A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. Proc 10th International Conference on the Semantic Web - Volume Part I. Berlin, Heidelberg: Springer-Verlag: 370–388.

Lupiana D, O'Driscoll C & Mtenzi F. (2009) Defining smart space in the context of ubiquitous computing. Ubiquitous Computing and Communication Journal (UbiCC) 4(3): 516–524.

Manzaroli D, Roffia L, Cinotti TS, Azzoni P, Ovaska E, Nannini V & Mattarozzi S. (2010) Smart-M3 and OSGi: The Interoperability Platform. Proc IEEE Symposium on Computers and Communications (ISCC). Riccione, Italy: 1053–1058.

Martin D, Burstein M, Hobbs J, Lassila O, McIlraith S, Narayanan S, Paolucci M, Parsia B, Payne PT, Sirin E & Sycara K. (2004) OWL-S: Semantic Markup for Web Services, W3C Member Submission. URI: http://www.w3.org/Submission/OWL-S/. Cited 2014/3/18.

Murth M & Kühn E. (2009a) Knowledge-Based Coordination with a Reliable Semantic Subscription Mechanism. Proceedings of the 2009 ACM Symposium on Applied Computing. New York, NY, USA: ACM: 1374–1380.

Murth M & Kühn E. (2009b) A Heuristics Framework for Semantic Subscription Processing. In: Anonymous 6th Annual European Semantic Web Conference (ESWC2009). : 96–110.

Masuoka R, Parsia B & Labrou Y. (2003) Task Computing - the Semantic Web Meets Pervasive Computing. In: Fensel D, Sycara K & Mylopoulos J (eds) The Semantic Web - ISWC 2003, Lecture Notes in Computer Science. : Springer Berlin Heidelberg: 866–881.

McCulloch W & Pitts W. (1943) A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 5(4): 115–133.

Minsky M. (1974) A Framework for Representing Knowledge. MIT-AI Laboratory Memo 306 URI: https://web.media.mit.edu/~minsky/papers/Frames/frames.html. Cited 2014/3/19.

Mokhtar SB, Georgantas N & Issarny V. (2007) COCOA: COnversation-based service COmposition in pervAsive computing environments with QoS support. Journal of Systems and Software 80(12): 1941–1955.

Morandi F, Roffia L, D'Elia A, Vergari F & Salmon Cinotti T. (2012) RedSib: a Smart-M3 Semantic Information Broker Implementation. Proc 12th conference of FRUCT Association. Oulu, Finland: 86–98.

Mozer MC. Lessons from an Adaptive House. In: Cook M & Das R (eds) Smart Environments: Technologies, Protocols, and Applications. : J. Wiley & Sons: 273–294.

Niezen G, Vlist B, Jun H & Feijs LMG. (2010) From Events to Goals: Supporting Semantic Interaction in Smart Environments. IEEE Symposium on Computers and Communications (ISCC). Riccione, Italy: 1029–1034.

Nixon P, Dobson S & Lacey G. (2000) Managing Smart Environments. Proc Workshop on Software Engineering for Wearable and Pervasive Computing. Limerick, Ireland.

Palviainen M, Kuusijärvi J & Ovaska E. (2012) Framework for End-user Programming of Cross-Smart Space Applications. Sensors 12(11): 14442–14466.

Pane JF, Myers BA & Ratanamahatana CA. (2001) Studying the Language and Structure in Non-programmers' Solutions to Programming Problems. Int J Hum -Comput Stud 54(2): 237–264.

Pantsar-Syväniemi S, Purhonen A, Ovaska E, Kuusijärvi J & Evesti A. (2012) Situation-based and self-adaptive applications for the smart environment. Journal of Ambient Intelligence and Smart Environments 4(6): 491–516.

Pellegrino L, Huet F, Baude F & Alshabani A. (2013) A Distributed Publish/Subscribe System for RDF Data. In: Anonymous Data Management in Cloud, Grid and P2P Systems. : Springer: 39–50.

