Semantic Interoperability in Smart Spaces

Sachin Bhardwaj

Semantic Interoperability in Smart Spaces

Sachin Bhardwaj

Printing: ProefschriftMaken || www.proefschriftmaken.nl

ISBN: 9789464698107

Semantic Interoperability in Smart Spaces

PROEFONTWERP

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de rector magnificus prof.dr. S.K. Lenaerts,
voor een commissie aangewezen door het College voor Promoties,
in het openbaar te verdedigen
op dinsdag 27 februari 2024 om 11:00 uur

door

Sachin Bhardwaj

geboren te Delhi, India

De documentatie van het proefontwerp is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

Voorzitter: prof.dr. E.R. van den Heuvel

Promotoren: prof.dr. J.J. Lukkien
dr. T. Ozcelebi

leden: prof. dr. ir. L.M.G. Feijs
prof. dr. P. Corcoran (National University of Ireland)
prof. dr. S.C. Mukhopadhyay (Macquarie University)
dr. A. A. Syed (Philips Electronics)

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Table of Contents

Acknowledgements

As a Ph.D. candidate at the IRIS group (formerly the SAN group), I can reflect on the significant changes I have undergone. Confronting numerous challenges has not only made me stronger, more independent, and professional but has also endowed me with a vast amount of invaluable experience that will undoubtedly prove priceless for my future. Interacting with a multitude of interesting individuals has played a crucial role in shaping both my personal growth and scientific development. I would like to express my gratitude to everyone who has assisted me throughout this enduring journey.

First and foremost, I extend my deepest gratitude to my first promoter, Prof. dr. Johan J. Lukkien, for providing me with the invaluable opportunity to embark on my Ph.D. journey at the IRIS group. I vividly recall the memorable Christmas day when you sent me an informal email, announcing my selection for the SOFIA project. Your official email was to follow after the holiday, making it one of the most joyous holiday seasons I have ever experienced. Your unwavering support was evident when you praised my first publication in IEEE PERCOM during the initial year of my Ph.D., marking it with an 'Excellent' remark. This encouragement fueled my enthusiasm to persevere in my doctoral studies. Our discussions, often conducted at the whiteboard, pushed me to delve deeper into my thoughts. Your guidance not only improved my conceptual understanding but also encouraged profound thinking. I am grateful for your critical comments and thought-provoking questions on my thesis, which contributed significantly to its remarkable improvement. Your patience in thoroughly reading and providing constructive feedback underscored your commitment to my academic growth. I am also thankful for your understanding of my family situation and your willingness to afford me the time needed to manage personal challenges. It is difficult to encapsulate the depth of my gratitude for you in a single paragraph. Your mentorship has been pivotal in my successful completion of various research projects. Overall, I sincerely thank you for entrusting me with promising projects and for being an indispensable guide throughout my academic journey.

Further, I would like to thank my second promoter, Dr. Tanir Ozcelebi. You cannot imagine how much I wish I could have an elder brother like you, always helping me out when I overthink, comforting me when I feel depressed, and supporting me when I lack confidence. I feel so lucky to have you in my life and start my Ph.D. project with you. Thank you for teaching me the most important rules of scientific writing, for polishing my manuscript with great patience, for sharing your gained experience, and for dropping me ideas. Without you, I could not have graduated so successfully and fruitfully. Your positive attitude toward everything is inspiring. As I remember, just before starting the second phase of my Ph.D., you motivated me and gave time for discussion even when I was not at TU/e. I sincerely thank you for your constant support.

Next, I would like to express my sincere gratitude to all the esteemed committee members (Prof. dr. Ir. L.M.G. Feijs, Prof. dr. P. Corcoran, Prof. dr. S.C. Mukhopadhyay, Dr. A. A. Syed) for their dedicated efforts in carefully reviewing my thesis. Your invaluable time and insightful feedback have significantly contributed to the refinement and enhancement of my work.

I also want to express my gratitude to Dr. Richard Verhoeven. You have been incredibly supportive of my experiments and pilot demos. I remember the countless late-night discussions and collaborative work we engaged in. Your doors were always open for discussions and to facilitate experiments. Thanks a lot.

Thank you very much, Prof. dr. Nirvana Meratnia, for motivating me to complete my Ph.D. I caringly recall your consistent encouragement whenever we met, advising me to expedite the thesis completion process. I appreciate the opportunity you provided me to conduct yoga lessons for the cluster members during the COVID period. While I cannot speak for others, I personally benefitted greatly from practicing yoga during that time, which helped alleviate a significant amount of stress. Once again, thank you for all your support.

I would like to express my gratitude to my colleagues (Mike, Pieter, Bram, Luis, Leila, Aaqib, Hrishikesh, Nan) at the IRIS cluster for engaging in insightful discussions during coffee and lunch breaks. Special thanks to Cecile, Anjolein, and Jolande for efficiently managing all administrative processes. The bi-weekly games and drinks after our group presentations were also enjoyable. Thanks to everyone for being such wonderful colleagues.

Most importantly, I would like to extend my heartfelt gratitude to my wife, Yogita, for unwavering support throughout the journey of my thesis. Managing our children, Samaira and Prisha, single-handedly during times when I could not devote time to them was indeed a challenging task. Your numerous sacrifices, including the periods when we had to live apart due to my Ph.D. commitments, did not go unnoticed. I genuinely attribute the successful attainment of this Ph.D. degree equally to your contributions.

I would like to express my gratitude to Anupam Krishna Dasa Prabhu Ji for guiding me and my family on the path of Krishna consciousness and introducing us to ISKCON. Your motivational discussions have been instrumental in helping me to stay focused on my Ph.D. goal while also fostering Krishna Bhakti. Your guidance has uplifted us in Krishna Bhakti, striking a balance with my pursuit of the Ph.D. Thank you very much for your valuable association.

I express my heartfelt gratitude to Sundaripriya Mata Ji and Bhagvat Dhama Prabhu Ji for providing me with parental care during my time in Eindhoven. You not only provided me with the chance to live in an environment infused with Krishna consciousness in Eind-

hoven, but you also offered support alike to family. I lovingly remember the time when we first arrived in Eindhoven and were searching for Tulsi leaves for the bhoga offering to Krishna. Mata Ji, with a single call to you, Tulsi leaves became readily available. Your efforts made us feel at home in Eindhoven. Your encouragement in completing my Ph.D. as soon as possible is deeply appreciated. Thank you very much for your love and support.

I would also like to extend my thanks to Arundhati Mata Ji and Divya Mohan Prabhu Ji for being pillars of support, creating a sense of complete family. Divya Mohan Prabhu Ji, your lectures served as a constant source of motivation, keeping me aligned with my objectives. The soul-stirring Krishna's kirtan that you led always touched my heart deeply. Arundhati Mata Ji, your support felt like that of a sister, and I could freely discuss any problem with you. I vividly recall instances when I felt depressed, and a simple call to you provided the energy I needed to pursue my goals.

I express my gratitude to my family members, including Amit, Anjali, Sumit, Reena, Lalit, Ruby, Navneet, and Chinky, for their unwavering support. Additionally, I would like to extend my thanks to my friends, namely Ragini, Pranav, Radhe Govind, Manu, Nitin, Shubendu, Sandeep, Gopika, Sahith, Roopali, Gayatri, Mohit and and all others whom I may not remember while writing this, for their encouragement and companionship.

Finally, I extend my heartfelt gratitude to my parents, and my mother-in-law, for consistently showering me with their blessings, which have been a source of inspiration in achieving my life goals.

Jai Srila Prabhupada! Hare Krishna!

Summary

The vision of ubiquitous computing is to facilitate people's tasks via the use of applications that run on embedded devices of various sizes, features and capabilities. One area of ubiquitous computing is the still emerging concept of smart spaces. A smart space is a physical environment that contains cooperating "nodes" (i.e., embedded devices) that continuously and autonomously monitor the environment, interact with users and adapt their behavior (services) to enhance user experiences using contextual reasoning. Such reasoning is based on information gathered either from the physical environment (e.g., via sensors) or digitally, e.g., from the Internet (e.g., via user profiles). Smart spaces are most often used to enhance individuals' living standards and to improve their quality of life, and they are characterized by an application or set of applications that function with a common objective. Smart space applications go beyond those applications that can be offered by a single device. Their potential to enhance the living standards of people is now being widely investigated in real-world implementations. In facilitating applications that are tailored to user preferences and are adaptive with respect to user contexts, smart spaces must conform to applications' resource needs. These objectives are achievable by interoperation and cooperation among nodes. Supporting interoperability, therefore, is a necessity for smart space architecture designs. In general, it is a property that enables multiple nodes to work together to achieve a shared objective in a smart space and can be achieved at different levels. For example, at the network level communication interoperability is essential, allowing nodes to send messages to each other. However, this is not enough by itself. The nodes in a smart space are diverse and use various types of technologies, yet they need to understand each other (i.e., the contents of the messages exchanged). The capacity to communicate meaningful information (semantics) across nodes and to share a common understanding of such information, across different heterogeneous networks, is called semantic interoperability. Semantics in a smart space are described using an ontology language.

This thesis work started about 10 years ago, when the smart space design and implementation efforts were only at their infancy, with the motivation that the existence of well-defined architectural concepts, components and mechanisms for semantic interoperability among smart space nodes and networks was a key towards the success of smart spaces. The work was unfortunately interrupted in 2013 close to finalization due to personal circumstances. In 2017, the work resumed, and we had a chance to carry out an extensive literature survey to rediscover the positioning of the work after the years that have passed in between. We found that, even though there have been many more attempts towards realization of more advanced smart spaces, space architecture designs out there are still very much application-specific, and their concepts and components are defined almost from scratch based on the specific needs of one target application. To

this date, semantic interoperability in smart spaces remains a challenge and the way to achieve it in a generic way is tightly coupled with the need for defining a generic semantic interoperability architecture, which defines the software components, their physical deployment and the protocols that they use for this.

To address the semantic interoperability challenge in smart spaces in a generic way, the first thing that is needed is a precise smart space definition clearly describing its fundamental concepts and properties. Prior to this thesis work, such a detailed formal definition of a smart space was non-existent. This set a barrier in front of achieving technological breakthrough in generic smart space design. In this thesis, we formally and rigorously define general smart space concepts and discuss related architectural components (both hardware and software) in detail. We propose candidate architectures, namely, centralized, decentralized and distributed smart space architecture options to choose from.

In developing smart spaces, starting from a generic semantic interoperability architecture can reduce the design effort significantly thanks to its reusability. We present a comparative analysis of the architectural designs proposed in the literature thus far. We use the insights gained from this analysis and summarize the discriminating properties that a smart space must possess, as well as the basic components and services necessary to realize these properties. Consequently, we introduce a semantic interoperability architecture where semantic interoperability as a key smart space property is central to the design. We propose semantic interoperability mechanisms based on this semantic interoperability architecture, enabling semantic interactions among nodes. Candidate architectures for a new design are then variants of this semantic interoperability architecture, tuned towards certain performance metrics relevant for the application at hand. We formulate the choice of the best architecture amongst candidate smart space architectures as a multi-objective optimization problem.

Smart spaces often include many resource-poor (low-capacity) nodes (e.g., sensors, actuators) that are not capable of performing complex tasks, e.g., due to insufficient computation, memory, energy and communication resources. We take a gateway approach in integrating these nodes into a smart space, wherein the gateway node does complex tasks on their behaves and enables their semantic interactions with each other and with other resource-rich (high-capacity) nodes.

We test the proposed semantic interoperability architecture solution through the implementation of a smart lighting application for specific smart lighting use cases; i.e., context-adaptive smart lighting and power-managed smart lighting. For this purpose, we first introduce the concepts that are applicable in the application setting. Therefore, we propose a novel smart lighting model that is used to continuously monitor and control

illumination in an activity space by means of sensing and actuation. Then after, we discuss the smart space life cycles and the implementation process of smart applications. We experiment with and verify the usability of this model on the context-adaptive smart lighting use case. Moreover, the power-managed smart lighting use case is implemented using the same smart lighting model, the proposed semantic interoperability mechanisms, and the proposed architecture. The various communication and hardware technologies used in these use cases can easily achieve a common objective by using the proposed semantic interoperability mechanisms in the semantic interoperability architecture.

Overall, the thesis provides formal definitions of smart space concepts and proposes several candidate architectures. We can choose a suitable smart space architecture for an application by employing the proposed multi-objective optimization problem formulation. In addition, the proposed semantic interoperability architecture can deliver a generic solution to the implementations of smart space applications. The semantic interoperability property of the semantic interoperability architecture simplifies the sharing of semantics among heterogeneous nodes and networks. The semantic interoperability architecture also enables semantic interoperability between low and high-capacity nodes via the gateway approach. Moreover, a novel smart lighting model has been proposed and successfully implemented, utilizing the semantic interoperability architecture. Although it is extensively tested for lighting applications, the semantic interoperability architecture is generic and provides a competitive solution also for implementing smart space applications other than smart lighting applications.

Chapter 1

Introduction

It has been nearly thirty years since the introduction of ubiquitous computing, a paradigm that has paved the way for numerous applications aimed at enhancing human living standards. Contemporary studies have yielded substantial solutions, continuously improving to achieve a satisfactory level of ubiquitous systems. In this chapter, we emphasize the significance of smart spaces by introducing the concepts of ubiquitous computing and ubiquitous systems. Furthermore, we delve into the research questions pertaining to smart spaces and outline our main contributions in addressing these questions. Lastly, we provide an overview of the organizational structure of the thesis.

1.1 Motivation

In the digital information age, the majority of people, particularly in developed countries, possess the knowledge of operating a computer or, at the very least, accomplishing basic tasks like browsing the Internet and using social media. Nevertheless, the relationship between computers and human beings has not always been as straightforward. There was a time when considerable technical training and experience were required to perform even a simple arithmetic operation and obtain the result on a standalone computer. During those days, individuals needed to be proficient in writing low-level machine code just to carry out the most basic tasks. Fortunately, we are now more privileged. Despite the increasing complexity of technology, users are shielded from its intricate details through abstraction. We have made significant progress, transitioning from an era of complex interactions with massive standalone supercomputers occupying entire rooms to an era of simplified interactions with personal computers and networks comprised of *embedded devices*.

Networked and embedded computing technology, along with the way we interact with it, have evolved hand in hand over time. Nowadays, many modern buildings are equipped with networks of embedded devices. For instance, consider a presence sensor appropriately positioned on the ceiling of an office room, connected to a lighting service. The primary task of this setup is to automatically turn the lights on and off based on the presence of users. Ideally, the sensor and the lighting service seamlessly blend into the environment, relieving employees of the need to worry about manually controlling the

lights upon entering or leaving the office. The presence sensor, in collaboration with an embedded computing device that implements a control logic, effectively assumes the responsibility of reducing energy consumption by avoiding wastage, thereby minimizing the involvement of users. As the lighting service operates smoothly, employees may eventually forget about both the sensor and the service. Nevertheless, with each interaction, valuable data are generated for the purpose of further learning. This instance exemplifies the contemporary phenomenon of *ubiquitous computing*, highlighting the seamless and automated nature of human-computer interaction.

In 1991, Mark Weiser introduced the concept of ubiquitous computing in his description of *the computer for the 21st century* [1.1]. Various synonyms are used in the literature for ubiquitous computing, including the post-PC era [1.2], pervasive computing [1.3], ambient intelligence [1.4], disappearing computing [1.5], mixed-mode systems [1.6], tangible bits [1.7], and real-time enterprise [1.8].[1] Although these terms have subtle differences in emphasis, they all pertain to the same vision of ubiquitous computing as defined by Weiser as follows:

> *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."*

In this vision, *everyday objects* that are part of a human user's environment incorporate embedded computational devices, systems, or technology that are intentionally concealed from the user's perception. In this way, the user can interact with such a system without consciously acknowledging its presence or complexity. This can be observed in the case of the office lighting service example, where users interact with the system effortlessly and without needing to be knowing of its inner workings.

> **Definition (Ubiquitous System):** A ubiquitous system is an infrastructure that seamlessly and invisibly supports human users through services and applications offered by its networked and heterogeneous embedded computational devices.

Invisibility is an important property of ubiquitous systems. In [1.10], Weiser explains what he means by invisible:

> *"A good tool is an invisible tool. By invisible, I mean that the tool does not intrude on your consciousness; you focus on the task, not the tool. Eyeglasses are a good tool – you look at the world, not the eyeglasses."*

1 For a more detailed analysis of these terms and their nuances, please refer to [1.9].

A ubiquitous system can exhibit heterogeneity, meaning that the embedded devices interconnected within it may differ from one another in various aspects. Weiser and his colleagues envisioned and developed prototypes of three distinct types of ubiquitous devices, namely wearable tabs resembling post-it notes, handheld pads, and large displays [1.11]. Today, we can observe striking examples of Weiser's vision in the form of smartphones, electronic tablets, and intelligent interactive boards, corresponding to tabs, pads, and displays, respectively. However, it is important to note that these examples do not encompass the entire range of devices suitable for ubiquitous computing. In 2009, Stefan Poslad extended the list of ubiquitous computing devices [1.9], taking into account the device miniaturization trend in the field of embedded systems. The taxonomies introduced by Weiser and Poslad primarily focused on the physical form and size of the devices. In this thesis, we distinguish embedded devices based on their *hardware platforms*, *software platforms*, and *networking capabilities*. It is crucial for heterogeneous embedded devices with varying capabilities to interoperate effectively, enabling the provision of robust distributed services and applications.

Firstly, in a ubiquitous system, it is crucial for different *hardware platforms* to be capable of interoperating. These platforms should have the ability to execute various components of a distributed application. Typically, a significant portion of embedded devices within a ubiquitous system consists of hardware components with limited resources, such as computing power, memory, storage, and energy capacity. Conversely, other embedded devices may not face such stringent limitations regarding these resources.

Secondly, the *software platforms* utilized in such embedded systems, including operating systems, software implementing application logic, and drivers, can vary widely. These software platforms may be tailored to specific user interests and application domains under consideration. Additionally, it is common for these embedded devices to have primitive low-level user interfaces or no user interface at all. In a ubiquitous system, applications should be able to run on embedded devices regardless of their specific software platforms.

Lastly, *networking capabilities* are essential for the embedded devices within a ubiquitous system to facilitate communication among themselves. This is achieved using various wired technologies (e.g., Ethernet, broadband, serial) and wireless technologies (e.g., Wi-Fi, Bluetooth, IEEE 802.15.4). Different embedded devices may employ one or more of these networking technologies, but not necessarily all of them.

Overall, a true ubiquitous system enables users to express their goals in an abstract manner and configures its embedded devices to fulfill those goals. The system takes into account user preferences, the current context (i.e., the context recorded during a user's ongoing activities), and stored contexts (i.e., contexts recorded and stored in a database

during a user's past activities) to determine the optimal state of the ubiquitous system that best satisfies the user's goals, while respecting privacy policies.

In order to illustrate the potential of this concept, we present the following example of a ubiquitous healthcare system as described in [1.12] and depicted in Fig 1.1.

Example Scenario (Ubiquitous Healthcare System): Consider a ubiquitous healthcare system that supports independent monitoring of patients. This healthcare system records a patient's movements using portable bio-electronic devices such as SpO_2, Electrocardiogram (ECG), Temperature, and Electromyography (EMG) sensors. These devices transmit the collected data via WiFi or Bluetooth to a smartphone, which further transmits it to the central base station. At the base station, the received bio-signals are analyzed and stored in the healthcare database. If critical information is detected, the patient's doctor receives this information for further treatment and care. The doctor can then immediately send assistance to the patient. Additionally, the doctor can query/access the stored history of the patient in the healthcare database.

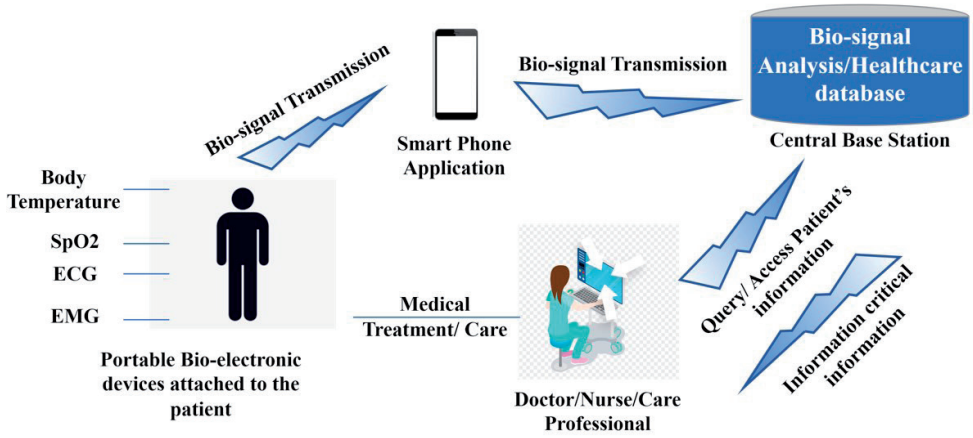


Figure 1.1. Ubiquitous healthcare system.

This scenario illustrates the use of a ubiquitous computing system in the healthcare domain. The system operates seamlessly, hiding the complexity of embedded devices from the patient and facilitating ubiquitous information sharing without requiring the patient's active involvement. In addition to healthcare, ubiquitous computing finds applications in various domains such as intelligent lighting, home care, environmental monitoring, robotics, sports, and transportation.

1.2 Smart Space Introduction and Background

One of the prominent areas of ubiquitous computing is the concept of *smart spaces*, which is currently being extensively researched [1.13]. In [1.14], the authors presented a fundamental conceptual view of a smart space within the context of intelligent environments. Smart space applications span across diverse fields of interest. To illustrate this, we provide a few examples of smart space scenarios in various applications, as shown in Table 1.1.

Table 1.1. Smart space application examples and scenario's descriptions.

Smart Space Application Example	Scenario description
Smart Classroom [1.15]	A smart classroom can assist lecturers in evaluating students, accessing assignments, and managing teaching plans. It can also support students in determining learning outcomes and accessing lecture materials in a comprehensive and organized manner. Additionally, it facilitates interactions between lecturers and students for questions and answers.
Smart Home Appliance Control System for Physically Disabled People [1.16]	A smart home can assist individuals with physical disabilities by enabling digital control of home appliances. A person with disabilities can use a smartphone to control various appliances. For instance, they can request the smartphone to turn on the microwave by simply saying, 'Turn on the microwave'.
Web Services and GSM based Smart Home Control System [1.17]	A user can remotely control home appliances using the Internet or GSM (Global System for Mobile Communications). For instance, critical values of temperature and gas levels can trigger notifications to a user through the Internet on their smartphone. Similarly, the notifications via GSM can immediately alert a user about critical environmental values, such as the presence of gas in the home.
Smart Home Gardening Management System [1.18]	A user can maintain their home garden by controlling parameters such as humidity, temperature, fertilizer compositions, and more. A cloud-based system is used for analysis, generating alerts, and notifying the user of critical values that require appropriate action.
Smart Healthcare Monitoring System using a Smartphone Interface [1.19]	The vital health parameters of a patient, including ECG, body temperature, blood pressure (BP), heart rate, glucose level, and galvanic skin response, can be collected and evaluated using smart devices. The data is then transmitted wirelessly for further analysis. The patient's vital statistics are communicated to a doctor's smartphone, allowing remote monitoring without the need for physical presence. Additionally, if any variations or critical signs are detected, automated notifications are received on the doctor's smartphone application.
Smart Healthcare System Considering Older Adults' Healthy [1.20]	A smart healthcare system can assist older adults aged 60 or above in their daily needs and help improve their lifestyle. This system is based on a survey approach in which older adults actively participate. The information collected through the survey enables the smart healthcare system to provide personalized assistance and support to older adults, ultimately enhancing their overall well-being and lifestyle.

Table 1.1. Continued

Smart Space Application Example	Scenario description
Smart Medicine Dispenser [1.21]	A smart medicine dispenser can assist patients in taking their medication on time, in a stress-free manner, without the risk of missing doses or accidentally overdosing. The dispenser includes an Android application that provides a user interface, controls the medication dispenser, and manages the user's schedule. An integrated alarm system is also in place to remind patients when it is time to take their medicine. Ultimately, the smart medicine dispenser greatly improves medication adherence, ensuring better treatment effectiveness and potentially saving lives.
Smart Lighting in Home Environments [1.22]	Smart lighting can enhance the lighting in a home by automatically adjusting the illumination based on light sensors. For example, the lights can be turned on, off, or dimmed according to user preferences. Smart lighting also contributes to energy-saving efforts.
Marketplace for Smart Cities [1.23]	A smart city can foster the growth of a local economy by facilitating the sharing of data from Internet of Things (IoT) devices related to market products. For instance, product owners can use IoT technology to decentralize the posting of product information on a marketplace, enabling potential buyers to access this information.
Smart Driving [1.24]	Smart vehicles play a crucial role in smart cities, with cars being the foremost type of vehicle that significantly impacts human life. For instance, drivers can enjoy a comfortable, safe, and easy driving experience in smart cars, thanks to the information provided by sensors installed both within the cars and in the surrounding infrastructure. This advanced technology reduces the likelihood of accidents and enables more convenient travel.
Smart Fire Management [1.25]	Fire protection is of utmost importance in ensuring the safety and security of large indoor garages or parking areas. Implementing smart management systems for fire incidents can greatly enhance the level of protection provided to the parked vehicles. For instance, gas and temperature sensors can be integrated with sprinkler systems in indoor parking spaces, following a carefully planned vehicle parking layout.
Streetlight Control for Smart Cities [1.26]	Streetlights are controlled using sensors and actuators that respond to human presence and luminous intensity. For instance, the streetlights turn on or provide adequate illumination when human presence is detected. This not only helps in energy savings but also ensures suitable lighting for people walking on the streets.
Smart Logistics using Smart Contracts [1.27]	Smart logistics assist in supply chain management within industries by incorporating smart contracts, logistics planners, and asset condition monitoring. For instance, the entire process from purchase to delivery can be efficiently managed using time management techniques and smart contracts, minimizing the involvement of service providers and users.
Smart Homes and Assisted Living as an Additional Service Offered to the Users [1.28]	A smart home not only promotes energy savings but also provides innovative services aimed at enhancing the quality of life for end-users. For instance, the integration of sensors and actuators in the smart home helps to prevent unnecessary electricity usage. Additionally, the infrastructure supports users in fulfilling their desires and preferences.
Smart Office [1.29]	Smart offices have the capability to control the power status of various electronic devices, including lamps, projectors, room temperature devices, computers, and security systems. For instance, these devices can be turned off when they are not required by the user in the office, and they are turned on only based on user requirements.

From the users' perspective, the goal of a smart space is to enhance people's living standards by offering services that go beyond what can be provided by a single device. To achieve this goal, a smart space consists of cooperating embedded devices that seamlessly sense events in their physical environment and monitor relevant contextual information for services that are interesting for users. This information can be processed and utilized to deliver personalized services based on user interests and preferences. As a result, users perceive the smart space as a unified entity that interacts with them and adapts its services to enhance their experience through reasoning. This reasoning can be based on contextual information collected from the physical environment (via sensors) and information obtained from databases (via user profiles). The application responsible for the reasoning can run on a single embedded device, processing information locally and making actuation decisions. Alternatively, in a smart space, distributed collaborative application logic can be used, which requires information exchange and semantic interoperability between embedded devices and their services. Semantic interoperability ensures that the exchanged information is interpreted with the same meaning. For instance, in a distributed application where one user's smartphone exchanges information with another user's smartphone, it is essential that both smartphones interpret the information in the same way. Achieving semantic interoperability in smart spaces remains a significant challenge.

From a system perspective, a smart space is a specific type of ubiquitous system. The example of a ubiquitous healthcare system in Section 1.1 explained the scenario in a mix of deployment aspects and a set of functions. In essence there are many ways that a functional scenario can be mapped on a deployment architecture (i.e., the operational part of the architecture). In a smart space, the functional part of a scenario should be the concern for a designer of the application and the smart space must realize its operational part (i.e., the exact roles of every component and the communications involved) transparently. Being a form of a ubiquitous system, a smart space is usually heterogeneous, i.e., made up of embedded devices of many vendors with various hardware, software and communication platforms.

Unfortunately, there is a lack of global ambition for universal standardization of semantic interoperability. Large vendors seeking market dominance do not recognize the need for interoperability with other vendors, while small vendors are often specialized in specific products and lack the resources and influence to promote standardization. As a result, there is a proliferation of data formats, taxonomies, and ontologies, resulting in fragmented smart space solutions that lack easy interoperability. It is worth noting that this is true even for smart space solutions that achieve communication interoperability with each other. Secondly, the capabilities of devices in a smart space vary in terms of sensing, actuation, processing, storage, and networking. For instance, a wireless sensor-actuator

network may consist of low-power sensors and actuators communicating over the IEEE 802.15.4 [1.30] wireless standard, which can be bridged to a network of quad-core smartphones and laptops through a gateway. This heterogeneity within a smart space network highlights the diverse nature of devices and their capabilities.

Using the analogy of a smartphone, a smart space can be seen as a programmable platform capable of running multiple applications concurrently. This platform concept, which employs an open interface, has been widely acknowledged by various authors and recent projects, and is considered crucial for advancement. In terms of progress, multimedia applications have made significant strides in this direction, with initiatives such as the Digital Living Network Alliance (DLNA) [1.31], Spotify [1.32], and Google technologies [1.33]. In smart space design, it is important to recognize that there are multiple stakeholders involved and that there are various criteria to consider beyond just functionality. Therefore, we advocate for smart space design that takes into account the perspectives of multiple stakeholders and considers multiple qualities assessed by different metrics.

Finally, smart spaces are also associated with the utilization of IoT technology. IoT technology is built on the concept of enabling connectivity anytime, anywhere, and for anything, which holds great promise for the future of computing and communication. It forms a vast network of Internet-connected objects, including embedded devices, home appliances, and sensors. Smart spaces can leverage IoT-enabled technologies to establish an infrastructure for human-computer interactions. For instance, a comprehensive approach that combines smart spaces and IoT technology is presented in [1.34]. This approach proves to be effective and facilitates the division of the global IoT into manageable smart spaces. IoT provides a unified infrastructure that accommodates diverse devices, technologies, and protocols, thereby enabling the development of applications for smart spaces that integrate ubiquitous computing and IoT.

1.3 Problem Statement and Research Questions

The presented vision of ubiquitous computing and smart spaces stimulates researchers to design and build working and living environments that improve user experiences via applications that run on top of embedded devices. However, research contributions and implementations in this direction are quite dispersed. This is mainly due to the fact that *there is no generic framework in the literature that defines how to design and implement a smart space*. Smart space architecture designs in the literature are mostly application specific, their concepts and components defined based on the specific needs of one application. Therefore, this thesis addresses the following leading Research Questions (RQs) related to this problem.

In order to formally define a smart space and establish a sharp domain of this research we formulate the following RQ:

RQ_1: What are the concepts, properties and architectural design alternatives of smart spaces?

RQ_{1A}: What are the fundamental concepts, building blocks and properties of a smart space?

RQ_{1B}: How can we compare potential smart space architectural design alternatives?

RQ_{1C}: What is an appropriate architectural solution for realizing these concepts? Which smart space properties are the most essential for this architectural solution?

Clearly, answering RQ_1 requires a thorough analysis and comparison of the smart space architecture designs available in the literature as well as a thorough examination of (potential) applications.

Regarding semantic interoperability in a smart space with various types of embedded devices and networks, we formulate the following RQ:

RQ_2: How can we establish semantic interoperability of heterogeneous embedded devices (nodes) in smart spaces?

RQ_{2A}: What are the processes and dependencies for semantic interactions among components that will enable semantic interoperability?

RQ_{2B}: What are the additional mechanisms and components needed for semantic interoperability in smart spaces? How can we achieve semantic interoperability with resource-poor (low-capacity) devices in a smart space?

RQ_{2C}: How can we achieve semantic interoperability with multiple applications in a smart space?

To experiment with semantic interoperability in a practical application, we have chosen to implement smart lighting applications in this thesis. For these smart lighting applications, we require an appropriate illumination model that can be integrated into the proposed semantic interoperability approach. To identify and implement our proposed approaches for RQ_1 and RQ_2, we have formulated the following research question:

RQ_3: How can our proposed solutions be applied in real systems?

RQ_{3A}: How can we map the concepts and properties of smart spaces in real systems (including an illustration of how to measure the performance of a smart space architecture)?

RQ_{3B}: What are the smart space life cycles?
RQ_{3C}: How can semantic interoperability be established in real systems?

1.4 Research Context and Methodology

Before describing the technical contributions in detail, we explain the research context and research methodology adopted in this thesis.

1.4.1 Research context

The research in this thesis was done in two phases. Initially, the thesis work was originated from the European Union Artemis project Smart Objects For Intelligent Applications (SOFIA) [1.35]. SOFIA was executed from 2009-2012, with the aim of developing an Interoperability Platform (IOP) that enables collaboration and data exchange between devices of multiple vendors in various target environments. The main research and development objectives of the SOFIA project were as follows:

- *Objective 1*: To develop a reference IOP, enabling the interoperability levels for exchanging information between multivendor devices.
- *Objective 2*: To develop interaction models to support a variety of smart spaces and users.
- *Objective 3*: To introduce methods, technological and economical structure and toolkits for the development of smart environments and their services and applications.
- *Objective 4*: To demonstrate the capabilities of the proposed IOP and interaction models using scenarios and use cases based on personal spaces, indoor spaces and cities.

In line with the aforementioned objectives, SOFIA offered an IOP solution for smart spaces involving multiple vendors, devices, and domains, known as Smart-M3 (Multi-vendor, Multi-device, Multi-domain). The Smart-M3 solution allows for the integration of information across diverse applications and domains, spanning from the embedded domain to the Internet domain. It boasts flexibility and modularity, making it compatible with various transport technologies, application development environments, and programming languages.

Within the scope of the SOFIA project and this Ph.D. research, we conducted experiments on smart lighting applications. To accomplish this, we closely collaborated with the TU/e department of Industrial Design, NXP Semiconductors, Philips Research, and CONANTE Advanced Interface Solutions. The outcome of our collaboration was a combined demonstration of a SOFIA Smart Home Pilot utilizing Smart-M3, as explained in Appendix A. Additionally, we introduced and developed a novel model for smart lighting applications, which we implemented using Smart-M3.

Years after the completion of the SOFIA project in 2012, the second phase of this thesis work commenced in 2017. The objective of this phase was to develop and precisely define general smart space concepts, building blocks, properties, and architectures. Additionally, we addressed the challenge of semantic interoperability in smart spaces, drawing from the experiments conducted in the first phase and the findings of the literature review.

1.4.2 Research Methodology

To identify and address the research questions mentioned in Section 1.3, the research methodology shown in Fig. 1.2 was employed. The methodology comprises several stages including problem analysis, proposed solutions, solution and implementation mechanisms, implementations and evaluations, and concluding remarks with future directions.

In the first phase, known as the problem analysis phase, we conducted a thorough review of the relevant literature on smart spaces. This review aimed to identify research questions that have not been formally or sufficiently addressed in previous studies. The research questions identified during this phase are listed in Section 1.3. The subsequent phase involved proposing solutions to these research questions. This was followed by designing solution mechanisms and implementing the proposed solutions on a hardware infrastructure testbed. Based on the implementations, we further refined the proposed solutions. Additionally, we evaluated the performance of the implementations and made improvements accordingly. We then discussed the answers to the research questions based on the implementation of the smart space solution. Finally, we provided concluding remarks and outlined future directions for research.

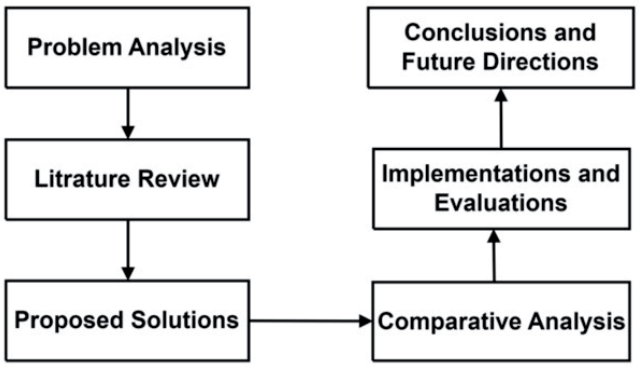


Figure 1.2. Research methodology.

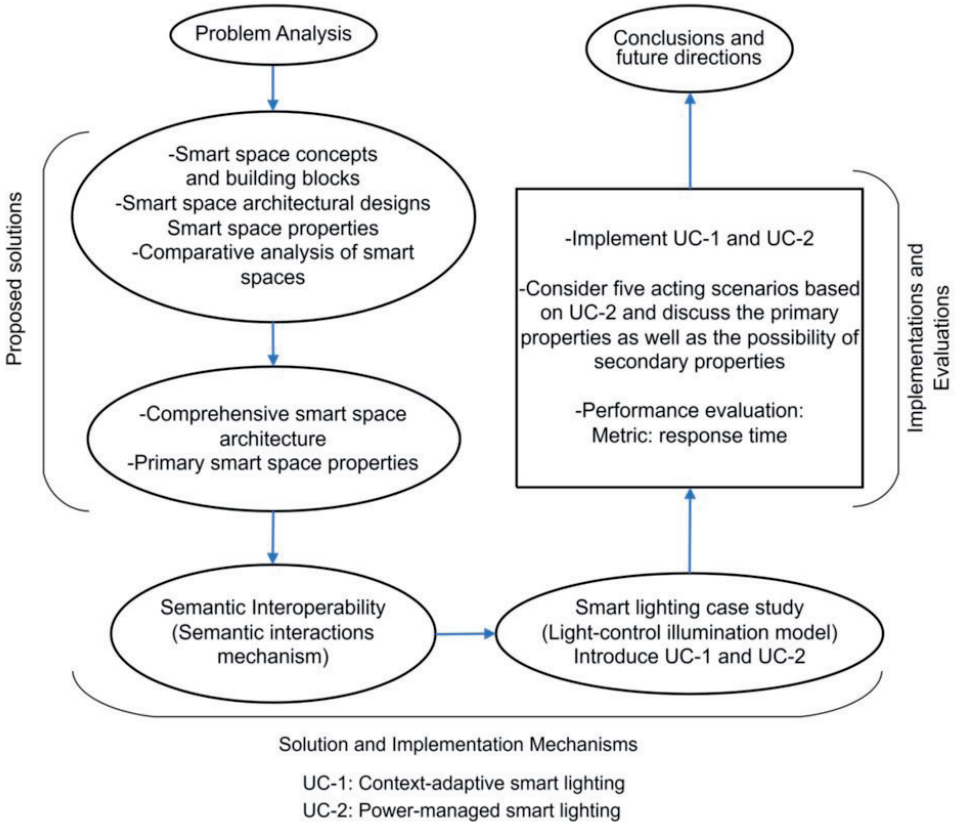


Figure 1.3 Contributions in the thesis.

1.5 Contribution and Organization of the Thesis

In this section, we outline the main contributions of the thesis, along with the organization of the remaining chapters. The main contributions of this thesis are summarized in the following six chapters and depicted in Fig. 1.3.

Chapter 2 addresses **RQ_{1A} and RQ_{1B}**. In this chapter, we identify the concepts, building blocks, and architectural views of a smart space. We provide formal definitions of a smart space and its concepts, and introduce the main components of its architecture. Additionally, we propose new architectural designs for smart spaces, including centralized, decentralized, and distributed architectures, which are then compared with the current state-of-the-art designs. A comparative analysis of smart space designs in the literature and the proposed designs is presented in terms of hardware and software capabilities.

Chapter 3 addresses **RQ_{1C} and RQ_{2A}:** In this chapter, we aim to identify the best smart space architecture based on an objective evaluation metric. We provide a detailed description of six smart space properties: adaptation, communication interoperability, semantic interoperability, openness, extendibility, and self-management. We propose a semantic interoperability architecture that offers a generic solution for smart space design. Our analysis determines adaptation, communication interoperability, and semantic interoperability as the primary properties for the proposed architecture. These primary properties are selected based on the essential needs of smart spaces (i.e., adaptation and communication interoperability) and the challenge of achieving semantic interoperability. We discuss the processes and dependencies among components for semantic interactions, which contribute to achieving semantic interoperability.

We provide guidelines for creating smart spaces using smart embedded devices and their basic modules introduced in the proposed semantic interoperability architecture. We provide suitable mechanisms of interaction among smart embedded devices in smart spaces, thereby achieving semantic interoperability.

Chapter 4 addresses **RQ_{2B} and RQ_{2C}**: In this chapter, we present a detailed explanation of semantics and reasoning, highlighting their role in achieving semantic interoperability. We introduce ontologies and discuss their significance in facilitating semantic interoperability. We explore the mechanisms of semantic interactions within a smart space and between smart spaces. Furthermore, we introduce an additional component, namely a gateway node, in the semantic interoperability architecture, which enables interoperability between low-capacity and high-capacity smart embedded devices.

Chapter 5 and **Chapter 6** address **RQ_3**. These chapters focus on applying the proposed solutions discussed in chapters 2, 3 and 4 in practice, by mapping them to real use cases. We first present a case study of smart lighting applications within the scope of the SOFIA project, requiring the development of a novel smart lighting model. The establishment of semantic interoperability is thoroughly discussed using the semantic interoperability architecture. Furthermore, two use cases, namely UC-1: *context-adaptive smart lighting* and UC-2: *power-managed smart lighting*, are introduced. UC-1 primarily emphasizes semantic interoperability between low-capacity and high-capacity smart embedded devices using the smart lighting model. UC-2 demonstrates semantic interoperability between multiple smart spaces composed of various nodes with different computing and communication platforms, forming a heterogeneous network. Lastly, in **Chapter 6**, we propose and discuss the life cycles of smart spaces. The life cycle of smart spaces involves several stages, from initial conceptualization to ongoing maintenance and potential decommissioning. We use these life cycles in the development of smart spaces.

Chapter 7 describes our implementation of the use cases, namely UC-1 and UC-2. To achieve this, we examine five example scenarios of lighting conditions based on the experimental results from the implementation of these use cases. In addition to discussing the primary properties, we also address other properties using these example scenarios. Furthermore, we calculate a performance metric, specifically the end-to-end delay between smart embedded devices. Finally, we evaluate the results to identify suitable solutions in smart spaces.

The origin of Chapters 2, 3, 4, 5, 6 and 7 is based on the following publications:

[P1] S. Bhardwaj, K. M. Lee, J. Lee, "An adaptive framework for applying machine learning in smart spaces", SAC 19, Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp.1263-1270, April 2019.

[P2] S. Bhardwaj, T. Ozcelebi, J. J. Lukkien, and K. M. Lee, "Smart Space Concepts, Properties and Architectures", IEEE Access, Vol. 6, pp. 70088-70112, Nov, 2018.

[P3] S. Bhardwaj, T. Ozcelebi, J. J. Lukkien, and K. M. Lee, "Semantic Interoperability Architecture for Smart Spaces", International Journal of Fuzzy Logic and Intelligent Systems, Vol. 18, Issue 1, pp. 50-57, 2018.

[P4] S. Bhardwaj, T. Ozcelebi, O. Ozunlu and J. J. Lukkien, "Increasing Reliability and Availability in Smart Spaces: A Novel Architecture for Resource and Service Management", IEEE Transactions on Consumer Electronics, (TCE), Vol.58, Issue 3, August, 2012.

[P5] S. Bhardwaj, T. Ozcelebi, C. Uysal and J. J. Lukkien, "Resource and Service Management Architecture of a Low-Capacity Network for Smart Spaces", IEEE Transactions on Consumer Electronics (TCE), Vol.58, Issue 2, May, 2012.

[P6] S. Bhardwaj, T. Ozcelebi, R. Verhoeven and J. J. Lukkien, "Smart Indoor Solid-State Lighting Based on a Novel Illumination Model and Implementation", IEEE Transactions on Consumer Electronics (TCE), Vol.57, Issue 4, November, 2011.

[P7] S. Bhardwaj, A. A. Syed, T. Ozcelebi, and J. J. Lukkien, "Power-managed Smart Lighting Using a Semantic Interoperability Architecture", IEEE Transactions on Consumer Electronics, Vol.57, Issue 2, 2011.

Finally, **Chapter 8** concludes this thesis by providing a summary of the key results and the main contributions of our work in relation to our research questions. Additionally, we highlight potential future research directions for the problems addressed and the solutions presented.

Chapter 2

Smart Space Concepts and Architectures

In this chapter, we focus specifically on the concept of smart space, aiming to define it rigorously, identify its distinguishing characteristics, and propose universally applicable architectural concepts. Initially, we offer formal definitions of a smart space and its fundamental computational components, namely, smart nodes and information objects. Subsequently, we present potential architectural designs for smart spaces. Additionally, we provide a comprehensive review of related work and conduct a comparative analysis of the proposed architectures. Finally, we conclude the chapter by summarizing our findings and presenting key insights.

2.1 Introduction

The advancements in the smart space research bring us closer and closer to a future, in which the living standards of people are greatly enhanced. Many strongly believe that user needs constitute the main driving force behind technological development, but sometimes it is the other way around. Technological advancements change the ways that people interact, perform activities, and connect with their environments. For example, a smart television can now do much more than simply receiving and displaying video signals. With Internet Protocol (IP) connectivity, advanced software, and plenty of computational power, smart televisions allow users to surf the Internet, browse movie libraries of local media servers, stream and play movies in various encoding formats, play online games, join video chat sessions, and much more. Nowadays, smart televisions come with a very capable built-in operating system and a variety of ('killer') entertainment applications that are so easy to use and so difficult to miss out on. Similarly, touch-screen-enabled smartphones with 4G and 5G connectivity have changed the way mobile users view cellular communication technology. Smart spaces also represent another area in which the driving force is mainly a technological push, i.e., the functionality is not called for by a globally widespread "killer application" and the utility of such spaces must be understood over time by experience. The technological push in this instance is the increased prevalence of electronic devices around and on people combined with the fact that these are networked and produce valuable information. *Smartness* here refers to the ability of these devices

to perform (collective) behaviors perceived as advanced and useful in some sense. Since the number of devices is rapidly exceeding the number of human users, this smartness implies (self-) management and configuration capabilities in a network setting.

Although smart spaces have not been established very precisely, many smart (space) applications have been developed both as project showcases [2.1-2.4] and for real deployments. These applications are characterized by the fact that the hardware and software elements of the system are dedicated to and specifically developed for the application at hand. Examples can be found in healthcare (e.g., patient monitoring, elderly monitoring), smart lighting, media use, and environmental monitoring. Special-purpose systems, however, are costly and do not lead to the commoditization of the system components. Realizing general-purpose smart spaces of the future requires a comprehensive list of generally applicable smart space architectural concepts and building blocks. For that, we first need precise definitions of the things that make up a smart space and the definition of a smart space.

2.2 Smart Space Concepts and Building Blocks

In this section, we first define the terminology regarding smart space concepts. The proposed list of smart space concept definitions will be used in describing a smart space formally, allowing to make a distinction between what is a smart space and what is not. A smart space delivers services such as context-aware information services [2.5-2.6], as well as physical services through actuation. Note that service refers to a set of functionalities such as the retrieval of identified information or the execution of a set of processes or actions. A smart space is generally known by its physical extent, embedded electronics, embedded networks, and software. In the literature of networked embedded systems an *object* is a combination of an embedded networked device (also called a node) with the software running on top. In this thesis, for clarity of presentation, we distinguish between *smart nodes* (hardware) and the software modules running on top that produce and consume information. We will discuss what smartness entails in detail but let us first discuss these software modules and their roles. In smart spaces, we refer to the software modules hosted by smart nodes as *information objects* (*iO*s). A given smart node can host various *iO*s.

Definition (Information Object): An information object is a software unit that can communicate with other information objects and has a local state, which it can change and, on which it can perform digital computations.

In general, an *iO* has both a *state* and a *behavior*. The state of an *iO* is the present condition of that *iO*. Changes in the state of an *iO* can be autonomous, which models a sensing capability. Similarly, some changes of an *iO* state can result in changes in the physical world, which models an actuation capability. The state of each *iO* may change over time and based on a set of (timed) events in which it can engage. Events are recognized by the changes in the *iO* state, which may include sensing, actuation, user interface events, and communication with other *iO*s.

Definition (*iO* State): The state of an *iO* is given by the set of values taken by its properties that may be static (e.g., *iO*'s identity, fixed coordinates of an immobile hosting device) or dynamic (e.g., dynamic IP address of the hosting device, values of dynamic variables).

Definition (*iO* Behavior): The series of messages sent and received by an *iO*, its state changes (events), actuations and the associated timing relations together form the *iO's behavior*.

A certain behavior of an *iO* may cause (trigger) a particular behavior of another *iO*. The behavior of an *iO* corresponds to state changes, which occur when an action or reaction takes place. For example, actions are based on sensing illumination by a light sensor, while reactions are triggered by messages received by a lamp to dim/on/off. The joint behaviors of the light sensor and the lamp cause the actions and reactions by *iO*s. For example, an action can be observed at the light sensor (starts observing illumination reading) and the reaction can be modifying the state of the light lamp (dim/on/off) in return. Note, such an *iO* is a producer or consumer of digital information, has a (dynamic) state and can communicate with other *iO*s.

All *iO*s are deployed on smart nodes.

Definition (Smart node): A smart node, also known as a smart embedded device, is a dedicated computational hardware component that hosts information objects. For a smart node *n*, *n.iO* denotes the set of *iO*s running on *n*. As smart spaces are the focus of this thesis, we will use the term *node* to refer to *smart nodes* in the rest of this chapter and in further chapters.

We consider that the (smart) nodes in an environment are placed in a limited physical area, called an *E_Space*, which is connected to the virtual world of information provided by its nodes.

Definition (*E_Space*): An *E_Space* is an infrastructure of nodes and network connections between them in a given physical environment. An *E_Space* $\mathcal{E}$ is modeled by a graph given by

$$\mathcal{E} = (\mathcal{E}.N, \mathcal{E}.C)$$

where

1. $\mathcal{E}.N$ denotes the set of nodes, where nodes further contain a set of *iO*s, $\mathcal{E}.C$ denotes the set of connections (edges) between the nodes. Note that $\mathcal{E}.C$ may denote a set of individual physical links or a fully connected network based on some network layer abstraction. In the latter case, $\mathcal{E}.C$ would include multi-hop end-to-end network layer connections between nodes and be a fully connected graph.
2. An element *e* of $\mathcal{E}.C$ is denoted by a pair (*e.s,e.t*) with *e.s(ource)* and *e.t(arget)* chosen in $\mathcal{E}.N$.

An *E_Space* may evolve over time because of new nodes that join and existing nodes that fail or leave, e.g. due to mobility. Fig 2.1 illustrates an example *E_Space* with three nodes n_A, n_B and n_C with associated *iO*s iO_A, iO_B and iO_C, respectively.

The edges between n_A *and* n_B, n_B *and* n_C, and n_C *and* n_A are given by e_{AB}, e_{BC} and e_{CA}, respectively and are detailed as follows:

$$e_{AB} = (e_{AB}.s, e_{AB}.t) = (n_A, n_B)$$
$$e_{BC} = (e_{BC}.s, e_{BC}.t) = (n_B, n_C)$$
$$e_{CA} = (e_{CA}.s, e_{CA}.t) = (n_C, n_A)$$

For two *iO*s to communicate they either need to be on the same node or their nodes should be able to exchange messages over a network. In the graph representation this corresponds to having a solid edge between the nodes. For example, an iO_A on n_A can send messages to another iO_B on n_B only when $e_{AB} = (n_A, n_B)$ is an element of $\mathcal{E}.C$.

We shall now answer the question: What makes a smart space *smart*? The dictionary definition of "smart" commonly refers to possessing a mental state that enables human beings (or animals) to demonstrate quick thinking and intelligence as an individual. Hence, the ability to react to changes adequately and timely is embedded in the notion of smartness. The perception of smartness in this sense in a smart space typically comes from the interactions between *iO*s, i.e. a certain behavior of an *iO* that leads to a particular behavior of another *iO* as a reaction based on interactions between them. An example is the detection of occupancy in a room by a presence sensor node, triggering a state change of the sensor *iO* followed by message transmissions to a light controller *iO*, which in turn sends actuation commands to an actuator *iO* on a light source node as shown in Fig 2.2.

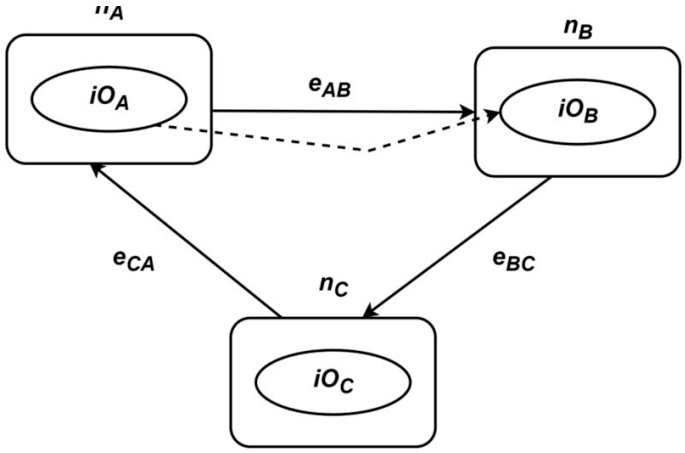


Figure 2.1. The graph illustrates the *iO*s (iO_A, iO_B and iO_C) deployed on nodes n_A, n_B and n_C.The solid arrows represent the edges in $\mathcal{E}.C$ and their directions while the dotted arrow represents a process level connection between the *iO*s of n_A and n_B.

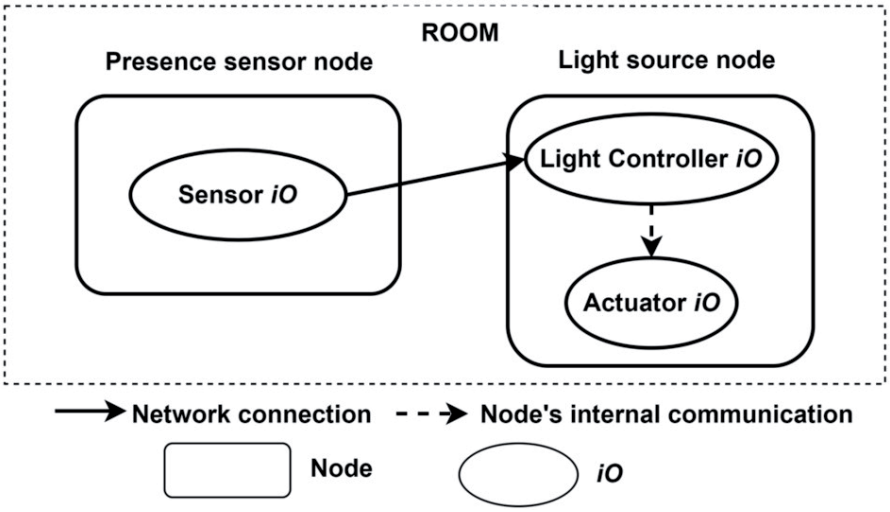


Figure 2.2. An example of a smart behavior via interactions between *iO*s.

Note that smartness is also possible to realize even with resource-poor nodes. In fact, even the so-called passive nodes that require external power sources to become active are considered to be smart nodes in this sense, for example, Radio Frequency Identification (RFID) tags. Nowadays a single smart node can contain many sensors and actuators, act as both a producer and consumer of several types of information and participate in multiple

applications. Smart nodes exhibit node-to-node, node-to-cloud, node-to-gateway, and back-end communication patterns.

A smart space is further characterized by the scenarios collectively realized by its *iO*s, which are sequences of events (state changes). We refer to these scenarios as the (potential) applications of a smart space.

> **Definition (Smart Application)**: A smart (space) application (A) is a set of scenarios realized by communicating *iO*s that together aim to serve and interact with application users and the electronics these users carry.

The application context A_c of a smart application A is the collective state of all *iO*s taking part in A, including external states monitored by these *iO*s that may influence the behavior of A. Therefore, we can say that the collective behavior of *iO*s that form and influence Adefines the application behavior A_b. Thus, the application behavior A_b depends on A_c and is, in fact, fully defined by it. In interactions between the stakeholders and the application A, events are triggered, e.g., a button is pressed, some user interface input or output is given, or some action is performed. These events affect the corresponding *iO*s whose states are part of the application context A_c. We refer to these as *interface objects*, and their states form a subset of A_c. Whether an *iO* is an interface object is subjective (influenced by the abstraction level of a stakeholder's interactions), but we typically restrict ourselves to *iO*s that affect the function of the application A. If the application behavior A_b is fully defined by these interface objects, we say that the application is *context-independent*; otherwise, we call it *context-dependent*.

Smartness and other qualities are perceived by stakeholders through interactions and state observations and are therefore properties of behaviors. We associate these properties with metrics. A *metric m* in this setting is a mapping from the application behavior A_b to a non-negative real number. When a lower value is better, the metric is called a *cost function*. An application A is called adequate with respect to a metric *m* if A satisfies a certain requirement on *m*, for all behaviors in A_b.

We can now define what we mean by a smart space.

> **Definition (Smart Space)**: A smart space is an *E_Space* with embedded information and communication technologies running a set of smart applications.

As a platform, a smart space must contain and build up knowledge about its capabilities and resources, its state (contexts) and history. To do so, it employs sensing, user interaction, resource monitoring, communication, computation, cooperation, and services on

the Internet. Furthermore, a smart space utilizes a variety of architectural components to manage applications and the available resources as well as to provide security and ensure the privacy of the smart space knowledge considering access rights. Further on, this chapter presents the fundamental hardware and software building blocks that make up smart spaces, starting with a classification of smart nodes.

2.2.1 Classification of smart nodes

The available node classes in networks of resource-constrained nodes were defined in [2.7-2.8] based on an investigation of the commercially available chips and embedded system designs. In this taxonomy, class 0 (*C*0) nodes are strictly constrained in terms of processing and memory (dynamic memory and permanent storage much less than 10 KiB and 100KiB respectively) capabilities. Therefore, these nodes are not able to run the IP stack. We call some nodes belonging to this class *passive nodes* since they depend on event-based energy harvesting for sending a few messages back-to-back into the network before their harvested energy is fully depleted again. A battery-less wireless light switch is an example of this. *C*0 passive nodes typically do not facilitate any management or security services other than pairing with other nodes over a trusted proxy. *C*0 nodes that are not passive can handle keep-alive messages and provide basic node state information. *C*1 nodes have roughly at least 10 KiB of memory space and 100 KiB of storage space, and they are mostly battery powered. Such resource limitations still do not allow running complex network protocol stacks such as the full IP stack. Nevertheless, there are low-resource (lightweight) protocol stacks specifically designed for this node class. For example, these nodes can use CoAP (Constrained Application Protocol) [2.9] over UDP (User Datagram Protocol) and employ 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) as adaptation layer to communicate directly with the Internet. Functionally (including security), with very careful use of resources, they would act like ordinary IP endpoints. However, in terms of network latency, throughput, and computational performance, these nodes perform poorly due to resource optimization techniques such as radio duty cycling. With around 50 KiB of memory and 250 KiB of storage space, *C*2 nodes can run conventional network protocol stacks. However, in practice, *C*2 nodes also employ low-resource protocol stacks as a performance precaution.

Naturally, such taxonomy needs frequent updates as the classes and their capabilities change continually thanks to developments in silicon technology. With this in mind and based on the architectural requirements for involvement in a smart space, we define two broad categories of smart nodes with respect to the available resources and communication capabilities: *high-capacity smart node* (HSN), and *low-capacity smart node* (LSN). HSNs are members of *C*2 or more capable. Typical examples of HSNs are smart phones, tablets, personal computers, and other high-capacity embedded devices. Contrary to LSNs, HSNs can employ complex services and protocols and perform relatively high-level

and complex reasoning. LSNs are resource-poor passive nodes of *C*0 such as RFID tags and battery-less switches and also the members of *C*1 such as sensor nodes (SNs) and actuator nodes (ANs).

2.2.2 Types and logical structure of *iO*s

A smart space allows seamless information sharing among *iO*s, i.e., information such as contexts. Smart space applications consist of sets of scenarios that deliver information services and their associated services through their *iO*s. The logical structure of relations between various *iO*s in a smart space is depicted in Fig. 2.3. The types of *iO*s shown in Fig. 2.3 are described in Table 2.1.

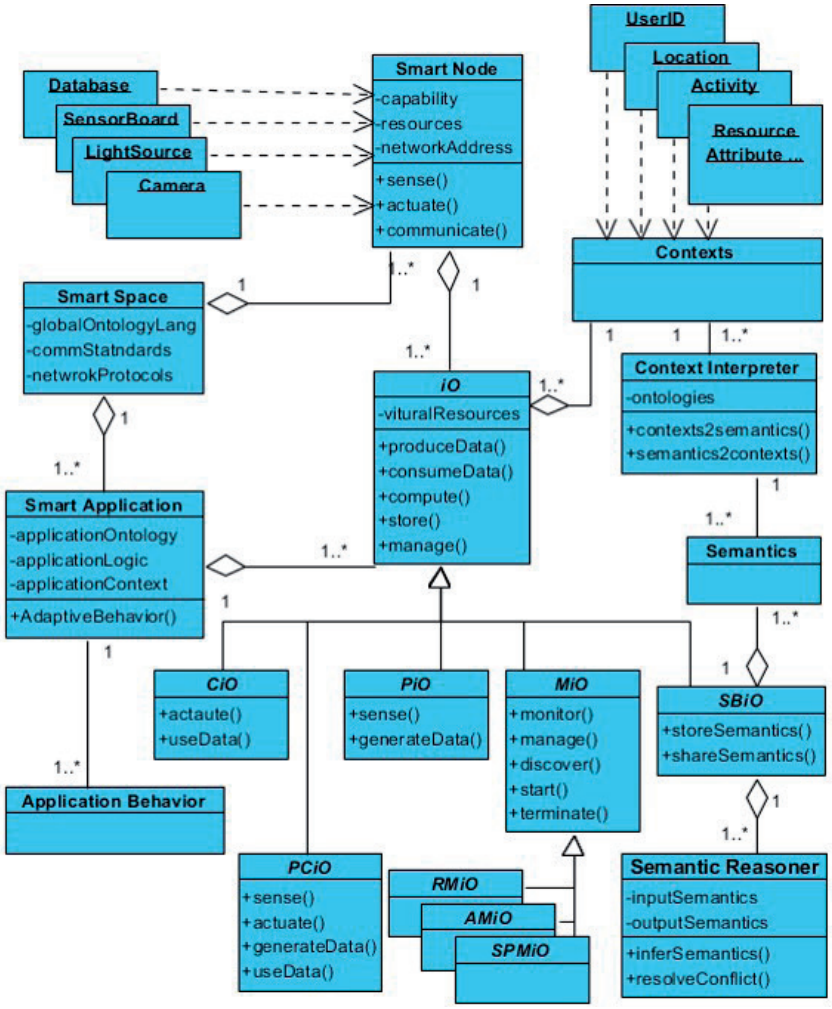


Figure 2.3. A diagram depicting the logical structure of relations between *iO*s in a smart space.

Table 2.1. Types of *iO*s shown in Fig. 2.3.

Item	Description
PiO	produces information for other *iO*s.
CiO	consumes information of other *iO*s.
SBiO	a module that stores and shares semantics (data with certain meaning)
MiO	manages resources, applications, security and privacy.
RMiO	an *MiO* instance that manages resources that *iO*s can access.
AMiO	an *MiO* instance that manages applications.
SPMiO	an *MiO* instance that provides security and privacy services.

Figure 2.4 shows the possible deployment of *iO*s on nodes. When we deploy *SBiO* and *MiO* on any node, the node becomes a semantic broker node of a smart space. In Fig 2.4(a), we deploy *PiO* and *CiO* on separate nodes; then the nodes become producer and consumer nodes in a smart space, respectively. It is also possible to deploy both *PiO* and *CiO* on the same node, in which case the node becomes both a producer and a consumer of information in a smart space. In Fig 2.4 (b), we can deploy *SBiO*, *MiO*, and *PiO* on the same node, and *CiO* on a different node. This choice makes the semantic broker node also a producer node. In Fig 2.4(c), *PiO* and *CiO* are both deployed along with *SBiO* and *MiO*. This means the semantic broker node also becomes both a producer and consumer in a smart space. Finally, in Fig 2.4 (d), we deploy *PiO* on a separate node and the others on the same node. This choice makes the central entity also a producer node in a smart space. We can choose

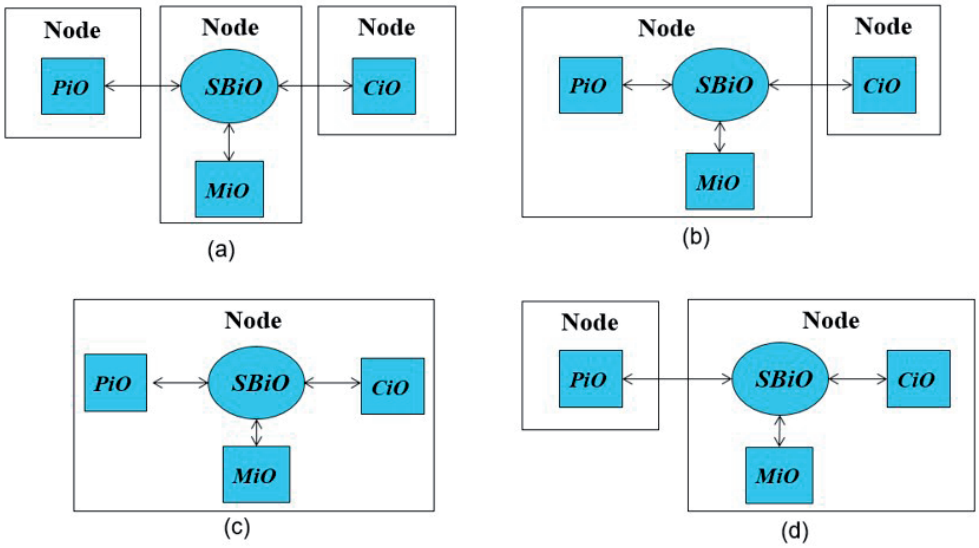


Figure 2.4. *iO*'s deployment possibilities on nodes. The arrows in the figure indicate a two-way message exchange.

any of the deployment options based on the requirements of smart space applications. We will discuss the smart space architectural designs in the next section, where we can choose how to deploy *iO*s accordingly. Later in Chapter 3, we choose Fig 2.4(a) as the option to deploy *iO*s in a smart space architectural design. This choice makes the smart space distributed for *iO*s, allowing information sharing by the semantic broker node.

An application A has application-specific contexts, application logic, and application ontology. The application context A_c is a set of contexts captured from the environment, which are represented in the states of *iO*s (*such as user ID, location, activity, and resource attribute value*) and are input to the application logic. The application logic has a set of instructions (a program code of implementation logic) to execute application-specific scenarios. The application ontology defines the set of concepts that are relevant, allowing to provide a flexible description of the application context A_c. The application A realizes its application behaviors by the collectively adaptive behaviors of *iO*s taking part in A. A smart node has its own capabilities, resources, and network address(es) to associate with other smart nodes, and to provide *sense*(), *actuate*(), and *communicate*() services. *Producer iOs* (*PiOs*) on a smart node, e.g. a light sensor node, produce knowledge in a smart space, whereas *consumer iOs* (*CiO*s) on a smart node, e.g., a light controller node, utilize such knowledge. Note that a smart node may host multiple *iO*s, allowing it to both produce knowledge and utilize knowledge produced by others.

A *context interpreter* gathers raw data from one or more *iO*s of an application A, summarizes them as A_c and converts the results into the established *semantics* [2.10] of a smart space. For example, a context interpreter participating in an application A involving user interaction can take raw accelerometer data and convert them into gesture semantics (*such as "pointing upwards"*) and further into corresponding control semantics (*e.g., increasing the light intensity by 10%*). It can also perform conversion in the reverse direction, i.e., from semantics to contexts. Semantics are represented using a high-level description language such as the Resource Description Framework (RDF) [2.11]. Contexts are expressed in a standard way using semantics and can be exchanged between *iO*s without loss of meaning. The semantics are then stored by a *semantics broker iO* (*SBiO*), where they are accessible by *iO*s that can perform reasoning on high-level application data. The stored semantics are also accessible by manager *iO*s (*MiO*s), whose instances are application manager *iO*s (*AMiO*s), resource manager *iO*s (*RMiO*s), and security and privacy manager *iO*s (*SPMiO*s). These instances are explained as follows:

- *AMiO* refers to a software module responsible for managing the installation, configuration, deployment, and monitoring of applications within a smart space computing environment. It plays a essential role in ensuring that applications run efficiently and reliably.

- *RMiO* is responsible for allocating resources within a smart space computing environment. It ensures that resources such as CPU, memory, storage, and network bandwidth are efficiently utilized by various applications and processes.
- *SPMiO* is responsible for ensuring that security and privacy measures are implemented and maintained within a smart space computing environment. It protects sensitive information, preventing unauthorized access, and complying with privacy regulations and policies.

Note: An *SBiO* is the core software component, i.e., an *iO* of the architecture, where the semantics are stored and shared by employing an ontology graph. We apply a specific reasoning at the *SBiO* for exchanging semantics or to produce results on queries in a smart space. Therefore, semantic interoperability is supported by the *SBiO*s in smart spaces. We will explain this in detail in Chapter 4.

Smart spaces may employ *management* to deal with factors that may cause an application to fail, such as attacks, lack of resources, hardware and software failures, and network topology changes. As trusted systems [2.12], they must ensure data privacy and enforce security measures by employing *SPMiO*s. We develop smart spaces in physical environments that present opportunities for offenders to trigger security incidents and cause harm. For example, in 2015 [2.13], offenders were able to access the Ukrainian power grid network to contaminate various nodes, such as a serial-to-Ethernet converter and power breakers, with malware. The power outage due to this affected more than 200,000 customers. In [2.14], the authors proposed a meta-model to represent security incidents in smart spaces, especially in smart buildings. They developed an automated approach to report incidents across a management team of the smart building. They presented a security incident that occurred in the smart building. An offender entered the toilet and then connected to a smart light source (installed in the toilet) using a smartphone. This gave the offender access to the installation bus, allowing to sniff unencrypted network traffic and collect application data (e.g., data exchanged between a Workstation and an HVAC). The offender then sent a targeted malware (e.g., exploiting the vulnerabilities present in HVACs) to disable the HVAC, subsequently causing the servers to heat up. A model with an automated approach to reporting such incidents is developed in [2.14]. The mechanisms for security and privacy (as treated by *SPMiO*) are beyond the scope of this thesis work, but we discuss *SPMiO*s as an option with the proposed architectural designs in the next Section 2.3. The *RMiO*s monitor the available resources and allocate them to *iO*s as necessary. They re-orchestrate *iO*s when certain *iO*s fail, change their behavior, or leave the smart space. In this way, failures are handled at the *iO* level before they cause application failure. “Failure” also includes poor application behavior or, equivalently, failure to adhere to specifications. In this thesis, we focus on *MiO*s to establish the con-

nection between smart spaces. The functionalities and operations of *MiO*s are discussed in detail in Chapter 4.

2.3 Smart Space Architectural Designs

The architectural design of a smart space entails the physical components (i.e., smart nodes), the connections between them, the deployment of *iO*s and the corresponding allocation of functionality to a smart node. We discriminate among three types of smart space architectural designs concerning the physical deployment of *iO*s on smart nodes, i.e., centralized, decentralized, and distributed smart spaces. Table 2.2 gives a summary of the smart node types used in smart space architectural designs. We added a communication component to each smart node that describes its networking capability based on its capacity for communication. For example, the communication component of a smart node of type HSN is capable of running complex protocols, therefore, able to establish standard network protocols (e.g. the full IP stack). On the other hand, the communication protocol of a smart node of LSN type is not capable of running complex protocols but can run some standards for LSNs such as Bluetooth [2.15] and Zigbee [2.16]. Therefore, a node of the LSN type always depends on a gateway smart node (GSN) for its communication with HSNs in a smart space, as well as for creating and interpreting semantics that describe a context. Thus, a GSN implements the communication standards of all networks that it bridges together and it is responsible for context interpretation for LSNs [2.17] using a gateway *iO* (*GWiO*). A *GWiO* is a specific type of *iO* on a GSN that translates contexts into semantics and vice versa between an LSN and a GSN. In the following three types of smart space architecture designs, we establish the network using Transport Control Protocol (TCP) over IP and Smart Space Access Protocol (SSAP) (a solution provided by the SOFIA project) is used as message transmission protocol for all HSNs. In the case of LSNs, the communication technology must be suitable for low-resource smart nodes, e.g., consider Zigbee and Bluetooth standards. It is the responsibility of the gateway node to establish communication between HSNs and LSNs, by means of translating SSAP messages to messages that are suitable for LSNs and back. Note that we will discuss the SSAP protocol in more detail in Section 4.2.2.

In a centralized smart space, most *iO*s do little or no computation (*perhaps some pre-processing or post-processing*), and they merely realize the tasks of input, output, sensing, and actuation. Most of the computation is performed at a *central iO* (*CTiO*), which is deployed on a special high capacity node referred to as central smart node (CSN) from here on. In practice, the *SBiO* is also placed on the same CSN as shown in Fig. 2.5. A *CTiO* must be able to interpret the contexts of other *iO*s that depend on it and convert them into the semantics of the smart space, which are then stored in the *SBiO*. Adequate application behavior is then imposed by the *CTiO*. In other words, a *CTiO* corresponds to an

Table 2.2. Types of smart nodes.

Item	Description
LSN	**Low-capacity smart node**: a low-capacity node in terms of resources and computation, such as a sensor.
HSN	**High-capacity smart node**: a high-capacity node in terms of resources and computation, such as a smart phone.
PSN	**Passive smart node**: a resource-poor passive node, such as an RFID tag which may require an external power source.
CSN	**Central smart node**: a smart node running all *iO*s of a single smart space application.
PXSN	**Proxy smart node**: a node that performs the simple task of transferring services from one *iO* to another.
SBSN	**Semantics broker smart node**: a smart node that stores and shares semantics using an *SBiO* in a smart spacee.
GSN	**Gateway smart node**: a smart node that translates contexts into semantics and vice versa with the help of a *GWiO*. It also provides a capability of communication between LSNs and HSNs, basically a gateway for LSNs to connect with HSNs.
MSN	**Manager smart node**: a node that does management tasks with the help of one or more *iO*s (i.e. *MiO*, *AMiO*, *RMiO*, *SPMiO*), based on the services' requirements in a smart space.

application in a centralized smart space and it is where the corresponding application logic lives. A smart node can host one or more *CTiO*s and each *CTiO* realizes the corresponding application behavior by communicating with other smart nodes.

Definition (Centralized smart space): A centralized smart space refers to a smart space where a single node, has control and authority over all resources, decisions, and functionalities within the smart space. This entity is responsible for making decisions, processing information, and managing the overall operation of the smart space.

The *iO*s at PSNs do not perform pre-processing. Instead, they are connected to a *CTiO* at the CSN over a trusted proxy (PXSN), where pre-processing refers to the actions or operations carried out by a proxy node before data is transmitted to other nodes, respectively. For example, a message needs to be translated by the PXSN for it to be understandable to the PSNs. In practice, the tasks of the proxy can be moved to the CSN. Therefore, the CSN and the associated *CTiO* are involved in almost all communications and computations, which creates a performance bottleneck in the architecture.

The dependency on the *CTiO* is a concern, as deploying an *iO* at the CSN requires either reservation of the needed resources and admission control or compatibility of the *iO* with a dynamic (best effort) resource management regime. In the latter case adequacy depends on the availability of resources and may be jeopardized with the increasing number of *iO*s that are deployed. In other words, resource management becomes easier as the entire set of resources are allocated at the CSN for an application in a network. This means that it is easy to manage the resources in case the number of *iO*s increases in a smart space.

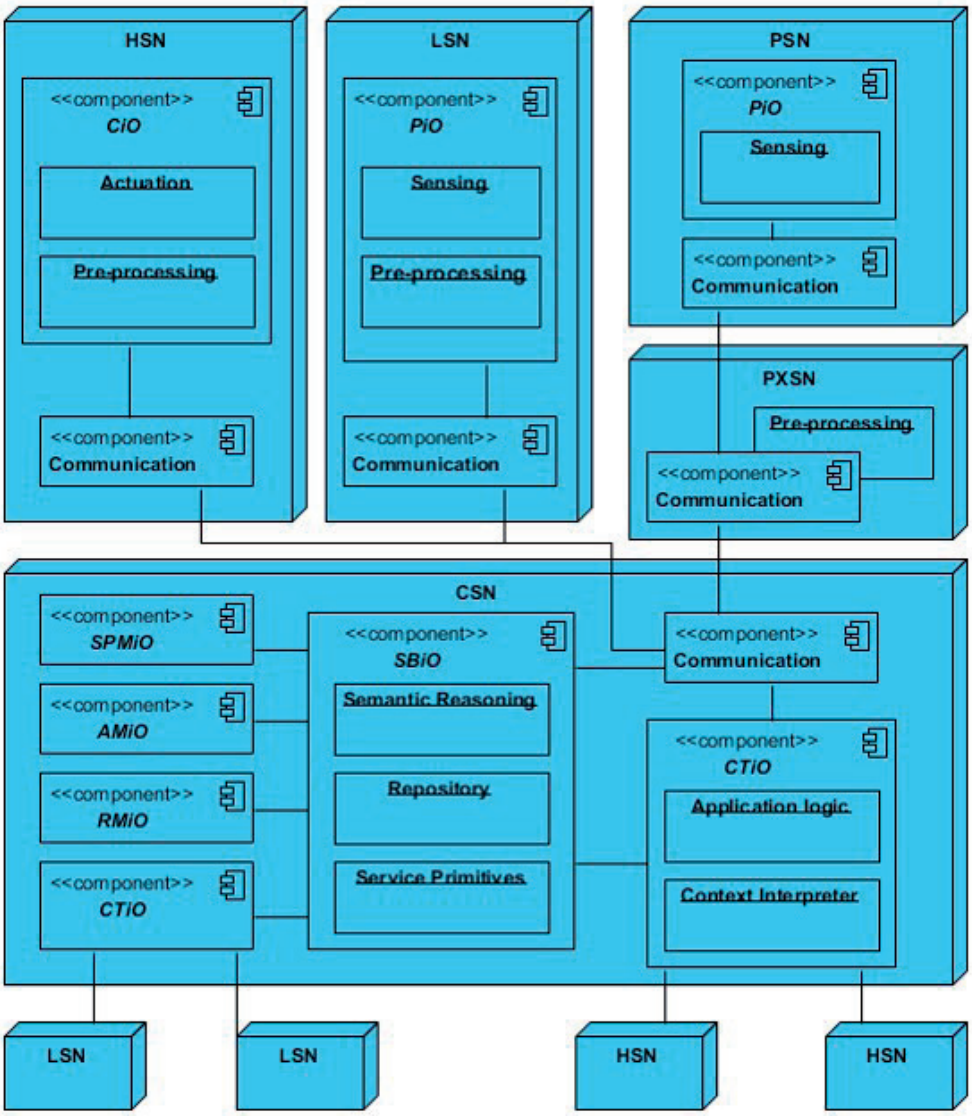


Figure 2.5. Architectural design (deployment view) of a centralized smart space.

Finally, a centralized smart space provides easier installations and updates of an application logic, updating the application logic requires updating it at the CSN instead of at many smart nodes. On the other hand, if the application logic is placed at the CSN, then the smart nodes must access the application logic from the central server each time it runs. As a result, it consumes more bandwidth in a network, and latency can also potentially be an issue. Delays in processing or transmitting data within the centralized smart space can lead to slower response times and may affect the overall performance. This can be par-

ticularly critical in applications where real-time or near-real-time processing is required. Therefore, a decentralized smart space represents a prospective substitute.

A decentralized smart space architecture is one in which the networked smart nodes do most of the computations necessary to realize user applications in a distributed way as shown in Fig. 2.6.

Definition (Decentralized smart space): A decentralized smart space is a smart space where multiple nodes work together without a central controlling node. Instead of relying on a single point of control, decision-making, or resource allocation, a decentralized smart space distributes these functions across various nodes. However, the decentralized smart space does not have a single point of control; instead, it follows a hierarchy that pertains to the distribution of authority and decision-making power. While there may be nodes with varying levels of authority, the ultimate goal is to avoid a single point of control. Decisions are made at various levels, but no single node has absolute control over the entire smart space. Instead, each node in a decentralized smart space possesses a degree of autonomy and can communicate directly with other nodes.

To realize this, it is important that *iO*s utilize shared communication standards and have access to enough computational resources. Here the network scope can vary across smart space implementations, e.g., nodes that have remote connections over the Internet, or link local connections within an IP domain, or VPN connections. The *SBiO* has service primitives and semantic reasoning to enable information sharing. Service primitives refer to a set of basic operations or functions provided by a communication service or protocol to enable communication between different *iO*s in a decentralized smart space. Semantic reasoning refers to the process of using formal logic and knowledge about the meaning of information (semantics) to draw conclusions or make inferences about information represented in a system. As a result, the functionality in a decentralized smart space (application) is much more scattered. The main differences from a centralized smart space are as follows.

i) Unlike a centralized smart space, in a decentralized smart space the total computational capacity increases with the number of smart nodes that take part. As a result, the computation bottleneck that exists in a centralized smart space due to running applications on a CSN is effectively eliminated. With careful application design, this may improve the computational performance significantly for smart applications, especially for those that deal with large cumulative streams of data.
ii) A decentralized smart space eliminates the communication bottleneck that exists in its centralized counterpart as well. By pushing computations towards locations where the data are generated, the performance dimensions related to latency (response times,

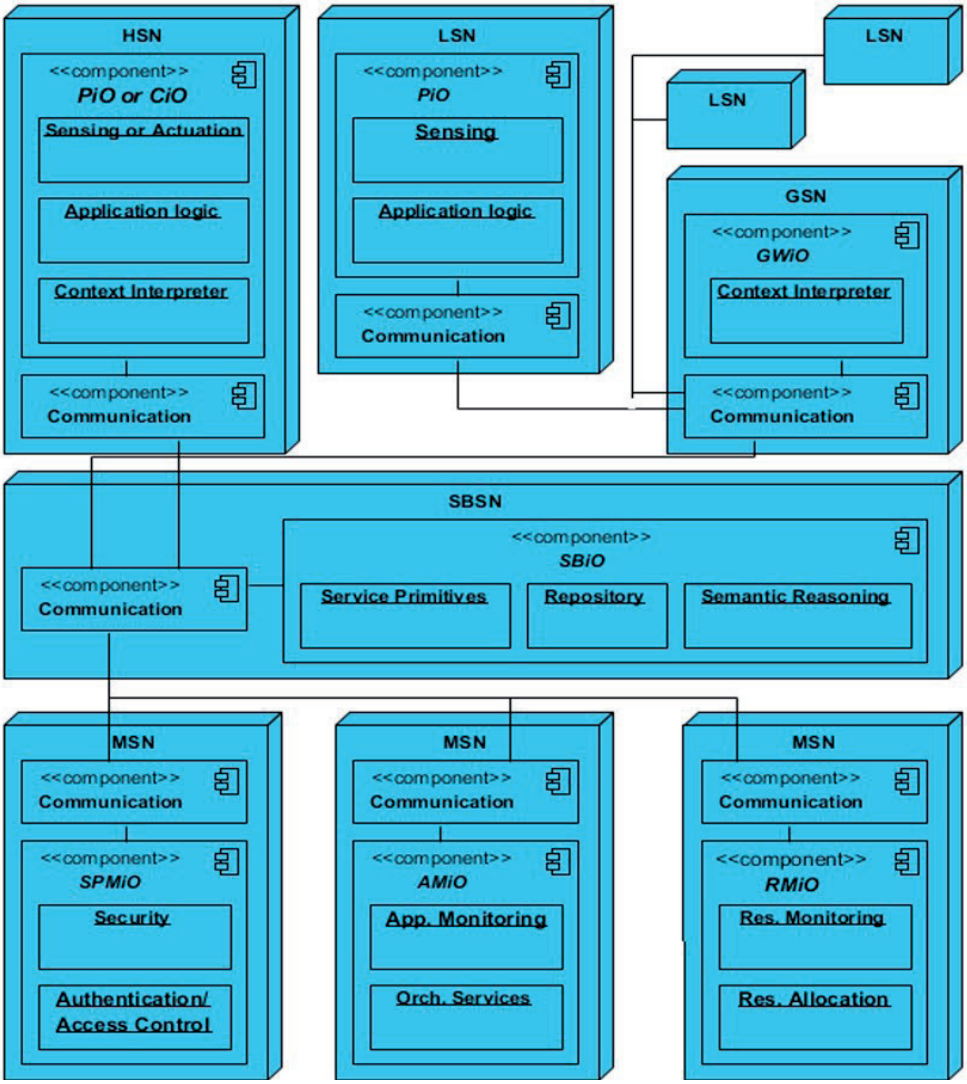


Figure 2.6. Architectural design of a decentralized smart space.

latency jitter) can improve dramatically. This improvement can mainly be attributed to the resulting reduction in the need for communication.

iii) A disadvantage of the decentralized architecture is the increased overall complexity of management. Resource management becomes more difficult as the entire set of resources that are allocated to an application are now decentralized in a network. Application management becomes much more difficult as well, for example, updating the application logic now requires coordinating the update across many smart nodes

where the application logic lives. Security and privacy are bigger concerns as it is more difficult to monitor and control access to individual smart nodes.

iv) Placing the application logic on smart nodes means that smart nodes and *iO*s are partially independent and thus can make decisions locally and perform computations accordingly. They do not always depend on a central unit to make application-level decisions. This may reduce communication regarding the application logic through the network, which is an advantage for bandwidth efficiency in comparison to a centralized smart space.

Services supported by the cloud are becoming increasingly popular and can be utilized in decentralized smart spaces directly or indirectly. The cloud services are available via a remote cloud computing server rather than an on-site server. In direct utilization [2.18], the cloud services must have the same ontology format and semantics as the smart space itself and each cloud service acts as an *iO*. Using cloud services will shorten the time from designing an architecture to its deployment. Because cloud computing provides the facilities of on-demand services to a shared podium of configurable computing resources (e.g. server, applications, services, storage, and networks) that can be quickly accessed with minimum administrative efforts. Furthermore, the smart nodes that communicate directly with the cloud services must be individually reachable (e.g., IP-to-the-leaves using 6LoWPAN [2.19]). A cloud service architecture can be based on models such as Software as a Service (SaaS), Data as a Service (DaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). In this thesis, we emphasize a semantic approach, where an *iO* can be used as indirect utilization of the cloud services. In indirect utilization [2.20], a cloud gateway between the cloud and smart space runs an *iO* that provides the semantic interface to cloud services, as shown in Fig. 2.7. For example, an SBSN can further connect to a cloud gateway for availing cloud services, where the cloud gateway can have a semantic interface to those services. Suppose, a smart node in a smart space requires services from a cloud server. Firstly, it needs to make a service request at the SBSN. Consequently, the SBSN is further connected to the cloud gateway, where the cloud gateway translates the request (which is translated in semantics) into a specific message that is suitable for the cloud platform and connects to the cloud server. The cloud server returns the query result to the cloud gateway and further communicates to the SBSN. As a result, the smart nodes connected to the SBSN can access the services provided by the cloud server.

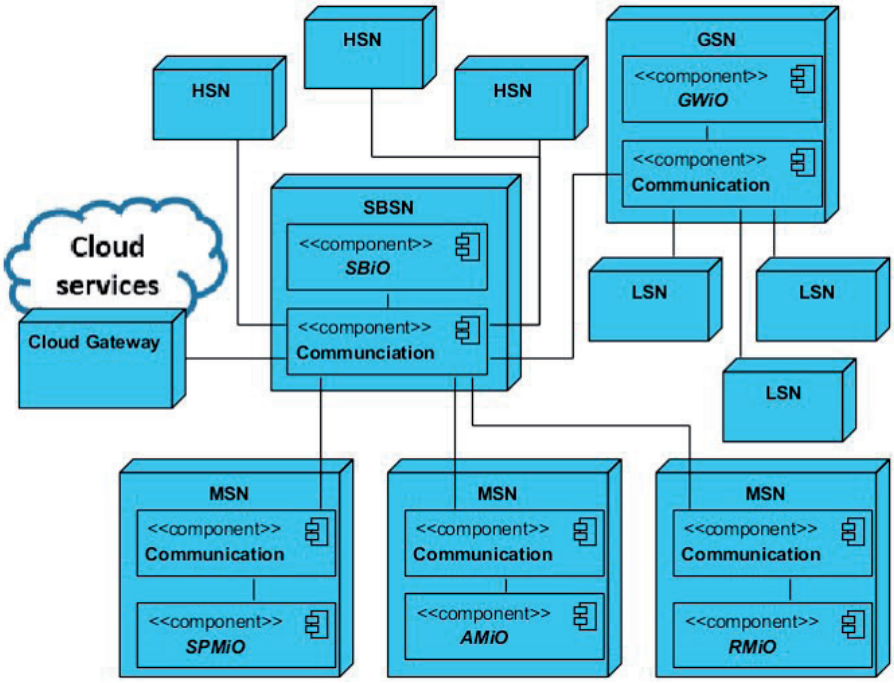


Figure 2.7. Cloud services for smart spaces.

A distributed smart space consists of smart nodes without a certain hierarchical structure. Each node contains the software components necessary to realize self-management, and distributed application logic as shown in Fig. 2.8.

Definition (Distributed smart space): A distributed smart space is a smart space in which nodes are interconnected and collaborate to achieve a common goal. In a distributed smart space, multiple autonomous nodes, often geographically dispersed, work together to perform tasks, share resources, and provide services. These nodes communicate and coordinate their activities through a network, enabling them to operate as a single, cohesive system. In a distributed smart space, unlike in a decentralized smart space, there may not be a clear hierarchy of authority or control. Instead, nodes often interact on a more equal basis, with each having a degree of autonomy and making decisions based on local information. This absence of a strict hierarchy can lead to a more collaborative and dynamic network, where nodes work together based on their capabilities and the information available to them.

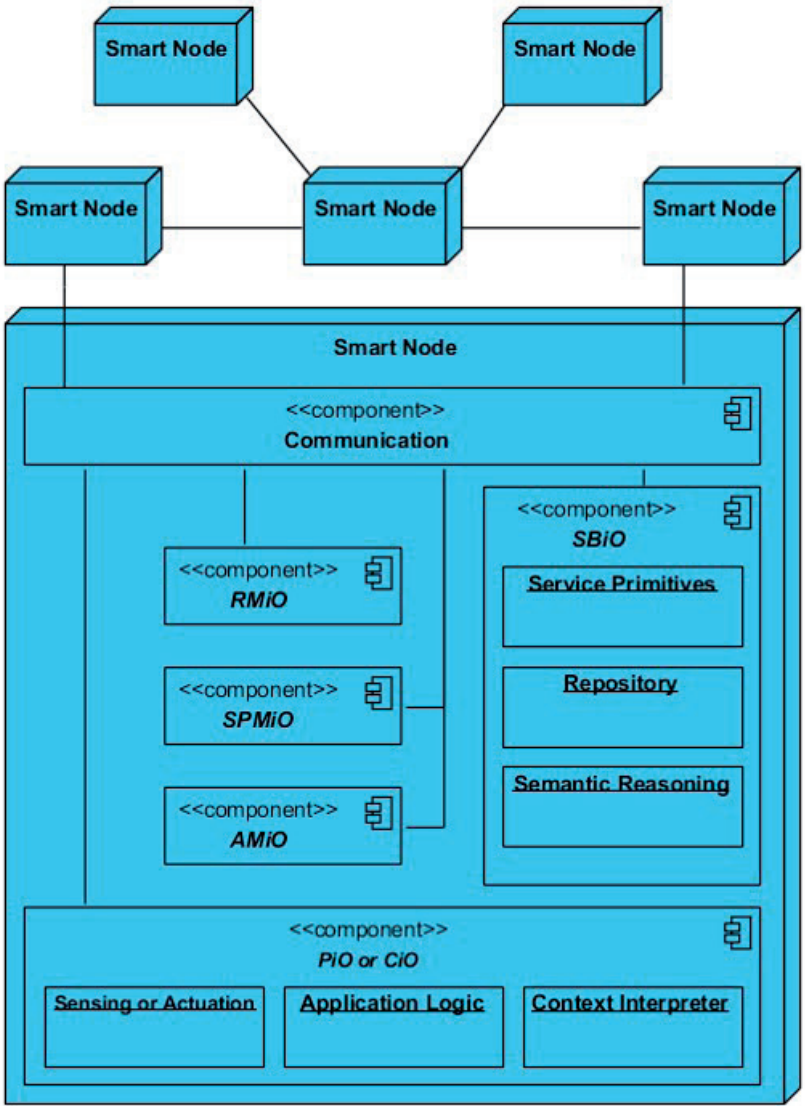


Figure 2.8. Architectural design of a distributed smart space.

This architectural design is strongly dependent on the application's requirements and is loosely coupled with *iO*s. Smart nodes handle the application logic individually by communicating and cooperating over a network. The distributed smart space architecture has the following advantages and disadvantages:

- The smart nodes in the distributed architecture are loosely coupled and do not share their memory. They communicate messages over a communication network using a protocol (a middleware) to execute smart applications. The distributed architecture provides a set of concurrent processes and communication channels between smart

nodes. Because of this, we can add an unbounded number of smart nodes based on a network capacity without any effect on the whole architecture. We can also manage the failures, for example, the failure of one smart node does not lead to the failure of the entire distributed smart space. On the other hand, scalability can lead to performance issues due to limitations in network capacity, such as bandwidth and range, when accommodating a large number of nodes within a single network. As an example, message overload can occur if a large number of nodes share information in a network. For this purpose, we need to evaluate the tradeoff between scalability and performance for calculating message overhead in distributed smart spaces.