Pfisterer D, Romer K, Bimschas D, Kleine O, Mietz R, Cuong Truong, Hasemann H, Kröller A, Pagel M, Hauswirth M, Karnstedt M, Leggieri M, Passant A & Richardson R. (2011) SPITFIRE: toward a semantic web of things. Communications Magazine, IEEE 49(11): 40–48.

Preuveneers D & Berbers Y. (2008) Encoding Semantic Awareness in Resource-Constrained Devices. IEEE Intelligent Systems 23(2): 26–33.

Quillian RM. (1967) Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities. Behav Sci 12: 410–430.

Rich E. (1983) Artificial Intelligence. New York, NY, USA: McGraw-Hill, Inc.

Rinne M, Nuutila E & Törmä S. (2012) INSTANS: High-Performance Event Processing with Standard RDF and SPARQL. Proc International Semantic Web Conference (ISWC), Posters & Demonstrations Track. Boston, USA.

Rinne M. (2012) SPARQL Update for Complex Event Processing. Proc International Semantic Web Conference (ISWC), Lecture Notes in Computer Science 7650: 453–456.

Rosenthall L & Stanford V. (2000) NIST Smart Space: Pervasive Computing Initiative. Proc IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE). Gaithersburg, MD : 6–11.

Saha D & Mukherjee A. (2003) Pervasive computing: a paradigm for the 21st century. Computer 36(3): 25–31.

Sathish S & di Flora C. (2007) Supporting Smart Space Infrastructures: A Dynamic Context-Model Composition Framework. Proc 3rd International Conference on Mobile Multimedia Communications (ICST). Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering. Brussels, Belgium, Belgium 67:1–6.

Seaborne A. (2004) RDQL - A Query Language for RDF. W3C Member Submission 9 January 2004. URI: http://www.w3.org/Submission/RDQL/. Cited 2015(10/20).

Shelby Z & Bormann C. (2010) 6LoWPAN: The Wireless Embedded Internet. : Wiley Publishing.

Shelby Z, Hartke K & Bormann C. (2013) Constrained Application Protocol (CoAP) draft-ietf-core-coap-18, RFC 7252. URI: http://datatracker.ietf.org/doc/draft-ietf-core-coap/. Cited 2014/4/5.

Sheth A, Henson C & Sahoo SS. (2008) Semantic Sensor Web. IEEE Internet Comput 12(4): 78–83.

Singh R, Bhargava P & Kain S. (2006) State of the Art Smart Spaces: Application Models and Software Infrastructure. Ubiquity 7(37): 2–9.

Sirin E, Parsia B, Grau BC, Kalyanpur A & Katz Y. (2007) Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2): 51–53.

Sohraby K, Minoli D & Znati T. (2007) Wireless Sensor Networks: Technology, Protocols, and Applications. Wiley-Interscience.

Soldatos J, Dimakis N, Stamatis K & Polymenakos L. (2007) A breadboard architecture for pervasive context-aware services in smart spaces: middleware components and prototype applications. Personal and Ubiquitous Computing 11(3): 193–212.

Song Z, Cárdenas AA & Masuoka R. (2010) Semantic Middleware for the Internet of Things. Proc IEEE Internet of Things (IOT). Tokyo, Japan: 1–8.

Su X, Riekki J & Haverinen J. (2012) Entity Notation: Enabling Knowledge Representations for Resource-Constrained Sensors. Personal and Ubiquitous Computing 16(7): 819–834.

Suomalainen J, Hyttinen P & Tarvainen P. (2010) Secure information sharing between heterogeneous embedded devices. Proc 4th ACM European Conference on Software Architecture (ECSA): Companion Volume. Copenhagen, Denmark: 205–212.

Tamma V & Payne TR. (2008) Is a Semantic Web Agent a Knowledge-Savvy Agent? Intelligent Systems, IEEE 23(4): 82–85.

Thomson G, Bianco S, Mokhtar SB, Georgantas N & Issarny V. (2008) Amigo Aware Services. In: Mühlhäuser M, Ferscha A & Aitenbichler E (eds) Constructing Ambient Intelligence, Communications in Computer and Information Science. Springer Berlin Heidelberg: 385–390.

Tolk A. (2004) Composable Mission Spaces and M&S Repositories - Applicability of Open Standards. Proc Spring Simulation Interoperability Workshop (SIW). Arlington, VA, USA: 1–14.

Vazquez JI, de Ipina DL & Sedano I. (2006) SoaM: A Web-powered Architecture for Designing and Deploying Pervasive Semantic Devices. International Journal of Web Information Systems 2(3): 212–224.

Vermesan O & Friess P (eds) (2013) Internet of Things - Converging Technologies for Smart Environments and Integrated Ecosystems. River Publishers.

Vestal S. (2007) Preemptive Scheduling of Multi-Criticality Systems with Varying Degrees of Execution Time Assurance. Proc, 28th IEEE International Real-Time Systems Symposium (RTSS). Tucson, AZ, USA: 239–243.

Vlist B, Niezen G, Jun H & Feijs LMG. (2010) Semantic Connections: Exploring and Manipulating Connections in Smart Spaces. Proc IEEE Symposium on Computers and Communications (ISCC). Riccione, Italy: 1–4.

Vlist B, Niezen G, Rapp S, Hu J & Feijs LMG. (2013) Configuring and controlling ubiquitous computing infrastructure with semantic connections: a tangible and an AR approach. Personal and Ubiquitous Computing 17(4): 783–799.

W3C. (1999) Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation. URI: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. Cited 2014/9/9.

W3C Government Linked Data Working Group. (2014a) The Organization Ontology. W3C Recommendation 16 January 2014. URL: http://www.w3.org/TR/vocab-org/. 2015(10/25).

W3C Government Linked Data Working Group. (2014b) Data Catalog Vocabulary. W3C Recommendation 16 January 2014. URL: http://www.w3.org/TR/vocab-dcat/. 2015(10/20).

W3C OWL Working Group. (2012) OWL 2 Web Ontology Language Document Overview (Second Edition), W3C Recommendation. URI: http://www.w3.org/TR/owl2-overview/. Cited 2014/3/21.

W3C RDF Working Group. (2014a) RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation. URI http://www.w3.org/TR/rdf11-concepts/. Cited 2014/3/20.

W3C RDF Working Group. (2014b) RDF Schema 1.1, W3C Recommendation. URI: http://www.w3.org/TR/rdf-schema/. Cited 2014/3/20.

W3C RDF Working Group. (2014c) RDF 1.1 XML Syntax, W3C Recommendation. URI: http://www.w3.org/TR/rdf-syntax-grammar/. Cited 2014/9/9.
W3C RDF Working Group. (2014d) RDF 1.1 N-Triples, A line-based syntax for an RDF graph, W3C Recommendation. URI: http://www.w3.org/TR/n-triples/. Cited 2014/9/9.
W3C RDF Working Group. (2014e) RDF 1.1 Turtle, Terse RDF Triple Language, W3C Recommendation. URI: http://www.w3.org/TR/turtle/. Cited 2014/9/9.
W3C SPARQL Working Group. (2008) SPARQL Query Language for RDF, W3C Recommendation 15 January 2008. URI: http://www.w3.org/TR/rdf-sparql-query/. Cited 2014/9/9.
W3C Semantic Sensor Network Incubator Group. (2011) Semantic Sensor Network Ontology. URL: http://purl.oclc.org/NET/ssnx/ssn. 2015(9/15)
W3C SPARQL Working Group. (2013a) SPARQL 1.1 Query Language, W3C Recommendation. URI: http://www.w3.org/TR/sparql11-query/. Cited 2014/3/24.
W3C SPARQL Working Group. (2013b) SPARQL 1.1 Update, W3C Recommendation. URI: http://www.w3.org/TR/sparql11-update/. Cited 2014/3/24.
Wang X, Dong JS, Chin CY, Hettiarachchi SR & Zhang D. (2004) Semantic Space: an infrastructure for smart spaces. Pervasive Computing, IEEE 3(3): 32–39.
Weiser M. (1999) The computer for the 21st century. SIGMOBILE Mob.Comput.Commun.Rev. 3(3): 3–11.
Woods WA. (1975) What's in a Link: Foundations for Semantic Networks. In: Bobrow DG & Collins A (eds) Representation and Understanding. : Academic Press.
Zadeh LA. (1965) Fuzzy sets. Information and Control 8(3): 338–353.