- If the smart nodes in a distributed system are scattered across different geographical domains with various technologies, it complicates the establishment and management of a coherent security policy. It causes a direct risk of exposure of confidential information in the uncontrolled, unprotected use of communication networks between smart nodes for information exchange. Therefore, all smart nodes and their network connections are required to be secured in the large scale of distributed smart spaces. Sometimes the exposure of information is required by an administrator of the system, for example, to preserve the administration management and usage history of the system for maintenance of distributed smart spaces. Finally, if a secure distributed smart space is required, encryption and access controls are essential.

Overall, a suitable architectural design must be chosen based on the target applications, their goals, and the corresponding performance metrics. We discuss an example for selecting an architectural design in Section 3.1 of Chapter 3.

2.4 Comparative Analysis of Smart Spaces

Many architectural designs for smart spaces have been proposed by researchers. In this section, we compare a selection of them, specifically those that provide (at least nearly) complete solutions with respect to the smart space concepts that we have introduced in Section 2.2 and Section 2.3. The analysis takes as the basis the information on the deployment of *iO*s (or rather counterparts of *iO* types that we define) on smart nodes, and their contributions to smart space architectural designs as given in Table 2.3. We have drawn three major observations.

i.) Observation on architectural designs: Most of the smart space architectures in the literature are decentralized by design, while fully centralized or fully distributed architectures are very rare. We will explain a few examples in the following.

An example of distributed smart space architecture is called PERSIST [2.23], which provides the overall design of a personal smart space (PSS). PSSs are smart spaces based on personal area networks that follow the user as she moves. PSSs consist of various smart nodes such as personal computers, mobile devices, wearable sensors, or other wearable devices. They ensure a minimum level of basic pervasiveness and context-awareness facilities anytime and anywhere. To provide connectivity to PSS owners, PSSs can operate in both infrastructure and ad-hoc network modes, allowing wide integration of a multitude of smart nodes. PSSs can interoperate with other smart spaces for exchanging information. They allow smart nodes of different smart spaces to adapt to new environments automatically in satisfying users' needs. Software modules in a PSS can be mapped to our *iO* concepts introduced in this thesis work, where all contexts are managed at the *AMiO* that stores and retrieves context information in a distributed manner. The facilities it provides to integrate multiple applications, however, are limited.

A centralized smart space design is given in [2.22]. In this design all contexts from the smart space and its physical environment are collected at a central unit. The received contexts are processed by a central reasoning module to provide outputs to the *iO*s, which also reside in the same central unit. The main focus of [2.22] is on the use of an ontology graph to enable a reasoning component for various scenarios. For this purpose, a user can select the services by querying the central unit and can make subscriptions for service updates. Furthermore, there is no opportunity to integrate LSNs into the design, i.e., no architectural description of a gateway solution incorporated into the design.

Most of the decentralized architectural designs from [2.24] to [2.37] in Table 2.3 employ *AMiO*s and *SBiO*s, while *RMiO*s and *SPMiO*s are mostly not considered in these designs. In the following, we elaborate on two examples, SPITFIRE [2.27] and CISE [2.21]. There are two issues that should be noted in the decentralized architecture solution of SPITFIRE. Firstly, it involves a semi-automatic process of creating semantic sensor descriptions for LSNs. The semi-automatic process is developed for annotating newly deployed sensors by considering the hypothesis that sensors with similar semantic descriptions would produce similar output. For example, two light sensors should produce similar time series if they are deployed in the same room. Consider one light sensor (first sensor) is already annotated with its room number, and later, another light sensor (second sensor) is deployed in the same room. If the illumination output of the second sensor is strongly correlated with the first sensor over some time, then it is concluded that both sensors are located in the same room. And the semantic descriptions of the first sensor are copied to the second sensor. If both sensors do not find any strong correlation, then the user must provide the semantic descriptions manually to the second sensor. Note that our proposed architectural design for decentralized smart spaces allows the semantics to be fetched directly and automatically if it is available through the *GWiO*, without requiring manual or

Table 2.3. Comparison of architectural designs of smart spaces based on *iO*s deployment on smart nodes.

Article	**Year**	**Architectural Style**	*PiO/CiO*	*GWiO*	*SBiO*	*AMiO*	*RMiO*	*SPMiO*
CISE (Context-Based Infrastructure for Smart Environments) [2.21]	1999	Decentralized	Yes	Yes	A context-based local server is used and reasoning is performed based on the specified inference rules.	The focus is LSN management at the application-level (*iO*s).	-----	-----
Goh et al. [2.22]	2007	Centralized	Yes	No	Reasoning is performed to maximize the use of contexts and minimize their conflicts.	Context is received from *iO*s and the application contexts are managed at a centralized manager.	-----	-----
PERSIST (PERsonal Self-Improving SmarT spaces) [2.23]	2009	Distributed	Yes	Yes	Domain-specific inference rules and predictions are made to interact with PSSs.	Applications are distributed and managed in several PSSs individually.	Manages *iO* resources for services residing in each PSS.	Applies privacy control mechanisms at each PSS (as in social network communities).
GUPSS (A Gateway-Based Ubiquitous Platform for Smart spaces) [2.24]	2009	Decentralized	Yes	Yes	Reasoning is performed on a standard web server.	Application contexts are managed either on a *GWiO* or an *iO* of the application.	-----	-----
(van der) Vlist et al. [2.25]	2009	Decentralized	Yes	No	Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed.	Contexts are managed at *iO*s first and are further interpreted for application logic at an *SBiO.*	-----	-----
Song et al. [2.26]	2010	Decentralized	Yes	No	Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed.	Application contexts are managed at *iO*s and further interpreted at a web server.	Service discovery protocols are used to manage resources and services.	-----

Table 2.3. Continued

Article	**Year**	**Architectural Style**	*PiO/CiO*	*GWiO*	*SBiO*	*AMiO*	*RMiO*	*SPMiO*
SPITFIRE [2.27]	2011	Decentralized	Yes	Yes	Knowledge representations into semantics are collected and stored at IoT-based server, where semantic reasoning is performed.	A query-based management system is in an *SBiO*, which manages semi-automatic creation of semantics and provides manual integration of sensor semantics.	-----	-----
INSTANS [2.28]	2012	Decentralized	Yes	No	Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed.	Contexts are managed at *iO*s and further interpreted at a web server; event-based processing to manage events is proposed.	-----	-----
Morandi et al. [2.29]	2012	Decentralized	Yes	No	Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed.	Application contexts are managed at *iO*s and further interpreted at a web server. Additionally, a subscription mechanism is added to improve query results.	-----	-----
Natalia et al. [2.30]	2013	Decentralized	Yes	No	Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed. An additional component integrates fuzzy reasoning and learning algorithms.	Application contexts are managed at *iO*s and further interpreted at a web server.	-----	-----

Table 2.3. Continued

Article	Year	Architectural Style	*PiO/CiO*	*GWiO*	*SBiO*	*AMiO*	*RMiO*	*SPMiO*
Eila et al. [2.31]	2014	Decentralized	Yes	No	Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed.	Application contexts are managed at *iO*s and further interpreted at a web server. A framework for managing runtime quality attributes via adaptation is proposed.	-----	Ontology for information security is used for security.
Jussi et al. [2.32]	2014	Decentralized	Yes	No	IoT based products are used. Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed.	Application contexts are managed at *iO*s and further interpreted at a web server.	-----	-----
Zeng et al. [2.33]	2015	Decentralized	Yes	No	A platform library server is introduced for data flow in a smart space.	-----	-----	----
Sergey et al. [2.34-2.35]	2017	Decentralized	Yes	No	Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed.	Application contexts are managed at *iO*s and further interpreted at a web server.	-----	-----
Andrey [2.36]	2017	Decentralized	Yes	No	Knowledge representations into semantics are produced and stored in an ontology language at an *SBiO*, where semantic reasoning is performed.	Application contexts are managed at *iO*s and further interpreted at a web server. Extra functionality is added to manage subscriptions at runtime for scalability improvement.	-----	-----

Table 2.3. Continued

Article	Year	Architectural Style	*PiO/CiO*	*GWiO*	*SBiO*	*AMiO*	*RMiO*	*SPMiO*
Shabir [2.37]	2018	Decentralized	Yes	No	A cloud server is employed for storing semantics and for semantic reasoning.	A virtual device manager is introduced to manage services in cloud-based web applications	Manages resources and services in cloud.	-----
William et al. [2.45]	2019	Centralized	Yes	No	Data acquisition through IoT and stored at the central entity and derived semantics by the big data analysis	Application contexts are managed at *iO*s and further interpreted at central entity.	-----	-----
Zeynep et al. [2.46]	2020	Centralized	Yes	No	A central entity, where all nodes are plug in and receives information for further analysis to conclude semantics.	Application contexts are managed at *iO*s and further interpreted at central entity.	-----	-----
Tekler et al. [2.47]	2022	Centralized	Yes	No	Information received at the central entity and semantics are derived using a deep learning algorithm	Application contexts are managed at *iO*s	-----	-----
Badr et al. [2.48]	2023	Decentralized	Yes	No	IoT based platform for sharing information	Application contexts are managed at *iO*s	-----	IoT Standard protocols for security and access control

semi-automatic creation of semantics. This means that the sensors will update contexts to the *GWiO* and the *GWiO* is self-managed to convert contexts into semantics. Secondly, SPITFIRE directly connects sensors to the semantic web, and the sensor data are updated frequently, which causes heavy network traffic and leads to performance issues. In our decentralized architectural design, this is easily solvable by locally processing the sensor data and sharing only the high-level semantics in the smart space over the *GWiO*.

A similar solution is provided in the CISE architecture, which focuses on the requirements for dealing with smart space contexts. CISE provides an architectural solution with the following components: contexts, a context server (*which is responsible for context aggregation*), a context interpreter (*which is responsible for context interpretation*), and a communication module for information sharing. CISE hides the context details of LSNs (such as sensors), and can be replaced or added without affecting the smart application if the new sensor is capable of performing the same services provided by the removed sensor. It also facilitates the addition of contexts to existing applications. Although this solution provides many facilities for building smart spaces, it does not support high-level interoperability (i.e. semantic interoperability) for generic infrastructures suitable for any application in smart spaces. The information shared only includes the basic descriptions of nodes. A common ontology language based on a semantic web is required for large-scale applications.

ii.) Observation on the deployment of smart nodes: The smart nodes considered in various smart space designs are summarized in Table 2.4, which shows that HSNs are used dominantly in most smart space designs. Improved gateway approaches are necessary to accommodate large numbers of LSNs in smart spaces, which is one of the main issues to be considered in this thesis (initial approaches available in [2.30] and [2.31]).
iii.) Observation on the deployment of *iO*s: It can also be seen from the comparison of various smart space architectures in the literature that *RMiO* and *SPMiO* are rarely ever considered. This is a huge problem for mainstream adoption of smart spaces, considering ethical aspects and gradually introduced laws enforcing data privacy in many countries. Dependability of smart space applications is also a main concern for user experience, e.g., changing an electric circuit (manual) light switch that is almost 100 percent reliable with a smart switch that works 99 percent of the time is not acceptable. For this, it is essential to integrate resource and service management mechanisms into smart spaces, which is also an area for improvement.

In addition to Table 2.3, some researchers such as Zeng et al.[2.38] explained the smart space concept based on a flow model. The flow model is a coarse-grained computational model called HyperspaceFlow, where a smart space is differently modeled than our approach using physical flow, data flow, and human flow components. The physical flow

Table 2.4. Smart nodes considered in various smart space designs.

Article	LSN	HSN	SBSN	GSN	MSN
CISE [2.21]	-	+	+	-	-
Goh et al. [2.22]	-	+	-	-	-
PERSIST [2.23]	+	+	-	-	-
GUPSS [2.24]	+	+	+	+	-
(van der) Vlist et al. [2.25]	-	+	+	-	-
Song et al. [2.26]	+	+	+	-	+
SPITFIRE [2.27]	+	+	+	-	+
INSTANS [2.28]	-	+	+	-	-
Morandi et al. [2.29]	-	+	+	-	-
Natalia et al. [2.30]	-	+	+	-	+
Eila et al. [2.31]	-	+	+	-	+
Jussi et al. [2.32]	+	+	+	+	-
Zeng et al. [2.33]	+	+	+	-	+
Sergey et al. [2.34-2.35]	+	+	+	+	-
Andrey [2.36]	+	+	+	-	-
Shabir [2.37]	+	+	+	-	+
William et al. [2.45]	-	+	+	-	-
Zeynep et al. [2.46]	-	+	+	-	-
Tekler et al. [2.47]	-	+	+	-	-
Badr et al. [2.48]	-	+	+	-	+

*Considered (+) and Not Considered (-)

specifies the relations between the cyberspace and the physical space. The data flow involves computations and communication related to the cyberspace. The human flow is utilized to model the interaction between the cyberspace and the human space. In addition, a system-level smart space design method using the HyperspaceFlow model was proposed. The feasibility and the effectiveness of this method were examined in a healthcare case study, which indicated that the specifications of a smart space can be further transformed into the underlying architecture by employing the HyperspaceFlow model. This proposed designed approach offers relatively user preference of a design solution. However, it allows modeling of a smart space only for a single user. In addition, heterogeneous wireless networks are not considered, which would enable tailoring for more complicated communication scenarios when designing smart spaces.

Some researchers have tried to explain smart space architectures using hierarchical models as in MavHome [2.39]. The smart space architecture in MavHome is realized by providing a complete solution to a smart home. The architecture proposed MavHome agents that are similar to our *iO*s. These agents are the software components deployed on

nodes and placed in four cooperating layers: a decision layer, an information layer, a communication layer and a physical layer. The decision layer selects the actions for an agent (an *iO*) to execute based on the information supplied by the other layers. The information layer gathers, stores, and generates knowledge useful for decision making. The communication layer facilitates the communication of information, requests, and queries among agents. The physical layer corresponds to the agents within the smart home. These layers provide the features necessary for self-managing smart home automation. The functional process of these layers is a bottom-up approach. For example, a sensor monitors the environment (e.g. light information) in the physical layer and transmits the information to another agent through the communication layer. Further, the information layer updates this information to maintain records and the predictions or decisions by the decision layer. This approach is reversed, i.e., Top-down when the actions are executed. The decision layer selects an action (turn on the lights) and relays the action to the information layer. After updating the information layer, the communication layer routes the action to an appropriate agent at the physical layer. Moreover, MavHome was implemented using a common object request broker architecture [2.40] interface between software components and powerline technologies such as X10 [2.341] and Lon Works [2.42] as electric devices. Although this architecture enables the integration of several technologies in a smart home, it fails to provide a solution for LSNs, efficient interoperability and adequate responses. It addresses only context-based interoperability and avoids any management of resources and services.

Finally, we discuss a similar project, i.e., ISHEWS [2.43] that explains application development in a smart space based on the concepts of interoperability. In ISHEWS, a smart home environment is introduced with five main sub-systems: surveillance and access control, home automation systems, digital entertainment systems, assistive computing systems, and an energy management system. These sub-systems interoperate in three levels: basic connectivity interoperability, network interoperability, and syntactic interoperability. The basic connectivity interoperability provides a common standard for data exchange between two sub-systems and establishes communication links. It represents the physical and data-link layers of the standard Open Systems Interconnection (OSI) model. Ethernet, the IEEE 802.11 family of Wi-Fi protocols, and Point-to-Point Protocol (PPP) are the common standards for basic connectivity interoperability. Network interoperability enables message exchange between systems across a variety of networks in a smart home environment. It is represented by the network, transport, session, and application layers of the OSI model. TCP, UDP, File Transfer Protocol (FTP), Address Resolution Protocol (ARP), and IP/IPv6 are the common standards for network interoperability. Syntactic interoperability refers to the agreement of rules that manage the format and structure for encoding information exchange among the sub-systems. Simple Object Access Protocol (SOAP) encoding, Representational State Transfer (REST) [2.44] encoding, and message exchange

patterns such as asynchronous publish/subscribe patterns are the common standards for syntactic interoperability. These interoperability solutions increase the complexity of integrating all three levels in a single smart space. Our proposed architectural design alternatives avoid this complexity and provide a solution for integration at the semantic level. Therefore, we can conclude that there is a demand for a semantic interoperability architecture for the following main reasons.

- Data Integration: In complex smart spaces, data often originates from various sources, including different databases, software applications, and devices. Semantic interoperability allows these diverse data sources to communicate and integrate seamlessly, enabling a more holistic view of information.
- Meaningful Data Exchange: Semantic interoperability ensures that data exchanged within a smart space is not only syntactically compatible (i.e., the format matches) but also semantically compatible (i.e., the meaning is understood). This ensures that data is interpreted correctly and consistently.
- Cross-Platform Compatibility: In a world with numerous hardware and software platforms, semantic interoperability allows information to be shared and understood across different systems and technologies. This is essential for data sharing and collaboration.

We will introduce a semantic interoperability architecture in Chapter 3 and discuss it in detail in Chapter 4.

2.5 Conclusions

We formally defined fundamental smart space properties, the components that make up a smart space, and the smart space concept. The roles of information objects and smart nodes are discerned in the development of smart space designs. Moreover, the smart space architectural designs presented in this chapter generalize over a broad range of smart space implementations in the literature. Based on the comparison with the state-of-the-art, we conclude that most of the smart space designs are decentralized and used mostly high capacity smart nodes. This analysis concludes that there is a demand for a semantic interoperability architecture, which can also integrate low capacity smart nodes efficiently for sharing information in smart spaces.

Chapter 3

Smart Space Properties and Semantic Interoperability Architecture

When studying smart space architectural designs for smart applications, starting with a generic template smart space architecture with semantic interoperability support can significantly reduce the design effort thanks to its reusability. Such architecture should facilitate the fundamental properties of a smart space, which we first need to define. We can quickly build candidate architectures as variants of the semantic interoperability architecture, tuned towards certain performance metrics relevant for the application at hand. To accomplish this, we require a measurable objective and a systematic way of choosing amongst candidate smart space architectures based on this objective. In this chapter, we achieve this by formulating a multi-objective optimization problem for selecting the best smart space architecture. Additionally, we present an inventory of the fundamental properties of a smart space. Finally, we propose a semantic interoperability architectural design for smart spaces.

3.1 Example Selection of Smart Space Architecture

The selection of the architectural design is dependent on the metrics considered and evaluated with respect to the smart space applications under consideration. For many smart applications timing is very important due to their interactive nature. For example, for an intelligent lighting application that controls lights based on users' presence, it is an annoyance factor when a user walks into a dark room and the lights are not turned on in her presence or their actuation is delayed, i.e. the quality of experience (measurable, e.g., by checking adherence to a maximum response time constraint) drops significantly. In general, the selection of the best smart space architecture for a set of applications can be formulated as a multi-objective optimization problem (MOP) where each objective represents a performance metric to be satisfied. The smart space architecture that gives the best optimization trade-off, i.e., an average over all possible contexts (or a weighted average if contexts vary in frequency or importance), is the winner and should be selected for the application(s) under consideration. For instance, an application's average performance in some objective can be improved by adapting *iO*s' behaviors appropriately to

changing contexts of the application. Candidate architectures will vary in how well they facilitate such adaptation while preserving performance in other objectives.

Consider the selection of the best smart space architecture for a ChairLight application in an office room with the infrastructure shown in Fig. 3.1. The users (η) with identities 1, …, η have regular team meetings in this room and a given meeting may be chaired by any one of the η users. The application goal is to provide the preferred light settings of the user who is chairing a meeting. In the following, we examine this example in more detail and illustrate how the ChairLight application's performance changes across different smart space architectures.

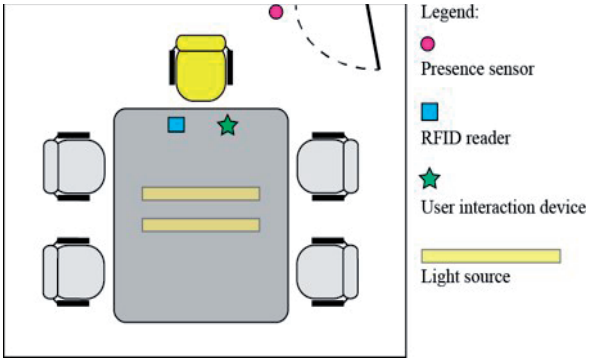


Figure 3.1. Bird's eye view of a smart lighting infrastructure of a meeting room in an office building. There is a table in the center of the room, and five seats are placed around it. Near the seat of the meeting chair, there is an RFID reader to identify the session chair. The users carry RFID tags. Two light sources are on the ceiling. A calibrated presence sensor on the ceiling detects room occupancy, and a user interaction node can be used by the meeting chair to control the lights manually. All sensors, light sources and the user interaction node communicate over a network.

The application utilizes dimmable lamps that emit cool and warm white light (with brightness percentages *L1* and *L2*, and operated by L1_Brightness_*iO* and L2_Brightness_*iO*, respectively) and an RFID reader for identifying the user who is chairing the meeting. We consider two smart nodes, a presence sensor node with Presence_Detecting_*iO* for detecting events (with binary value *PS*) and a wireless user interaction node with Push_Button_*iO* having two states being *Push*=0 *(interaction button is released) and Push*=1 *(interaction button is pressed)*. The RFID reader is placed next to the seat that is designated for the chair of the meeting (from here on referred to as the primary seat) communicates with passive tags carried by the users. Based on the received signal strength, it tries to detect the serial number $f \in \{f_1, \ldots, f_\eta\}$ that is closest in physical distance to the primary seat. The serial number carried by a user may change over time, as the tags are likely to be replaced due to wear or loss. The mapping of serial numbers to actual user identities is maintained on a separate User Profile Server (UPS), together with their preferred lighting

set-points (SP_η for the user η). The users update their profiles and choose their favorite set-points via the web interface of the UPS.

For three users (η = 3), example set points are given in Table 3.1. Assume that the same application can run on two different smart space infrastructures SS_1 and SS_2, where the former has a local UPS, while the latter utilizes a remote cloud server (deployed with Cloud_Services_*iO*) through a gateway (deployed with a Translator_*iO*) for this purpose, as shown in Fig. 3.2 and the physical deployment of associated *iO*s on smart nodes is shown in Fig. 3.3.

Table 3.1. User preferences (set points) in the example with η = 3.

Set-point	***L1***	***L2***
SP_0 (*default*)	Off	Off
SP_1 (*of user* 1)	50%	50%
SP_2 (*of user* 2)	100%	Off
SP_3 (*of user* 3)	Off	100%

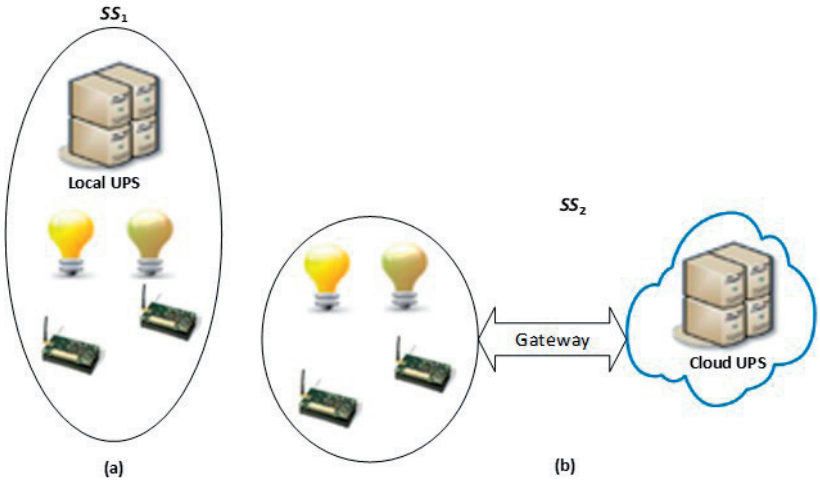
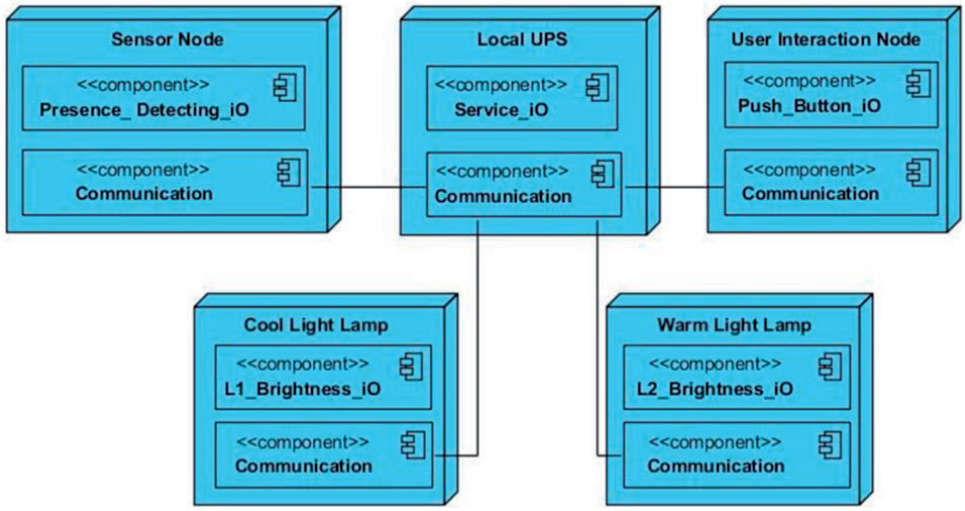
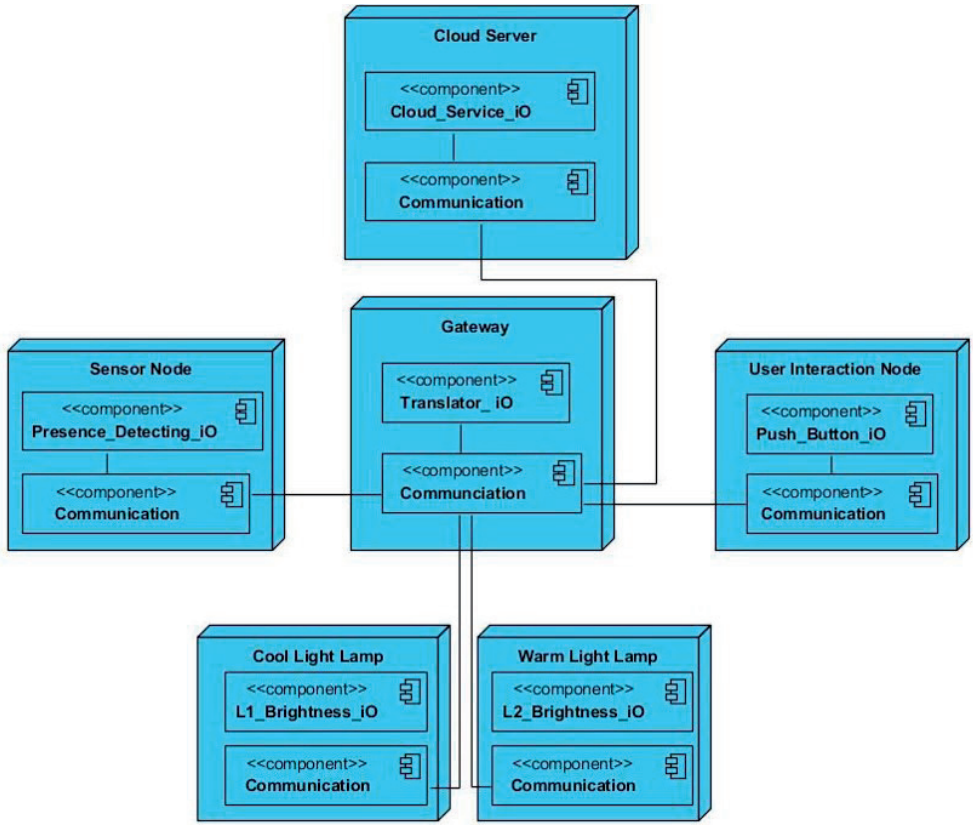


Figure 3.2. Physical configurations of (a) SS_1 and (b) SS_2. In (a), the ellipse shows the smart space nodes in a high capacity network (e.g., an IP network). In (b), the ellipse indicates a low capacity network behind a gateway, or it may indicate an IP sub-network behind an edge router.



(a)



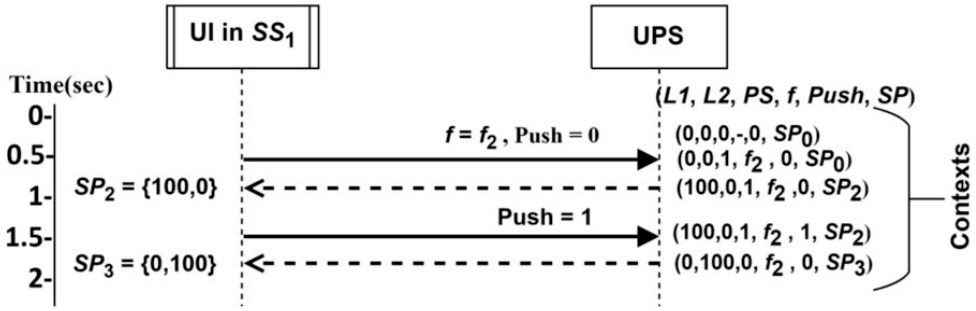
(b)

Figure 3.3. Physical deployments of *iO*s in (a) SS_1 and (b) SS_2.

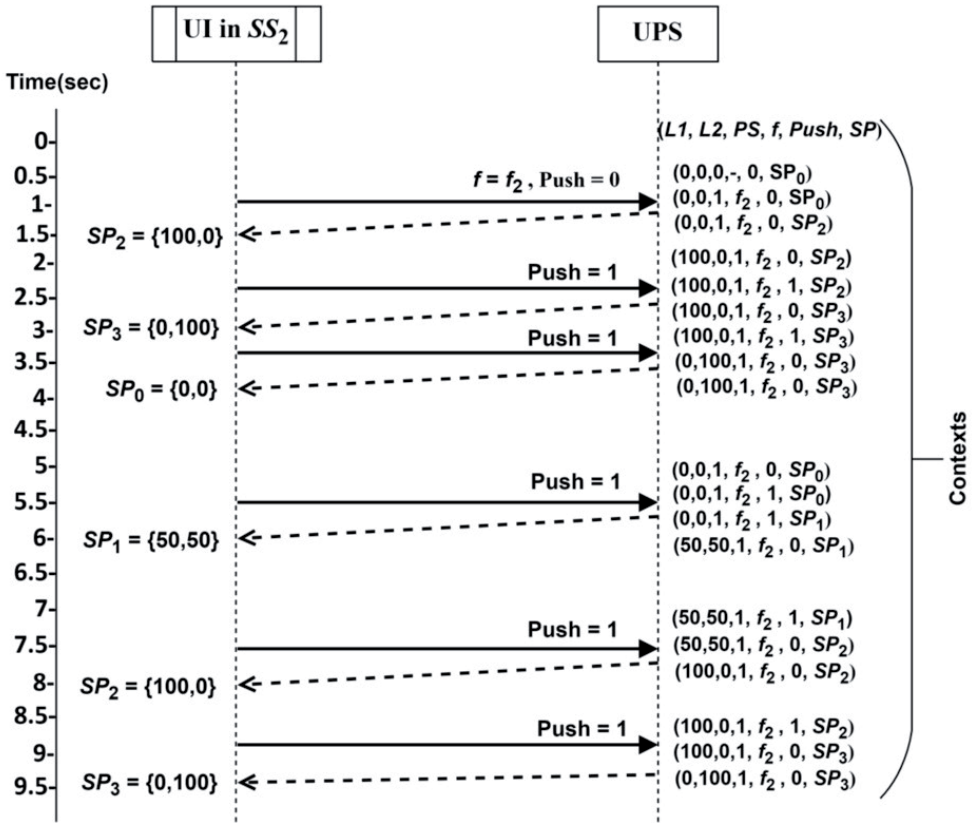
When the current light settings are not in line with the preferences of the user chairing the meeting (e.g. as a result of the wrong user identity being detected), the user has the option to push the user interface button to toggle between set-points. This action communicates to the UPS that the light set point should be switched from SP_i to $SP_{i+1}(modulo(\eta + 1), 1 \leq i \leq \eta)$. Because this service is highly interactive, it is necessary for the user to observe the changes resulting from the interaction nearly immediately after the button is pressed. Otherwise, i.e., if it takes a long time to observe the change, the user tends to push the node again, switching the set-point for a second time before actuation can occur.

There are two cost functions m_1 and m_2 associated with this application. m_1 is the number of user interventions per use, whose upper limit is 1 (application constraint). m is the time it takes until a meeting chair is presented with her preferred lighting set-point, whose upper limit is 1 sec. Referring to the architectures depicted in Fig 3.2, suppose all the lights in the two rooms of the two smart spaces are initially off. User 3 is the meeting chair, she takes her seat at time t = 1 sec, and the RFID tag of user 2 is *incorrectly* detected as the closest tag to the RFID reader in both smart space infrastructures. Figure 3.4 illustrates the changes in contexts in the two infrastructures, with the contents of the messages exchanged and timing relations, i.e., the ChairLight application's behaviors in SS_1 and SS_2. For simplicity of presentation, we assume that the sensing, actuation, communication, and computation within SS_1 and SS_2 are all instantaneous, except for the network propagation delay between the remote cloud server and the other nodes of SS_2, which is 0.5 sec.

In this example scenario, the costs of user intervention per use, m_1 (SS_1, A) and m_1 (SS_2, A), are 1 and 5, respectively, where the smart application A is the ChairLight Application for both smart space infrastructures. The delay costs of reaching the preferred light set-point, m_2 (SS_1, A) and m_2 (SS_2, A) are 0.5 sec and 9 sec, respectively. Hence, the ChairLight application in SS_1 satisfies the application constraints with respect to both performance metrics, and we say that its behavior is adequate in SS_1. Since the upper limits allowed by the application are exceeded in SS_2 for both metrics, the ChairLight application is not adequate in SS_2. It may be argued that adequateness may be achieved in SS_2 if the application maintains a cache of the most recent pairs of RFID serial numbers and light set-points at the RFID reader, which sends actuation commands to the two lamps immediately when there is a cache hit. Such a cache is adaptive by nature. The selection of the meeting chair may not be completely random, e.g., the boss is the chair when he joins a meeting. The application can also make use of such information collected at runtime and learn to improve its performance (e.g., if in doubt prefer the RFID tag that was most frequently the chair's in the past).



(a)



(b)

Figure 3.4. Contexts (*L1*, *L2*, *PS*, *f*, *Push*, *SP*) and behaviors in Smart Space Infrastructures (a) SS_1 and (b) SS_2.

In general, consider an application A with *V* objectives (performance metrics) m_v to be optimized subject to *L* constraints ∂_l (*e.g., speed, accuracy, response time, and resource utilization*). m_v may also be conflicting, in which case a tradeoff must be considered in choosing the best smart space architecture *SS** among a set of λ candidate architectures $\{SS_1, SS_2, \dots, SS_\lambda\}$. A comparison of multiple smart space architectures with respect to the performance of A can be formally represented as a MOP. Without loss of generality, we can assume that all the objectives are cost functions to be minimized as given in the following formalization.

$$\begin{gathered}\underset{i=\{1,\dots,\lambda\}}{\text{minimize}}\, m_v(SS_i, \mathrm{A}) \,|v \in \{1, \dots, V\} \\ \text{subject to} (\delta_1, \dots, \delta_L)\end{gathered} \tag{3.1}$$

The optimization goal is to find SS_i that minimizes all the cost functions. In practice (some of) the objectives under consideration may be conflicting, i.e., optimizing for one cost function may worsen the performance in others. In that case the optimization shall find the best tradeoff. This can be done, for example, via aggregated (requires assigning proper weights to objectives) or constrained optimization using Lagrange multipliers [3.1] or via Pareto analysis [3.2]. The specific MOP solution techniques are beyond the scope of this thesis work and the readers are referred to the provided literature for more information on this topic.

It may be the case that none of the candidate system architectures satisfies the given set of constraints and then we say the application A is *not feasible* in these architectures. We can apply this formalization to our example with only two candidate architectures, leading to the following MOP (λ = 2, *V* = 2, and *L* = 3):

$$\begin{gathered}\underset{i=\{1,2\}}{\text{minimize}}\, m_1(SS_i, \mathrm{A}) \\ \underset{i=\{1,2\}}{\text{minimize}}\, m_2(SS_i, \mathrm{A}) \\ \text{subject to} \begin{pmatrix} \delta_1 : m_1(SS_i, \mathrm{A}) \leq 1\ interaction/day \\ \delta_2 : m_2(SS_i, b(\mathrm{A})) \leq 1\ sec \\ \delta_3 : 0 \leq \eta \leq \eta_{req} \end{pmatrix}\end{gathered} \tag{3.2}$$

where the last constraint indicates that the application A is required to support up to η_{req} users, such that adequate behavior of application, i.e., A_b is guaranteed if $\eta \leq \eta_{req}$ holds. This does not necessarily imply that the application A will fail whenever $\eta > \eta_{req}$.

The example illustrates i) that when the application behavior A_b is context dependent two identical sets of smart nodes may show entirely different behaviors in different contexts, and ii) that the chosen smart space architecture significantly impacts the performance of smart space applications.

3.2 Smart Space Properties

According to our literature survey, the fundamental properties of smart space designs in the literature are as shown in Table 3.2. In the following, we define these properties, i.e., *adaptation, communication interoperability, semantic interoperability, openness, extendibility* and *self-management*. Depending on whether a thorough consideration of each of these properties exists in candidate smart space architectures or not and looking at performance metrics that are important for an application A, it can be determined to which extent a smart space architecture is suitable for A.

Table 3.2. Properties of smart space solutions in the literature.

Project / Article	Adaptation	Communication Interoperability	Semantic Interoperability	Openness	Extendibility	Self-management
CISE [2.21]	+	+	+	-	+	-
Goh et al. [2.22]	+	+	+	+	+	+
PERSIST [2.23]	+	+	-	+	+	+
GUPSS [2.24]	+	+	-	+	+	+
(van der) Vlist et al. [2.25]	+	+	+	+	+	-
Song et al. [2.26]	+	+	+	+	+	-
SPITFIRE [2.27]	+	+	+	+	-	+
INSTANS [2.28]	+	+	+	+	+	-
Morandi et al. [2.29]	+	+	+	+	+	+
Natalia et al. [2.30]	+	+	+	+	+	+
Eila et al. [2.31]	+	+	+	+	+	+
Jussi et al. [2.32]	+	+	+	+	+	+
Zeng et al. [2.33]	+	+	+	+	-	+
Sergey et al. [2.34-2.35]	+	+	+	+	+	-
Andrey [2.36]	+	+	+	+	+	-
Shabir [2.37]	+	+	+	+	+	+
William et al. [2.45]	+	+	-	+	+	-
Zeynep et al. [2.46]	+	+	-	+	+	+
Tekler et al. [2.47]	+	+	-	+	+	-
Badr et al. [2.48]	+	+	-	+	+	-

* (+) means that the solution includes a thorough consideration of the respective property and (-) means that it does not.

When an application A is context-dependent, it may adapt the application behaviors A_b and trigger services as required by the scenario in question. Let the context domain of a smart space be given by C_s such that a particular context domain for an application A is

included, i.e., $\mathrm{A}_c \in C_s$. It is possible to dynamically change A_b at runtime to cope with performance issues that are stemming from changes in A_c.

Definition (Adaptation): An application A is said to exhibit adaptation with respect to metric *m* if it behaves to increase *A*'s adequacy with respect to *m* as a response to changes in application's contexts $\mathrm{A}_c \in C_s$.

Adaptation may be user initiated or autonomous. The changes in application's contexts A_c are due to direct interactions of users with smart nodes when the adaptation is user initiated, and the users are at best notified about the variations in contexts A_c in case of autonomous adaptation. For example, user-initiated adaptation occurs when a user enters a building: a sensor identifies the presence of the user and sends a command to turn the lights on. In this example, the user directly interacts with the smart nodes in the smart space. In contrast, autonomous adaptation occurs when a music player changes the light presets based on the music that is playing in the background. In this example, the users do not interact directly but may (or may not) be directly or indirectly notified about the changes. In this thesis, we take both user initiated and autonomous adaptations into consideration.

These adaptations are realized based on *iO*s' state changes to adapt application's contexts A_c. The changes in states of *iO*s to adapt A_c adequately need to occur within a limited time interval. One of the main reasons is that slow adaptation of actions taken by any *iO* may prevent timely adaptations of other *iO*s within application A, effectively leading to inadequate execution of A. Therefore, for adaptive applications, the latency of an event to changes in contexts of any *iO* is often a critical performance metric, i.e., a cost function to be minimized. Latency is the time interval between stimulation of an *iO* belonging to A and the time at which the corresponding changes take effect in A_c. In computing the total adaptation latency, we need to account for the latencies added by all *iO* tasks. The *iO*s perform the following tasks for adaptation: i) monitor contexts in the application environment, ii) analyze the contexts and find the needs for adaptation, iii) execute the adaptation. These three steps followed by any *iO* will result in a change of A_c. Based on these observations, we can classify the *iO*'s adaptation into the following three types:

1.) **Periodic Adaptive *iO*:** Gathering contextual information from an environment of a smart space and periodically updating it to the *iO*, potentially changing its state.
2.) **Triggered Adaptive *iO*:** Triggering a certain event based on context information received from another *iO* in a smart space, the result of which may change the receiving *iO*'s state.
3.) **Controlled Adaptive *iO*:** Controlling the *iO* state explicitly to take decisions for achieving a certain application goal in a smart space.

We consider an adaptive lighting example (Fig. 3.5) to explain the classification of adaptive *iO*. Let the adaptive lighting example have the following goal and assumptions:

Goal: The example system needs to achieve the goal of maintaining light intensity according to user preferences.
Assumptions: An *SiO* (deployed on an SN) serves as a light sensor and an *AiO* (deployed on an AN) acts as a luminary with an actuator, both connected to the *GWiO* (deployed on a GSN) in a room infrastructure. The *SiO* updates current light intensity values periodically, every 10 seconds. The *AiO* needs to control its light output based on the commands given by the *GWiO* and a user sets the preferred light intensity at the *GWiO*.

The *SiO* updates the light intensity periodically at the *GWiO* every 10 seconds, allowing the *SiO* to exhibit the periodic adaptive *iO*. The *GWiO* at the GSN calculates the actuation commands based on user's set preferences and the light intensity, demonstrating the controlled adaptive *iO*. Finally, the *AiO* at the AN receives actuation commands from the *GWiO* and actuates the light output accordingly, showcasing the triggered adaptive *iO*. The goal of the adaptive lighting application example is achieved by maintaining the light intensity at the desired level, as defined by user preferences.

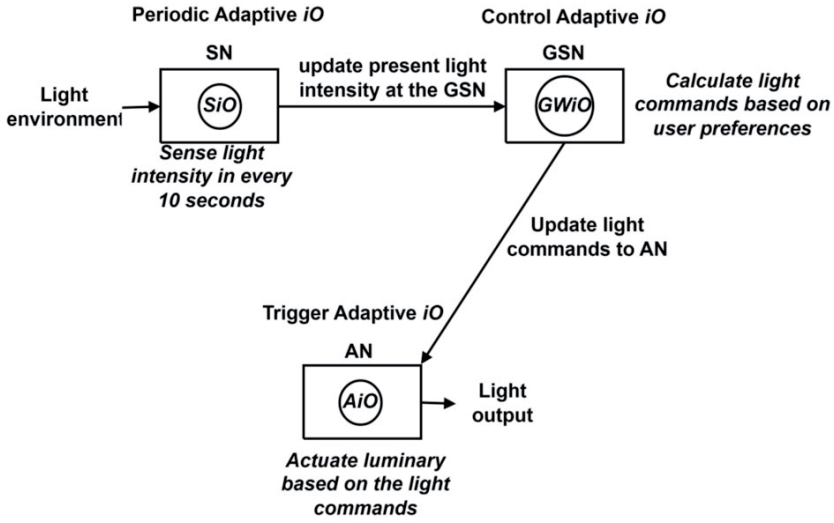


Figure 3.5. An adaptive lighting example to explain the classification of adaptation in *iO*s.

To conclude the discussion on adaptation, we can state that a smart application A may adapt and reconfigure according to changes in states as demanded by the encountered scenarios. The joint behaviors of a set of *iO*s in A, aimed at achieving an application goal, result in adaptive application A_c.

Smart nodes consist of diverse hardware and software platforms and interoperability among them is a requirement. Interoperability, which refers to the ability to exchange information at different levels, between two *iO*s requires compatibility of the hardware and software platforms used by their smart nodes. Broadly, interoperability is achieved in two ways: communication interoperability and semantic interoperability.

Definition (Communication Interoperability): The ability of smart nodes in a smart space to exchange messages over a network, following certain (communication) protocols, message formats and syntax, is called communication interoperability.

Once communication interoperability is established in a smart space, semantic interoperability plays a crucial role in exchanging semantically meaningful information. Achieving semantic interoperability means representing contexts as structured knowledge that holds meaning for the *iO*s sharing them. Semantics are constructed through syntax and pragmatics. Syntax refers to the principles and processes by which symbols are constructed in specific languages, while pragmatics explores the relationships between the symbols of a language and their meanings.

Many researchers in the fields of linguistics and programming languages are currently engaged in semantics [3.3-3.6], which traditionally focuses on the study of the meaning of (parts of) words, phrases, sentences, and texts. According to Euzenat [3.7], semantics '*provide the rules for interpreting syntax, which does not directly provide meaning but constrains the possible interpretations of what is declared*'. In other words, semantics involve interpreting information applied to specific types of data structures designed for representing information content.

In linguistics, semantics refers to the '*study of meaning*', aiding in understanding human expression through the elements of a language. These elements, known as symbols, are physical entities (such as character strings) used to convey meaning. Meaning is derived from the relationships between symbols, concepts, and real-world entities. A concept can be expressed in two different sets of symbols, but the meaning of these symbols aids in understanding the concept. This implies that two or more symbols can have the same meaning in the real world. For instance, '*human*' in English is known as '*menselijk*' in Dutch and '*humain*' in French, yet the meaning of these symbols remains consistent across all languages.

In literature [3.8-3.9], semantic interoperability is defined as the ability of computer systems to exchange data with unambiguous meaning, referring to the exchange of the meaning of data. In this thesis, we study the meaning of contexts, which is referred to as the semantics of those contexts. The translation of contexts into these meaningful repre-

sentations, i.e., semantics, is important for achieving semantic interoperability. Semantic interoperability can be established between two *iO*s in a smart space, if the meaning of contexts maps to the shared ontology of the smart space. When the meaning of contexts maps with the shared ontology, it means that the understanding of the situations or environments matches the agreed-upon framework of concepts and their relationships within that specific domain. We will discuss the properties related to mapping in Chapter 4 with semantic reasoning.

Definition (Semantic Interoperability): The ability of *iO*s in a smart space to interoperate using only the meaning (semantics) of contexts is called semantic interoperability. Semantic interoperability of *iO*s is built on top of communication interoperability and is typically achieved by utilizing a shared ontology.

The next smart space property is openness. For openness, the hardware and software platforms of smart nodes in a smart space should allow third parties to develop and implement *iO*s that can be used in the smart space. The smart space architecture must support portability of *iO*s and enable interoperability in heterogeneous networks of smart nodes. Let the set of protocols, message formats and syntax in smart space SS_i be denoted by SyntaxFormat (SF), i.e., SF_i. Consider a new *iO* (iO_{new} that has just joined SS_i. Let the set of protocols, message formats and syntax employed by iO_{new} be given by SF_{new}. The set SF_{new} needs to be a subset of SF_i or the smart space needs to facilitate a translation between SF_i and SF_{new} (e.g. by means of a communication gateway) for openness. In addition, semantic interoperability of the iO_{new} in SS_i is achieved if the additional condition holds that the ontologies of iO_{new} are a subset of the ontologies of SS_i. Therefore, openness exists in the architecture when the joining of new smart nodes in the architecture can be established by a third party.

Definition (Openness): A smart space architecture is said to be open if its protocols, data formats and syntax are well-described such that third parties can design new *iO*s and nodes that will integrate with an existing smart space.

The next smart space property is extendibility. A smart space architecture must also be extendible to allow for additions to the smart space, e.g., to enable smart nodes to access the smart space easily in new applications. It must enable programmers to develop applications without having to interact with the physical world of embedded devices. In other words, the smart space must provide an interface that can decouple programming and application development from physical infrastructure deployment and integration. Extendibility indicates that new nodes and applications can easily be inserted into a smart space; meaning installation and bootstrapping should be with minimal technical expertise involvement.

Definition (Extendibility): A smart space is said to be extendible if new smart nodes can connect and new applications can be installed to the smart space.

A smart space is typically composed of heterogeneous smart nodes and networks. Smart nodes have varying capabilities in terms of resources such as communication bandwidth, computational power, memory and energy. LSNs are examples of resource-poor nodes such as SNs and ANs, whereas HSNs are not limited in terms of resources. LSNs can execute simple protocols and simple tasks. This requires a management mechanism to handle LSNs' resources and their associated services to facilitate efficient and timely responses to changes in contexts.

Definition (Self-management): Self-management is the capability of smart space to monitor and manage its resources and services.

Self-management services can be assigned to a particular smart node to be accessed by all *iO*s. Alternatively, these services can also be distributed. A smart space should give ample opportunities to the associated applications for enhancing their persistence by self-management as a failure-free (dependable) operation of smart space applications is key to user satisfaction.

Self-management for resources and services in smart space architectures can be extended to include management for security and privacy. For example, the *iO*s taking part in smart applications not only require access to sensitive data and services, but also insert their own data and services into the smart space, which calls for security measures to be taken (e.g. using encryption). Protecting privacy properties related to its users is an important concern for a smart space. A privacy property is a mapping from an information receiver *IR* and a data item *D* to a data handling property D_p: "*IR* will only do D_p with *D*". A smart space should, therefore, aim to guarantee and enforce privacy properties while exchanging contextual knowledge with *iO*s.

Finally, we call the first three properties, namely adaptation, communication interoperability and semantic interoperability, the primary properties of a smart space. They represent the minimum requirements for a smart space to function. Specifically, *i*) the behavior of each *iO* needs to adapt according to the state changes of smart applications, a requirement derived from our definition of a smart space; *ii*) communication is a basic necessity for exchanging information among smart nodes that produce and consume it; and *iii*) semantic interoperability is required to establish a shared meaning for the exchanged information. Secondary smart space properties, namely openness, extendibility and self-management can be implemented as needed based on application requirements within smart spaces.

3.3 Semantic Interoperability Architecture

Before we propose the semantic interoperability architecture in smart spaces, we would like to discuss the following observations on challenges and considerations in the architecture based on the discussion so far in this thesis.

***Semantic Interoperability as a challenge in smart space architectures*:** We have introduced two types of interoperability: communication interoperability and semantic interoperability. Communication interoperability in smart spaces can be easily established by using standard network protocols or by employing a gateway to connect with LSNs. Example of standards include powerline technologies like X10 and LonWorks, wireless technologies such as IEEE 802.15.4 for wireless sensor networks, and CAT5 for audio, video or data communication. Additionally, middleware solutions like Jini [3.10], HAVi [3.11], UPnP [3.12], and IoT middleware [3.13] can be utilized to connect with smart nodes. For a comprehensive overview of communication interoperability utilized in various smart space projects or applications, refer to Table 3.3.

The core challenge in smart space architectures is achieving semantic interoperability, because different systems may use different terminologies, taxonomies, or ontologies to represent information. Furthermore, the semantic interoperability architecture consists of heterogeneous smart nodes that comprise a network of interconnected nodes with varying capabilities, features, or specifications. These nodes may have different processing power, memory capacity, communication protocols, or sensor types. To address this, we need to develop semantic interoperability on top of communication interoperability to facilitate information sharing among these *iO*s. This task is not trivial due to the heterogeneity of smart nodes, presenting a research challenge.

For instance, contexts from *iO*s must first be translated into semantics and then shared using a common ontology model. Each *iO* in the architecture should be capable of performing this translation and aligning with the common ontology model. However, not all *iO*s (e.g., those of LSNs) can directly perform semantic translation. In such cases, indirect methods, such as using a gateway node to translate contexts into semantics, can be employed. Consider a distributed application where a body sensor node, attached to a user, exchanges information with the user's smartphone. Achieving a common understanding between the body sensor node and the user's smartphone requires knowledge representation. However, this task is not straightforward due to resource limitations (energy, processor, memory, and storage) in the body sensor node, presenting another research challenge.

Among the primary properties of smart spaces, we prioritize solving semantic interoperability within the semantic interoperability architecture. Semantic interoperability also

Table 3.3. Communication interoperability related work.

Article	Communication Interoperability
CISE [2.21]	For lightweight communication, application layer protocols like HyperText Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP) are used for sending and receiving messages, with the default language being Extensible Markup Language (XML).
Goh et al. [2.22]	The request and response messages are transmitted using SOAP, which utilizes application layer protocols like HTTP and SMTP.
PERSIST [2.23]	Various connectivity technologies, including Ethernet, Wi-Fi, Global Positioning System (GPS), and 3G, are utilized for different types of nodes such as mobile phones, home appliances, and surveillance cameras.
GUPSS [2.24]	Communication with sensors was established using the low-power IEEE 802.15.4 standard, which defines the physical and Medium Access Control (MAC) layers for a low-power Personal Area Network (PAN). The sunSPOT system and technologies are employed for data transmission schemes that are similar to the UDP and TCP protocols used on the internet.
(van der) Vlist et al. [2.25]	Basic communication is achieved by using Ethernet and Wi-Fi.
Song et al. [2.26]	Interoperability is established by combining services of devices based on the non-IP Bluetooth and UPnP specifications.
SPITFIRE [2.27]	To integrate the sensor into the web, 6LoWPAN in combination with CoAP is introduced, as it can provide what are known as RESTful web services.
INSTANS [2.28]	The connectivity is established using TCP over IP, and SSAP is utilized as the message transmission protocol.
Morandi et al. [2.29]	The connectivity is established using TCP over IP, and SSAP is utilized as the message transmission protocol.
Natalia et al. [2.30]	The connectivity is established using TCP over IP, and SSAP is utilized as the message transmission protocol.
Eila et al. [2.31]	The connectivity is established using TCP over IP, and SSAP is utilized as the message transmission protocol.
Jussi et al. [2.32]	The connectivity is established using TCP over IP, and SSAP is utilized as the message transmission protocol.
Zeng et al. [2.33]	The connectivity is established through Bluetooth, Zigbee and WiFi.
Sergey et al. [2.34-2.35]	WiFi is commonly used as the wireless technology to interconnect various IoT devices. Additionally, SSAP is employed as the message transmission protocol.
Andrey [2.36]	The connectivity is established using TCP over IP, and SSAP is utilized as the message transmission protocol.
Shabir [2.37]	Communication between the cloud and IoT toolbox and the cloud and real devices uses an HTTP approach, whereas CoAP is used to communicate between the cloud and the IoT toolbox, as well as between the cloud and real devices, utilizes an HTTP approach. On the other hand, CoAP is used for communication between the toolbox and actual resources.
ISHEWS [2.43]	Basic connectivity in communication is established through Ethernet, Wi-Fi, and PPP protocols. CAT5 cables are used for audio, video, and data communications. Network interoperability for communication is achieved through protocols such as TCP, UDP, FTP, ARP, IP, and IPv6.

facilitates adaptation in smart spaces through the adaptation in *iO*s. For instance, multiple *iO*s collaborate to achieve specific goals in an application and dynamically adjust their contexts to meet the requirements of these goals. These *iO*s adapt their contexts at runtime without the need for prior configuration.

***Integration of LSNs as a challenge in the semantic interoperability architecture*:** First, we ensure that the LSNs are compatible for integration with the gateway in terms of communication protocols and hardware. To achieve this, we install any required drivers or software on both the LSNs and the gateway to facilitate communication and data exchange. The gateway requires an internet connection to further connect with a smart space. Once the connection is established, we install *iO*s to produce or consume information within a smart space.

We deploy *SiO*s and *AiO*s on LSNs which are typically not capable of doing complex computations due to power and memory constraints. There is a wide range of potential applications such as those in smart homes [3.14], smart healthcare [3.15], transport and logistics management [3.16], inventory and product management [3.17], firefighting systems [3.18], social networks [3.19], smart cities [3.20], and smart lighting systems [3.21], to name just a few. However, the translation of contexts into semantics poses a bottleneck for LSNs. In some infrastructures [3.22-3.28], LSNs are capable of semantically sharing information using semantic web technologies such as the Sensor Web Enablement (SWE) specification. SWE defines sensor data representation in the Sensor Model Language (SensorML), which utilizes an XML-based structure. SensorML describes the semantics and relationships between different data elements of sensor nodes using XML representations. SWE provides models and interfaces to handle sensor data (represented in SensorML) in applications based on heterogeneous sensor networks. It aims to enhance the practicality of producing semantics for smart nodes and establish interoperability with the semantic web. Similar approaches exist for LSNs to comply with the semantic web, including Sensorpedia [3.29], SensorWare [3.30], and SensorMap [3.31].

However, while these approaches are effective in integrating LSNs with the semantic web, they yield unfavorable results when LSNs themselves need to represent semantics, as this significantly increases their power consumption and relies heavily on processing capacity. This tradeoff between the lifespan and computational capability of LSNs creates an opportunity to integrate an external unit—a gateway node—that can provide the necessary computational power and knowledge representation capabilities to enable information sharing with other smart nodes. The GSN in the semantic interoperability architecture offers an excellent solution to address the computation and memory bottleneck of LSNs. By deploying a *GWiO* on the GSN, we can compute knowledge representations for LSNs and process them to achieve improved results. Consequently, the GSN becomes a power-

ful smart node capable of performing computations and executing semantics through the *GWiO*, using the contexts received from resource-limited LSNs.

***Consideration of management tasks in the semantic interoperability architecture*:** Management tasks in the semantic interoperability architecture, such as application management, resource management, and security and privacy management, depend on the specific requirements of the applications. The application management task is performed by the *iO*s themselves (explained in chapter 4); eliminating the need for a separate component called *AMiO* in the proposed architecture. Resource management involves representing the more abstract concept of smart nodes and their service discovery as defined in the architecture. For example, when a smart node leaves an application, its services should be handed over to another smart node within the application. Another example of resource management is workload balancing among *SBiO*s. This involves distributing the workload based on the number of *iO*s communicating with a single *SBiO*. If an *SBiO* becomes overloaded, some services can be transferred to another *SBiO*. It is important to note that resource management is beyond the scope of this thesis; however, we propose a workload balancing mechanism for *SBiO*s in [3.32].

Finally, the security and privacy management task in the architecture involves a concept of membership with different authorization levels. To address this, we propose a single management component, the *MiO*, associated with each *SBiO* in the semantic interoperability smart space architecture.

Developers can implement tasks related to *AMiO*s, *SPMiO*s and *RMiO*s as needed, based on the application requirements. By centralizing all management tasks within the *MiO*, developers can choose and configure the specific management task required for their application. In this thesis, the management task of *MiO* is to deploy and maintain a basic ontology graph at the *SBiO*. The role of *MiO* becomes more apparent in Chapter 4 when we delve into the discussion of ontologies.

We now propose the design of a semantic interoperability architecture in smart spaces, as depicted in Fig. 3.6. This architecture generalizes the smart space architectures studied in Section 2.3. Depending on the requirements of the application at hand, users can employ specific instantiations of this semantic interoperability architecture. Figure 3.7 illustrates the component diagram, depicting the processes and dependencies among *iO*s in the architecture.

In a smart space application, various *iO*s are present. All *iO*s convert contexts into semantics using a semantic representor. The semantic representor refers to a module that is responsible for representing semantic information in a structured format. This involves

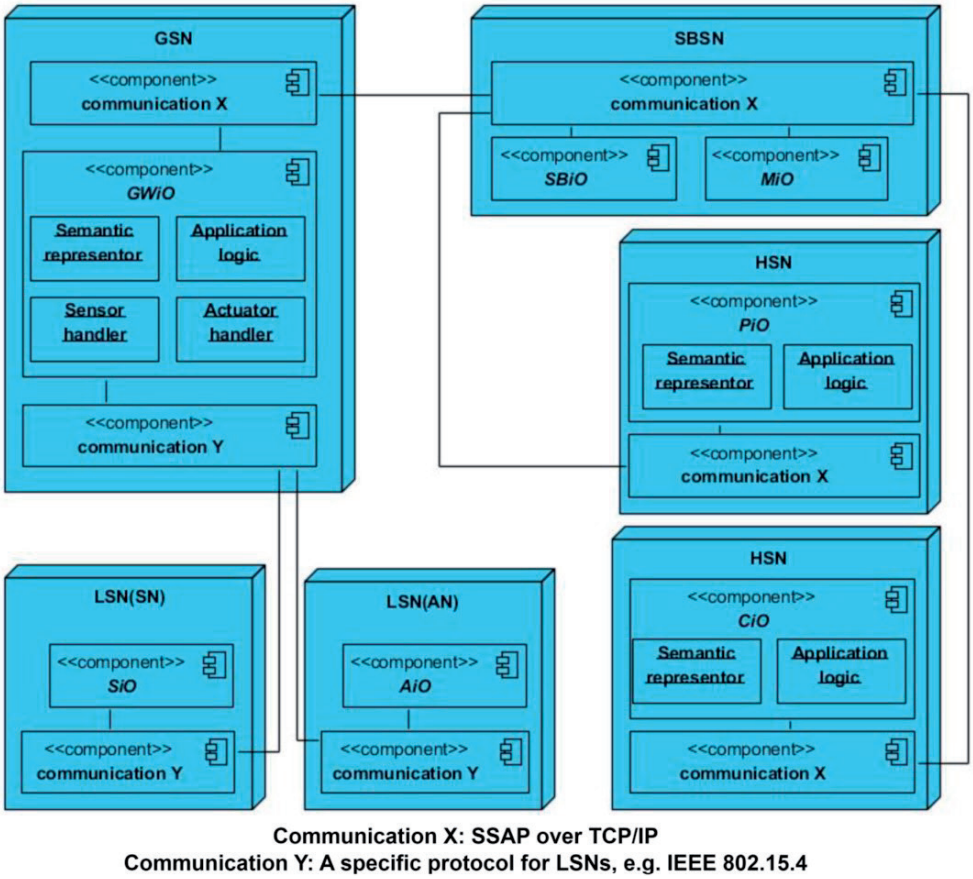


Figure 3.6. Semantic interoperability architectural design.

converting contexts into a standardized semantic representation, such as RDF, which allows for the explicit representation of meaning and relationships between entities. The *PiO*s rely on the contexts received from the environment and generate semantics using the semantic representor that are intended to be shared with other *iO*s. The updated semantics at the *SBiO* are processed by semantic reasoning (discuss in Chapter 4), utilizing an ontology graph that defines the relationships between different semantics.

Further, these semantics are stored in the *SBiO* repository and can be accessed by *CiO*s. This enables meaningful interaction between a source *iO* and receiver *iO* through the exchange of semantics via the *SBiO* in a smart space. Please note that we will delve into the details of semantic interactions and the ontology graph in Chapter 4. The *GWiO* component consists of two handlers: a sensor handler and an actuator handler. The sensor

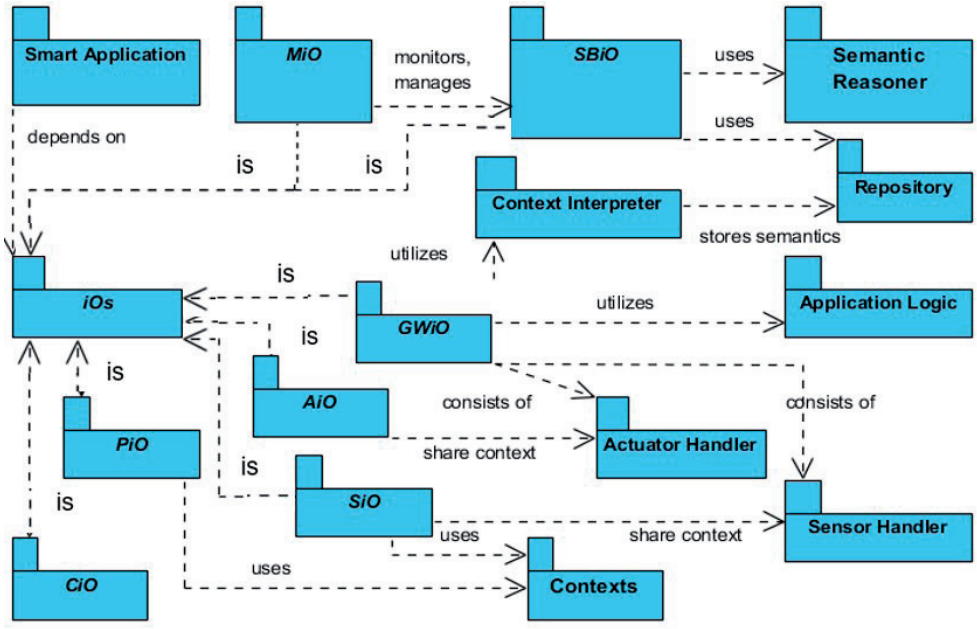


Figure 3.7. Processes and dependencies among *iO*s in the semantic interoperability architecture.

handler processes information from *SiO*s, while the actuator handler processes information towards *AiO*s. These handlers rely on input and output information from sensors and actuators, respectively.

Application logic refers to the set of rules, procedures, and operations that govern the behavior and functionality of a smart application. It determines how the application processes data, responds to events and user inputs, performs calculations, and carries out specific tasks or functions. The contexts received from an *SiO* are transferred to the application logic through the *GWiO*, and *AiO* receives the actuation commands from the application logic via the same *GWiO*. The application logic concludes actuation commands based on the contexts received from *SiO*s and the semantics received from the *SBiO*. Before being used by the application logic, the semantics received from the *SBiO* is first translated into contexts by the *GWiO*. Lastly, an *MiO* manages communications between smart spaces and is installed on the same node as an *SBiO*.

In this proposed architecture, consisting of an SBSN, a set of distributed HSNs or GSNs along with LSNs, represents a single smart space. Multiple smart spaces are formed by a collection of different SBSNs, with each SBSN corresponding to a single smart space.

According to our definition of a smart space, the following relationships hold for smart nodes in a smart space *SS* and denoted by *SS.N*.

$$SS.N = n_{SB} \cup n_H \cup n_{GW} \cup n_S \cup n_A \tag{3.3}$$

where n_{SB}, n_H, n_{GW}, n_S and n_A denote the SBSN, the set of HSNs, the set of GSNs, the set of SNs and the set of ANs in *SS.N*, respectively. Note that in a smart space it is necessarily the case that $SS.N \neq \emptyset$. More specifically, in a smart space there is one SBSN and a non-empty set of smart nodes, i.e., $n_H \cup n_{GW} \cup n_S \cup n_A \neq \emptyset$. Furthermore, logic dictates that $n_{GW} \neq \emptyset$ if and only if $n_S \neq \emptyset$ or $n_A \neq \emptyset$.

Each smart node has an *iO* or a set of *iO*s. The set of all *iO*s in a smart space is denoted by *SS.iO*:

$$SS.iO = SB \cup MO \cup P \cup C \cup GW \cup S \cup A \tag{3.4}$$

where *SB*, *MO*, *P*, *C*, *GW*, *S* and *A* denote the *SBiO*, the *MiO*, the set of *PiO*s, the set of *CiO*s, the set of *GWiO*s, the set of *SiO*s and the set of *AiO*s respectively. The terms on the right-hand side of the equation denote the sets of specific types of *iO*s hosted by smart nodes as follows:

- The combination of *sb* and *mo* are together hosted by a node $n_{sb} \in n_{SB}$, where $sb \in SB$ and $mo \in MO$; and more than one combination of this can be hosted by the node n_{sb}.
- *p* or *c* are hosted by a node $n_h \in n_H$, where $p \in P$ and $c \in C$; and more than one *p* or *c* can be hosted by the node n_h. This is also possible that the combinations of *p* and *c* can be hosted by the node n_h.
- *gw* is hosted by a node $n_{gw} \in n_{GW}$, where $gw \in GW$; and more than one *gw* can be hosted by the node n_{gw}.
- *s* is hosted by $n_s \in n_S$, where $s \in S$; and more than one *s* can be hosted by the node n_s. Similarly, *a* is hosted by $n_a \in n_A$, where $a \in A$; and more than one a can be hosted by the node n_a.

Note: *s* and *a* cannot communicate and share semantics directly, they first always need to communicate with a *gw* to translate contexts into semantics and to communicate with other *iO*s in a smart space.

3.4 Conclusions

In this chapter, we have discussed the process of selecting the best smart space architecture, specifically tailored to meet a specific performance metric, i.e., response time relevant to smart applications. To achieve this, we formulated a multi-objective optimization problem to find an optimal architectural solution among feasible alternatives based on the specific requirements of applications in a smart space. We concluded that the choice of a smart space architecture is highly dependent on the specific requirements of the application, taking into consideration performance metrics that may vary depending on the context. Additionally, we explored the various properties of smart spaces and proposed the semantic interoperability architecture. Through this discussion, we identified communication interoperability, semantic interoperability, and adaptation as the primary properties of smart spaces, while others such as openness, extendibility, and self-management are considered secondary properties. Lastly, we recognized that achieving semantic interoperability and integrating low-capacity devices within the architecture pose the most challenging issues to be resolved.

Chapter 4

Semantic Interoperability

We proposed the semantic interoperability architecture in Chapter 3 where we emphasized that semantic interoperability is a property that is both vital (a primary smart space property) and challenging to realize. In this chapter, we lay out the details of how to achieve semantic interoperability between iOs in a smart space and present iO interaction mechanisms. To accomplish this, we discuss fundamental concepts such as semantics and reasoning, and explore SSAP transactions for facilitating semantic interactions in smart spaces. Additionally, we provide scenario examples to illustrate semantic interactions in two parts: i) semantic interactions among iOs of LSNs and HSNs to enable adaptations within a single smart application of a smart space, and ii) semantic interactions among iOs for adaptations with multiple interconnected smart applications of a smart space.

4.1 Introduction

Interoperability refers to the ability of two systems to operate together. In [4.1], interoperability is structured into multiple levels, including the *no connection* level, *technical* level, *syntactical* level, *semantic* level, *dynamic* level, and *conceptual* level interoperability. This 6-level model of interoperability is quite complex. Other researchers have proposed simplified models with fewer levels. For instance, [4.2] presents an interoperability model with node, service, and information levels.

In this thesis work, we adopt the simple two-level interoperability model from the SOFIA project [1.35]. In this model, interoperability is defined at the *connection* level (communication interoperability, as explained in Chapter 3) and at the semantic level (semantic interoperability, as defined in Chapter 3). The connection level interoperability of nodes in the SOFIA project utilizes the standard OSI model, ranging from the physical layer to the transport layer, for process-to-process data transfer. In general, communicating nodes may use different hardware and software platforms, as well as different communication protocols, resulting in a heterogeneous network. However, communication interoperability alone does not guarantee that the exchanged information can be understood and utilized by the receiver. To achieve that, the *iO*s need to interoperate at a higher level, namely the semantic level interoperability (or simply semantic interoperability). Semantic

interoperability ensures that the meaning of the exchanged information is consistent across *iO*s that have access to it.

For semantic interoperability, sharing information in a smart space requires a suitable abstraction technology that specifies how to express shared concepts and their relationships, as well as an ontology that establishes the mapping between the exchanged information and the corresponding knowledge within a specific domain, ensuring its validity.

There is a considerable body of literature on information sharing in the semantic web using web technologies. One common approach to implementing semantic web technologies is through the utilization of the blackboard architecture style [4.3]. The blackboard architecture style is analogous to a classroom blackboard used for collaborative problem-solving. It is a data-directed and partially data-driven architecture. The architecture is divided into two main components: the blackboard (which stores data) and knowledge sources (representing the knowledge of a local domain), as shown in Fig 4.1. Knowledge sources do not interact with each other directly; rather, they interact and respond to the blackboard. It is also possible to introduce an additional component, known as the controller, to coordinate the interactions between knowledge sources and the blackboard.

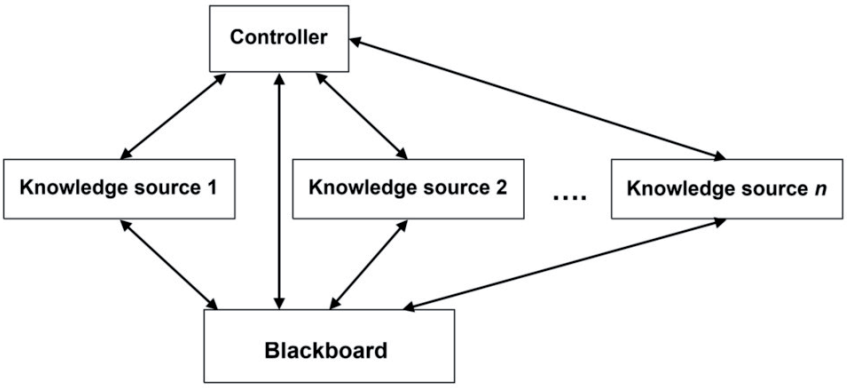


Figure 4.1. Blackboard architecture style.

In the experimental section of this thesis, we utilize the solution provided by SOFIA for publishing and subscribing semantics in the proposed semantic interoperability architecture. SOFIA employs a publish-subscribe pattern based on the blackboard architecture style for semantic solutions in smart spaces. It consists of two entities: the Knowledge Processor (KP) as a knowledge source and the Semantic Information Broker (SIB) as a blackboard. SOFIA offers a platform for interoperability across domains, devices, and vendors, known as Smart-M3[2] (Multi-vendor, Multi-device, Multi-domain). It enables

2 https://github.com/smart-m3/sib-tcp

the integration of information domains that participate in smart applications using web technologies. For instance, an information domain can be represented as an ontology. SOFIA enables the integration of ontologies at SIB, where KPs participate in producing or consuming information through SIB. We leverage the Smart-M3 architecture to align with the proposed semantic interoperability architecture. In this context, HSNs and GSNs be like the KPs, while the SBSN corresponds to the SIB. Therefore, the *PiO*s, *CiO*s and *GWiO*s do not directly communicate with each other; rather, they interact and communicate with the *SBiO* using the SSAP protocol, as depicted in Fig 4.2. An additional *iO*, the *MiO*, serves as a controller responsible for managing semantics at an *SBiO* and is colocated with the *SBiO* on the same node. It operates differently from the controller in the blackboard architecture style, as the *MiO* does not directly interact with other *iO*s in a smart space. Its tasks include deploying and managing application ontologies at the associated *SBiO* on the SBSN node. Lastly, the *MiO* has the capability to modify or delete RDF triples, thereby managing the semantics stored at the *SBiO*.

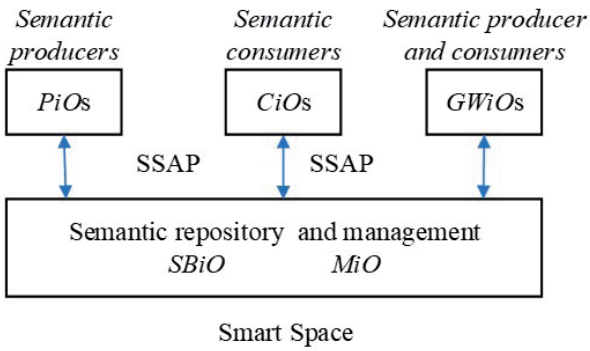


Figure 4.2 Interactions of *iO*s with the *SBiO* and *MiO*.

In this chapter, we address research question **RQ_2**, aiming to achieve semantic interoperability within the proposed semantic interoperability architecture. It is natural to question what conditions are necessary to achieve semantic interoperability. In this context, we discuss the following three necessary conditions.

1.) First and foremost, in order to establish a shared understanding of the information being shared, the participating entities must possess a common understanding of the knowledge domain. This necessitates a precise description of the domain, such as a taxonomy. A taxonomy serves as a classification scheme and functions as a knowledge map. An ontology, on the other hand, is a tool used to describe such a taxonomy, defining its concepts and the relationships between them. To formally specify an ontology for a domain and enable communication using that ontology, two key components

are required: i) a syntax for referencing concepts and their direct relationships, and ii) a language for representing the knowledge encapsulated within the domain taxonomy. The former allows for expressing any semantics and relationships, regardless of whether they are valid within the domain taxonomy, while the latter enforces the validity of the expressed information. For instance, consider the taxonomy statement "Brown is a Doctor", where Brown belongs to the human class and Doctor belongs to the profession class within the domain taxonomy. Consequently, a valid expression would be "Brown is a human name, Brown is a Doctor", whereas an invalid expression would be "Brown is a color". In Section 4.2.1, we discuss RDF triples in detail, which consist of a subject, predicate, and object. The predicate is used to validate the property type of the subject. If "Brown" is the name, the predicate will represent the "name" property, and if "Brown" is the type, the predicate will represent the "type" property. Thus, the validity of an RDF triple depends on how the statement is properly represented and aligned with the intended semantics.

In the proposed semantic interoperability architecture outlined in Chapter 3, we introduced the concept of knowledge representation. RDF (Resource Description Framework) is employed to represent knowledge, while the Ontology Web Language (OWL) is utilized to represent the knowledge embodied in a domain taxonomy, e.g. in the domain of lighting applications. In Section 4.2.1 of this chapter, we provide a comprehensive explanation of the underlying concepts of semantics and reasoning.

2.) The second essential condition for achieving semantic interoperability is that the communicating entities must reach a consensus on a domain ontology that will govern their interactions. The domain ontology defines the necessary specifications for an *iO*, such as the information or services expected to be shared by the *iO* within the smart space. To fulfill this requirement, *iO*s that communicate semantics to other *iO*s must adhere to the constraints imposed by the designated domain ontology.

In this thesis, we adopt an approach that utilizes application ontologies for designing smart lighting applications within the proposed semantic interoperability architecture. Consequently, we confine the implementation of ontologies to the smart lighting domain. In Section 4.2.1, we provide a comprehensive explanation of the smart lighting domain and its relevance to the application ontologies employed in the proposed semantic interoperability architecture.

3.) Thirdly, for semantic interactions between the communicating entities in smart spaces, it is essential to establish a protocol that facilitates their interaction. This protocol should encompass a precise description of how connectivity is managed and semantics are shared between *iO*s. Regarding the protocol for purposeful interactions

among *iO*s, three key components are required: i) a set of transactions that outline the procedures for joining and leaving a smart space, ii) a set of transactions for updating, querying, and subscribing to semantics by *iO*s, and iii) a transaction mechanism within the protocol that enables seamless interaction between *iO*s. To fulfill these requirements, we utilize the SSAP protocol, which enables *iO*s to modify and access semantics. Section 4.2.2 provides a detailed explanation of the SSAP protocol and its role in facilitating semantic interactions among *iO*s.

In Section 4.2, we introduce the fundamental concepts of semantic interoperability, specifically semantics and reasoning, as well as SSAP transactions. Section 4.3 delves into the mechanism of semantic interactions among *iO*s in a smart application, covering two aspects: i) semantic interactions of *PiO*s and *CiO*s with an *SBiO*, and ii) semantic interactions of a *GWiO* with an *SBiO*. Finally, in Section 4.4, we present a mechanism for semantic interactions among multiple smart applications within a smart space.

4.2 Fundamental Concepts of Semantic Interoperability

In this section, we discuss the concept of semantics and reasoning in Section 4.2.1 and SSAP transactions in Section 4.2.2. These concepts are based on three necessary conditions of semantic interoperability introduced in Section 4.1.

4.2.1 Semantics and reasoning

In chapter 3, we defined semantic interoperability as the extraction of semantics from the information to be shared by *iO*s in a smart space. The sharing of semantics in a smart space is made possible by *SBiO*, where *SBiO* generates the results of queries and subscriptions through semantic reasoning. Therefore, in this section, we first define and discuss semantics, and then we explain semantic reasoning.

We introduced the concept of semantics, where knowledge is represented using RDF graphs. An RDF graph is composed of graph_nodes and directed edges. Graph_nodes represent objects in the domain of discourse while edges represent relationships indicated by the label. We represent the combination of graph_nodes and directed edges as RDF triples. An RDF triple, or simply a triple, is an object of the form **{rdf_subject, rdf_predicate, rdf_object}**. The rdf_subject and rdf_predicate are resources (Uniform Resource Identifier (URI)s) defined by graph_nodes in an RDF graph, while the rdf_object is another resource identified by a URI or a literal and also defined by a graph_node. Literals have datatypes that define the range of possible values, such as strings, numbers, and dates. An example of RDF graph is shown in Fig 4.3.

For instance, the context set A_c of a smart application A = {α_1,α_2}= {“light has an intensity in unit”, “light has color”} is represented as a set of RDF triples A_s ={σ_1,σ_2,σ_3}, where $\mathrm{A}_c \subseteq C_s$ and $\mathrm{A}_s \subseteq O_s$

- σ_1 = (“Light”, “hasLightIntensity”, “LightIntensity”),
- σ_2 = (“LightIntensity”, “hasUnit”, “LightUnit”),
- σ_3 = (“Light”, “hasColor”, “LightColor”)

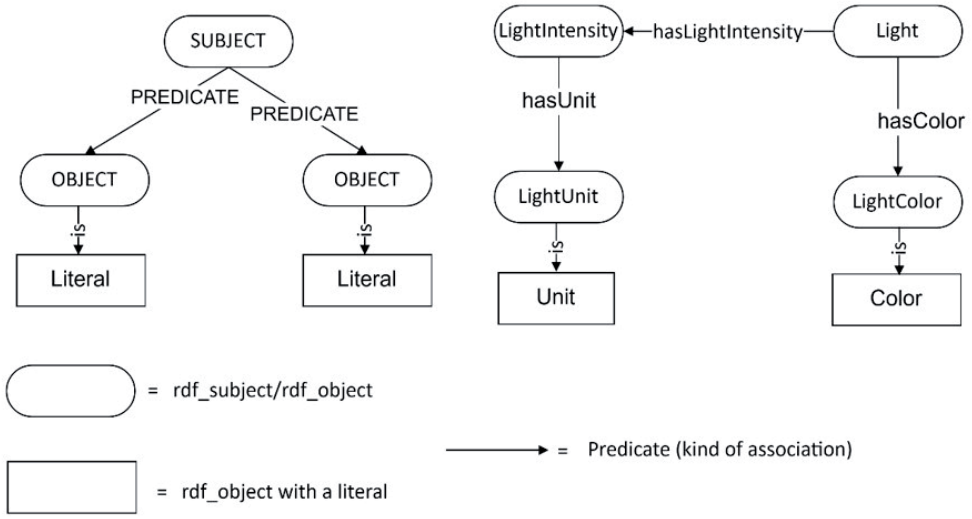


Figure 4.3. A graph of RDF triples with example.

Now we can give a formal definition of ontology as follows:

Definition (Ontology): An ontology is a collection of interconnected RDF graphs that defines a set of concepts and categories in a particular domain, illustrating their properties and relationships. It is also commonly referred to as an ontology graph.

This means that we represent concepts as RDF triples, allowing us to define a model for representing concepts and their relationships in ontologies. In this thesis, we consider application ontologies to design smart lighting applications within the semantic interoperability architecture, which is introduced with the smart lighting model in Chapter 5. These application ontologies represent a specific application or a set of related applications. They define concepts and relations for smart lighting applications and are made available at the *SBiO*. Every concept can be expressed with all possible relationships. For example, a sensor ontology graph can include all conceivable concepts and relations illustrated in Fig 4.4. This figure is self-explanatory and can be easily correlated with Fig 4.3.

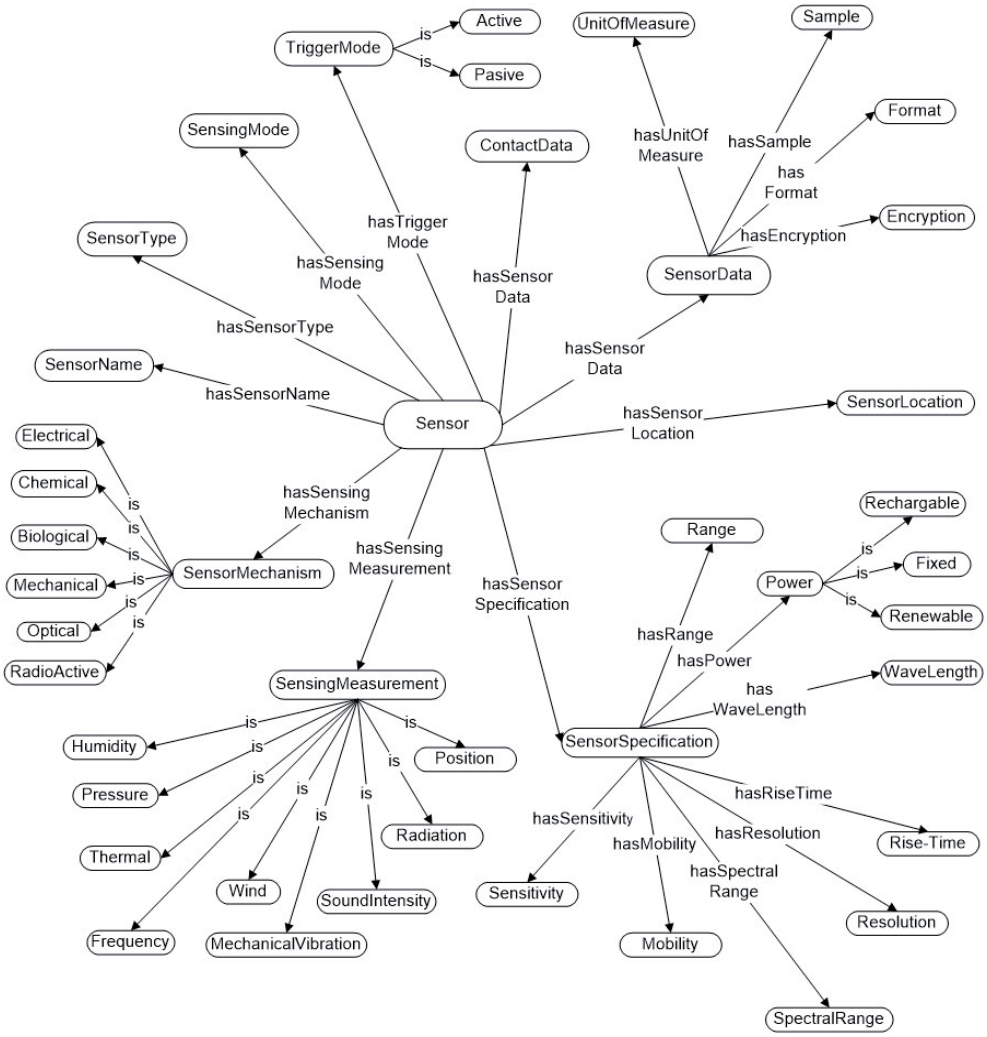


Figure 4.4. An example of the sensor ontology graph in the smart space ontology.

In this thesis, we adopt the approach of using application ontologies in smart spaces. It is important to note that our focus will specifically be on the lighting domain, considering the implementation of smart lighting applications in subsequent chapters, which will utilize the proposed semantic interoperability architecture. To achieve this, we design ontologies using OWL to express concepts in smart spaces. An example of an application ontology for a smart space, also applicable for smart lighting applications, is presented in Fig 4.5.

In an ontology, classes and subclasses help organize information and relationships within a domain. They provide a structured framework for representing knowledge, making it easier to reason about and query information related to that domain. Classes are catego-

ries that represent a set of individuals or things in a particular domain. Subclasses are specific categories that are more specialized and specific than the broader classes. The taxonomy within the application ontology describes the following classes and subclasses. It is important to note that these classes and subclasses are based on the necessary concepts and relationships required for semantic interoperability in later chapters. Specifically, the relationships between nodes and *iO*s, node locations, and the interaction states of *iO*s are highlighted as necessary concepts for achieving semantic interoperability in a smart space. The interaction state of an *iO* defines the state used to share semantics in a smart space for producing and consuming information. For example, a lamp can have a possible state: 'brightness level'. Additionally, a sensor can have a sensor value as the interaction state for sharing in a smart space.

The superclass of the ontology graph shown in Fig 4.5 is 'SmartSpace'. This class is further connected to the classes of smart nodes through the predicate 'hasSmartNode'. The smart node classes define their associated *iO*s as subclasses by establishing a relationship using the predicate 'hasiO'. A smart node is also associated with a specific location, defined by a relationship using the predicate 'hasLocation' within the smart space. The location of the node is also specified by a literal value called 'location'. All *iO*s have an interaction state, represented by a literal value called 'state'. These interaction states are used to execute specific scenarios in smart applications. For instance, a sensor node with a *PiO* has a state that contains sensor values. This state is considered the interaction state in the ontology.

It is important to note that we have included only the relationships in the graph that are necessary to explain application scenarios in later chapters. In principle, there can be many associated relationships, as exemplified in Fig 4.4 with the sensor ontology graph. Therefore, we consider this graph as the basic ontology graph for a smart space, based on the definition provided in Chapter 3. In fact, we will establish additional relationships as predicates in subsequent chapters as needed to map smart lighting applications.

We deploy the basic ontology graph at *SBiO* with the help of *MiO* in a smart space. The deployment view of the application ontology in a smart space is shown in Fig. 4.6. All *iO*s are able to represent contexts into RDF triples using a semantic representor. The semantic representor translates contexts into semantics on *iO*s and creates RDF triples to produce or access information. In the case of LSN, the corresponding *GWiO* takes over the task of translating contexts into semantics on behalf of *iO*s of LSNs. We also refer to the *GWiO* as a translator, responsible for translating contexts into semantics (RDF triples) and vise vera.

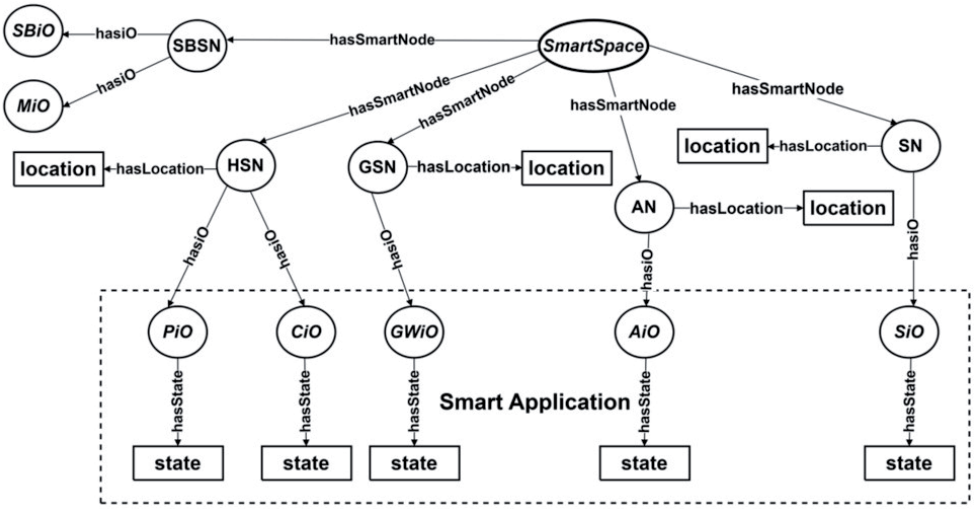


Figure 4.5. An ontology graph for a smart space.

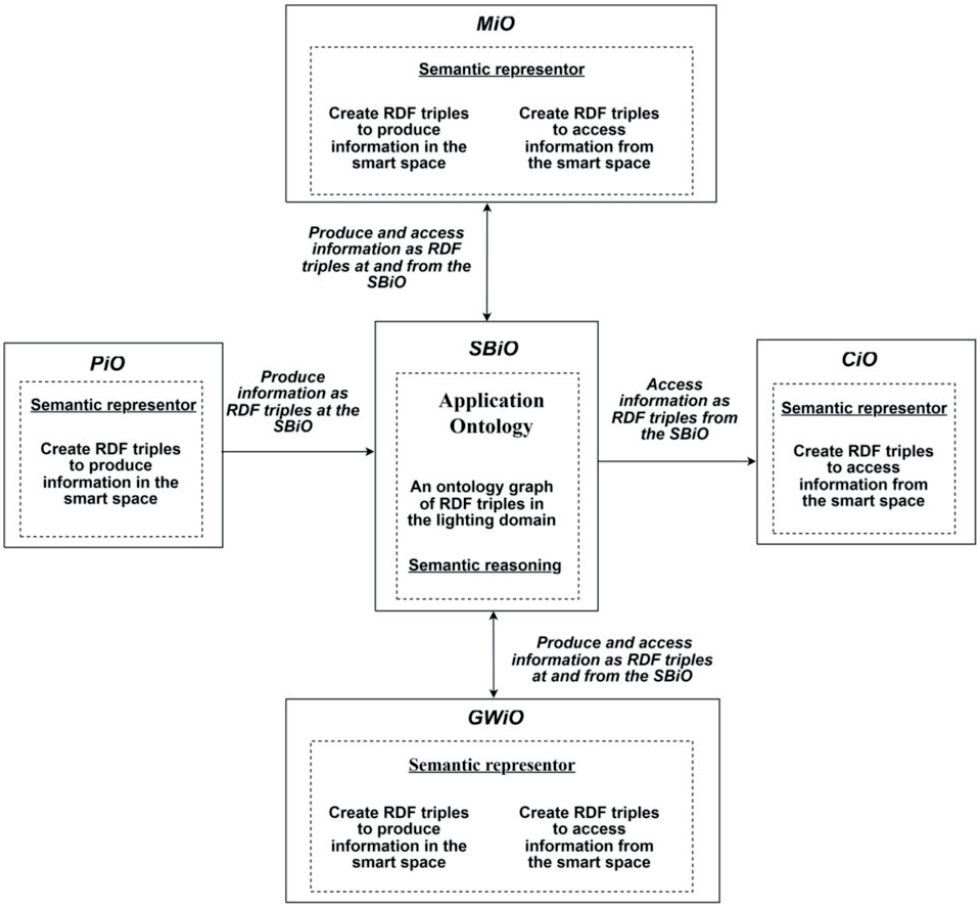
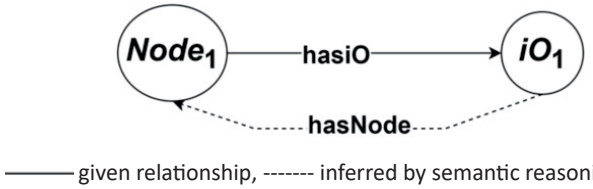


Figure 4.6. Deployment view of application ontologies in a smart space.