Original articles

I Kiljander J, Eteläperä M, Takalo-Mattila J & Soininen J. (2010) Opening Information of Low Capacity Embedded Systems for Smart Spaces. Proc IEEE 8th Workshop on Intelligent Solutions in Embedded Systems (WISES). Heraklion, Greece: 23¬¬–28. doi: 10.1109/WISES.2010.5548438.

II Kiljander J, Eteläperä M, Takalo-Mattila J, Soininen J & Keinänen K (2011) Autonomous File Sharing for Smart Environments. Proc 1st International Conference on Pervasive and Embedded Computing and Communication Systems. Algarve, Portugal: 191–196. doi: 10.5220/0003362501910196.

III Takalo-Mattila J, Kiljander J, Eteläperä M & Soininen J. (2011) Ubiquitous Computing by Utilizing Semantic Interoperability with Item-Level Object Identification. In: Pentikousis K, Agüero R, Garcìa-Arranz M & Papavassiliou S (eds.) Mobile Networks and Management, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg: 198–209. doi: 10.1007/978-3-642-21444-8_18.

IV Kiljander J, Takalo-Mattila J, Eteläperä M, Soininen J & Keinänen K. (2011) Enabling End-Users to Configure Smart Environments. Proc IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT). Munich, Germany: 303–308. doi: 10.1109/SAINT.2011.58.

V Ylisaukko-oja A, Hyttinen P, Kiljander J, Soininen J & Viljamaa E. (2011) Semantic Interface for Resource Constrained Wireless Sensors. Proc SciTePress International Conference on Knowledge Engineering and Ontology Development (KEOD). Paris, France: 505–511. doi: 10.5220/0003698905050511.

VI Viljamaa E, Kiljander J, Soininen J & Ylisaukko-oja A. (2011) A Smart Control System Solution Based on Semantic Web and uID. Proc IARIA 5th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM). Lisbon, Portugal: 105–110. isbn: 978-1-61208-171-7.

VII Kiljander J, Ylisaukko-oja A, Takalo-Mattila J, Eteläperä M & Soininen J. (2012) Enabling Semantic Technology Empowered Smart Spaces. Journal of Computer Networks and Communications 2012: 1–14. doi: 10.1155/2012/845762.

VIII Kiljander J, Morandi F & Soininen J. (2012) Knowledge sharing protocol for smart spaces. International Journal of Advanced Computer Science and Applications 3(9): 100–110. doi: 10.14569/IJACSA.2012.030915.

IX Roffia L, Morandi F, Kiljander J, D'Elia A, Vergari F, Viola F, Bononi L & Salmon Cinotti T. A Semantic Publish-Subscribe Architecture for the Internet of Things. Manuscript.

X Kiljander J, D'Elia A, Morandi F, Hyttinen P, Takalo-Mattila J, Ylisaukko-oja A, Soininen J & Salmon Cinotti T. (2014) Semantic Interoperability Architecture for Pervasive Computing and Internet of Things. IEEE Access 2: 856–873. doi: 10.1109/ACCESS.2014.2347992.

Original publications are not included in the electronic version of the dissertation.

Book orders:
Granum: Virtual book store
http://granum.uta.fi/granum/

UNIVERSITY OF OULU P.O. Box 8000 FI-90014 UNIVERSITY OF OULU FINLAND

ACTA UNIVERSITATIS OULUENSIS

UNIVERSITY of OULU
OULUN YLIOPISTO

ISBN 978-952-62-1080-3 (Paperback)
ISBN 978-952-62-1081-0 (PDF)
ISSN 0355-3213 (Print)
ISSN 1796-2226 (Online)