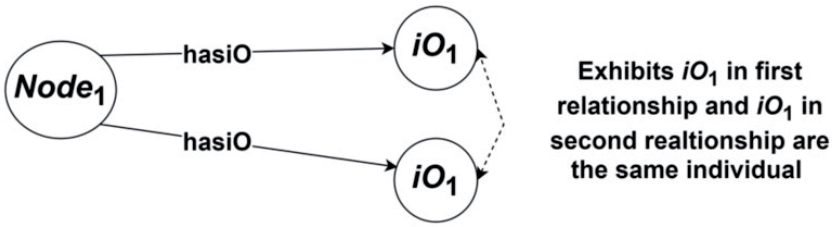
The *iO*s share semantics within a smart space through the *SBiO*, which is supported by semantic reasoning. Semantic reasoning involves RDF matching operations and the application of rules or axioms to existing relationships among concepts in order to infer new relations within an ontology graph. The *SBiO* implements semantic reasoning using semantic reasoners, which are software tools capable of matching RDF triples and deducing additional concepts and relationships from the original descriptions of concepts. OWL provides a comprehensive set of RDF matching operations and predefined axioms to facilitate this process. Some of the key functionalities of a semantic reasoner defined using OWL, are as follows:

1. **Consistency checking**: A reasoner can verify the logical consistency of concepts in an ontology graph, ensuring the absence of contradictions or conflicts. In the context of OWL, logical consistency refers to the ontology's lack of contradictions or conflicts. A consistent ontology ensures the absence of logical errors, such as assertions that violate defined rules or lead to contradictory statements. When inconsistencies are detected during the consistency checking process, the reasoner will identify and report the specific conflicting statements or logical errors within the ontology. This feedback aids ontology developers in identifying and resolving inconsistencies in their ontologies.

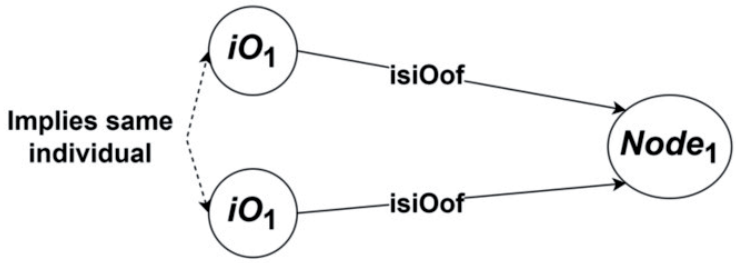
2. **Inference of logical relationships**: A reasoner can infer logical relationships between entities based on defined properties and their characteristics. This encompasses the inference of subclass relationships, property restrictions, and inverse relationships. In OWL, there are several key logical relationships that can be inferred, including:
   - ***Subclass relationships***: A reasoner can infer subclass relationships between classes based on the defined class hierarchy and the properties of the classes. For example, if it is explicitly stated that "A" is a subclass of "B" and "B" is a subclass of "C" the reasoner can infer that "A" is also a subclass of "C".
   - ***Equivalent class relationships***: A reasoner can infer equivalent class relationships, where two or more classes are semantically equivalent. If it is explicitly stated that "A" is equivalent to "B", the reasoner can infer that instances of "A" are also instances of "B" and vice versa.
   - ***Property relationships***: A reasoner can infer relationships between properties based on their characteristics and the defined restrictions. We list property relationship inferences that are commonly integrated with semantic reasoning as follows:
     - **Inverse property**: An inverse property in ontology refers to a relationship between two properties where if one property connects entity A to entity B, the inverse property connects entity B to entity A. For instance, 'hasNode' is inverse of 'hasiO'.



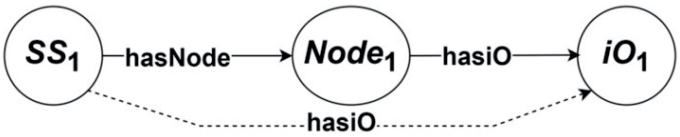
- **Functional property**: A functional property in ontology associates each instance of a class with at most one unique value of another class. $Node_1$ has an *iO* called iO_1. If there is another instance of the same relation, iO_1 exhibits the existence of the same individual.



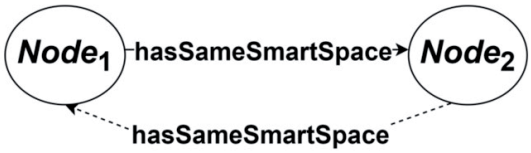
- **Inverse functional property**: An inverse functional property in ontology associates each value of a class with at most one unique instance of another class. If iO_1 is the *iO* of $Node_1$ and another instance of the same relation implies that iO_1 is the same individual.



- **Transitive property**: A transitive property in ontology refers to a relation that applies not only to a given element but also to all elements related to that element. If SS_1 has a node called $Node_1$, and $Node_1$ has an *iO* named iO_1, then it can be inferred that SS_1 has iO_1.



- **Symmetric property**: A symmetric property in ontology refers to a relation between two elements where if the relation holds between the first element and the second, it also holds between the second element and the first. *Node*$_1$ and *Node*$_2$ have the property that they have the same smart space.



- **Asymmetric property**: An asymmetric property in ontology is a relation between two elements where if the relation holds between the first element and the second, it does not hold in the reverse direction.

· ***Property domain and range***: A reasoner can infer the domain and range of a property based on the assertions and restrictions defined in the ontology. For example, if it is explicitly stated that the property "hasChild" is defined on the class "Person", the reasoner can infer that if an individual has a child, that individual must be a person.

· ***Disjointness relationships***: A reasoner can infer disjointness relationships between classes, indicating that they cannot have any common instances. If it is explicitly stated that "A" and "B" are disjoint classes, the reasoner can infer that no instance can be both of "A" and "B".

3. **RDF matching mechanism**: A reasoner incorporates an RDF triple matching mechanism. This mechanism involves matching RDF triples against specific criteria or patterns. SPARQL (SPARQL Protocol and RDF Query Language) is used to express these patterns or queries. RDF triple matching compares the rdf_subject, rdf_predicate, and rdf_object components of triples with the corresponding components in the patterns or queries. A match occurs when all components of a triple align with the pattern or query. Various operations are involved in the matching process, including Equality Comparison, Wildcard Matching, Range Matching, and Pattern Matching. Here is a brief explanation of these matching operations:

- ***Equality comparison***: Equality comparisons in RDF triple matching involve comparing values within the rdf_subject, rdf_predicate, or rdf_object components of RDF triples. These comparisons are used to determine if the values match specific criteria or conditions. When performing equality comparisons, URIs or literals are compared to check for equality. For example, when checking if the rdf_subject of a triple matches a specific URI, an equality comparison is made between the rdf_subject value and the URI being compared. If the URIs are the same, the comparison evaluates as true, indicating a match. Similarly, equality comparisons can be applied to rdf_predicates and rdf_objects within RDF triples. For rdf_predicates, URIs are compared to check for a specific URI match, while for rdf_objects, literals are compared to determine if they match a particular value or condition.
- ***Wildcard matching***: Wildcard matching involves using wildcard symbols or variables to represent unknown or unspecified components of RDF triples. Wildcards enable flexible pattern matching, where the specific value of an rdf_subject, rdf_predicate, or rdf_object component is either unknown or unnecessary to specify. The most commonly used wildcard symbol is the question mark "?" (also known as a variable or placeholder). When a question mark is used as a component in an RDF triple pattern or query, it matches any value for that component during the matching process. For instance, when the question mark "?" is used as a wildcard for the rdf_subject component, it matches any rdf_subject value when the pattern is matched against actual RDF triples. As long as the rdf_predicate and rdf_object components match the specified criteria regardless of any rdf_subject value, will be considered an RDF triple match.
- ***Range matching***: Range matching involves comparing values in the rdf_object component of triples against specified ranges or conditions. It is used to determine if the rdf_object value falls within a specific range or satisfies certain criteria. The rdf_object component can contain literal values such as strings, numbers, dates, etc. Range matching enables the filtering or selection of RDF triples based on the value range of the rdf_object component. For example, range matching can be applied to the rdf_object values when matching the pattern against actual RDF triples. Let us consider the scenario where we want to match only those triples where the rdf_object value is a number between 10 and 20. In this case, the range matching operation compares the numeric value of the rdf_object against the specified range. If the value falls within the range, the triple is considered a match. Range matching can be extended to other data types as well, such as dates or strings. For instance, range matching on dates could involve checking if the rdf_object value is within a specific date range, while range matching on strings could involve performing string comparisons.
- ***Pattern matching***: Pattern matching refers to the process of matching the structure and relationships between the rdf_subject, rdf_predicate, and rdf_object compo-

nents of RDF triples to a given pattern or template. A pattern refers to a specific arrangement or structure that is searched for within a dataset. Patterns are defined by a set of rules or criteria that specify what is being sought. For example, in a text search, a pattern might be a sequence of characters that is desired to be found within a larger body of text. Let us consider a list of email addresses and the objective is to extract all the email addresses ending in ".com". This scenario exemplifies the application of pattern queries. Wildcard matching is a specific type of pattern matching that employs wildcard characters to represent unknown parts of a string. Pattern matching, on the other hand, is a broader concept that encompasses finding specific patterns or structures within data using defined rules or patterns.

In this subsection, we discussed the fundamental concepts of semantics and reasoning. Ontology tools such as OWL and OWL2 come equipped with features like consistency checking, inference of logical relationships, and a specific matching operation. In this thesis, we use OWL2 for semantic reasoning with a wildcard matching operation at *SBiO* in the proposed semantic architecture to retrieve query and subscription results in the smart space. Let us consider an example query from an *iO* where the query is ("$Node_1$", "hasiO", "?") to the *SBiO*. We assume that the *SBiO* contains the following RDF triples: ("$Node_1$", "hasiO", "PiO_1") and ("$Node_1$", "hasiO", "CiO_1"). Matching RDF triples involves determining if two triples (the querying triple and a triple at *SBiO*) are equivalent or compatible based on their rdf_subject, rdf_predicate, and rdf_object components. To begin the matching operation, we first start matching the rdf_subject of two triples. If the rdf_subjects are the same, it indicates a match. The comparison is based on the URIs of the rdf_subjects. Next, we compare the rdf_predicates of the two triples. If the rdf_predicates are the same, it suggests a match. The comparison is based on the URIs of the predicates. Finally, we compare the rdf_objects of the two triples. The specific comparison depends on the type of rdf_object.

- If the rdf_objects are URIs, we compare their URIs. If the URIs are the same, it indicates a match.
- If the rdf_objects are literals, we compare their literals. If the values are the same, it suggests a match.

In this example, the rdf_object is a placeholder. Therefore, the matching operation yields two possible triples as results: ("$Node_1$", "hasiO", "PiO_1") and ("$Node_1$", "hasiO", "CiO_1").

In Section 4.2.2, we will discuss the format of query and subscription transactions along with other SSAP transactions. Additionally, the literature offers a wide range of other semantic reasoners with similar functions and operations; a few of them are listed in Table 4.1.

Table 4.1 Semantic Reasoners.

Items	Description
Pellet [3]	Pellet is an open-source reasoner based on Java and OWL 2. It provides functionalities that explain inferences and answer SPARQL queries; check the consistency of ontologies and compute a classification hierarchy.
Apache Jena [4]	Apache Jena is an open-source semantic web framework for Java. The RDF graphs are represented in an abstract model and are queried through SPARQL.
Prova[5]	Prova is an open-source programming language and rule-based scripting middleware. It supports java-based inference rules and provides users an ontology development platform.
Flora-2[6]	Flora-2 is an open-source semantic rule-based reasoner and supports for knowledge representation and reasoning.
Gandalf[7]	Gandalf is an open-source decision engine based on PHP and is used for big-data analysis.
KAON2[8]	KAON2 is an inference engine for answering conjunctive queries using SPARQL.
Apache Marmotta[9]	Apache Marmotta is a rule-based reasoner and support for SPARQL queries
SPIN[10]	SPIN (SPARQL Inferencing Notation) supports advanced properties and rules for semantic reasoning. SPIN is a W3C[11] Member Submission created and manipulated by TopQuadrant[12]. The SPIN rules are expressed in SPARQL that allows creating new individuals and finding individuals using CONSTRUCT and WHERE keywords, respectively. The SPIN rules can run directly on RDF databases.
HermiT[13]	HermiT is an open-source reasoner for determining consistency and identifying subsumption relationships between classes.

4.2.2 SSAP transactions

The *iO*s must support a standard protocol to modify and access semantics at the *SBiO*. Therefore, we use SSAP transactions for exchanging semantics by *iO*s with the *SBiO* in the proposed semantic interoperability architecture. These transactions include JOIN, LEAVE, INSERT, REMOVE, UPDATE, SUBSCRIBE, UNSUBSCRIBE and QUERY. When an *iO* sends a transaction request to the *SBiO*, it receives a transaction confirmation. However, in the case of QUERY, SUBSCRIBE, and UNSUBSCRIBE transactions, the *iO* also receives a separate message containing the result of the subscription or query, in addition to the confirmation message, as shown in Fig. 4.7. All transactions must be executed atomically by the following two aspects: i) Ensuring that the operations either complete successfully or are completely rolled back. We have acknowledgment systems in transactions to

3 https://www.w3.org/2001/sw/wiki/Pellet.
4 https://jena.apache.org/.
5 https://prova.ws/.
6 http://flora.sourceforge.net/.
7 https://gndf.io/.
8 http://kaon2.semanticweb.org/.
9 http://marmotta.apache.org/ind.
10 http://www.spinrdf.org.
11 https://www.w3.org/.
12 https://www.topquadrant.com/.
13 http://www.hermit-reasoner.com/.

ensure this. ii) Ensuring that no other transaction operates on the triple that is being used by another transaction at the *SBiO*. This is similar to a binary semaphore in an operating system, which is used to control access to shared resources. The transaction involving a specific RDF triple receives a signal when it needs to be executed; otherwise, it waits until the current transaction on the triple finishes. This ensures that only one transaction can be executed at a time on a specific triple at the *SBiO*. This is particularly important for queries, which may require multiple accesses to the underlying data structures within the *SBiO*. Therefore, each transaction involves only two entities: an *iO* (a *PiO* or a *CiO* or a *GWiO*) and an *SBiO*.

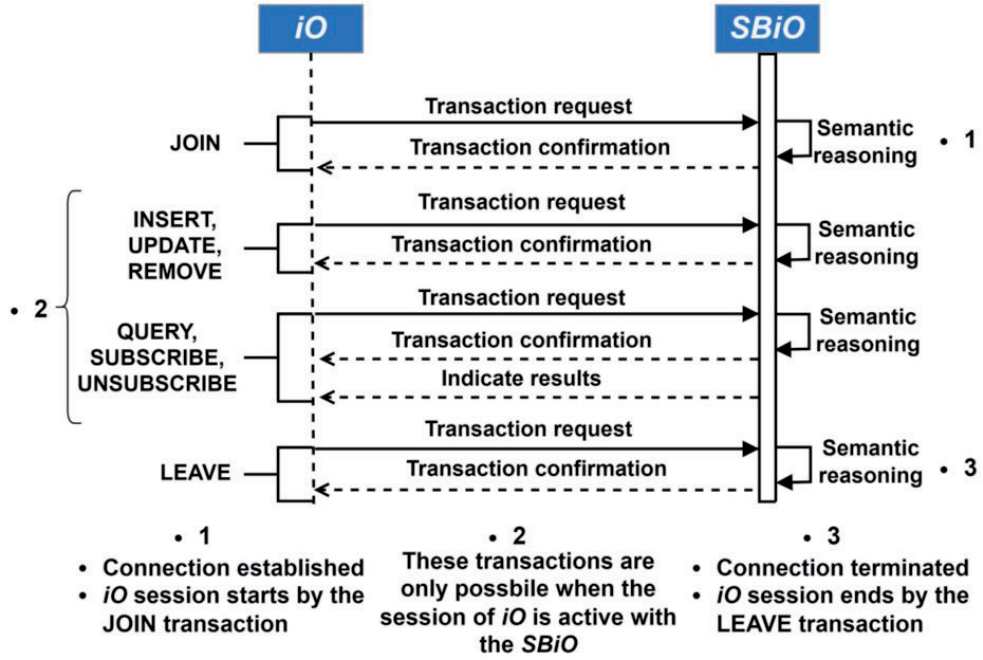


Figure 4.7. The basic operations of all transactions by *iO*s at the *SBiO*.

In more detail, the rules and functionalities of these transactions are explained as follows:

- JOIN and LEAVE transactions are (eventually) employed by all *iO*s. An *iO* starts a session with the *SBiO* using a JOIN transaction and ends the session by using a LEAVE transaction. The session provides access to the semantics stored at the *SBiO* using other transactions. No further transaction requests are accepted from the connection until a new JOIN transaction is invoked after the LEAVE transaction.
- INSERT, UPDATE and REMOVE transactions are employed by *PiO*s or *GWiO*s
  - By using an INSERT transaction, a *PiO* or *GWiO* inserts RDF triples at the *SBiO*, causing the *SBiO* to generate a specified graph node in the ontology graph.
  - A *PiO* or *GWiO* can remove and update the application ontology graph at the *SBiO* by means of REMOVE and UPDATE transactions, respectively. UPDATE transaction

is an atomic combination of REMOVE and INSERT transactions, where REMOVE is performed first.

- QUERY, SUBSCRIBE and UNSUBSCRIBE transactions are employed by *CiO*s or *GWiO*s.
  - A *CiO* or *GWiO* can query for RDF triples at the *SBiO* by using a QUERY transaction and get results from the *SBiO* in reply.
  - Any *CiO* or *GWiO* in a smart space can subscribe to specific RDF triples, which means placing a persistent query for triples stored at the *SBiO* by using the SUBSCRIBE transaction. In this case, the *CiO* or *GWiO* is notified of changes in the corresponding stored triples. Similarly, an UNSUBSCRIBE transaction terminates the persistent query at the *SBiO*.

*PiO*s or *GWiO*s can modify semantics in the ontology graph at the *SBiO* using SSAP transactions. When a new *PiO* joins a smart space, a specified URI for RDF triples associated with the new *PiO* is generated within the application ontology graph using the INSERT transaction. Afterwards, the new *PiO* can modify RDF triples using the UPDATE transaction.

Smart applications utilize these SSAP transactions within a smart space to realize their intended behavior. The producing and consuming *iO*s employ these transactions with the *SBiO* to execute scenarios in applications. We will now explain these transactions performed by *iO*s to the *SBiO*. Let us consider a first example: a smart space (SS_x) has a smart application ($\mathbb{A}_x$) consisting of two light sensor nodes; n_a with a *PiO* $\boldsymbol{p_a}$, and n_b with a *PiO* $\boldsymbol{p_b}$. Both p_a and p_b have two interaction states to represent the status of light information i.e., 'on' and 'off'. These nodes are connected to the smart space SBSN node (n_x), which itself has two *iO*s: an *SBiO* (sb_x) and an *MiO* (mo_x). The initial design of the basic application ontology graph, created by the smart space developer and stored at $\boldsymbol{sb_x}$ using $\boldsymbol{mo_x}$, is shown in Fig 4.8.

An *iO* has the ability to perform various SSAP transactions at $\boldsymbol{sb_x}$, including adding, removing, updating, querying, and subscribing to any triple. In order to successfully execute these transactions, the *iO* must have knowledge of the basic structure of the information stored in the ontology graph at $\boldsymbol{sb_x}$. For instance, if the *iO* intends to query the rdf_object in a triple, it needs to have information about the rdf_subject and rdf_predicate associated with that triple. Now, let's explain some transaction examples that can be performed by an *iO* at $\boldsymbol{sb_x}$:

(i) Add and remove semantics: To add semantics (RDF triples) to the basic application ontology graph at $\boldsymbol{sb_x}$, we utilize the INSERT transaction through the *PiO*s (p_a and p_b). When the interaction states of *PiO*s are off, the *PiO*s add corresponding off states to the graph. The graph, with the newly added RDF triples, is illustrated in Fig 4.9, where the

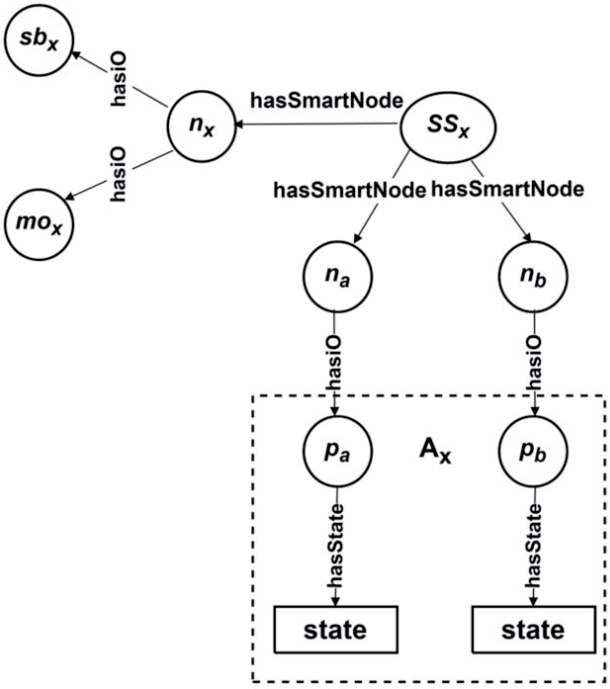


Figure 4.8. The basic application ontology graph at sb_x.

RDF triples are *INSERT*{("p_a ","*hasState*","*OFF*")} and *INSERT*{("p_b ","*hasState*","*OFF*")} by p_a and p_b, respectively.

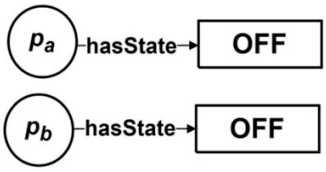


Figure 4.9. The basic application ontology graph of Fig. 4.8 added with RDF triples.

The *PiO*s can also utilize the REMOVE transaction to remove RDF triples from the basic application ontology graph. The graph, after the removal operation, is depicted in Fig. 4.10 and the RDF triples are *REMOVE*{("p_a ","*hasState*","*OFF*")} and *REMOVE*{(p_b ","*hasState*","*OFF*")} by p_a and p_b, respectively.

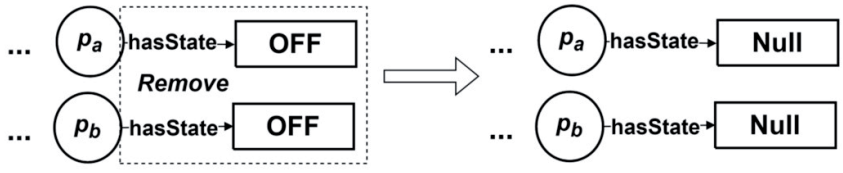


Figure 4.10. The basic application ontology graph with removed RDF triples.

(ii) Update semantics: The *PiO*s have the capability to update RDF triples in the basic ontology graph using the UPDATE transaction. The update operation involves using a wildcard to identify the field that needs to be removed or updated. The UPDATE transaction provides a persistent update operation, where the REMOVE/INSERT part of SPARQL Update is executed only if the WHERE pattern produces a solution. This ensures that the UPDATE transaction does not result in duplicate inserted triples at the *SBiO*, since the REMOVE transaction is applied first. In this way, the UPDATE transaction is guaranteed not to have the same inserted triples at the *SBiO*. The ontology graph in Fig 4.9, with the updated RDF triples, is depicted in Fig 4.11.

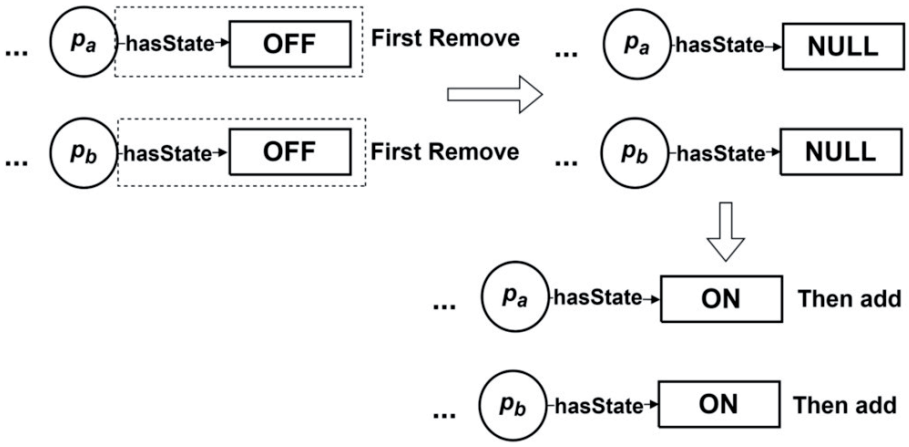


Figure 4.11. The ontology graph of Fig 4.9 with updated RDF triples.

For example:

$$UPDATE\{("p_a","hasState"," ?")|("p_a","hasState","ON")\} \text{ by } p_a$$

$$UPDATE\{("p_b","hasState"," ?")|("p_b","hasState","ON")\} \text{ by } p_b$$

Now let us discuss accessing semantics from $\boldsymbol{sb_x}$ using *CiO*s. In a particular application scenario, a user may want to retrieve information from the smart space SS_x. To achieve this, the user needs to connect to $\boldsymbol{sb_x}$ through a *CiO*. Let's consider an example where the user's smartphone has an interface with a *CiO* ($\boldsymbol{c_{s\_phone}}$) and establishes a connection with $\boldsymbol{sb_x}$.

(iii)Query semantics: $\boldsymbol{c_{s\_phone}}$ can query RDF triples at $\boldsymbol{sb_x}$ using the QUERY transaction after joining SS_x. The QUERY transaction allows for the use of wildcard entries in the triples for which we need answers to the query. When $\boldsymbol{c_{s\_phone}}$ initiates a query, $\boldsymbol{sb_x}$ matches the query using the pattern matching operation and retrieves the corresponding URIs

from the ontology graph. It then lists all possible RDF triples in the query answer. Here are a few examples:

Query: ***QUERY***{("SS_x","?","?")}
Results: {("SS_x","*hasSmartNode*","n_x"),("SS_x","*hasSmartNode*","n_a"),
("SS_x","*hasSmartNode*","n_b") }

Query: ***QUERY*** {("p_a","*hasState*","?"),("p_b","*hasState*","?")}
Results: {("p_a","*hasState*","*ON*"),("p_b","*hasState*","*ON*")}

(iv)Subscribe semantics: $\boldsymbol{c_{s\_phone}}$ can subscribe to RDF triples at $\boldsymbol{sb_x}$ using the SUBSCRIBE transaction. Similar to the QUERY transaction, we use wildcards in the RDF triples for which we want to establish the subscription. $\boldsymbol{sb_x}$ matches the wildcard URI using the pattern matching operation and retrieves the corresponding RDF triples from the ontology graph. It then lists all possible RDF triples in the subscription notification. Once subscribed, $\boldsymbol{c_{s\_phone}}$ will receive notifications on any updates to the subscribed RDF triples until the subscription is terminated using the UNSUBSCRIBE transaction.

Similar to the JOIN and LEAVE transactions, all other transactions also receive a confirmation message (*cnf*). When a *CiO* successfully performs a QUERY, SUBSCRIBE, or UNSUBSCRIBE transaction, it receives the notification (a list of all possible RDF triples) of subscribed triples. In the case of SUBSCRIBE transaction, a *CiO* receives the notification of the subscribed triples at the *SBiO*, while the *CiO* receives the notification as the list of last updated triples (for the subscribed triples at the *SBiO*) before terminating the subscription using the UNSUBSCRIBE transaction. When a *CiO* performs a QUERY transaction, it receives the notification as the query results from the *SBiO*.

4.3 Semantic Interactions in a Smart Application

In this section, we discuss the semantic interactions among *iO*s within a smart application in a smart space. To denote these interactions, we assign labels to each transaction as follows: JOIN (T_j), LEAVE (T_l), INSERT (T_i), UPDATE (T_u), REMOVE (T_r), QUERY (T_q) SUBSCRIBE (T_s) and UNSUBSCRIBE (T_{us}) transactions. The format of semantic interactions, which involves the exchange of RDF triples using specific transactions, between an *iO* and the *SBiO*, is depicted in Fig 4.12. In this thesis, we adopt this format due to its simplicity in expressing semantic exchanges in the interaction diagrams of smart spaces. This format contains the transaction type, message type and the message. Transaction types are T_j, T_l, T_i, T_u, T_r, T_q, *Ts* and T_{us}, whereas the message types are 'request' or 'result'. The message includes RDF triples. For instance, if we have a triple like ("Lamp", "hasState", "ON"),

the corresponding message would be denoted as σ: ("Lamp", "hasState", "ON"), where σ represents a triple. However, it is important to note that JOIN and LEAVE transactions differ in their content. Unlike other transactions, they do not contain RDF triples; instead, their message consists of the ID of the joining *iO*.

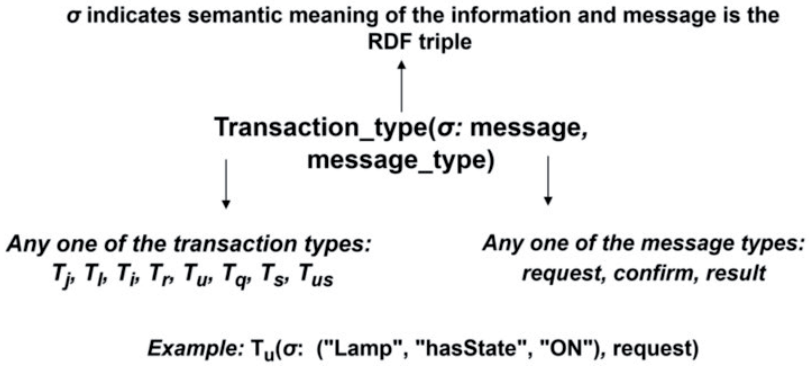


Figure 4.12. The format of sematic interactions between an *iO* and the *SBiO*

As per the rules outlined in the previous section, T_j and T_l transactions for semantic interactions are performed by all *PiO*s, *CiO*s and *GWiO*s. *PiO*s exclusively execute the T_i, T_u and T_r transactions, while *CiO*s handle the T_q, T_s and T_{us} transactions. It is important to note that if both types of *iO*s (*PiO* and *CiO*) are deployed on the same smart node, then that node can perform all semantic transactions. For instance, a *GWiO* possesses the functionalities of both *PiO* and *CiO* types, enabling it to utilize all transactions.

Note: The INSERT and REMOVE transactions involve the atomic insertion or removal of RDF triples from the *SBiO*. On the other hand, the UPDATE transaction utilizes two lists: one for deletion and another for insertion. This approach helps resolve conflicts that may arise when attempting to remove and insert the same RDF triple content. In the case of the QUERY and SUBSCRIBE transactions, they can comprise a set of triples where one or more elements of a triple may contain a wildcard entry. Importantly, these operations are assured to be executed in the same order as they were performed at the *SBiO*. For operations performed by parallel or distributed *iO*s, the *iO*s must join a particular namespace at the *SBiO*. This ensures that the *SBiO* does not process any later operation before processing the earlier ones, thereby resolving potential conflicts.

4.3.1 Semantic interactions by *PiO*s and *CiO*s with an *SBiO*

We will now explain semantic interactions between *PiO*s and *CiO*s through an *SBiO* in a smart space. Consider a second example of a smart space (SS_{sl}) that includes a basic smart lighting application (A_{sl}). The objective of this application is to turn a luminary 'on' when a user presence is detected in a room. In the smart space SS_{sl}, we consider two nodes: a presence sensor node (n_{p_s}) and a luminary node with an actuator (n_{l_a}). The presence

sensor node n_{p_s} has a *PiO* ($\boldsymbol{p_{p\_s}}$) because it needs to produce information about the user presence in the smart space SS_{sl}. $\boldsymbol{p_{p\_s}}$ implements the application logic of event-based detection of the user presence in the room, with the interaction states YES and NO (where YES indicates user presence and NO indicates the absence of the user in the room). On the other hand, the node n_{l_a} has a *CiO* ($\boldsymbol{c_{l\_a}}$) because it needs to consume information of the user presence from the smart space SS_{sl}. The application logic at $\boldsymbol{c_{l\_a}}$ turns on the luminary when the user presence information is received, where the interaction states are on and off (on means the luminary is being turned on and off means it is off). Thus, the objective of the application $\mathbb{A}_{Sl}$ is to respond to the event of user presence generated by $\boldsymbol{p_{p\_s}}$ and perform the associated action of turning the luminary on performed at $\boldsymbol{c_{l\_a}}$ in.

We consider an SBSN (n_{sl}) in the smart space SS_{sl}, with an *SBiO* ($\boldsymbol{sb_{sl}}$) and an *MiO* ($\boldsymbol{mo_{sl}}$). The nodes of the application $\mathbb{A}_{Sl}$ (i.e., n_{p_s} and n_{l_a}) are attached to n_{sl} using TCP/IP and exchange semantics using SSAP transactions. We develop a basic application ontology graph (O_graph_{sl}) for the smart space SS_{sl} that is shown in Fig 4.13 and deploy at $\boldsymbol{sb_{sl}}$ using $\boldsymbol{mo_{sl}}$. The graph O_graph_{sl} includes the description of nodes, deployment of *iO*s and interaction states of *iO*s in the smart space SS_{sl}, and it includes the following RDF triples.

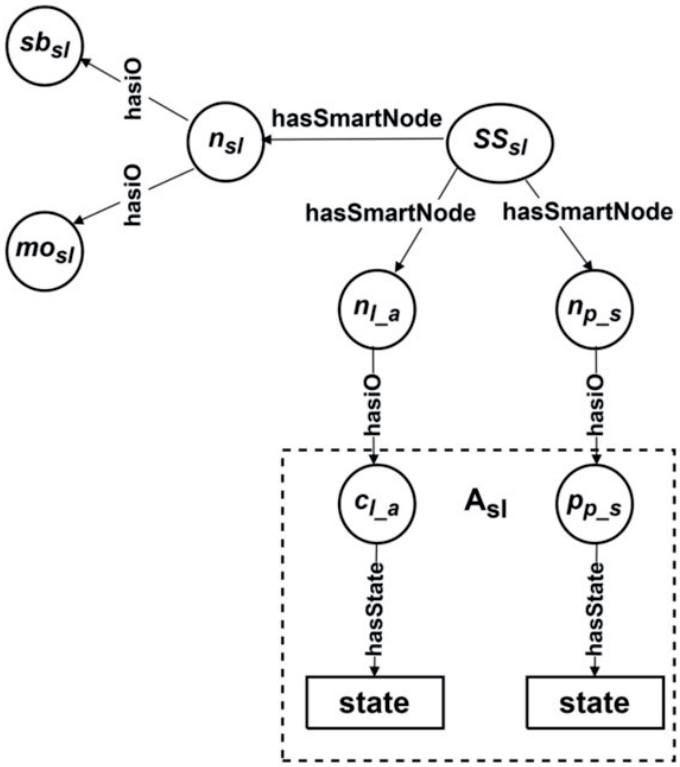


Figure 4.13. The ontology graph O_graph_{sl} inserted by $\boldsymbol{mo_{sl}}$ at $\boldsymbol{sb_{sl}}$.

$$\{, (\text{"}SS_{sl}\text{"}, \text{"}hasSmartNode\text{"}, \text{"}n_{sl}\text{"}), (\text{"}\mathbb{A}_{sl}\text{"}, \text{"}hasSmartNode\text{"}, \text{"}n_{p\_s}\text{"}), (\text{"}\mathbb{A}_{sl}\text{"}, \text{"}hasSmartNode\text{"}, \text{"}n_{l\_a}\text{"}),$$
$$(\text{"}n_{p\_s}\text{"}, \text{"}hasiO\text{"}, \text{"}p_{p\_s}\text{"}), (\text{"}n_{l\_a}\text{"}, \text{"}hasiO\text{"}, \text{"}c_{l\_a}\text{"}), (\text{"}n_{sl}\text{"}, \text{"}hasiO\text{"}, \text{"}sb_{sl}\text{"}), (\text{"}n_{sl}\text{"}, \text{"}hasiO\text{"}, \text{"}mo_{sl}\text{"}),$$
$$(\text{"}p_{p\_s}\text{"}, \text{"}hasState\text{"}, \text{"}state\text{"}), (\text{"}c_{l\_s}\text{"}, \text{"}hasState\text{"}, \text{"}state\text{"})\}$$

The deployment view of the smart space SS_{sl} is shown in Fig 4.14, and the semantic interactions between $\boldsymbol{p_{p\_s}}$ and $\boldsymbol{c_{l\_a}}$ through $\boldsymbol{sb_{sl}}$ are shown in Fig 4.15.

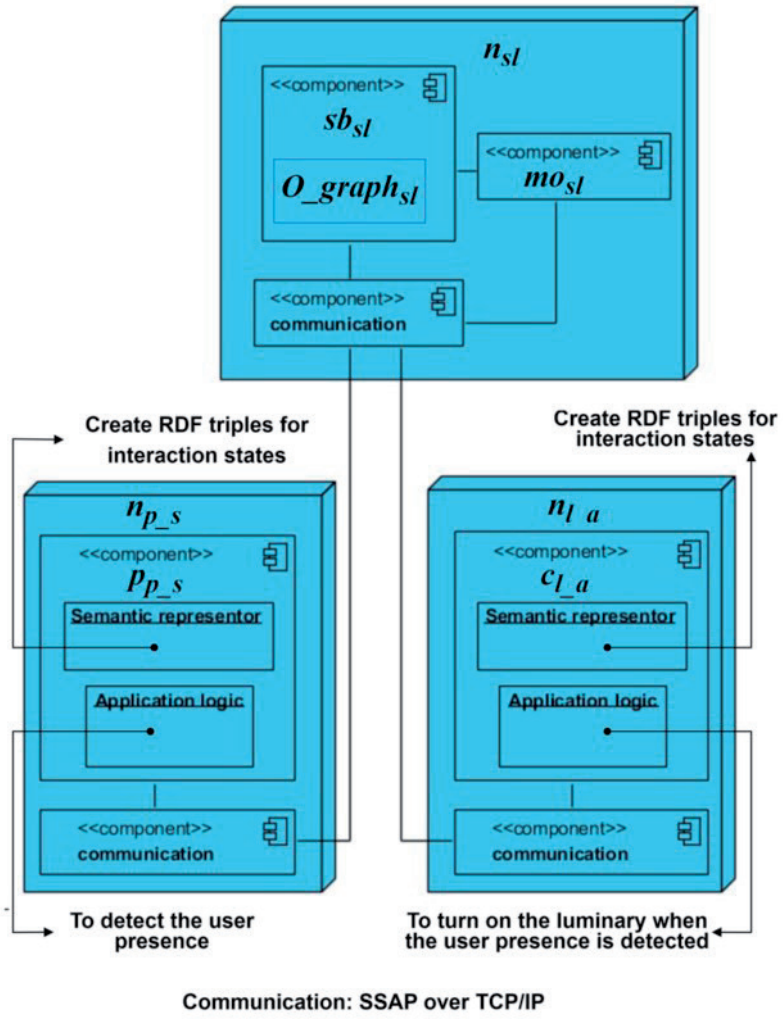


Figure 4.14. Deployment view of the smart space SS_{sl}.

In Fig 4.15 (a), we explain the following steps:

1. $\boldsymbol{p}_{p\_s}$ joins at $\boldsymbol{sb}_{sl}$
2. Joining of $\boldsymbol{p}_{p\_s}$ is confirmed after the session of $\boldsymbol{p}_{p\_s}$ starts at $\boldsymbol{sb}_{sl}$
3. $\boldsymbol{c}_{l\_a}$ joins at $\boldsymbol{sb}_{sl}$
4. Joining of $\boldsymbol{c}_{l\_a}$ is confirmed after the session of $\boldsymbol{c}_{l\_a}$ starts at $\boldsymbol{sb}_{sl}$
5. $\boldsymbol{c}_{l\_a}$ subscribes at $\boldsymbol{sb}_{sl}$ to the information of user presence in the room using the wildcard entry in the triple
6. $\boldsymbol{sb}_{sl}$ activates the subscription of $\boldsymbol{c}_{l\_a}$
7. $\boldsymbol{c}_{l\_a}$ receives the subscription confirmation
8. No user is present in the room
9. $\boldsymbol{p}_{p\_s}$ updates its state 'NO' at $\boldsymbol{sb}_{sl}$
10. The transaction by $\boldsymbol{p}_{p\_s}$ updates the triple state value to 'NO' at O_graph_{sl} (by removing any stored state value and inserting 'NO')
11. $\boldsymbol{p}_{p\_s}$ receives confirmation of the updated triple state 'NO' by $\boldsymbol{sb}_{sl}$
12. $\boldsymbol{sb}_{sl}$ notices the subscription result to be communicated to $\boldsymbol{c}_{l\_a}$ and notifies to the session of $\boldsymbol{c}_{l\_a}$) using internal D-Bus.
13. $\boldsymbol{c}_{l\_a}$ receives the results of subscription, i.e., state 'NO'
14. Luminary is being 'off' because of the user presence is not detected.

In Fig 4.15 (b), we explain the following steps of the example:

15. User presence is detected in the room
16. $\boldsymbol{p}_{p\_s}$ updates the triple of the state 'YES' at $\boldsymbol{sb}_{sl}$
17. The transaction by $\boldsymbol{p}_{p\_s}$ updates the triple state value to 'YES' at O_graph_{sl} (by removing any stored state value and inserting 'YES')
18. $\boldsymbol{p}_{p\_s}$ receives confirmation of the update
19. $\boldsymbol{sb}_{sl}$ notices the subscription result to be communicated to $\boldsymbol{c}_{l\_a}$ and notifies to the session of $\boldsymbol{c}_{l\_a}$ using internal D-Bus.
20. $\boldsymbol{c}_{l\_a}$ receives the results of subscription, i.e., state 'YES'
21. Luminary is being 'off' because of the user presence is detected.

Figure 4.15 describes the execution of application A_{Sl} in the smart space SS_{sl}. The luminary automatically adjusts its state based on the presence of the user. We observed that it is necessary to establish a user presence subscription from $\boldsymbol{c}_{l\_a}$ to $\boldsymbol{sb}_{sl}$. As a result, the luminary automatically turns to on when the event of user presence is updated at $\boldsymbol{sb}_{sl}$ by $\boldsymbol{p}_{p\_s}$. The user's presence information will be received at $\boldsymbol{c}_{l\_a}$ until the subscription is terminated by $\boldsymbol{sb}_{sl}$ using the transaction T_{us}. Thus, both $\boldsymbol{p}_{p\_s}$ and $\boldsymbol{c}_{l\_a}$ exchange semantics through $\boldsymbol{sb}_{sl}$ to achieve the objective of application A_{Sl}.

Note: $\boldsymbol{p}_{p\_s}$ can use the REMOVE transaction to remove the inserted or updated triples from $\boldsymbol{sb}_{sl}$ if they are no longer needed. If $\boldsymbol{p}_{p\_s}$ leaves without removing these triples, they will remain available at $\boldsymbol{sb}_{sl}$ without any further use. In such cases, it is the responsibility of $\boldsymbol{mo}_{sl}$ to manage the removal of these triples from $\boldsymbol{sb}_{sl}$ if they are no longer required. This is possible when $\boldsymbol{mo}_{sl}$ subscribes to the information of an *iO* who has left the smart space and further checks for the related triples updated by the *iO* at $\boldsymbol{sb}_{sl}$.

4.3.2 *GWiO* semantic interactions with an *SBiO*

SSAP was developed by SOFIA to work with the internet protocol for HSNs and is not practical for LSNs. Therefore, we propose a gateway approach for LSNs, utilizing a GSN to integrate them into a smart space. The GSN collects abstract information in the form of states from *SiO*s and *AiO*s. The *GWiO* deployed on the GSN translates received states into RDF triples. For instance, in smart lighting applications, we consider the states of

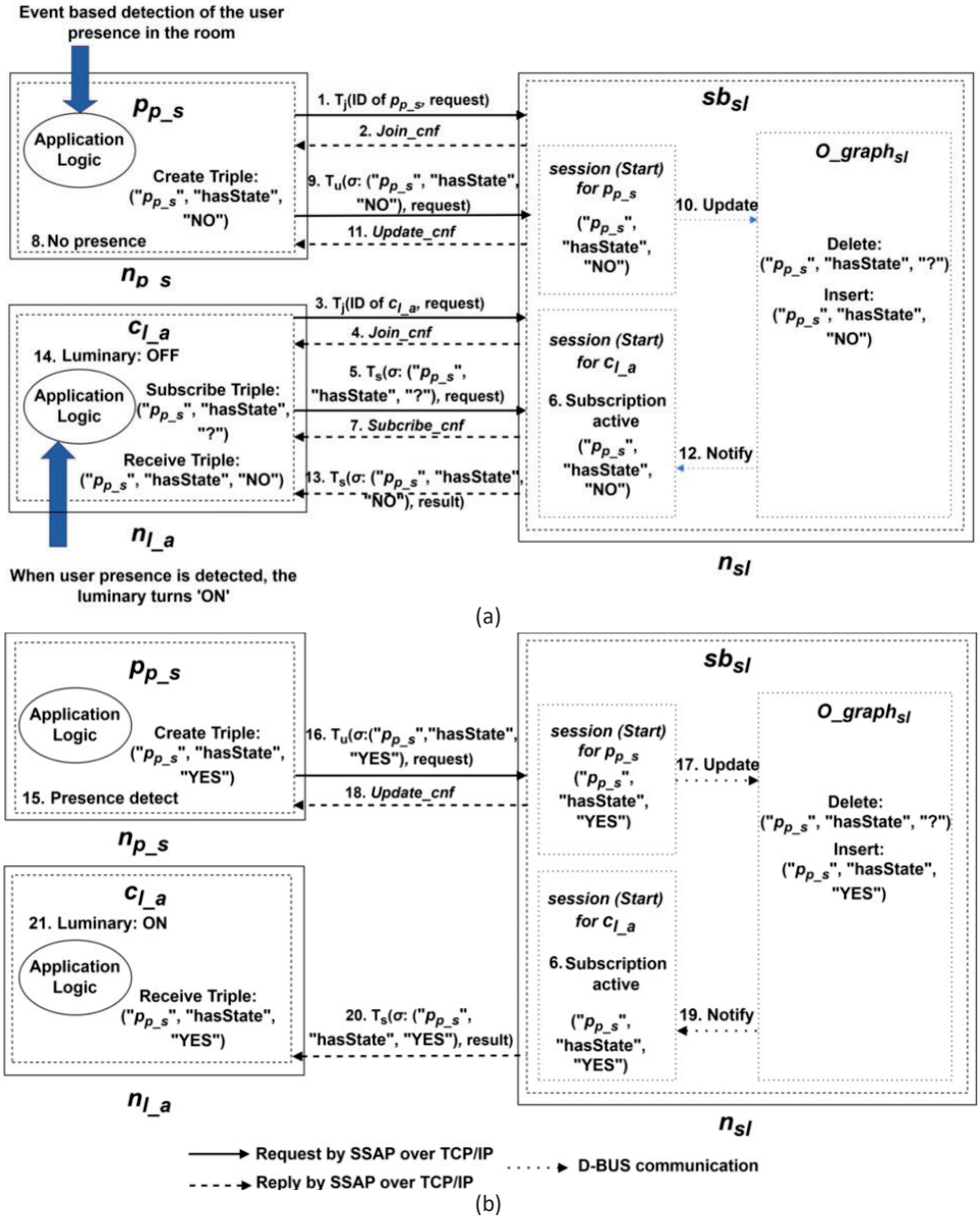


Figure 4.15 (a) and (b). Execution of the smart application A_{sl}.

*SiO*s as illumination readings within an activity space, and the states of *AiO*s as luminary brightness values represented as a percentage (0% for *off* and 100% for *on*). The *GWiO* functions as both a *PiO* and a *CiO*, enabling it to update or modify the semantics of states in RDF triples at the *SBiO* using transactions similar to those used by *PiO*s, as well as access semantics from the *SBiO* using transactions similar to those used by *CiO*s.

Generally, LSNs communicate with the GSN to update states using low-power communication protocols such as Zigbee over IEEE 802.15.4. Regardless of the communication protocol used by LSNs, they can share semantics in a smart space through the GSN. The translation of states into RDF triples by the *GWiO* is crucial for their participation in a smart space. Therefore, we do not specify a particular communication protocol to explain transactions between LSNs and the GSN in this section. It depends on the hardware or software used by LSNs and the GSN, as well as on a specific application in a smart space. In Chapter 5, we will consider the implementation of a specific communication protocol (Zigbee, based on the IEEE 802.15.4 standard). Thus, we present an abstract format of transactions that includes the transaction type with a message with state of *iO*s as shown in Fig 4.16. We introduce a transaction, referred to as an update transaction (T_{lsn}), which is accompanied by a message. The message has two types, i.e., *sense_value*() and *actuate_value*(). The *sense_value*() message facilitates communication between *SiO*s (deployed on SNs) and the *GWiO* (deployed on GSN) to report the states of *SiO*s, from *SiO*s to the *GWiO*. The *actuate_value*() message enables communication between the *GWiO* and *AiO*s to update states at *AiO*s based on actuation commands from the *GWiO*.

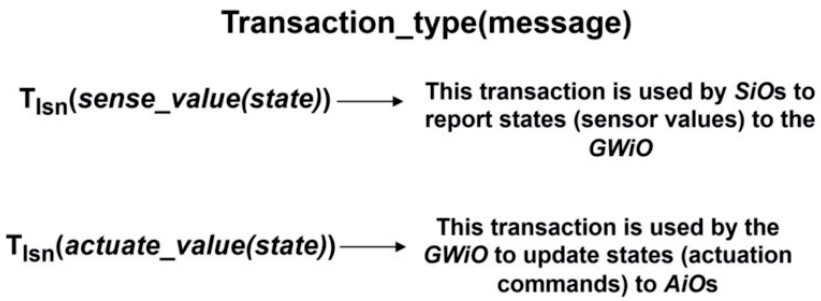


Figure 4.16. An abstract format of transactions that contain the transaction type with a message for LSNs.

We explain the semantic interactions between the *GWiO* and the *SBiO* in a smart space with the following example: "The user presence is detected, and the luminary in the room turns 'on' (Objective A). Furthermore, the user desires to obtain precise light information in the room (Objective B)". We implement an example that incorporates both LSNs and HSNs to illustrate the semantic interactions between them in a smart space. Let us consider the third example of the smart space (SS_{sr}) with a smart room application (A_{sr}) having the following nodes and *iO*s:

$SS_{sr}.N = \{n_{l\_s}, n_{l\_a}, n_{h\_p}, n_{h\_c}, n_{gw\_sr}, n_{sb\_sr}\}$ and $SS_{sr}.iO = \{s_{l\_s}, a_{l\_a}, p_{h\_p}, c_{h\_c}, gw_{sr}, sb_{sr}, mo_{sr}\}$;

Where,

- n_{l_s} is the LSN to measure light sensor readings and deployed with $\boldsymbol{s_{l\_s}}$, i.e., *SiO*. The state of $\boldsymbol{s_{l\_s}}$ is an illumination value of light in the room.

- n_{l_a} is the LSN luminary deployed with $\boldsymbol{a_{l\_a}}$, i.e., *AiO*. The states of $\boldsymbol{a_{l\_a}}$ are 'on' (being the luminary on) and 'off' (being the luminary off)
- n_{h_p} is the HSN to detect presence of a user and deployed with $\boldsymbol{p_{h\_p}}$, i.e., *PiO*. The interaction states of $\boldsymbol{p_{h\_p}}$ are 'YES' (the user presence detected) and 'on' (the user is not present).
- n_{h_c} is the HSN that can make queries in the smart space and deployed with $\boldsymbol{c_{h\_c}}$, i.e., *CiO*, for example a smartphone of the user. The interaction states of $\boldsymbol{c_{h\_c}}$ will depend on user's queries to $\boldsymbol{sb_{sr}}$.
- n_{gw_sr} is the GSN deployed with $\boldsymbol{gw_{sr}}$, i.e. *GWiO* and further connects with n_{l_s} and n_{l_a}.
- Finally, n_{sb_sr} is the SBSN of the smart space deployed with $\boldsymbol{sb_{sr}}$, i.e., *SBiO* and $\boldsymbol{mo_{sr}}$, i.e., *MiO*.

The smart nodes n_{h_p}, n_{h_c} and n_{gw_sr} are connected to n_{sr} using TCP/IP and exchange semantics using SSAP transactions. The smart nodes n_{l_s} and n_{l_a} are connected to n_{sr} using low-power communication protocols such as Zigbee over IEEE 802.15.4. We develop a basic application ontology graph (O_graph_{sr}) for the smart space SS_{sr} that is shown in Fig 4.17 and store at $\boldsymbol{sb_{sr}}$ using $\boldsymbol{mo_{sr}}$. The graph O_graph_{sr} includes the following RDF triples:

$$\{("SS_{sr}","hasSmartNode","n_{sr}"),("\mathbb{A}_{sr}","hasSmartNode","n_{h\_p}"),("\mathbb{A}_{sr}","hasSmartNode","n_{h\_c}"),,$$
$$("\mathbb{A}_{sr}","hasSmartNode","n_{l\_s}"),("\mathbb{A}_{sr}","hasSmartNode","n_{l\_a}"),("\mathbb{A}_{sr}","hasSmartNode","n_{gw\_sr}"),$$
$$("n_{h\_p}","hasiO","p_{h\_p}"),("n_{h\_c}","hasiO","c_{h\_c}"),("n_{l\_s}","hasiO","s_{l\_s}"),("n_{l\_a}","hasiO","a_{l\_a}")$$
$$("n_{sr}","hasiO","sb_{sr}")("n_{sr}","hasiO","mo_{sr}"),$$
$$("n_{gw\_sr}","hasiO","gw_{sr}"),("p_{h\_p}","hasState","state"),("a_{l\_a}","hasState","state"),$$
$$("s_{l\_s}","hasState","state"),("c_{h\_c}","hasState","state")\}$$

The deployment view of the smart space SS_{sr} is shown in Fig. 4.18, where the semantic interactions between *iO*s are shown in Fig 4.19.

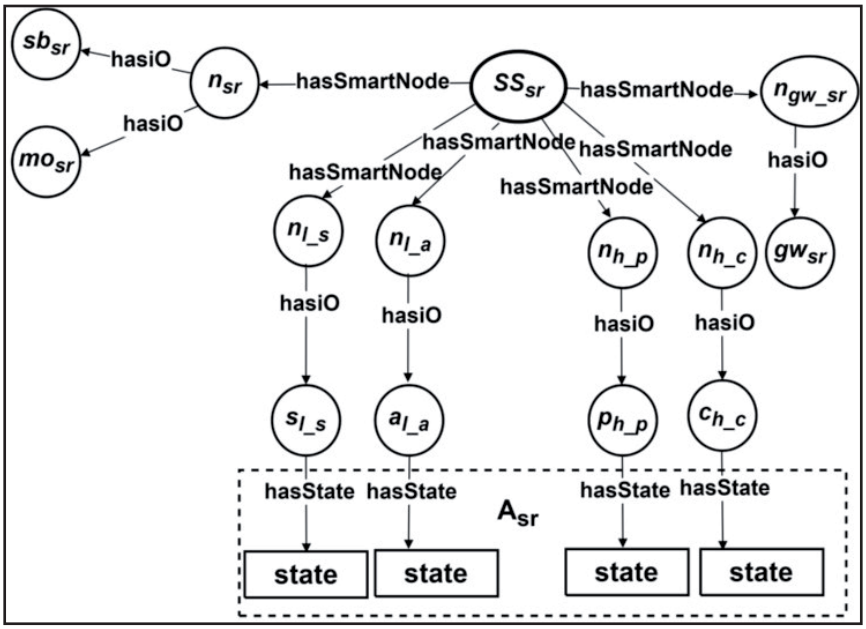


Figure 4.17. The ontology graph O_graph_{sr} inserted by $\boldsymbol{mo_{sr}}$ at $\boldsymbol{sb_{sr}}$.

Figure 4.19 explains the process of achieving objectives of the application A_{sr} in the smart space SS_{sr}. Figure 4.19(a) explains the following steps:

1. $\boldsymbol{gw_{sr}}$ joins at $\boldsymbol{sb_{sr}}$
2. Joining of $\boldsymbol{gw_{sr}}$ is confirmed after the session for $\boldsymbol{gw_{sr}}$ starts at $\boldsymbol{sb_{sr}}$
3. $\boldsymbol{p_{h\_p}}$ joins at $\boldsymbol{sb_{sr}}$
4. Joining of $\boldsymbol{p_{h\_p}}$ is confirmed after the session of $\boldsymbol{p_{h\_p}}$ starts at $\boldsymbol{sb_{sr}}$
5. $\boldsymbol{gw_{sr}}$ subscribes at $\boldsymbol{sb_{sr}}$ for the state of $\boldsymbol{p_{h\_p}}$ using the wildcard entry
6. $\boldsymbol{sb_{sr}}$ activates the subscription of $\boldsymbol{gw_{sr}}$
7. $\boldsymbol{gw_{sr}}$ receives the subscription confirmation
8. User presence detects in the room
9. $\boldsymbol{p_{h\_p}}$ requests at $\boldsymbol{sb_{sr}}$ to update the state 'YES'
10. The status 'YES' added to the O_graph_{sr} graph
11. $\boldsymbol{p_{h\_p}}$ receives confirmation of the updated state 'YES' by $\boldsymbol{sb_{sr}}$
12. $\boldsymbol{sb_{sr}}$ extracts the subscription result from the O_graph_{sr} graph and notifies to the session of $\boldsymbol{gw_{sr}}$
13. $\boldsymbol{gw_{sr}}$ receives the results of subscription, i.e., state 'YES'
14. $\boldsymbol{gw_{sr}}$ sends the command 'on' to $\boldsymbol{a_{l\_a}}$
15. The luminary turns 'on'

Figure 4.19(b) explains the following steps:

16. $\boldsymbol{c_{h\_c}}$ joins at $\boldsymbol{sb_{sr}}$
17. Joining of $\boldsymbol{c_{h\_c}}$ is confirmed after the session of $\boldsymbol{c_{h\_c}}$ starts at $\boldsymbol{sb_{sr}}$
18. $\boldsymbol{s_{l\_s}}$ measures illumination value (eg. 123) and sends to $\boldsymbol{gw_{sr}}$
19. $\boldsymbol{gw_{sr}}$ creates a triple of illumination value and requests to update at $\boldsymbol{sb_{sr}}$
20. The illumination value 123 added to the O_graph_{sr} graph
21. $\boldsymbol{gw_{sr}}$ receives the confirmation of updated triple from $\boldsymbol{sb_{sr}}$
22. $\boldsymbol{c_{h\_c}}$ queries at $\boldsymbol{sb_{sr}}$ for the illumination value in the room using the wildcard entry in the triple
23. $\boldsymbol{sb_{sr}}$ extracts the query result from the O_graph_{sr} graph and notifies to the session of $\boldsymbol{c_{h\_c}}$
24. $\boldsymbol{c_{h\_c}}$ receives the query confirmation by $\boldsymbol{sb_{sr}}$
25. $\boldsymbol{c_{h\_c}}$ receives the query results, i.e., the illumination value, i.e., 123

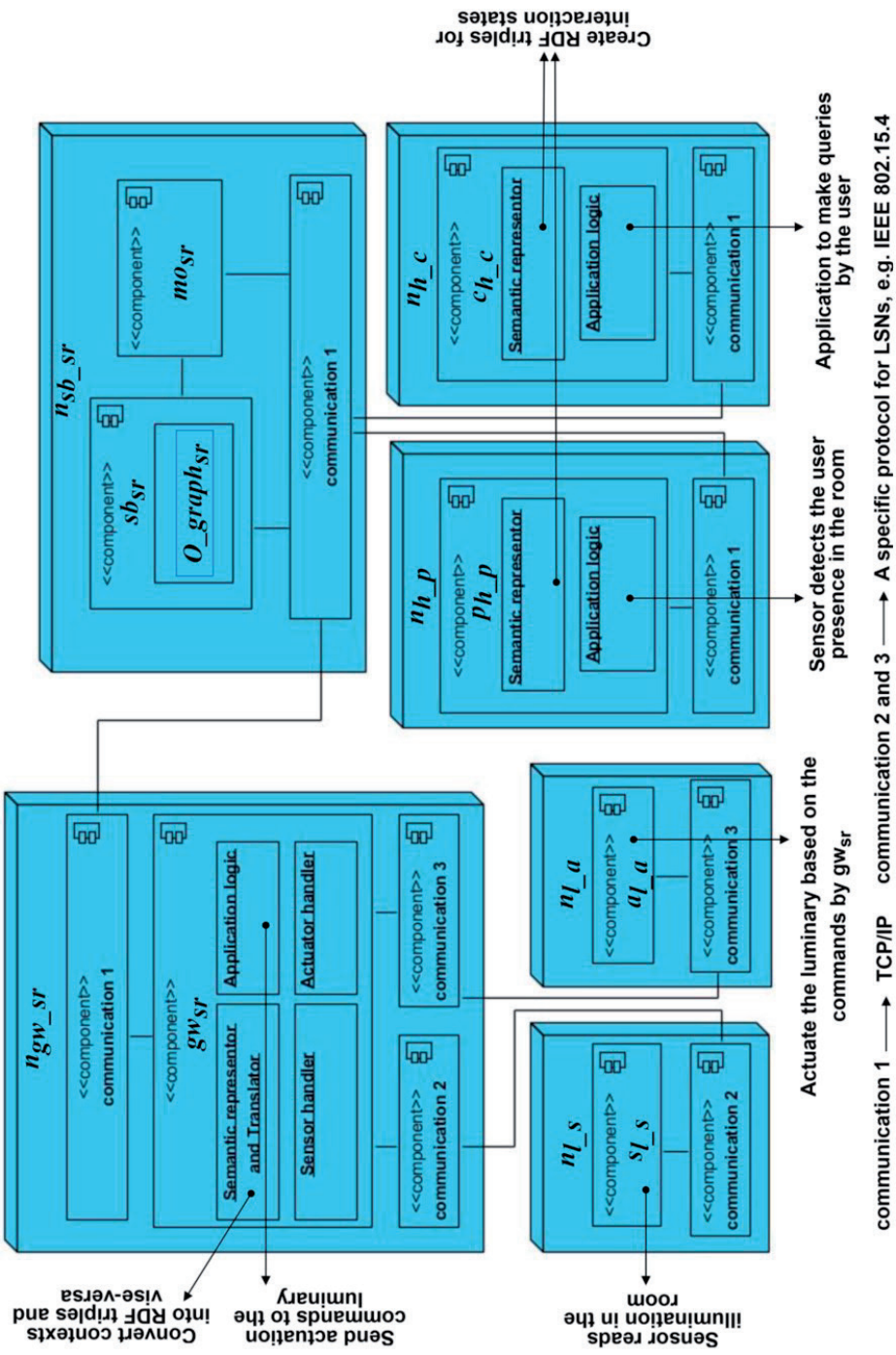
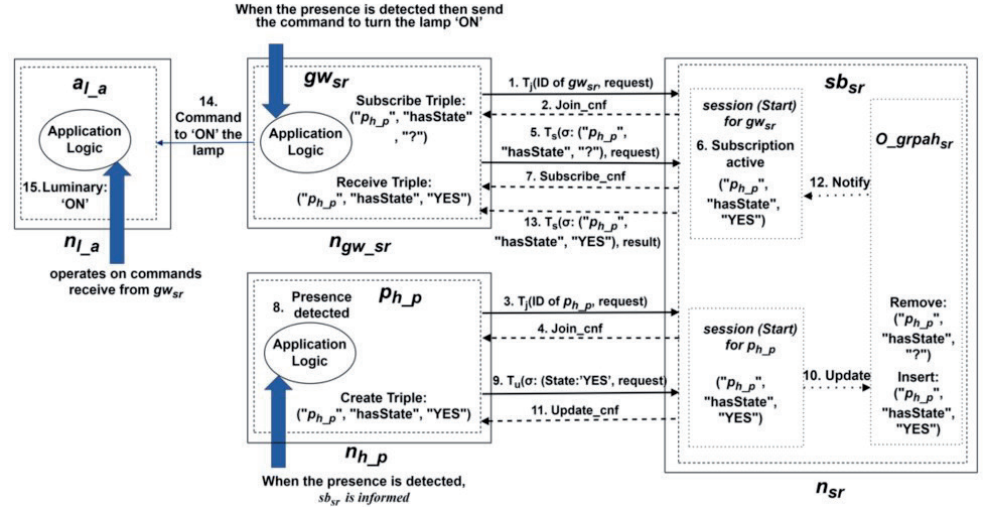
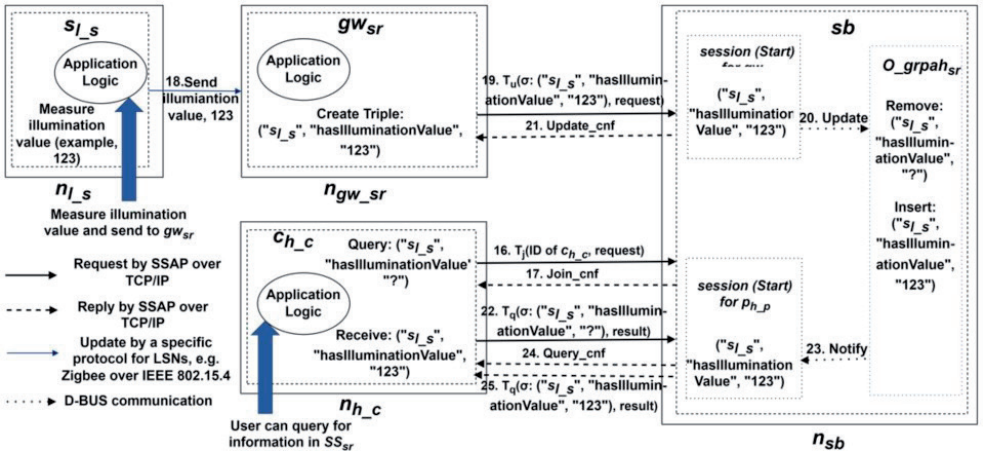


Figure 4.18. Deployment view of the smart space SS_{sr}.



(a)



(b)

Figure 4.19. Execution of the smart application A_{sl}, where (a) explains Objective A and (b) explains Objective B.

In Fig 4.19(a), we explained the execution of A_{sr} for Objective A in the smart space SS_{sr}. The event of the user's presence is generated at $\boldsymbol{p_{h\_p}}$, and as a result, the associated action is executed to turn on the luminary at $\boldsymbol{a_{l\_a}}$. On behalf of $\boldsymbol{a_{l\_a}}$, $\boldsymbol{gw_{sr}}$ established the necessary subscription at $\boldsymbol{sb_{sr}}$ to receive the user presence information. In Fig 4.19(b), we explained the execution of A_{sr} for Objective B in the smart space SS_{sr}. The user wants to get information on the illumination value in the room. Specific information on lighting (illumination) can be beneficial to the user, where the user can further change the lighting according to preferences. This can improve visibility or comfortability for the user. The illumination

control algorithm based on preferences will be discussed in the smart lighting model in Chapter 5. In Objective B, our focus is solely on the details of the light information (illumination value) in the room. The event is generated at $\boldsymbol{s}_{l\_s}$ to measure the illumination value, and the associated action is executed at $\boldsymbol{sb}_{sr}$ to add the illumination value to the ontology graph O_graph_{sr}. Subsequently, the user queries to get this information using $\boldsymbol{c}_{h\_c}$. Finally, the user receives the illumination value in the room from the smart space SS_{sr}.

In this example, $\boldsymbol{gw}_{sr}$ resolves the complexity for $\boldsymbol{s}_{l\_s}$ and $\boldsymbol{a}_{l\_a}$, eliminating the need for them to directly join at $\boldsymbol{sb}_{sr}$. Furthermore, $\boldsymbol{p}_{h\_p}$ and $\boldsymbol{c}_{h\_c}$ do not require direct interaction with $\boldsymbol{a}_{l\_a}$ and $\boldsymbol{s}_{l\_s}$, respectively. The exchange of information between both sides is facilitated through $\boldsymbol{gw}_{sr}$, allowing them to achieve the common objectives of the A_{sr} application.

4.4 Semantic Interactions with Multiple Applications

In this section, we explore semantic interactions within a smart space involving multiple applications. Each application represents the infrastructure of smart nodes associated with a specific application within the smart space. To illustrate this, we consider the following scenario example that involves two applications in a smart space:

"A user's smart space comprises two applications: the smart home and the smart office. Assuming that the user is outdoor and wants to know the light's information in both locations. If the luminaries are 'on' in both applications, the user proceeds to turn them 'off'."

Let us consider the fourth and last example of a smart space in this chapter as follows: Consider two applications, i.e. A_a for the smart home and A_b for the smart office in a user smart space (SS_u). The smart space SS_u contains the following nodes and *iO*s as follows:

$$SS_u.N = \{n_{p\_a}, n_{c\_a}, n_{p\_b}, n_{c\_b}, n_{sb\_u}\} \text{ and } SS_u.iO = \{p_a, c_a, p_b, c_b, mo_u, sb_u\}$$

where, p_a and p_b are *PiO*s deployed on the nodes n_{p_a} and n_{p_b}, respectively; c_a and c_b are *CiO*s deployed on the nodes n_{c_a} and n_{c_b}, respectively; mo_u is *MiO* and sb_u is *SBiO* deployed on the node n_{sb_u}. The nodes n_{p_a}, n_{c_a}, n_{p_b} and n_{c_b} are attached to n_{sb_u} using TCP/IP and are able to exchange semantics using SSAP transactions. The nodes n_{p_a} and n_{p_b} are the switches or smartphones (installed with an application to regulate a luminary). The nodes n_{c_a} and n_{c_b} are the smart home and the smart office luminaries, respectively. Further, the roles of $\boldsymbol{p}_a$, $\boldsymbol{p}_b$, $\boldsymbol{c}_a$, and $\boldsymbol{c}_b$ are as follows:

- $\boldsymbol{p}_a$ generates events for the home luminary in A_a, where each event can have the state either 'on' or 'off'. For example, a switch or a smartphone application can generate an

event to turn on/off the luminaries. Thus, p_a can publish the interaction state for the home luminary, i.e., either 'the home luminary gets on' or 'the home luminary gets off'.

- c_a subscribes to receive information about the current state of the home luminary at sb_u, enables the necessary action (either turning it on or off) for the home luminary in A_a. Consequently, c_a receives the interaction state, indicating either 'the home luminary is currently on' or 'the home luminary is currently off'.
- Similarly p_b generates events for the office luminary in A_b, where the interaction state can be 'the office luminary is turned on' or 'the office luminary is turned off'.
- Likewise c_b subscribes to receive information about the current state of the office luminary at sb_u, which helps c_b determine whether to turn the office luminary 'on' or 'off' in A_a. The interaction state received by c_b can indicate either 'the office luminary is currently on' or 'the office luminary is currently off'.

We design a basic ontology graph (*O_graph*$_u$) of SS_u and store at sb_u through mo_u where the graph is shown in Fig 4.20. The ontology graph *O_graph*$_u$ has the following initial RDF triples:

$$\{(\text{"}SS_u\text{"},\text{"}hasSmartNode\text{"},\text{"}n_{sb\_u}\text{"}),(\text{"}SS_u\text{"},\text{"}hasSmartNode\text{"},\text{"}n_{p\_a}\text{"}),(\text{"}SS_u\text{"},\text{"}hasSmartNode\text{"},\text{"}n_{c\_a}\text{"}),$$
$$(\text{"}SS_u\text{"},\text{"}hasSmartNode\text{"},\text{"}n_{p\_b}\text{"}),(\text{"}SS_u\text{"},\text{"}hasSmartNode\text{"},\text{"}n_{c\_b}\text{"}),$$
$$(\text{"}n_{sb\_u}\text{"},\text{"}hasiO\text{"},\text{"}sb_u\text{"}),(\text{"}n_{sb\_u}\text{"},\text{"}hasiO\text{"},\text{"}mo_u\text{"}),(\text{"}n_{p\_a}\text{"},\text{"}hasiO\text{"},\text{"}p_a\text{"}),(\text{"}n_{c\_a}\text{"},\text{"}hasiO\text{"},\text{"}c_a\text{"}),$$
$$(\text{"}n_{p\_b}\text{"},\text{"}hasiO\text{"},\text{"}p_b\text{"}),(\text{"}n_{c\_b}\text{"},\text{"}hasiO\text{"},\text{"}c_b\text{"}),(\text{"}p_a\text{"},\text{"}hasState\text{"},\text{"}state\text{"})$$
$$(\text{"}c_a\text{"},\text{"}hasState\text{"},\text{"}state\text{"}),(\text{"}p_b\text{"},\text{"}hasState\text{"},\text{"}state\text{"}),(\text{"}c_b\text{"},\text{"}hasState\text{"},\text{"}state\text{"})\}$$

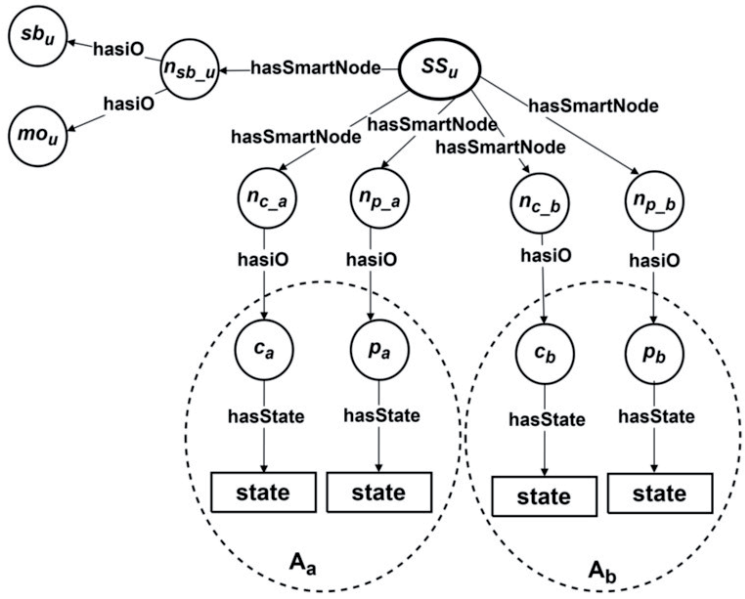


Figure 4.20. The ontology graph *O_graph*$_u$ inserted by mo_u at sb_u.

The deployment view of the smart space SS_u is shown in Fig 4.21, where the semantic interactions among *iO*s are shown in Fig 4.22.

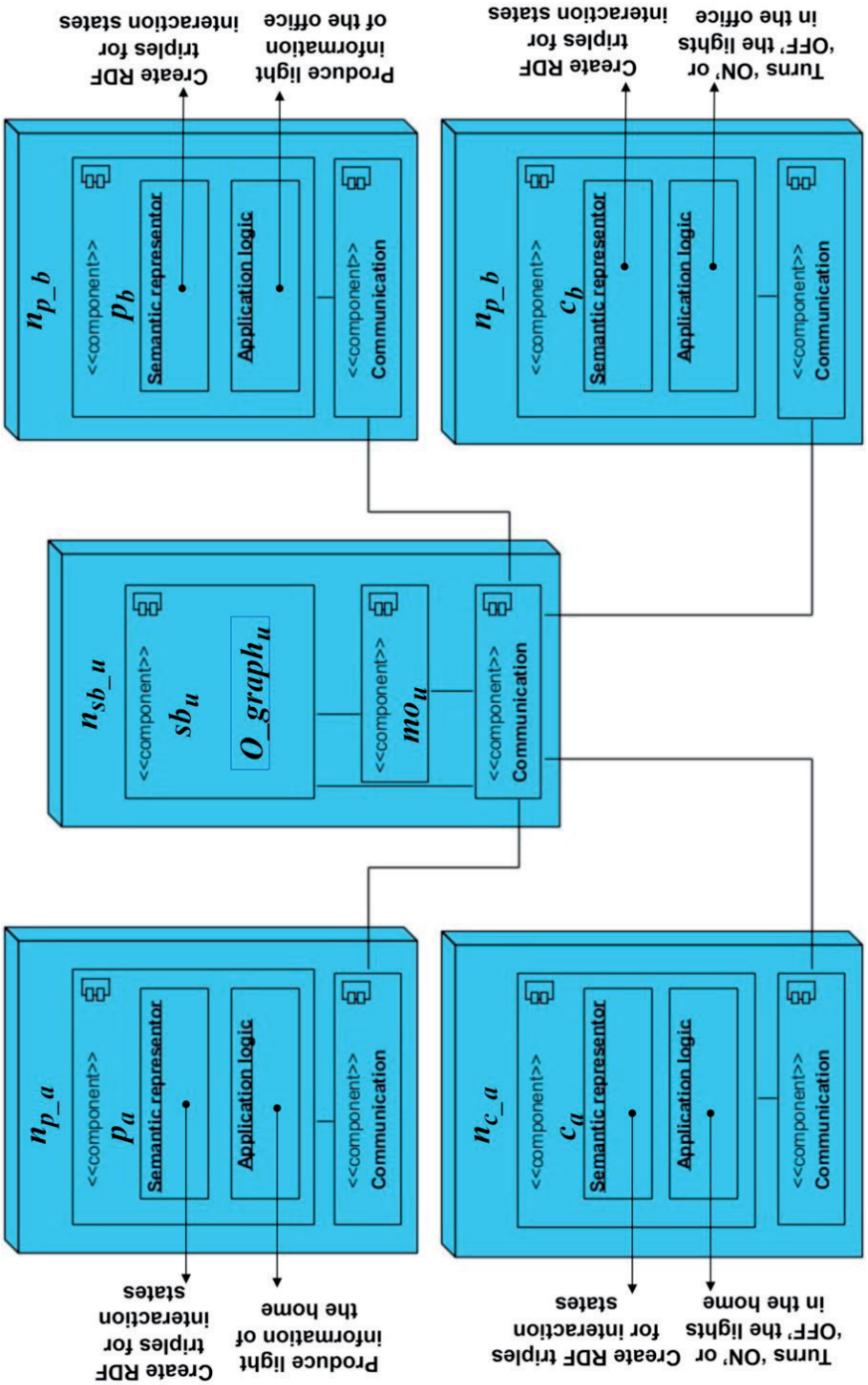


Figure 4.21. Deployment view of the smart space SS_u.

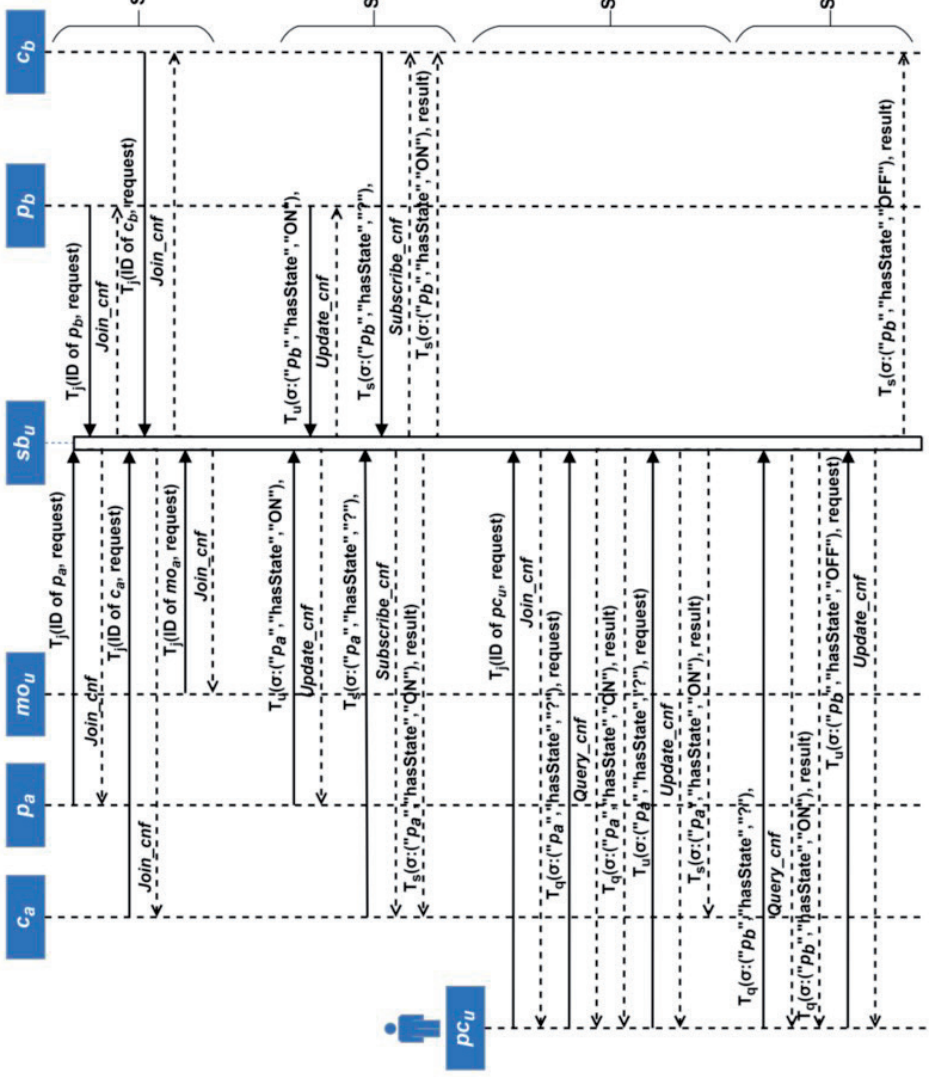


Figure 4.22. Semantic interactions among *iOs* of two applications in the smart space SS_u.

We explain the sharing of semantics among *iO*s of applications in Fig 4.22 using a four-step process. Step 1 illustrates the joining of all *iO*s at sb_u , establishing their connection to the smart space. In Step 2, we describe how p_a and p_b update the interaction states in the smart home and smart office, respectively. Additionally, c_a and c_b subscribe at sb_u to receive the latest updates on interaction states from p_a and p_b, enabling them to regulate the luminaries. As a result, the luminaries in both applications adapt based on the recently updated interaction states within the smart space SS_u.

In the example scenario, the user is outdoors and carrying a smartphone. The smartphone is equipped with a *PiO* and a *CiO* (i.e., pc_u) to enable the user to both produce and consume information within the smart space SS_u. The user's objectives are twofold: first, to obtain the status of luminaries in both applications (consuming information from the smart space), and second, to turn 'off' the luminaries if they are currently 'on' in both applications (producing information in the smart space). To achieve these objectives, the user joins the smart space SS_u through the joining of pc_u at sb_u. In Step 3, pc_u queries at sb_u for the interaction state of the home luminary, where the interaction state is 'on'. Consequently, the user updates the interaction state 'off' at sb_u for the home luminary, the luminary turning 'off'. Similarly, in Step 4, the user queries at sb_u for the interaction state of the home luminary using pc_u, where the interaction state is 'on'. As a result, the user updates the interaction state 'off' at sb_u for the office luminary, causing it to turn 'off'. In the example scenario above, we explained the execution of a smart lighting application involving two applications within a smart space. Throughout this scenario, we have observed the importance of regular updates of lighting information events in both applications, as they are required for user queries. Furthermore, the user has the possibility to modify and regulate the luminaires in both applications based on the information received from the smart space. The user's smartphone acted as both producer and consumer of information within the smart space. It produced semantics to switch off the lights in the applications and accessed information about the status of the lights from the smart space. It is worth noting that a smart space works in a similar way when more than two applications or smart applications are used.

4.5 Conclusions

In this chapter, we have explained the fundamental concepts of semantic interoperability and emphasized the importance of developing application ontologies in smart spaces. Semantic interoperability is enhanced by the use of semantic reasoning with matching operations to extract the results of queries and subscriptions. Semantic reasoning enabled the inference and matching of concepts based on existing ones within a smart space. In addition, we have elaborated on semantic interactions through the use of SSAP trans-

actions and have provided examples for both single and multiple applications in smart spaces. SSAP transactions facilitate the manipulation, querying, and subscription of RDF triples in a smart space.

Furthermore, we presented a mechanism for information sharing between resource-poor nodes and high-capacity nodes through a gateway approach in smart spaces. Looking ahead, we will conduct experiments on smart lighting applications using semantic interactions via SSAP transactions in subsequent chapters.

Chapter 5

Smart Lighting Case Study

We aim to demonstrate the practical utility of the smart space and semantic interoperability concepts, along with the proposed semantic interoperability architecture discussed in previous chapters. To achieve this, we apply these concepts to a case study of smart lighting applications, encompassing multiple use cases. In this chapter, we begin by examining the existing literature on smart lighting applications for indoor spaces. Subsequently, we propose a novel smart lighting model that is specifically designed to align with the smart space concepts and semantic interoperability architecture presented earlier. This model facilitates the control of illumination and the adaptation of lighting behaviors in indoor spaces, such as homes and office buildings. Additionally, we elucidate the mechanisms of semantic interoperability interactions among smart nodes employed in smart lighting applications. Finally, we present two use cases, namely context-adaptive smart lighting and power-managed smart lighting, which serve as the basis for our experimental evaluations.

5.1 Smart Lighting Applications and Related Work

The existence of sufficient light is crucial for any task one may want to carry out. Human beings feel comfortable with different levels of illumination for different tasks. They perceive, interact with, and are affected by lighting in many ways. The introduction of smart lighting applications, as a particular type of smart space application, not only facilitates better use of natural light sources (e.g. by actuating blinds to control exposure to daylight and moonlight), but also provides more control over man-made artificial light than ever. We provide a summary of typical smart lighting applications and their goals. Note that, many smart lighting applications cannot be classified purely in just one of the following categories. Some applications are designed to achieve multiple goals at the same time, and thus, have combined goals of multiple categories.

1.) Environment friendly lighting with energy and cost efficiency: Lighting constitutes around 19% of the total electricity consumption of the earth's population [5.1]. Smart indoor and outdoor lighting can increase the energy efficiency of lighting by dimming or turning off light sources where they are not needed based on occupancy and activity monitoring. It is not only the total amount of energy consumed that determines the energy bill. Since energy is very difficult to store, electricity providers encourage their

consumers to maintain a total power consumption that complies with a fixed energy budget, allowing planning of power generation capacity in a cost-effective manner. For this, they apply cheap rates within the energy quota and higher rates for the extra usage. Thus, keeping the energy consumption tightly below the quota can lead to significant cost savings for their customers. The energy consumption due to lighting can be very accurately controlled by a power-managed smart lighting [2.32] indoors as well as outdoors. The stakeholders of this type of application are building owners, municipalities, and energy companies.

2.) Lighting supporting user comfort and performance: User studies show that light affects people's feelings of comfort [5.2] [5.3]. It is also proven that the level of light illuminance affects performance-related human attributes such as alertness, vitality, and the level of sustained attention in tasks done. According to a study on perception of lighting quality in offices [5.4], office users' preferences for color temperature vary between 3000 K and 6000 K, and employee satisfaction is increased when they are in control of lighting in their personal area. Therefore, the goals of smart lighting applications that fall in this category are twofold. Firstly, they aim at autonomously choosing the light settings that maximize comfort and user performance in tasks carried out. Secondly, they aim to provide advanced interaction and control possibilities to users in their vicinity. Note that this may not be easy in today's typical living and working places. It is especially challenging in multi-user environments where there are conflicting preferences of users, e.g. in an open office space. In this type of smart lighting applications challenges lie with i) expressing user preferences and feedback through simple user interaction, ii) modeling and dealing with variations and inconsistencies in user preferences, iii) identifying application context and interpreting it, iv) resolving conflicts between concurrent activities and users, and v) representing lighting scenarios, i.e. 'how to actuate'. Interactivity requires high performance in terms of response times, e.g. average and mean time between pushing a button switch and a set of lights turning on, average and maximum response times within such a set, and smoothness of dimming. At a higher level, solutions need to consider user satisfaction (difficult to measure), productivity and the balance between automation and user interaction, i.e., too much automation and too much interaction are both undesirable.

***3.) Smart lighting as information medium*:** In this type of smart lighting application [5.5] visible and invisible (infrared) light coding and modulation are employed to use the lighting infrastructure for communicating information. Such communication may be for exchanging information among components that can observe each other's light emissions (via direct line of sight or via reflection from a surface), or for delivering useful information directly to users. The former approach is especially useful for taking some load from the wireless radio communication traffic, and for increasing physical

topology awareness in the system. Depending on the installation, nodes that can communicate would know that they either have a line of sight, or they lie nearby, e.g. on a flat ceiling.

There are a multitude of applications that employ smart lighting to directly inform a user. For example, a user may need subtle and personalized information reflected to the surfaces in their surroundings, such as the daily agenda of an office employee reflected on the wall across from his desk. It is possible to arrange a personalized light pattern such that only the employee would know what the patterns mean, and hence can read the agenda, whereas visitors see it as a decoration aspect. Another example is light being used as a communication medium for information to be shared by inhabitants of an environment. For example, a wall-wash lighting fixture that can change colors dynamically can be used as a meeting timer [5.6]. Lighting can be used for indoor navigation purposes as well, e.g. to show the exit direction in case of an emergency evacuation [5.7]. Coded light is also started to be used for indoor positioning of people and artifacts [5.8]. This technology requires a camera (e.g., of a smart phone) to decode the information in light.

4.) Smart lighting supporting well-being: The level of light illuminance affects heart rate and its variability [5.9] [5.10]. Light also influences (perception of) safety [5.11], mood [5.12] and the circadian rhythm [5.13]. While exposure to daylight is of utmost importance for physiological reasons [5.14], a lot of people stay indoors during daytime either at home or for work. It is well known that blue light has the biggest impact on the human circadian system, as the human brain has evolved to become alert (wake up) when we are exposed to blue light. The problem is, artificial light sources are known to emit a high intensity of blue light. Today people are already exposed to a lot of blue light by looking at computer screens and smart phones for many hours. On top of this, common behaviors like keeping lights on when they are not needed cause circadian rhythm shifts and sleeping difficulties as melatonin (sleeping hormone) production is suppressed, causing a person to feel weak during daytime. As proven in [5.15], blue light leads to melatonin suppression, even when a person is asleep, and his eyes are closed. This knowledge can be used to shift the sleep cycles of patients that suffer from certain sleeping disorders back to normal [5.16].

5.) Smart lighting influencing human behavior: Studies show that it is possible to influence human behavior using light. For example, light can be used to create atmospheric settings that regulate the speed at which people eat in a restaurant and a carefully chosen setting can decrease the number of calories taken in and increase their overall customer satisfaction of the food [5.17]. Light can affect social situations as well. In

Table 5.1. Smart lighting applications related work.

Article	Year	Objective	Communication Interoperability	Semantic Interoperability	Use Cases
Higuera et al [5.20]	2015	Smart lighting system for offices	ISO/IEC/IEEE 21451 standards and ZigBee Light Link are used.	---	Smart lighting contributing to an office use case scenario.
Kim et al. [5.21]	2016	Smart lighting system for displaying messages	Bluetooth beacon communication is used with DALI dimming. controller.	---	Display system through LED. lighting.
Prasetio et al. [5.22]	2016	Smart lighting system for occupancy-based scenarios	Communication over internet using Wi-Fi.	Semantic Sensor Network (SSN), IoT-Lite, and Semantic Actuator Network (SAN) ontologies used for semantics sharing.	Smart lighting contributed to a user presence detection scenario.
Liu et al [5.23]	2016	Smart LED lighting system	Ethernet RJ45 communication with DALI dimming technology.	---	Smart lighting contributing to an office use case scenario.
Lwin et al. [5.24]	2017	Design of a LED dimmer integrated with a light sensor to contribute to energy saving in a smart indoor lighting system	Zigbee based on IEEE 802.15.4 used on the sensor nodes to transmit data periodically.	---	Detecting user's presence and activities to actuate LED luminaries accordingly.
Viani et al. [5.25]	2017	Smart lighting in energy-efficient museums	Zigbee communication module used on sensor nodes, whereas Wi-Fi used to control LED luminaries.	---	Smart lighting contributing to a smart museum scenario
Barve et al. [5.26]	2017	Smart lighting for smart cities	GSM based communication is used between the street lighting poles and a server, with DALI interface for dimming	---	Smart street lighting control based on daylight, occupancy, time and web interface commands.
Khatavkar et al. [5.27]	2017	Smart street lighting	Zigbee module integrated to sensing and controlling units for communication.	---	Controlling street lighting based on daylight at several intervals and occupancy

Table 5.1. Continued

Article	Year	Objective	Communication Interoperability	Semantic Interoperability	Use Cases
Kumar et al. [5.28]	2017	Smart lighting system for a building	Zigbee communication module for controlling lighting fixtures.	---	Control of lighting in a building
Mathews et al. [5.29]	2018	An Open Architecture for Intelligent Solid-State Lighting Systems (OpenAIS) using IoT technologies	Any communication technologies compatible with IoT devices such as 6LOWPAN, Wi-Fi, IEEE 802.15.4, Ethernet, IPv6	---	Control of lighting in an office building
Sikder et al. [5.30]	2018	Designing a smart indoor and outdoor lighting system for energy saving	IEEE 802.15.4-based IoT communication protocols such as ZigBee, 6LoWPAN, and JenNET-IP are used for communication interoperability	---	Smart lighting contributing to save energy in multiple indoor scenarios (e.g., in home or offices) and outdoor scenarios (e.g., street lighting)
Amarillo et al. [5.31]	2020	Smart lighting application for academic environments	IoT devices used Wi-Fi as a communication protocol		Control of lighting in a classroom implementation scenario
Soheilian et al. [5.32]	2021	Smart lighting application for energy saving and user well-being	Three setups of communication protocols are used as follows: Wi-Fi, WiFi combined with Zigbee using a gateway, and bluetooth combined with DALI	---	Smart lighting contributing to energy saving and user well-being in the residential environment
Gowda et al. [5.33]	2021	Smart lighting system for smart cities	Wi-Fi and Zigbee based communication with DALI interface for dimming	---	Smart lighting contributing to energy saving in street lighting environments
Putri et al. [5.34]	2022	Smart lighting system for a meeting room	Bluetooth communication		Smart lighting environment for user comfort in the co-working space's meeting room

[5.18], they use dynamic outdoor lighting to diffuse escalating behavior of people on the street during nighttime.

6.) ***Smart lighting considering aesthetics***: Lighting is widely used to improve atmospheres. Sometimes it is easier to perceive an effect from a distance. Decorative adaptive dynamic lighting is an example of this. Interestingly, the atmospheric effect of dynamic lighting is large on the people that are observers in the surrounding and not so large for the people whose locality is being illuminated [5.19]. Another example application is to use smart outdoor lighting against light pollution. By dimming down the street lighting where it is not needed, light pollution in cities can be reduced, allowing a better view of the sky and the stars at night.

We explained the above categories of smart lighting applications seen in smart spaces. These applications are broadly considered in two categories, indoor and outdoor smart spaces. In this thesis, we focus on indoor lighting applications such as homes and offices in a smart space. We propose a smart lighting model based on the semantic interoperability architecture in this chapter. We now enlist existing implementations of smart lighting applications in Table 5.1, where the objectives, communication interoperability, and semantic interoperability of the applications are introduced together with use cases.

As listed in Table 5.1, Agung et al. [5.22] proposed a semantic-based model using SSN and SAN ontologies. However, the model includes only LSNs (sensors and actuators with low resources). Our proposed semantic interoperability architecture covers both LSNs and HSNs. For HSNs, we used TCP/IP for communication interoperability. We integrated a gateway approach for LSNs using the communication technology Zigbee over IEEE 802.15.4 to accommodate low-capacity sensors and actuators in a smart space. On top of these communication technologies for HSNs and LSNs, we use SSAP for semantic interoperability in the proposed architecture that enables smart nodes to share knowledge using semantics. A GSN on behalf of LSNs solves the complexity of translating knowledge into semantics for smart spaces. Other lighting applications listed in Table 5.1 are application-specific and do not use the semantic approach for interoperability. In Section 5.2, we propose a semantic-based smart lighting model using the proposed semantic interoperability architecture. The semantic-based smart lighting model or the smart lighting model, as discussed in the next section, facilitates the sharing of lighting semantics among *iO*s to achieve specific objectives in controlling illumination in indoor smart spaces. We will delve into the smart lighting model in detail in Section 5.2.

5.2 Smart Lighting Model

Smart lighting applications execute with the help of collaborating *iO*s in a smart space. We consider those collaborating *iO*s in an application that are responsible for parameters such as user requirements (in terms of preferred illumination in activity spaces), illumination readings by light sensors, and brightness values of luminaries installed in the physical environments of smart spaces. Lighting applications aim to satisfy users to meet their requirements of preferred illumination in activity spaces or subspaces with the help of a suitable smart lighting model that can be deployed in a smart space. Therefore, we propose a novel smart lighting model, where the objective is to provide adaptive services for user satisfaction by automatically dimming and turning on/off the lights when required. We aim to achieve this objective by implementing the smart lighting model using the proposed semantic interoperability architecture in a smart space. We discuss two main assumptions that are considered for the proposed smart lighting model. First, several types of light sources are available in the market such as incandescent, fluorescent, LED luminaries. Therefore, we present our assumptions of light sources in Section 5.2.1. Second, we consider indoor spaces such as buildings, homes, and offices for the smart lighting model. The model needs a proper mechanism to place light sensors and luminaries in indoor spaces. Therefore, we discuss the placement of light sensors and luminaries in indoor environments in Section 5.2.2. Finally, we discuss the illumination control algorithm in Section 5.2.3.

5.2.1 Light sources

The simplest form of smart indoor lighting is an automatic on/off control based on occupancy detection in a room. We discussed this example in Section 4.3.1. More complex forms of smart lighting systems [5.35]- [5.37] try to automatically adjust light levels according to the user activity performed. However, the solutions in [5.35]- [5.37] are initially optimized for traditional incandescent and fluorescent luminaries, which do not provide enough controllability. Note that it is difficult to dim fluorescent lamps and to control color and intensity simultaneously with incandescent lights, since increasing intensity raises the heat and shifts the color spectrum. Unlike incandescent lamps, which need to convert electricity into thermal energy first, LED illumination is achieved through LED luminance (a semiconductor crystal that can directly produce visible light in a desired wavelength range). Therefore, we use the LED luminaries as the light sources in our proposed smart light model. The LED luminaries comprise the following properties:

- LED luminaries can maintain lumen efficacy and light color in a larger sink while enabling digital control. LED luminaries at low temperatures around 2700-3000K give “warm” colors such as yellow and orange whereas at a higher temperature around

5000 K or more they give “cool” colors such as blue and green. Moreover, we can adjust the brightness of LED luminaires, either increasing or decreasing it.

- LED luminaries save energy, provide an extra-long lifetime, and help the environment by not emitting harmful gases [5.38], [3.39]. Most of the LED luminary manufacturers claim that the LED lifetime is around 50,000 hours. This is approximately 50 times longer than a typical incandescent luminary, 20-25 times longer than a typical halogen, and 8-10 times longer than a typical Compact Fluorescent Lamp (CFL).
- LED luminaries are highly energy efficient as compared to incandescent and fluorescent luminaries. For example, an 8 or 9-watt LED luminary emits as much light as a 60-watt incandescent luminary and a 14-watt fluorescent.
- LED luminaries concentrate the light into a small area using a focusing/rod lens in front of the LEDs.
- LED luminaries can reduce glare by using a micro-prismatic technology. It can develop special diffusers with this technology that disperse light from individual LEDs to give homogeneous light with optimum contrast by avoiding any direct or reflected glare.

5.2.2 Placement of light sensors and luminaries in indoor spaces

The illumination generated on a surface by an LED luminary is the superposition of the luminance generated by LEDs. Each LED in an LED luminary emits a directed beam of light with a width of 20-30 degrees. Therefore, several LED luminaries can provide localized illumination effects in an indoor space with low light leakage and glare. Illumination distribution of a group of LEDs is studied in [5.40]. In our lighting model, the area of illumination generated by luminance from one LED luminary is focused on a single activity subspace and the overflow onto the other activity subspaces is negligible. The activity subspace is a physical portion of the total physical area of a smart space where a user performs some activity such as reading a book or watching TV.

Producing a large amount of illumination on a surface requires several LEDs together. Therefore, an LED luminary normally consists of arrays of LEDs producing local illumination at target points from a distance. Thanks to the small size of LEDs, we can design LED luminaries of arbitrary size and shape (square, circle, rectangle, etc.). In an LED luminary, each LED has a certain radiation pattern and produces illumination on a target surface. The overall illumination distribution on the target surface depends on the distance. In [5.41], an analysis was performed on different shapes of LED luminaries (e.g. triangular, circular, square hexagonal, or rectangular). The analysis results show that a uniform illumination rendering can be achieved by a hexagonal LED array and can be approximated by a square-shaped LED array. We consider square-shaped LED array luminaries for illuminating square grid cells since a room in the model is divided into square grid cells.

The illumination of a single LED appears in the shape of a circle. However, given many LEDs together, the illumination of a square LED array approximates a square shape. Here we assume that the center-to-center distance between two neighboring LEDs in an LED luminary (d) is less than 1 cm, which is true for most LED luminaries available in the market. An LED emits a direct spotlight with an angle of light (θ) between 20 and 30 degrees as shown in Fig. 5. 1(a), where h is the distance between LED and illumination area, and r is the radius of illumination area, calculated by (5.1).

$$r = h \times \tan(\frac{\theta}{2}) \tag{5.1}$$

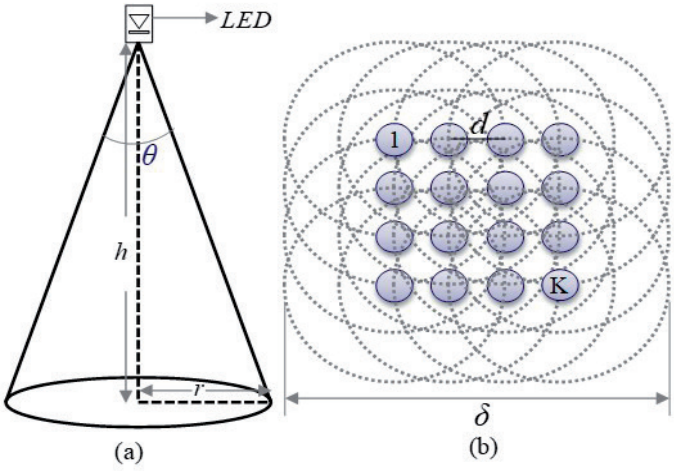


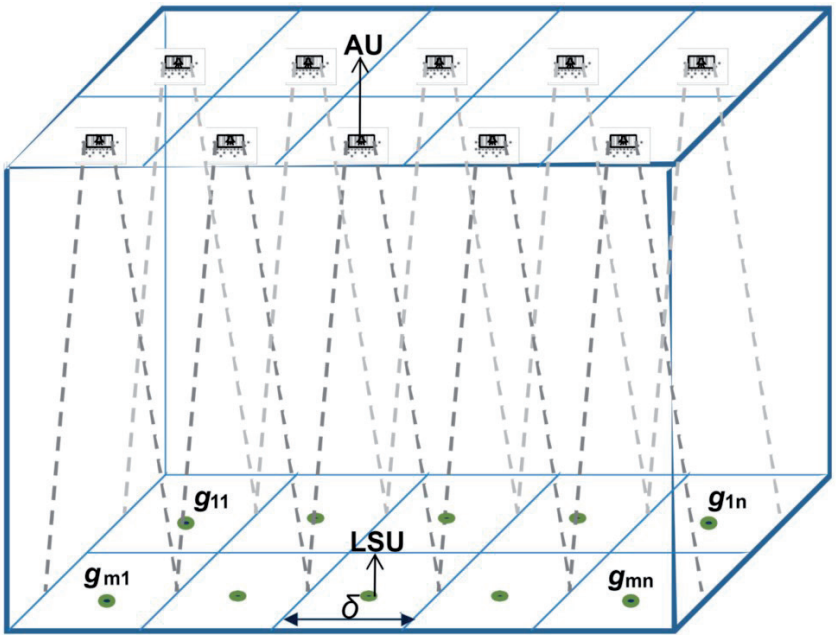
Figure 5.1. Illumination area of (a) single LED and (b) array of LEDs.

When the LED luminary is a square shaped array of LEDs, the illumination area is as seen in Fig. 5.1(b), where δ denotes side-length of the illumination area, calculated by (5.2).

$$\delta = 2r + (\omega - 1) \times d \tag{5.2}$$

where ω (≥2) is the number of columns and rows in an LED matrix. We consider placing light sensors and luminaries in an indoor space in the smart lighting model. For this purpose, an indoor space is divided into several square grid cells (g_{mn}), where 1≤m≤M and 1≤n≤N, resulting in a total of M×N square grid cells. We consider a light sensing unit (LSU) to measure illumination in an activity space and an actuating unit (AU) to actuate LED luminaries with certain brightness values. We place LSUs and AUs in a room such that one LSU and one AU are placed in each square grid cell as shown in Fig. 5.2., i.e., for $\forall m \in M, \forall n \in N$, g_{mn} has (LSU_{mn}, AU_{mn}). In our realization, we observe slightly more illumination at the corner of all square grid cells as compared to the center of the grid cells because of the overlapping illumination from the neighboring square grid cells. The increased illumination at the corners will possibly be less than 10 lux, therefore, we con-

sider it negligible. The illumination measured at the center of each grid by the light sensor is the actual illumination in each square grid cell.



(c)

Figure 5.2. LED luminary and sensor placement.

5.2.3 Illumination control algorithm

We propose and explain the Illumination Control Algorithm (ICA) for indoor spaces (e.g. home [5.42], office [5.43]). Regardless of the infrastructure of a smart space, we first explain the calculation of actuation commands in the algorithm. Later we map ICA to the proposed semantic interoperability architecture in Section 5.3. The ICA calculates the actuation commands for AUs based on the readings of LSUs and user preferences in all square grid cells. The actuation of LED luminaries depends on the relationship between the illumination at a given activity subspace and the corresponding luminous intensity of luminaries. The activity subspace is one or a group of square grid cells where the user performs an activity. A luminary can maintain illumination at a square grid cell of an activity subspace by autonomously adjusting light output in the 0% to 100% brightness range. An analysis of the relationship between illumination and luminous intensity has been given in [5.44].

In order to calculate the illumination (*E*) at a particular point using the inverse square law, the luminous intensity, *I* (lumens), by the luminaries and the distance between sensing nodes (LSUs) to the luminaries, *h* (meters), need to be known [5.45]. A LSU reads the illumination *E* in units of *lux*, i.e., lumens per square meter. We can measure the illumination *E* over a square grid cell, where the intensity *I* is distributed over a circle with the radius r. The basic relationship in this case can be formulated as:

$$E = \frac{I}{\pi(h \times \tan\frac{\theta}{2})^2} \tag{5.3}$$

In this thesis, we focus on the side length of a square grid cell which is equal to the side length of illumination area δ given in (5.2). Therefore, the side length of a square grid cell is given by (5.4) as follows:

$$\delta = 2\left(h \times \tan\frac{\theta}{2}\right) + (\omega - 1) \times d \tag{5.4}$$

Note:

- LSU's readings are considered without considering reflection, absorption, or loss. These are the actual illumination readings of the activity subspace (the square grid cells or cells) illumination even in case of reflection and absorption.
- The activity subspace may cover one or more square grid cells, depending on the activity of a user.

At any time, the average readings from square grid cells by LSU_{mn} give the average illumination without any barriers or occlusions, denoted by E_t.

$$E_t = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} E_{mn}}{M \times N} \tag{5.5}$$

Note:

- External light sources (e.g., daylight) affect the LSU's readings, having an influence on the way LED luminaries are to be dimmed.
- LSU in a square grid cell does not necessarily need to be placed exactly at the center. This is because the grid cell is approximately uniformly illuminated. We conducted a series of experiments by taking LSU's readings at various locations within the square grid cell to observe changes in illumination. In the end, we discovered that there were changes of less than 5% in illumination when we moved the LSU towards the corners.

The average illumination caused by only the LED luminaries on all g_{mn} is given by $\bar{E}_{mn}$. The light output from one LED luminary in g_{mn} is denoted by I_{mn}. In addition, external light sources (e.g., daylight) in all g_{mn} also contribute to the average illumination by E_{ext}. Therefore, E_t is given by (5.6).

$$E_t = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} E_{mn}}{M \times N} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} \bar{E}_{mn}}{M \times N} + E_{ext} \tag{5.6}$$

$$\therefore E_{ext} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} E_{mn}}{M \times N} - \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} \bar{E}_{mn}}{M \times N} \tag{5.7}$$

The values of E_{ext} cannot be controlled by the proposed model, but it can be changed manually or externally. For example, more daylight contribution is expected when the user opens curtains of the room. Therefore, the ICA in the proposed model can adjust E_t only by controlling $\bar{E}_{mn}$, where $\bar{E}_{mn}$ is controlled by the light output of an individual luminary I_{mn}. We can control the light output I_{mn} by adjusting the brightness percentage of the LED luminary B_{mn}, where the brightness percentage B_{mn} is operated by AU_{mn}. Here 0% and 100% correspond to the luminary being turned 'off' and being fully 'on', respectively. In the ICA, we regulate the brightness percentage B_{mn} to adjust the light output I_{mn} and corresponding illumination E_{mn} based on a user preference of illumination in every individual square grid cell. We define the user preference in a range between minimum and maximum desired illumination in each square grid cell due to the fluctuating nature of illumination. Note that fluctuation may occur due to many reasons, for example, when a user opens or closes the door or because of switching the TV on or off. Therefore, we define the preference ρ_{mn} for g_{mn} such that: $\rho_{mn} = [E_{mn|min}, E_{mn|max}]$, where, $E_{mn|min}$ and $E_{mn|max}$ denote the minimum and the maximum desired illumination in a square grid cell g_{mn}.

To meet the desired illumination range of preferences ρ_{mn}, we propose the following procedure of the ICA as shown in Table 5.2. We calculate the actuation commands in the form of target brightness levels B_{mn} for AU_{mn}. Let the smallest unit of increment and decrement of brightness percentage of a luminary at one update be ΔB in g_{mn}. We discriminate between the following two states (St_1, St_2) in Table 5.2.

Table 5.2. Procedure of the Illumination Control Algorithm (ICA).

No.	States
1	St_1: For g_{mn}: if ($E_{mn} < E_{mn\|min}$ and $0 \leq B_{mn} \leq 100 - \Delta B$)
2	then $B_{mn} \leftarrow B_{mn} + \Delta B$
3	hence I_{mn} is increased.
4	St_2: For g_{mn}: if ($E_{mn} > E_{mn\|max}$ and $\Delta B \leq B_{mn} \leq 100$)
5	then $B_{mn} \leftarrow B_{mn} - \Delta B$
6	hence I_{mn} is decreased.

The procedure in Table 5.2 will operate on B_{mn} to decrease or increase until E_{mn} meets the preferred range of illumination $[E_{mn|min}, E_{mn|max}]$. By this procedure, the illumination in each square grid cell is controlled independently by the corresponding LED luminary in the same square grid cell. If an activity is distributed over a group of square grid cells in an

activity subspace, the corresponding sets of LED luminaries work together to provide the required illumination in the activity subspace, i.e., each provides the required illumination in their corresponding grid cells.

It is important to note that for a seamless transition between different light settings, ΔB must be chosen appropriately. If ΔB is too large, users may notice rapid changes in light intensity, potentially leading to discomfort. Conversely, if ΔB is small enough, users may experience comfort due to the smooth transitions between different light settings. Additionally, the procedure in Table 5.2 depends on specific frequencies, including ΔB. These frequencies also have an impact on stability. LSUs provide illumination readings to the ICA with a certain sensing frequency, and ΔB calculated by the ICA is further communicated to AUs with a certain actuating frequency. The choice of both sensing and actuating frequencies significantly affects the operation of the ICA. In Chapter 7, we will also delve into the impact of choosing these frequencies together with ΔB in the smart lighting model.

5.3 Mapping of ICA to the Proposed Architecture

In this section, we describe the mapping of the ICA to the proposed semantic interoperability architecture. For this purpose, we make groups of similar capacity nodes in a smart space into two smart applications. This means a group of LSNs in one application and a group of HSNs in another application. In general, the deployment of LSNs and HSNs depends on the requirements of the smart space infrastructure. We choose to group them within individual applications for the sake of convenience and with a specific purpose as follows. First, we can easily group them within their network connected to a central node: LSNs communicate to a GSN and HSNs communicate to an SBSN. Second, we want to conduct separate experiments in depth on the deployment of LSNs in a smart space since one of the challenges mentioned in $\boldsymbol{RQ_{2B}}$ is the integration of LSNs in smart spaces. Note, we can also deploy mixed nodes of LSNs and HSNs in a smart space, and this depends on the smart space developer's choices.

Consider a smart space for smart lighting (SS_{SL}) with two applications: A_l and A_h. These applications A_l and A_h are running on their respective smart space infrastructures of LSNs and HSNs. The purpose of considering two applications here is to demonstrate the capabilities of LSN integration through a gateway node and HSN communication directly to an SBSN in a smart space. This capacity is associated with the deployment of the ICA in both types of applications. Later, we will delve into how the ICA is implemented in both applications.

The activity spaces of applications A_l and A_h are divided into square grid cells of equal sizes, denoted by $g_{l,mn}$ and $g_{h,mn}$ respectively. We consider the same number of square grid cells for both applications for convenience only, but it may differ based on the physical areas in both applications in real scenarios.

In the application A_l, we deploy two types of LSNs (one is a light sensor node $n_{l\_s,mn}$ and another is a LED luminary with actuator $n_{l\_a,mn}$) with two associated *iO*s (***s*****$_{mn}$** and ***a*****$_{mn}$**, respectively) in every single square grid cell. All LSNs are connected to a GSN (n_{gw_SL}) that hosts a *GWiO* (gw_{SL}) through a low power communication protocol, i.e., Zigbee. The interaction states of s_{mn} and a_{mn} are the illumination value ($e_{l,mn}$), i.e., $E_{mn} = e_{l,mn}$ and the brightness percentage ($b_{l,mn}$), i.e., $B_{mn} = b_{l,mn}$, respectively.

Similarly in the application A_h, we deploy two HSNs (one is a light sensor node $n_{h\_p,mn}$ and another is a LED luminary with actuator $n_{h\_c,mn}$) with two associated *iO*s (***p*****$_{mn}$** and ***c*****$_{mn}$**, respectively) in every single square grid cell. All HSNs are connected to an SBSN (n_{SL}) deployed with an *SBiO* (sb_{SL}) and an *MiO* (mo_{SL}). They communicate over TCP/IP and share semantics using SSAP. The interaction states of ***p*****$_{mn}$** and ***c*****$_{mn}$** are the illumination value ($e_{h,mn}$), i.e., $E_{mn} = e_{h,mn}$ and the brightness percentage ($b_{h,mn}$), i.e., $B_{mn} = b_{h,mn}$, respectively.

The static mapping between the square grid cells and the placement of *iO*s according to Fig. 5.2 is such that: $\forall m \in M, \forall n \in N$, ($s_{mn}$, a_{mn}) are placed in $g_{l,mn}$ and (p_{mn}, c_{mn}) are placed in $g_{h,mn}$. The smart nodes n_{gw_SL} and n_{SL} can be placed independently anywhere in the smart space SS_{SL}, provided that the placement depends on the reachability of connections from LSNs to n_{gw_SL} and HSNs to n_{SL}.

The basic application ontology graph of SS_{SL} (O_graph_{SL}) is shown in Fig 5.3. The graph O_graph_{SL} is deployed on ***sb*****$_{SL}$** through ***mo*****$_{SL}$**. The deployment view of $SS_{SL}$with both applications and the mapping of the smart lighting model to the semantic interoperability architecture is shown in Fig. 5.4. With respect to the smart lighting model, LSU_{mn} maps to ***s*****$_{mn}$** in the application A_l and ***p*****$_{mn}$** in the application A_h. AU_{mn} maps to ***a*****$_{mn}$** in A_l and ***c*****$_{mn}$** in A_h. The ICA executes at ***gw*****$_{SL}$** in A_l whereas at every individual ***c*****$_{mn}$** in A_h.

The user preferences for the individual square grid cell are $\rho_{l,mn}$ for $g_{l,mn}$ and $\rho_{h,mn}$ for $g_{h,mn}$ in applications A_l and A_h, respectively. They are defined in the range of minimum and maximum illumination values, such that, $\rho_{l,mn} = [\, e_{l,mn|min}, e_{l,mn|max}]$ and $\rho_{h,mn} = [\, e_{h,mn|min}, e_{h,mn|max}]$. We defined the range of user preferences with minimum and maximum illumination values because natural light and other external light sources (for example, light from objects like the TV in the room) typically varies around 10 lux. We aim to avoid unnecessary triggering of actuators to adjust luminaries in case of slight changes in illumination that are imperceptible to the human eye. These preferences are stored

at $\boldsymbol{sb_{SL}}$ and integrated in the ontology graph (O_graph_{SL}). We can store user preferences at $\boldsymbol{sb_{SL}}$ in two ways. First, we can store these preferences at $\boldsymbol{sb_{SL}}$ through $\boldsymbol{mo_{SL}}$, where this will add user preferences in the application ontology graph at the initial stage of implementation. These are more static updates of user preferences and do not change during execution of the smart space**.** Second, we can update dynamic changes in user preferences through a specific *PiO*. For example, a user's smartphone (installed with a *PiO*) can update preferences based on user activities. This specific *PiO* for user preferences first joins the smart space SS_{SL} and then gets knowledge of the application ontology structure by making queries at $\boldsymbol{sb_{SL}}$ to update user preferences. It will update user preferences whenever needed by the user using the UPDATE transaction. Therefore, the user can update preferences at $\boldsymbol{sb_{SL}}$ using the smartphone after joining the smart space SS_{SL} .

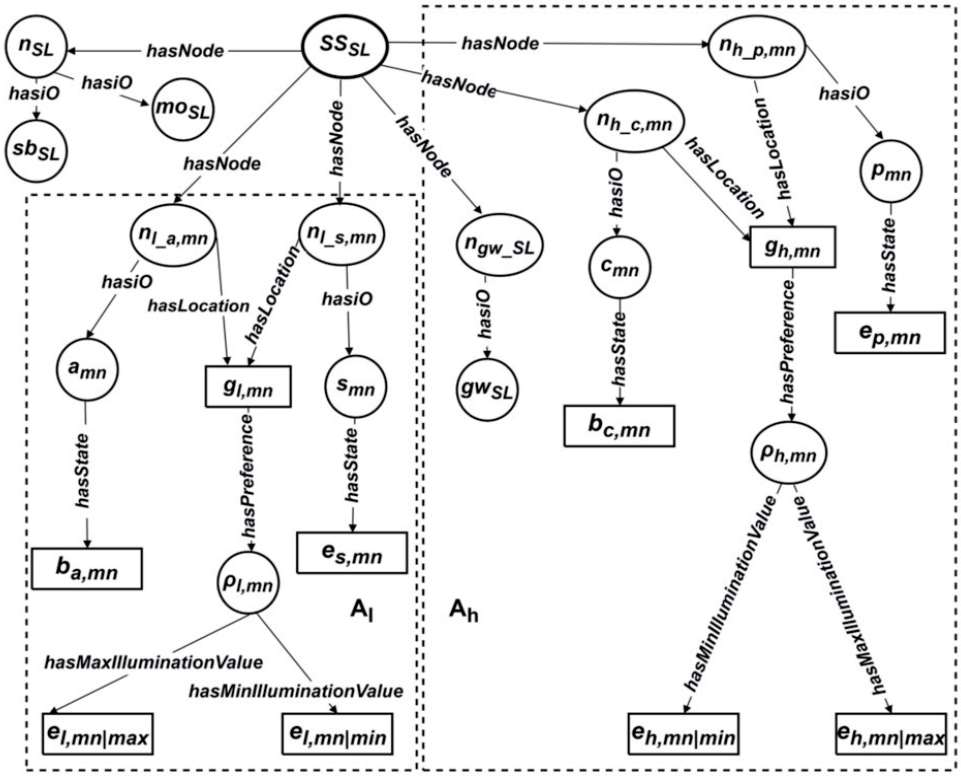


Figure 5.3. The basic application ontology graph of SS_{SL} (O_graph_{SL}).

We discuss the adaptation of LED luminaires based on user preferences in individual smart applications using semantic interactions in the following subsections, i.e., for the application A_l in Section 5.3.1 and for the application A_h in Section 5.3.2.

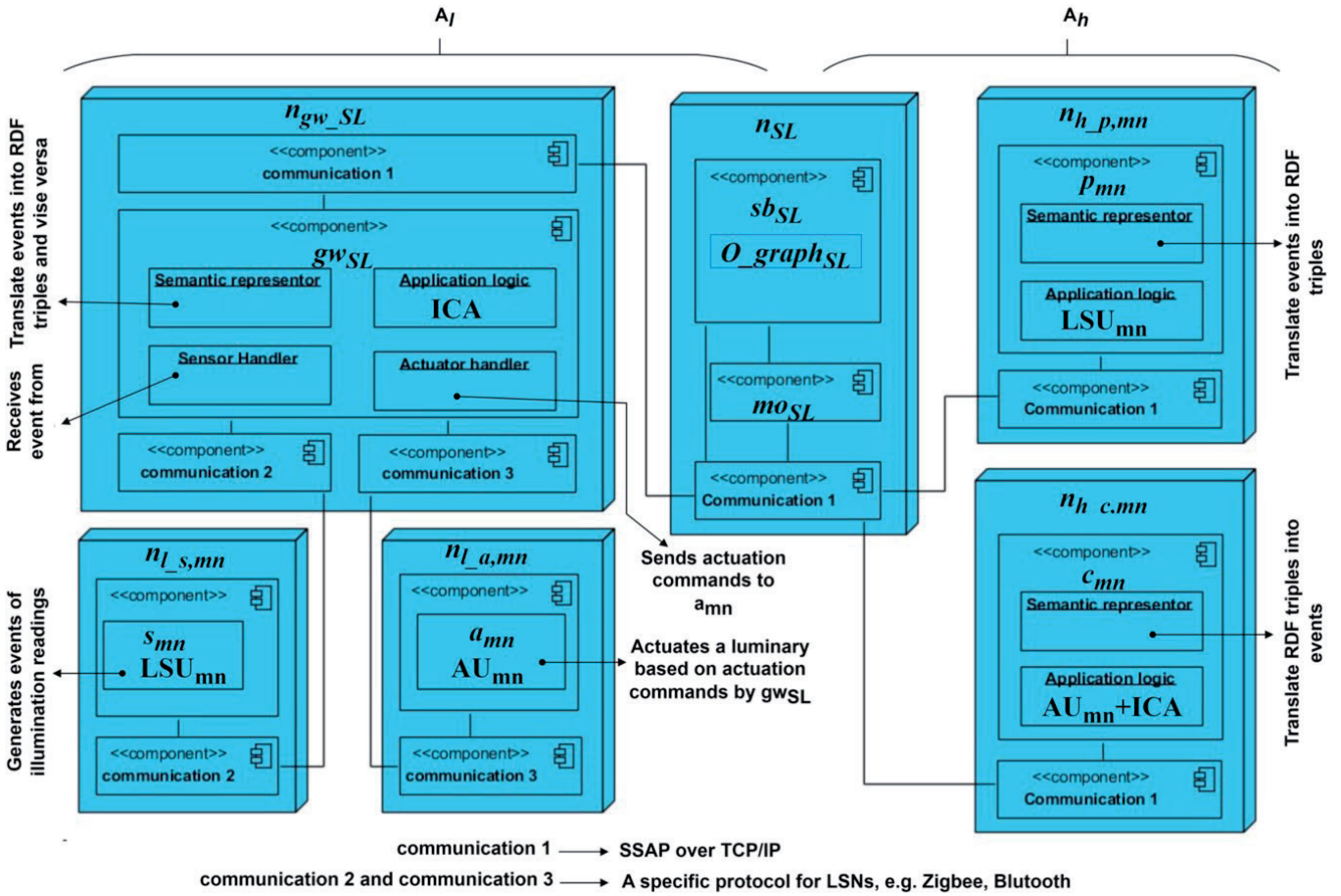


Figure 5.4. Deployment view of the smart space SS_{SL} with the mapping of the smart lighting model.

5.3.1. Adaptation of LED luminaries based on user preferences in A_l

Let us consider the following scenario to explain the adaptation of LED luminaries based on user preferences in A_l .

Scenario:
A change of illumination in $g_{l,mn}$ triggers an illumination reading event, resulting in the associated action of adjusting the luminary settings to maintain compliance with the user preferences.

Given: User preferences of minimum illumination ($e_{l,mn|min}$) and maximum illumination ($e_{l,mn|max}$) for a particular square grid cell $g_{l,mn}$, i.e., $\rho_{l,mn} = [e_{l,mn|min}, e_{l,mn|max}]$.

Objective: LED luminaries to adapt according to user preferences.

The sequence diagram of the scenario is shown in Fig 5.5. In Step 1, ***gw*$_{SL}$** joins at ***sb*$_{SL}$** and subscribes to get updates on user preferences, i.e., $\rho_{l,mn}$. In reply, ***gw*$_{SL}$** receives the JOIN and SUBSCRIBE confirmations. Therefore, ***gw*$_{SL}$** receives user preferences and will get further updates on user preferences from ***sb*$_{SL}$** .

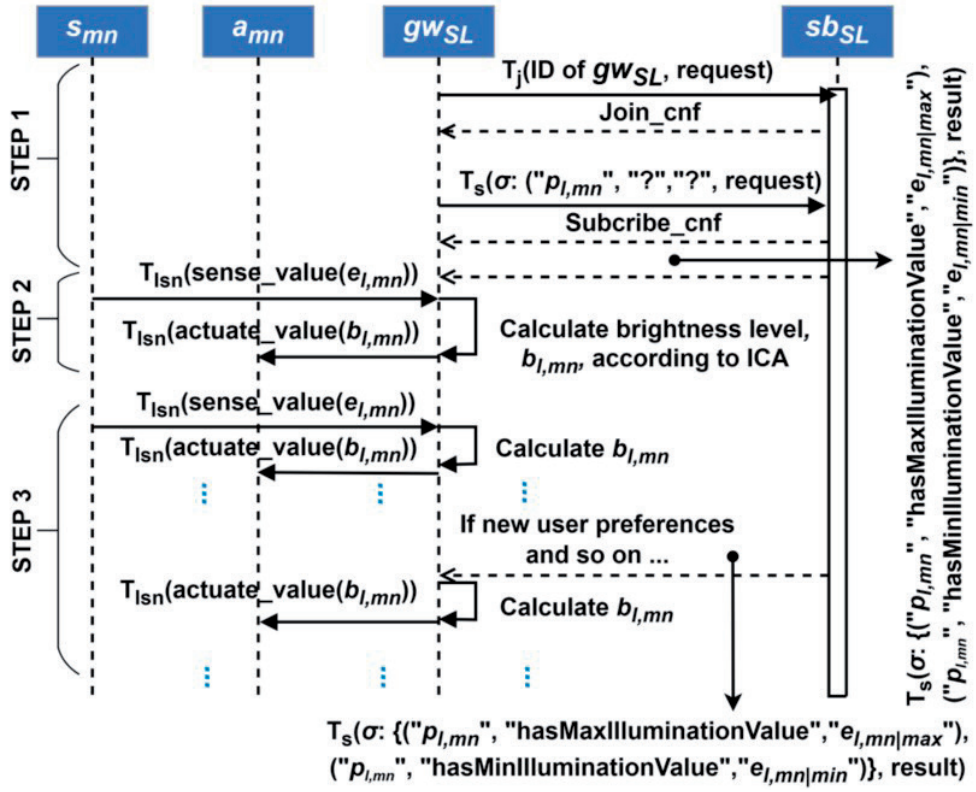


Figure 5.5. Execution of the scenario example in the application A_l.

In Step 2, $\boldsymbol{s_{mn}}$ generates events of illumination $e_{l,mn}$ and updates to $\boldsymbol{gw_{SL}}$ using the transaction T_{lsn}, i.e., $e_{l,mn} = E_{l,mn}$. Based on the user preferences $[e_{l,mn|min}, e_{l,mn|max}]$ and the illumination $e_{l,mn}$, $\boldsymbol{gw_{SL}}$ calculates the brightness level $b_{l,mn}$ according to the ICA, i.e., $b_{l,mn} = B_{l,mn}$ and updates to a_{mn} using the transaction T_{lsn}. Finally, the LED luminary attached to a_{mn} is readjusted to the brightness level $b_{l,mn}$.

In Step 3, $\boldsymbol{s_{mn}}$ updates the next event of illumination $e_{l,mn}$ at $\boldsymbol{gw_{SL}}$ when the illumination changes in g_(l,mn). Therefore, $\boldsymbol{gw_{SL}}$ calculates the new brightness $b_{l,mn}$ based on the new illumination $e_{l,mn}$ and the user preferences. The LED luminary again adjusts the brightness level $b_{l,mn}$. Similarly, if the user preference also changes by the user, for example, the new illumination range $[e_{l,mn|min}, e_{l,mn|max}]$ and updates at $\boldsymbol{sb_{SL}}$. Further, $\boldsymbol{gw_{SL}}$ receives the update of the user preference automatically because of the subscription established at $\boldsymbol{sb_{SL}}$. Then, $\boldsymbol{gw_{SL}}$ calculates the new brightness level $b_{l,mn}$ as in Step 2 and updates to $\boldsymbol{a_{mn}}$. Therefore, the luminary adapts the brightness level generated because of the new user preferences. The process of calculating new brightness level commands will continue as we get new events of illumination or new user preferences at $\boldsymbol{gw_{SL}}$.

Finally, LED luminaries can adapt the brightness levels calculated by $\boldsymbol{gw_{SL}}$ according to the user preferences in the application A_l. In this example scenario, we explained the mechanism of working with LSNs in A_l and further semantic interactions between GSN and SBSN. This discussion explained semantic interoperability within the counterpart of the semantic interoperability architecture.

5.3.2. Adaptation of LED luminaries based on user preferences in A_h

We consider a similar scenario to the one explained in Section 5.3.1 in order to explain the adaptation of LED luminaries based on user preferences in A_h.

Scenario Example:
A change of illumination in $g_{h,mn}$ triggers an illumination reading event, resulting in the associated action of adjusting the luminary settings to maintain compliance with the user preferences.

Given: User preferences of minimum illumination ($e_{h,mn|min}$) and maximum illumination ($e_{h,mn|max}$) for a particular square grid cell $g_{h,mn}$, i.e., $\rho_{h,mn} = [e_{h,mn|min}, e_{h,mn|max}]$.

Objective: LED luminaries to adapt according to user preferences.

The sequence diagram of the scenario execution in SS_H is given in Fig 5.6. In Step 1, $\boldsymbol{p_{mn}}$ and $\boldsymbol{c_{mn}}$ join at $\boldsymbol{sb_{SL}}$. In Step 2, $\boldsymbol{p_{mn}}$ generates the event of illumination $e_{h,mn}$ and updates at $\boldsymbol{sb_{SL}}$ using the transaction T_u, i.e., $e_{h,mn} = E_{h,mn}$. In Step 3, $\boldsymbol{c_{mn}}$ subscribes at $\boldsymbol{sb_{SL}}$ to get updates of both events $e_{h,mn}$ and user preferences $\rho_{h,mn}$ using the transaction T_s.

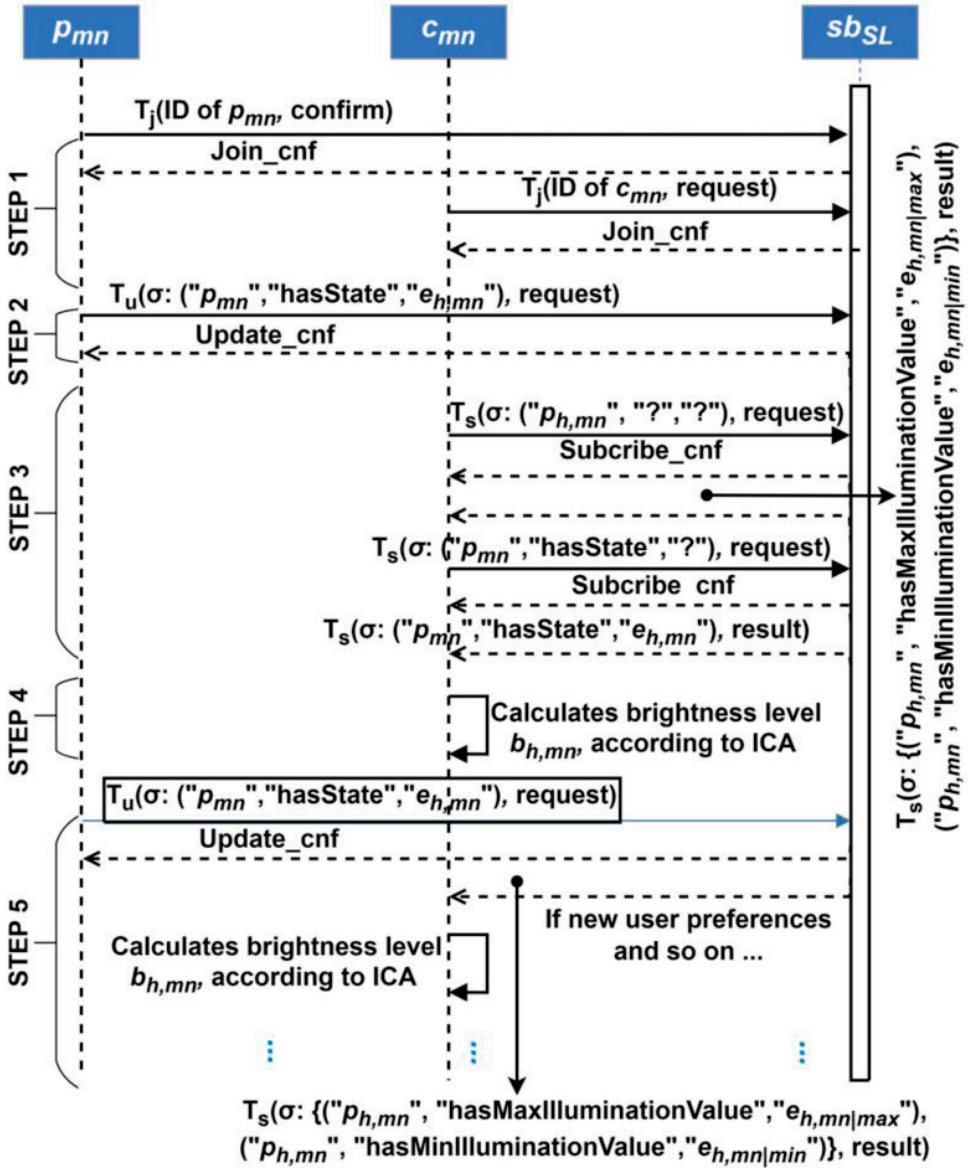


Figure 5.6. Execution of the scenario example in the application A_h.

In Step 4, based on the user preferences [$e_{h,mn|min}$, $e_{h,mn|max}$] and the illumination $e_{h,mn}$, $\boldsymbol{c_{mn}}$ calculates the brightness level $b_{h,mn}$ according to the ICA, i.e., $b_{h,mn}$ = $B_{h,mn}$ and updates the luminary output accordingly. Finally, the LED luminary attached to $\boldsymbol{c_{mn}}$ adjusts the brightness level $b_{h,mn}$. Thus, the LED luminaries adapt the brightness percentages according to the user preferences. Similar to the previous section in Section 5.3.1, the process of calculating new brightness level commands will continue until we get the next events of illumination in $g_{h,mn}$ or new user preferences from $\boldsymbol{sb_H}$ at $\boldsymbol{c_{mn}}$ as shown in Step 5.

Finally, LED luminaries can adapt the brightness percentages based on user preferences in the application A_h. In this example scenario, we explained the mechanism of working with HSNs in A_h and semantic interactions between HSNs and SBSN. This discussion explained semantic interoperability within the semantic interoperability architecture.

In real life, users and their preferences may change over time and a user may have different preferences for different activities performed in an activity subspace. Based on the user preferences for a given activity, the illumination in the activity subspace can be adjusted dynamically by changing the LED brightness levels using the smart lighting model. Different preferences of multiple users can cause conflicts in controlling the luminaries in the model. Generally, user preferences can be formulated in two ways. Firstly, consider the preferences from multiple users are taken into account in different activity subspaces within the smart space. In this situation, conflicts can be easily avoided by controlling the luminaries in those square grid cells that are specifically assigned to each user. This way ensures that the activity subspaces do not conflict with each other. Secondly, preferences from multiple users in a shared activity subspace can lead to conflicts. To address these conflicts, we propose a resolution method that prioritizes the highest-ranking user preference. To illustrate this conflict resolution method, we consider two users and their preferences within the same activity subspace.

- **Case 1**: Different user preferences for the same activity: In this case, the selection of user preferences in the smart lighting model is directly influenced by choosing a preference with the higher range of illumination values. This is because the activities are the same for both users.
- **Case 2**: Different user preferences for different activities: In this case, the choice of user preferences in the smart lighting model depends on both preferences and activities. Therefore, the activity with higher priority will take precedence over the one with lower priority. For instance, in a smart home within the same activity subspace, the activity of reading a newspaper will take precedence over watching TV. The priority levels of user activities can be defined based on general human perception. For example, it is understood that watching TV has a lower priority level than reading a newspaper in terms of lighting within the same activity subspace.

5.4 Smart Lighting Use Cases

We consider two smart lighting use cases: **Use Case 1 (UC-1)**: *Context-adaptive smart lighting* and **Use Case 2 (UC-2)***: Power-managed smart lighting*. UC-1 implements the smart lighting model for LSNs and semantic interoperability with an SBSN in a smart space. UC-2 implements the smart lighting model using the semantic interoperability architecture that

includes both LSNs and HSNs, where the nodes can share semantics over different applications in a smart space.

5.4.1 UC-1: Context-adaptive smart lighting

The existence of sufficient light is crucial for any task one may want to carry out. Human beings feel comfortable with different levels of illumination for different tasks or activities. For example, while low light levels are good for watching a movie, videoconferencing on a computer requires vivid lighting. This is because most webcams would not be able to capture good quality videos in low light conditions. Therefore, there is a growing demand towards smart lighting systems that can dynamically meet user requirements for different activities and in different places. In the context-adaptive smart lighting, the objective is to control illumination in activity subspaces based on user activities and preferences. To achieve this, we consider the following two user activities: reading a book that is laying on a table and Watching TV.

Hence, the key challenge lies in implementing this use case using the semantic interoperability architecture, incorporating LSNs and the smart lighting model. Ultimately, the luminaries must adjust light intensity according to various user preferences. The implementation of UC-1 is elaborated in Chapter 7.

5.4.2 UC-2: Power-managed smart lighting

Electric power usage constitutes an important source of financial outflow for building spaces. Therefore, electricity providers offer various schemes for billing. One scheme is that a building gets a quota of electric power that it can use. This way, electricity providers can cost-effectively plan their power generation. However, if a building uses more power than its assigned quota, a significantly higher electricity price is charged, leading to excessive power bills. Hence, the quota of electric power overflows must be avoided. The power usage in a building may vary based on the types of activities happening in the building at different times of the day. Let us take the simple example of office space with a single coffee machine that is used 12 hours per day to make 200 cups of coffee per week. In this typical setting, according to [5.46], a single coffee machine would consume around 200 kWh per annum (kWh/a) for actual use (spikes of power use while making coffee), 147 kWh/a for heating water when idle, and 18 kWh/a in stand-by mode. Note that more than half of the power consumed comes in the form of bursts (*i.e., when someone presses the button to make coffee*), while a rather smaller portion is consumed continuously and constantly due to idle operation. Therefore, in a huge office building with many of these machines, coffee break times together with other stuff could cause power consumption peaks, leading to power quota violations, if not handled properly. Thus, it is important to keep power usage just tightly below a given quota taking all power consumption variations into account. One approach to solve this problem is to purchase more power than

the building would need on average, creating a power quota margin for times of excessive electricity usage, which is suboptimal.

In UC-2, the power-managed smart lighting addresses this issue by implementing a priority mechanism to regulate luminaries in the rooms of a residential building. We focus specifically on the power usage of luminaries within the building in this use case. The priority mechanism ensures controlled power usage by luminaries, thereby adhering to a predetermined power usage quota for the entire building. In this approach, we divide rooms of the building into two categories: *i) high-priority rooms (HPR) that are allowed to use power according to whatever demand there is at that time, and ii) low-priority rooms (LPR) that are obliged to use the power that is leftover from the quota after the consumption of high-priority rooms.* If the maximum HPR power consumption by itself cannot exceed the available power quota of the building; then quota overflows can be prevented in all cases, while maximizing power utilization by allocating the remaining power budget to LPRs. To achieve this approach, we need to solve the following issues:

1) Firstly, individual rooms (either LPR or HPR) may contain smart nodes from various suppliers, possibly forming collaborative networks among themselves. To manage power for the entire building, collaboration of smart nodes in all rooms is necessary. This is difficult since smart nodes from different suppliers may operate over different system architectures due to lack of standardization. Ideally, a collaborative network that is installed using smart nodes of a given supplier should be easily extendable by products of another supplier. Hence, interoperability among different smart nodes becomes a key challenge. We can solve this challenge by the proposed semantic interoperability architecture.

2) Secondly, LPR should only include services that are low-priority by nature, since the functionality of LPR services depend on the availability of power budget remaining after power allocation to HPRs. Some examples of such low-priority services are low-priority lighting (*e.g., lighting in a parking lot*), and non-functional (*e.g., decorative, artistic*) lighting. Hence the key challenge is to control the luminaires in the building, where HPR gets priority to use the power quota assigned to the building, then LPR uses the remaining power quota. We can solve this challenge by the smart lighting model.

The challenge is to maintain the power quota of a residential building by controlling luminaries using the smart lighting model and exchanging semantics using the semantic interoperability architecture. We explain the implementation of UC-2 in Chapter 7.

5.5 Conclusions

This chapter introduced a novel smart lighting model that is usable together with the proposed semantic interoperability architecture, allowing for efficient deployment of the illumination control algorithm on both low and high-capacity smart nodes. This proposed architecture facilitates the mapping of the model with shared semantics within a smart space. Additionally, the model offers a solution for maintaining illumination in indoor spaces based on user preferences and activities. Moreover, the conflict resolution method in the smart lighting model can avoid conflicts of preferences by multiple users. Finally, we have drawn two use cases of smart lighting applications for experimenting in Chapter 7.

Chapter 6

Smart Space Life Cycles

In Chapter 5, we proposed and mapped the smart lighting model to the semantic interoperability architecture, presenting two use cases (UC-1 and UC-2) to illustrate the practical implementation of smart spaces. To lay the basis for the subsequent implementation detailed in Chapter 7, we shift our focus to the perspective of smart space developers in this chapter. Here, we delve into the life cycles of smart spaces, providing essential insights that serve as a foundational understanding before delving into the practical aspects in the later chapters. The primary focus of this chapter is on the life cycle steps of a smart space, facilitated by the proposed semantic interoperability architecture. Through this exploration, our objective is to evaluate the utility of the semantic interoperability architecture within the development process and for the developers involved. This investigation of smart space life cycles is aimed at assessing the practical implications and benefits that the proposed architecture brings to the overall development and implementation of smart spaces.

Some researchers are actively working to find solutions for the development of smart spaces, and a recent example is the 'Matter' project (formerly known as Connected Home over IP or CHIP)[14]. This collaborative initiative involves Amazon, Apple, Google, Samsung Smart Things, and the Zigbee Alliance, with the goal of simplifying the development of smart home products and enhancing compatibility among consumer devices. While Matter focuses on achieving interoperability between smart home devices and IoT platforms from different providers, it is primarily centered around the 'smart home' and does not emphasize semantic interoperability. In this thesis, we employed the SSAP protocol to achieve semantic interoperability among smart nodes from different providers. The SSAP protocol streamlines node connectivity by facilitating semantic interactions within a smart space. However, relying solely on the SSAP protocol is insufficient for designing a comprehensive smart space. Recognizing this, Chapter 3 introduced the semantic interoperability architecture, ensuring seamless collaboration among all smart nodes within smart spaces. This proposed architecture allows smart space developers to concentrate on creating a user-friendly application experience, freeing them from the burden of developing and maintaining proprietary protocols. Thus, from the perspective of smart space developers, we delve into the detailed design of smart spaces through the life cycle processes outlined in this chapter. For instance, a standard life cycle of a software system comprises six

14 https://csa-iot.org/all-solutions/matter/

distinct stages: analysis, design, implementation, testing, deployment, and maintenance. Examining the life cycle perspective of a system provides valuable insights into the tasks associated with each stage, the stakeholders accountable for and impacted by these tasks, and the challenges that may arise. In our investigation of life cycle requirements within smart space architectures, we follow the approach outlined in [6.1].

In this thesis, our focus centers on a case study involving the implementation of smart lighting applications within smart spaces utilizing the proposed architecture. Consequently, we introduce the smart space life cycle, explaining the details of designing a smart space while incorporating the deployment of the smart lighting model. The smart space life cycle comprises three integral components: the smart node life cycle, the smart service life cycle, and the smart application life cycle. These components form a holistic view that we will delve into in subsequent sections, providing an in-depth exploration of each life cycle to offer a comprehensive understanding of their interplay in the context of smart space implementation.

6.1 Smart Node Life Cycle

The life cycle of a smart node shown in Fig 6.1 can vary depending on the specific context and application. Determining the purpose of smart nodes is foundational to building a smart space that is cohesive, efficient, and capable of meeting its intended objectives. Therefore, first, we need to determine the purposes of smart nodes in a smart space. This involves identifying the specific functions and roles they serve within the smart space. Here are some important points to help determine the purposes of smart nodes:

- **Define overall objectives and identify use cases**: We begin by defining the overall objectives or goals of the smart space, clarifying what it aims to accomplish and the issues it seeks to address. Additionally, we identify various use cases or scenarios within the smart space that involve nodes. These may include applications such as environmental monitoring, smart home systems, energy management, or healthcare monitoring. Each use case may necessitate specific functionalities or capabilities from the smart nodes.
- **Analyze system requirements**: We conduct an analysis of the system requirements for the smart space to ascertain the essential functions and capabilities of the nodes. This encompasses factors like data collection, processing, communication, actuation, and integration with other nodes.
- **Consider interoperability**: We evaluate the interoperability requirements within the smart space, discerning how the nodes will communicate and cooperate with each

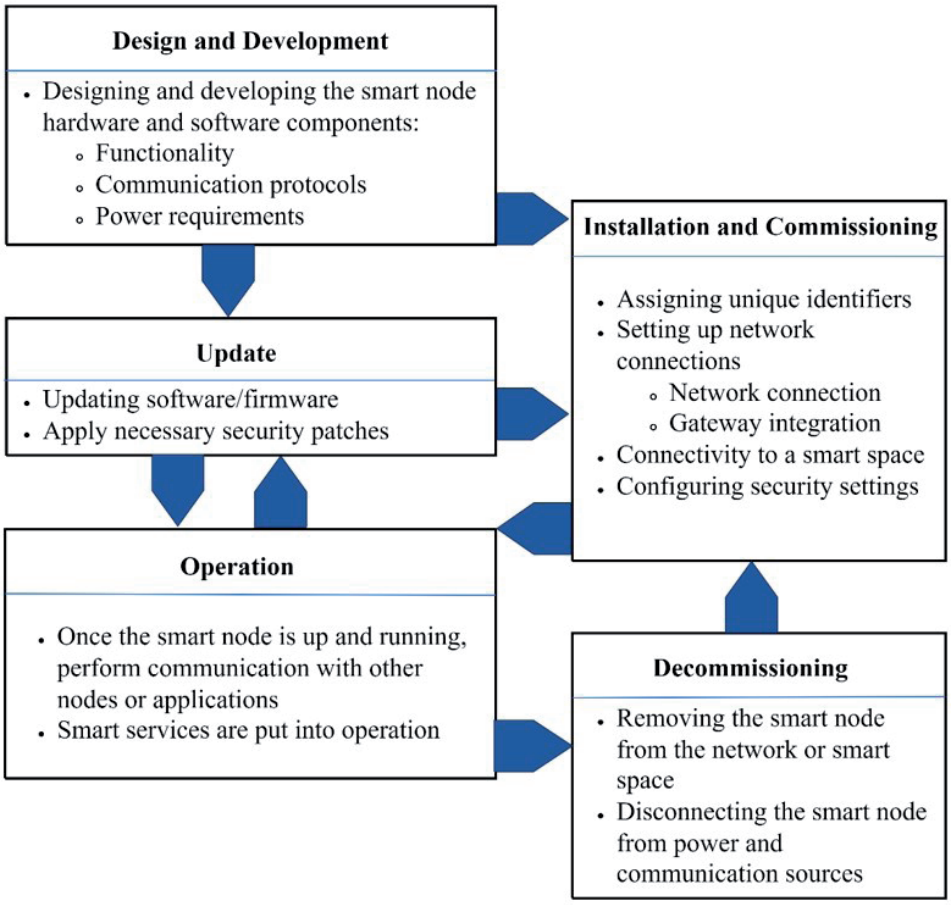


Figure 6.1. Smart node life cycle.

other, as well as with external systems. We also identify any particular protocols or standards that must be supported by the smart nodes.

- **Scalability and flexibility**: We take into account the scalability and flexibility requirements of the smart space, assessing whether the nodes should have the capacity to adapt to evolving needs, accommodate future expansions, or support various configurations and deployments.
- **Stakeholder input**: We collect input from stakeholders, which includes end-users, domain experts, smart space developers, and other pertinent parties. Their insights and perspectives are invaluable in identifying the specific needs, functionalities, or roles that the smart nodes should fulfil.

In this thesis, we delve into various smart lighting use cases for experimental analysis. To comprehensively explore these use cases, we identify light sensor and actuator nodes as

integral components of smart nodes within smart spaces. These nodes play a pivotal role in the implementation of lighting applications, allowing for the seamless integration of smart lighting systems. By focusing on these smart nodes, we aim to gain valuable insights into the practical applications and performance of smart lighting solutions. Additionally, our investigation extends beyond theoretical considerations to practical implementations, providing a holistic understanding of the role of light sensor and actuator nodes in enhancing the efficiency and adaptability of smart spaces.

6.1.1 Design and development of smart nodes

This step involves designing and developing the hardware and software components of the smart node. Once the purpose is determined, the design and development phase includes defining the functionality, communication protocols, and power requirements.

- **Functionality:** Smart nodes have various functionalities that enable them to collect, process, and transmit data, interact with other nodes, and contribute to the smart space. Here are some common functionalities of smart nodes:
  - **Data sensing and collection**: Smart nodes are equipped with sensors capable of detecting various types of data, such as temperature, humidity, motion, light, sound, and more. They gather data from their surroundings and convert it into digital signals for further processing.
  - **Data processing**: Smart nodes often have built-in microcontrollers or processors that can process the data collected from sensors. This processing may involve data filtering, aggregation, analysis, and even running simple algorithms to derive insights.
  - **Actuation**: In addition to sensing and processing data, some smart nodes have the ability to actuate and control other devices or systems based on the processed data or received commands. For example, a smart thermostat can control the heating or cooling system in a building.

In Chapter 2, we classified smart nodes into three different categories: *C*0, *C*1, and *C*2, based on their memory and processing power. These categories are further divided into two types of nodes within the semantic interoperability architecture: LSNs (*C*0 and *C*1) and HSNs (*C*2). HSNs host either *PiO*s or *CiO*s, while LSNs host either *SiO*s or *AiO*s. The smart nodes, both HSNs and LSNs, possess the capability to either produce or consume information within a smart space through sensing and actuating functionalities, respectively. To facilitate this, we configure software components by installing either *PiO* or *CiO* or both on smart nodes. Each node is associated with an intended service that corresponds to a specific *PiO* or *CiO*. For example, a light information service associated with a light sensor node generates light information using the installed *PiO*. More detail on services is discussed in Section 6.2.

- **Communication protocols**: The choice of communication protocol depends on factors such as the application requirements, range, power consumption, and the size of the smart space. Smart nodes are designed to communicate with other nodes or a central node within the smart space. They can use various communication protocols such as IP, Bluetooth and Zigbee to share data. For instance, we choose IP for HSNs and Zigbee for LSNs for experiments in Chapter 7.

- **Power requirements**: The power requirements of smart nodes in smart spaces can vary significantly depending on several factors, including the node's functionality, communication technology, operating environment, and power management techniques. Generally, smart nodes are designed to be energy-efficient to prolong battery life in wireless devices and reduce overall power consumption in wired systems. For instance, we use battery operated LSNs and external power supply for HSNs (explained later in this section in detail).

6.1.2 Installation and commissioning of smart nodes

Proper installation and commissioning ensure that the smart nodes function correctly and are integrated into the smart space. Here are some important points to consider for achieving successful installation and commissioning:

- **Assigning unique identifiers:** Assigning unique identifiers to smart nodes is a critical step in the manufacturing and deployment process. These unique identifiers help distinguish each smart node from others. Each smart node is assigned a unique serial number during the manufacturing process. This number can be a combination of letters or numbers. Serial numbers are usually printed, or stored electronically on the node. The smart space developer uses these numbers for reference. The unique ID is provided to each smart node by the developer itself. In smart spaces, developers use the taxonomy of nodes that is suitable for ontology development. The taxonomy of smart nodes for ontology development was explained in Chapter 4.
- **Setting up network connections**: Setting up network connections for smart nodes is needed for establishing communication between the nodes and the SBSN in smart spaces. The process can vary based on the communication protocols and network infrastructure involved. Generally for smart spaces, we look for the following points:
  - **Network connection**: In this thesis, an IP connection to HSNs is essential for participation in smart spaces, as it is required for SSAP. Additionally, a suitable communication protocol for LSNs is needed to connect with the gateway node.
  - **Gateway integration**: If the smart nodes communicate through a special gateway node, ensure that the nodes (LSNs) are properly integrated with the gateway node to facilitate communication with other nodes in the smart space. The gateway

node must have an IP connection to participate in the smart space on behalf of all other nodes (LSNs) connected to it.

- **Connectivity to the smart space**: When the network connection is active for smart nodes, it enables them to access smart space services, receive updates, and interact with other smart nodes or users with the help of the SBSN. For instance, all smart nodes are connected to the SBSN of the smart space through a joining process involving basic operations, as explained in Chapter 4 and depicted in Fig 4.7. The smart nodes can access the SBSN via the joining process using two methods. First, they can utilize the Avahi mDNS[15] service to discover the SBSN. Second, smart nodes can use the static IP address of the SBSN to join the smart space. Ultimately, the nodes within the smart space assume the role of either producers or consumers of information.

- **Configuring security settings**: When configuring security for nodes, it is important to consider several aspects to protect the integrity, confidentiality, and availability of the smart space environment. Some important key points for configuring security for smart nodes involve authentication, access control, encryption and more. In this thesis, security patches are installed on smart nodes when SSAP is installed on them. SSAP provides a secure communication channel with the necessary security settings over TCP/IP.

6.1.3 Operation of smart nodes

The operation of smart nodes refers to the functioning and behavior of individual nodes within a smart space. These smart nodes are equipped with sensors or actuators, processors, and communication capabilities. Once a smart node is deployed, a smart service is put into operation. For instance, a sensor starts collecting data, processing data, and delivering the associated service to it. The smart service is explained in detail in Section 6.2.

6.1.4 Updates on smart nodes

Updating software on smart nodes is a critical process to ensure the devices have the latest features, improvements, bug fixes, and security patches. This may involve writing new code, modifying existing code, or integrating third-party libraries. This applies to both LSNs and HSNs, when there is a need to update functionalities or services on the nodes.

Updates are also driven by other factors, such as battery and energy efficiency. As the number of smart nodes increases, energy efficiency has become a significant concern. Generally, manufacturers have been focusing on optimizing power consumption to extend

15 avahi - mDNS/DNS-SD

the operational life of battery-powered devices. For instance, this primarily applies to battery-operated LSNs.

6.1.5 Decommissioning

Decommissioning smart nodes refers to the process of retiring or removing these nodes from operation within a smart space. This process is essential when smart nodes are no longer needed, when demands of the application requirements to remove or replace, or when they reach the end of their useful life and become obsolete in the smart space. Smart nodes can exit the smart space through the installed *iO* by using the LEAVE transaction. In cases where they fail to execute the LEAVE transaction, physical removal becomes necessary. Some common reasons for failure include end-of-life scenarios, technical issues, and system upgrades.

6.1.6 Example of the smart node cycle used in UC-1 and UC-2

In both UC-1 and UC-2, the hardware units of LSNs are designed at the TU/e IRIS lab by configuring various embedded devices together. Figure 6.2 illustrates the configuration of an SN, which consists of a Body Sensor Network (BSN) node and a Phidget precision light sensor. The Phidget sensor is connected to an Analog to Digital Converter (ADC) channel of the BSN node. The BSN node runs TinyOS, and its hardware specifications are listed in Table 6.1. The Phidget sensor can detect fluctuations in light levels at higher frequencies than the human eye. Its specifications are also listed in Table 6.1.

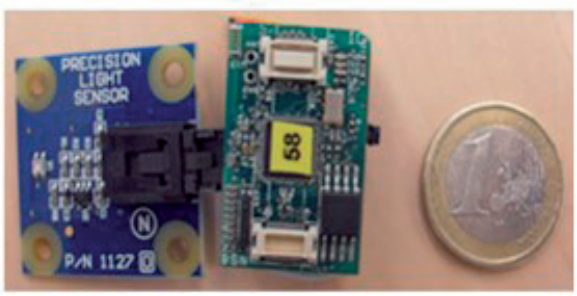


Figure 6.2. LSN: wireless sensor node SN.

Table 6.1. Hardware specification of the configured LSN: wireless sensor node.

Unit	Module	Parameter and specification
BSN	Processor (TI MSP430F1611)	Flash memory: 48 KB RAM: 10KB On-chip ADC resolution: 12 bits ADC channels: 8 channels DAC channels: 2 channels
	Radio Transceiver (TI CC2420)	Wireless communication standard: IEEE 802.15.4 (2.4 GHz) 250 kbps data rate Indoor range: 50 m Outdoor range: 125m
	EEPROM (AT 45DB321)	Flash memory: 4 MB SRAM buffers: 512/528 bytes Program/ Erase cycle: 100,000 cycles
Phidget	Phidget light sensor board (connected to the ADC channel of the BSN node)	Voltage input: (0-5 V) Response Time Max: 20 milliseconds Measurement Error Max: +-5% Light level min: 1 lux Light level max: 1000 lux (equivalent to TV studio lighting)

A LED luminary (referred to AN) is designed by combining a Phidget actuator with multiple LEDs attached to it. To assemble these hardware components into a single unit, we utilize a suitable case (a readily available plastic body cover in the market) for each LED luminary, as depicted in Fig 6.3. The Phidget actuator allows independent control of up to 64 LEDs. Each LED can have its current limit set individually, and its brightness can be adjusted from 0 to 100% within the set limit using current control. Standard LEDs typically have forward voltages below 2.75 Volts and can be easily used with the Phidget actuator by soldering them to a connector wire and inserting the wire into any board output. The default forward voltage is 2.75V, and the maximum current defaults to 20mA. The Phidget actuator board is controlled via USB, and its specifications are listed in Table 6.2.

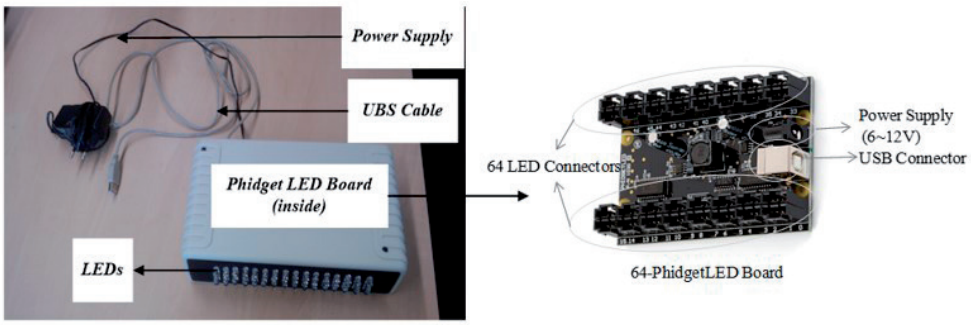


Figure 6.3. LSN: LED luminary (combination of an actuator and LEDs).

Table 6.2 The hardware specification of the configured LED luminary for the experiment purpose.

Phidget actuator board	Description
Controlled by	USB
Number of LED outputs	64
Recommended wire size (Power Terminal)	12 to 26 AWG
Supply voltage Min/Max	6/12 V DC
LED current limit Max	80mA

HSNs were designed by NXP for research purposes. Figure 6.4(a) and Figure 6.4(b) illustrate two types of HSNs: the LED luminary and the sensor node, respectively. The specifications of these HSNs are listed in Table 6.3.

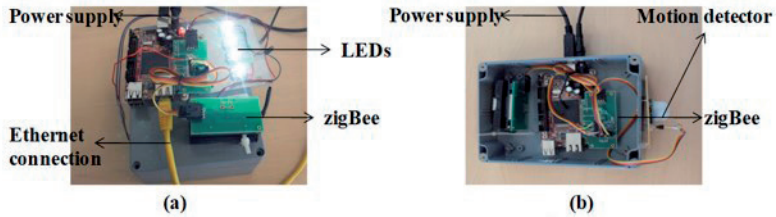


Figure 6.4 (a) HSN: LED luminary and (b) HSN: Sensor node.

Table 6.3 The hardware and software specification of HSNs (LED luminary and sensor node).

HSN	Features	Description
LED luminary	Luminary Function	Turn 'on' and 'off' and its brightness controlled
	Power source	AC220V
	Power conversion	LED 220V
	Connection	Zigbee and Internet
	Maximum light Output per luminary	525 (maximum 175 lumens per LED)
	Maximum power used per luminary	4500 milliwatts
	CPU	Arm cortex M0
	Operating system	µCLinux
Sensor Node	Sensor Type	Motion sensor
	Power source	AC220V
	Connection	Zigbee and Internet
	CPU	Arm cortex M0
	Operating system	µCLinux

The installation and commissioning of these designed smart nodes are carried out as follows: we assign a unique ID to each smart node using the ontology concept discussed in Chapter 4. For instance, when defining any smart node in the ontology, a URL is specified for that node. Additionally, we set up an IP connection for all HSNs and Zigbee for all LSNs.

HSNs become operational within a smart space when they join the SBSN using the JOIN transaction and initiate semantic exchanges. LSNs become operational within a smart space when they join to a gateway node (GSN) using Zigbee. The GSN further connects to the SBSN using the JOIN transaction. HSNs and GSN can exchange semantics through various transactions (REMOVE, UPDATE, SUBSCRIBE, and UNSUBSCRIBE) at the SBSN using the SSAP protocol, as detailed in Section 4.2.2. Smart nodes can be updated as needed to modify application logic. For this purpose, the developer responsible for the smart space identifies the update requirements necessary for the application logic. This involves a thorough understanding of the desired changes, whether it is bug fixes, the addition of new features, or additional functionalities. Once the update requirements are clear, the developer proceeds to programmatically update the smart nodes. This process ensures that the application's logic is in sync with the latest specifications and functionality of the smart nodes. After the update is completed, the installation and commissioning process will be repeated, and finally, the smart nodes become operational in the smart space again. The ability to update smart nodes dynamically allows for the adaptability needed in evolving smart spaces. Finally, decommissioning becomes necessary when smart nodes are no longer required in the smart space or encounter issues such as end-of-life scenarios or technical problems that cannot be repaired. As discussed in Section 6.1.5, smart nodes can exit the smart space by utilizing the LEAVE transaction through the installed *iO*. However, instances may arise where the execution of the LEAVE transaction fails, necessitating physical removal.

We employ these designed smart nodes for conducting experiments on the use cases UC-1 and UC-2, as discussed in Chapter 7. According to our definition of a smart space, the hardware unit used for node n_h is either a producer or consumer HSN, as shown in Fig 6.4(a) and Fig 6.4(b). The hardware units for nodes n_S and n_A are either producer or consumer LSNs, as depicted in Fig 6.2 and Fig 6.3, respectively. Additionally, every smart space includes an SBSN node and, if any LSNs are present, a GSN node. All SNs communicate with the gateway node GSN using the Zigbee over IEEE 802.15.4 protocol, while all ANs utilize USB for communication. The GSN is an HSN node and can establish communication with the central repository node SBSN through SSAP over TCP/IP. The SBSN node is a powerful node that comprises a repository, which can be either a laptop or a smartphone. Lastly, all HSNs also communicate with the SBSN via SSAP over TCP/IP.

6.2 Smart Service Life Cycle

Smart service is a foundational concept within the realm of smart spaces, denoting a set of functionalities provided by an *iO*. It is designed to perform specific tasks or operations that can be accessed and utilized by other *iO*s. For instance, a smart service influences

data collected from sensors, processes it through *iO*s, and produces light-related information in a smart space. A smart service can also initiate actuation processes on a specific *iO* based on the consumed light information in a smart space. Smart services play a crucial role in transforming physical spaces into smart spaces. For example, smart lighting systems adjust brightness based on occupancy, natural light, and user preferences, optimizing energy consumption and enhancing user comfort.

The smart service life cycle shown in Fig 6.5 refers to the different stages that a smart service goes through from its design to its termination. This life cycle involves various processes and activities that ensure the effective development, deployment, and management of smart services within a smart space. The typical stages in the smart service life cycle include: Design, Deployment and Integration, Operation, Updates and Termination.

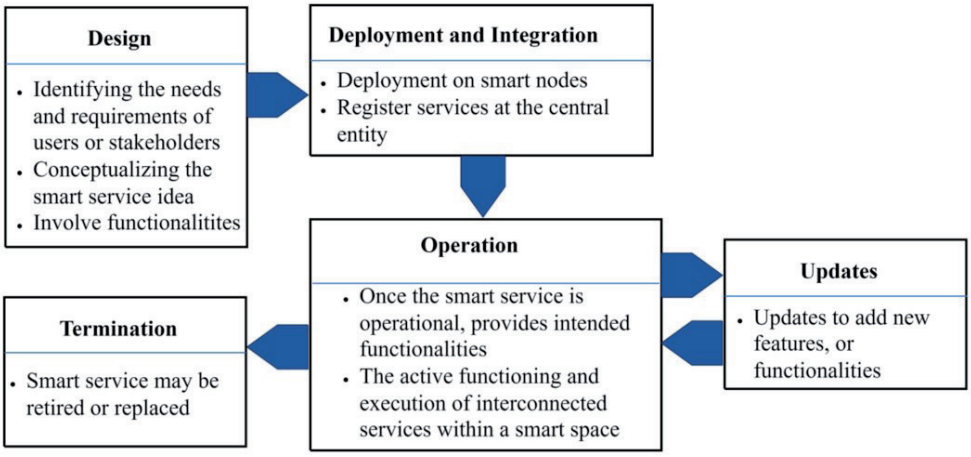


Figure 6.5. Smart service life cycle.

6.2.1 Design

The design of a smart service is a collaborative effort that involves designers, engineers, domain experts, and end-users. It is an iterative and evolving process that requires continuous refinement to meet the dynamic needs of the smart space and its users. The design process is vital as it lays the foundation for how the smart service will operate, interact with users and other smart nodes, and achieve its intended goals. Here are the key steps in the design of a smart service:

· Clearly defining the objectives of the smart service and understanding the specific needs and requirements of the target users or stakeholders are essential steps. This process helps in establishing the scope and goals of the service

- Designers need to adopt a user-centric approach, focusing on the needs, preferences, and behavior of the users who will interact with the smart service. User research and feedback play a crucial role in shaping the service design.

Next, we conceptualize the smart service idea to ensure that the intended goals are met. We consider the functionalities that a smart node can provide as a service. For instance, a smart node (namely light lamp) can have the functionalities of being on or off. These functionalities of the light lamp contribute to a service that we can call a 'light service'. Similarly, another smart node (namely presence sensor) can have the functionality of measuring the presence of a user. This functionality of the presence sensor is known as the 'presence status service'. Finally, the design of these services relies on the functionalities and application logics of smart nodes. This process is facilitated with the assistance of *iO*s, enabling the transformation of interaction states into RDF triples. The formation of RDF triples was explained in Chapter 4.

6.2.2 Deployment and integration

Once smart services are designed, we deploy them on smart nodes within a smart space. We register these smart services at the SBSN to integrate them into the smart space. After registration, these services become available for use within the smart space. For instance, the light and presence status services are deployed on smart nodes and then registered by them at the SBSN for further use by other nodes in the smart space.

6.2.3 Operation

Once the smart service is operational, it provides the intended functionalities. The operation of smart services in smart spaces refers to the active functioning and execution of interconnected services within a smart space. For instance, once the smart services are registered at the SBSN of the smart space, they become available for use by other smart nodes. These smart nodes can subscribe to these services to execute specific scenarios to interconnect the services. Let us consider an example scenario where the behavior of a light lamp is influenced by a presence sensor. The light lamp will subscribe to the presence status service, and based on the result, it will either turn on or off accordingly. As a result, both the light service and the presence status service will become operational within the smart space.

6.2.4 Updates

Smart services may undergo updates to add new features or functionalities. This stage involves updating the software firmware of smart nodes or making changes to the service. For instance, if we want to add a new functionality to the light lamp, allowing it to be operated with adjustable brightness levels, we can update the light service to include the brightness level along with the 'on' function.

6.2.5 Termination

At the end of its life cycle or when no longer relevant, a smart service may be retired or replaced. Proper decommissioning and data handling are crucial during this phase. For instance, the smart node itself can deregister by removing the registered service at the SBSN using the REMOVE Transaction.

6.2.6 Example of the smart service cycle used in UC-1 and UC-2

All light sensor nodes have the service to measure illumination at a specific point of their placement, as specified in Table 6.1 in Section 6.1.6. Additionally, all actuator nodes have the service to turn on with adjustable brightness levels and turn off, as specified in Table 6.2 in Section 6.1.6. For example, the actuator of the lamp has a smart service that defines a state as either on or off. This state needs to be expressed by a *PiO* in RDF triple format, i.e., (“Lamp”, “hasState”, “ON”) when the lamp is on and (“Lamp”, “hasState”, “OFF”) when the lamp is off. For deployment and integration, the *PiO* will register its service with the smart space using the INSERT transaction. Once registered, this service becomes available for other *PiO*s in the smart space. The service becomes operational when the lamp receives instructions from the smart space to either turn on or off. Additionally, the service can be updated to include new features, such as operating at a specified brightness level on the lamp. This new feature state must be expressed by the lamp in RDF triple format, i.e., (“Lamp”, “hasState”, “Brightness_level”), when the lamp is operating at a certain brightness level. Finally, the service is terminated when it is no longer registered in the smart space, achieved by removing the specific triples associated with the service using the REMOVE transaction.

In conclusion, the *iO*s create services for nodes based on the available features or functionalities of those nodes and express these features or functionalities in RDF triples. Following the creation of these RDF triples, the services must be registered at the SBSN by the *iO*s for subsequent use by other *iO*s. Finally, a smart application executes or interconnects these services in a smart space to run scenarios.

6.3 Smart Application Life Cycle

The smart application life cycle refers to the stages that a smart application goes through from its design to its deployment, operation, and eventual retirement. We defined smart application as a set of application scenarios. Throughout this life cycle, we delve into the process from the initial design and deployment to the eventual termination of application scenarios. The overview of the smart application life cycle is illustrated in Fig 6.6.

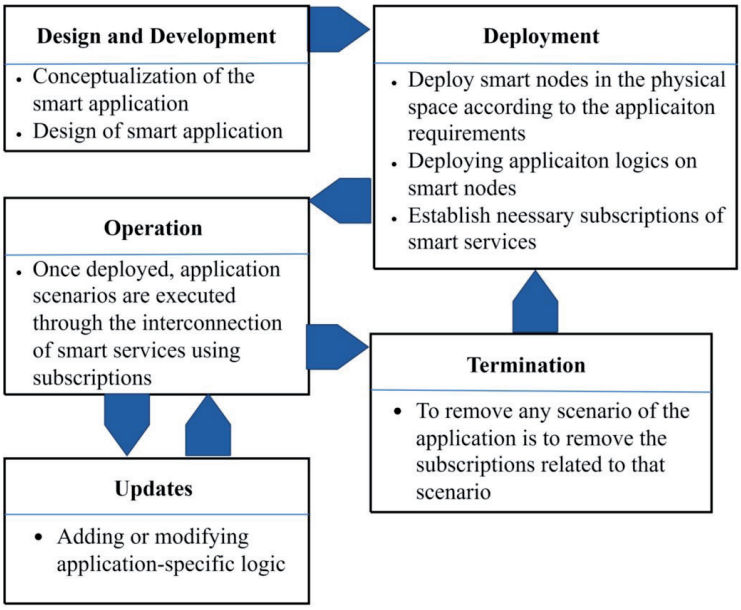


Figure 6.6. Smart application life cycle.

6.3.1 Design and development

The life cycle begins with the conceptualization and design of the smart application. During this stage, the objectives, requirements, and functionalities of the application are defined. The smart application is designed to address specific use cases and provide smart solutions within a smart space. In this stage, the actual development of the smart application takes place. Software developers, engineers, and domain experts work together to write the necessary code, create algorithms, for application logic.

6.3.2 Deployment

The smart application is deployed and integrated into the smart space. This involves installing the application logic on smart nodes to contribute to specific application scenarios, connecting them to the central entity or gateway, and ensuring smooth interaction with other smart services and nodes. To achieve this, we first deploy smart nodes in the physical space according to the requirements of the smart space and then deploy application logic to these smart nodes. Finally, we establish the necessary subscriptions of smart services required by nodes at the SBSN of the smart space.

6.3.3 Operation

Once deployed, the smart application becomes operational, delivering its intended services and functionalities. It undergoes execution through various application scenarios, during which smart nodes receive subscription results of the subscription made

at the SBSN. The interconnection of smart services through subscriptions facilitates the execution of these application scenarios seamlessly, ensuring that the smart application responds dynamically to the changing conditions within the smart space. Additionally, at any time, *iO*s can receive queries for information updates, further enhancing the flexibility and responsiveness of the smart application to changing requirements.

6.3.4 Updates

Smart applications may require updates to add new application logic based on newly added features and functionalities. For this purpose, we can update scenarios of the smart application by adding or modifying application-specific logic in the communicating smart nodes, where the communicating smart nodes are the nodes that participate in that particular scenario.

6.3.5 Termination

Eventually, a smart application may reach the end of its life cycle due to changing requirements, technological advancements, or obsolescence. In such cases, the application may be retired, replaced by a newer version, or replaced by a different application that better meets the current needs of the smart space. The straightforward way to remove any application scenario is to unsubscribe the smart nodes involved in that particular scenario.

6.3.6 Example of the smart application cycle used in UC-1 and UC-2

In the design and development step, we design application-specific scenarios by programming a dedicated application logic. This logic enables sharing information between different nodes, such as a luminary and a user's presence node in a room, through the SBSN. In the given example, the application logic is designed to determine whether the luminary should be turned on or off based on the user's presence status. Additionally, we develop an application ontology, building upon the basic concepts discussed in Section 5.3, as illustrated in Fig 5.3. In the deployment step, we position the sensor and actuator smart nodes within an indoor space. These nodes are paired in each square grid cell to facilitate the publishing and subscribing of information in the smart space. It is important to recall that when deploying the ICA algorithm, we install it on the gateway node in the LSN network within the smart space, and on every consumer node in the HSN network. We establish the necessary subscriptions required to run any scenario in the smart space. For example, if the luminary turns on or off based on the user's presence in the room, then the *iO* of the luminary smart node needs to subscribe to the user's presence status at the SBSN. The application scenario becomes operational when smart nodes initiate the sharing of information at the SBSN, thereby executing its intended objective: controlling the luminary based on the user's presence. In the update step, the application logic can be further updated to adjust the brightness level of the luminary along with turning it on based on user activity. This update can be implemented by making changes to the applica-

tion logic. Finally, we can terminate the application scenario by removing the established subscriptions for it. For instance, the subscription can be removed by the luminary smart node from the SBSN when removing this particular scenario.

The smart application life cycle is a continuous and iterative process. As technology advances and smart space requirements evolve, new scenarios of the applications are designed, developed, and integrated, while existing applications are maintained, updated, and retired as necessary. This dynamic life cycle ensures that smart applications stay relevant and continue to contribute to the efficiency and smartness of smart spaces.

6.4 Conclusions

In addressing the perspective of smart space developers, we have explained the life cycles inherent in smart spaces. These life cycles are divided into three key components: the smart node life cycle, the smart service life cycle, and the smart application life cycle. Each component is precisely explained, detailing the steps involved from the initial design phase through to termination. Building upon this foundation, Chapter 7 serves as the subsequent chapter, where we use these comprehensive life cycles to illustrate the implementation of two specific use cases, UC-1 and UC-2, introduced in Chapter 5. This approach allows for a seamless transition from understanding the theoretical aspects of smart space life cycles to their practical application in real-world scenarios.

Chapter 7

Implementations and Evaluations

In the previous chapter, we delved into the life cycles of smart spaces from the perspective of smart space developers. Building on this understanding, we utilize the smart space life cycle in the development and implementation of two use cases: context-adaptive smart lighting and power-managed smart lighting. This chapter takes a deep dive into the real-world implementation and evaluation of these use cases, presenting the results of our experiments. Additionally, we explore the properties of smart spaces by examining various example scenarios. A critical analysis and measurement of the delay performance metric between events and actions in these scenarios is conducted. In the final sections, we draw conclusions based on the outcomes of the experimental results and discussion about the potential and effectiveness of the proposed semantic interoperability architecture.

Firstly, let us discuss how the implementations and evaluations were conducted in this chapter. In Chapter 3, we introduced the semantic interoperability architecture, followed by a further elaboration on semantic interoperability in Chapter 4. Then, in Chapter 5, we proposed the smart lighting model and mapped it into the semantic interoperability architecture. Furthermore, Chapter 6 focused on the smart space life cycle, specifically involving smart lighting applications. Therefore, throughout these chapters, we have proposed two concepts: the semantic interoperability architecture and the smart lighting model built with it. For the implementation of these concepts, we consider the smart space SS_{SL} (as depicted in Fig. 5.4 in Chapter 5), where both UC-1 and UC-2 are implemented by following the life cycles. These two concepts are implemented and evaluated in the following manner:

- In UC-1 (context-adaptive smart lighting), our focus is on LSNs in smart spaces. Hence, we select the application A_l in the smart space SS_{SL} to implement UC-1, as explained in Section 5.3.1. We provide a detailed analysis of the novel smart lighting model, including experimental results and an assessment of the model's stability.
- In UC-2 (power-managed smart lighting), our focus is on sharing semantics across applications, which includes both LSNs and HSNs in smart spaces. Thus, we employ the smart space SS_{SL}, incorporating both applications A_l and A_h, to implement UC-2. The mechanisms for this implementation are explained in Section 5.3.1 and Section 5.3.2. Here, we present the semantic interactions across multiple applications in a smart

space. Additionally, we discuss the properties of the smart space and evaluate its performance based on the implementations in Section 7.3 and Section 7.4 respectively.

This illustration of the smart space encompasses the proposed concepts of smart spaces, the semantic interoperability architecture, and the smart lighting model. Finally, in UC-1, we present a demonstration of the smart lighting model in a smart space consisting of LSNs following the smart space life cycles. In UC-2, we present a demonstration of a comprehensive smart space that includes both LSNs and HSNs following the life cycles.

7.1 UC-1 Implementation and Evaluations

In UC-1, we defined the objective to achieve automatic light intensity adjustment by controlling illumination in activity subspaces based on user activities and preferences. Following the life cycle of smart nodes, we use LSNs: the light sensor node depicted in Fig 6.2 and the LED luminary shown in Fig. 6.3. The application logic defines the response and behavior of LED luminaries according to the light sensor's illumination reading and the user preferences. We consider following three application scenarios for the execution of the use case:

- *SC*_1: Reading activity under uniform lighting: This scenario assumes that all square grid cells have uniform lighting throughout the space.
- *SC*_2: Reading activity under non-uniform lighting: In this scenario, the square grid cells closer to the user have more lighting compared to the other grid cells.
- *SC*_3: Watching TV under uniform lighting: This scenario involves uniform lighting across all square grid cells specifically for watching TV.

A reading table of the user and TV are placed right in front of the chair and opposite the user, respectively. According to the smart lighting model, the activity subspace (table surface) is divided into 4 equal square grid cells represented by $g_{l,11}$, $g_{l,12}$, $g_{l,21}$ and $g_{l,22}$, where M=2 and N=2. Every square grid cell is installed with one light sensor and one LED luminary.

The deployment view of the smart space SS_{SL} (from Fig 5.4) based on the application A_l for UC-1 is shown in Fig. 7.1.

Given is: t the application A_l consisting of a set of scenarios (*SC*_1, *SC*_2 and *SC*_3) based on the following user preferences consisting of the minimum and maximum illumination values in the square grid cells, i.e., $\rho_{l,mn} = [e_{l,mn|min}, e_{l,mn|max}]$. The difference between the minimum and maximum values is 50 lux, and this selection of the range difference is

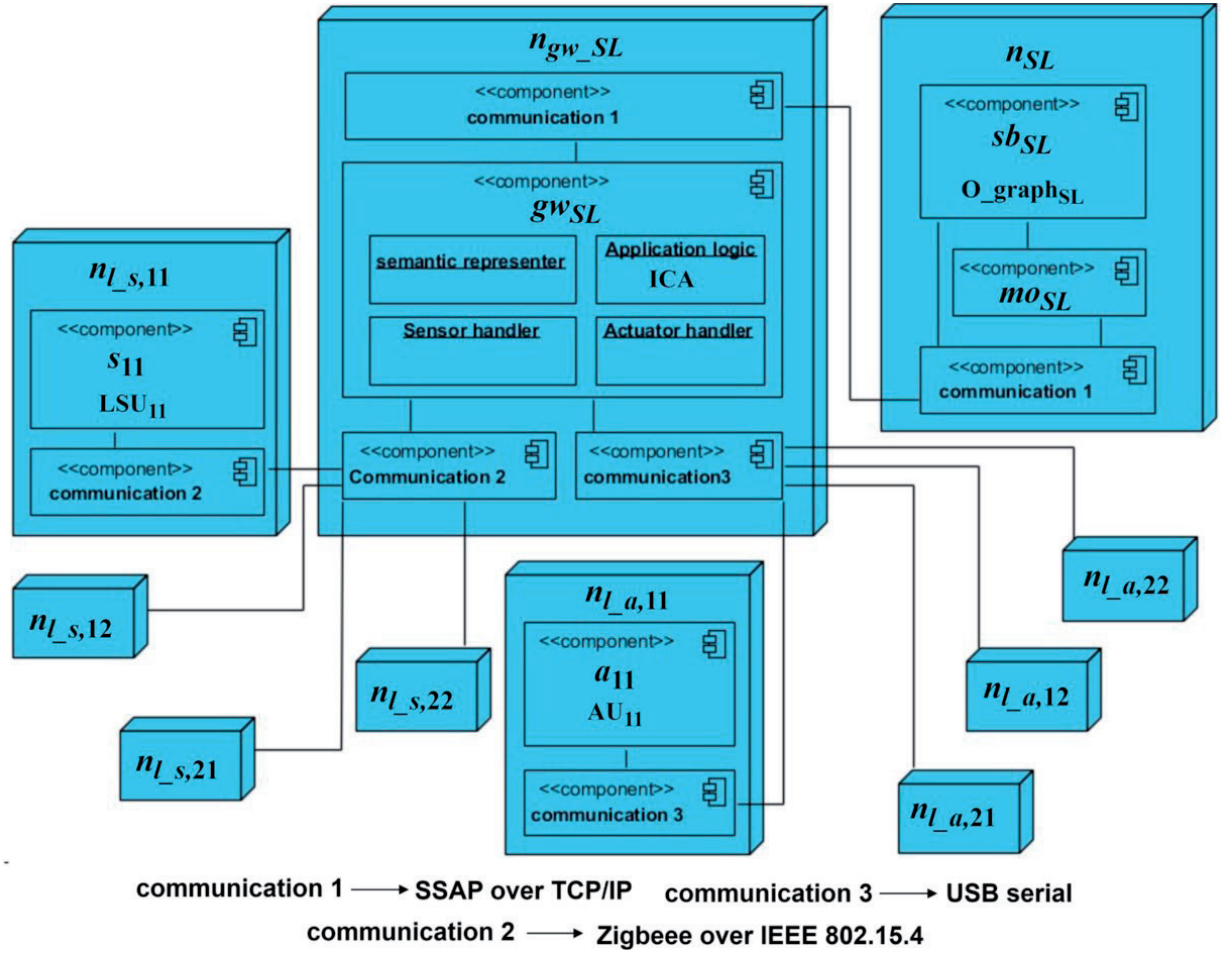


Figure 7.1. Extracted deployment view from the smart space SS_{SL} based on the application A_l.

random for these example scenarios. As discussed in Chapter 5, a small difference may lead to changes in illumination when there are slight variations in light intensity caused by external light or factors related to installed luminaries.

- In *SC*_1: $\rho_{l,mn}$ = {250 *lux*,300 *lux*} for all $1 \le m \le 2$ and $1 \le n \le 2$
- In *SC*_2: $\rho_{l,21} = \rho_{l,22}$ = {200 *lux*,250 *lux*}; $\rho_{l,11} = \rho_{l,12}$ = {250 *lux*,300 *lux*}
- In *SC*_3: $\rho_{l,mn}$ = {150 *lux*,200 *lux*} for all $1 \le m \le 2$ and $1 \le n \le 2$

The UC-1 prototype was constructed on the sixth floor of the Meta Forum building at the Technical University of Eindhoven in The Netherlands. We have deployed four light sensor nodes ($n_{l\_s,11}$, $n_{l\_s,12}$, $n_{l\_s,21}$, $n_{l\_s,22}$) with corresponding *SiO*s ($\boldsymbol{s_{11}}$, $\boldsymbol{s_{12}}$, $\boldsymbol{s_{21}}$, $\boldsymbol{s_{22}}$), as well as four luminaries with 36 LEDs in each device ($n_{l\_a,11}$, $n_{l\_a,12}$, $n_{l\_a,21}$, $n_{l\_a,22}$) with *AiO*s ($\boldsymbol{a_{11}}$, $\boldsymbol{a_{12}}$, $\boldsymbol{a_{21}}$, $\boldsymbol{a_{22}}$). These nodes are placed in alignment with the centers of square grid cells, as shown in Fig. 7.2. The interaction state of each light sensor node by $\boldsymbol{s_{mn}}$ represents the illumination value $e_{s,mn}$, while the interaction state of each luminary $\boldsymbol{a_{mn}}$ represents the brightness level $b_{a,mn}$. These *iO*s are connected to the gateway *iO*, $\boldsymbol{gw_{SL}}$, which is deployed on the gateway node n_{gw_SL}. Furthermore, $\boldsymbol{gw_{SL}}$ is connected to $\boldsymbol{sb_{SL}}$, which is deployed on the node n_{SL}.

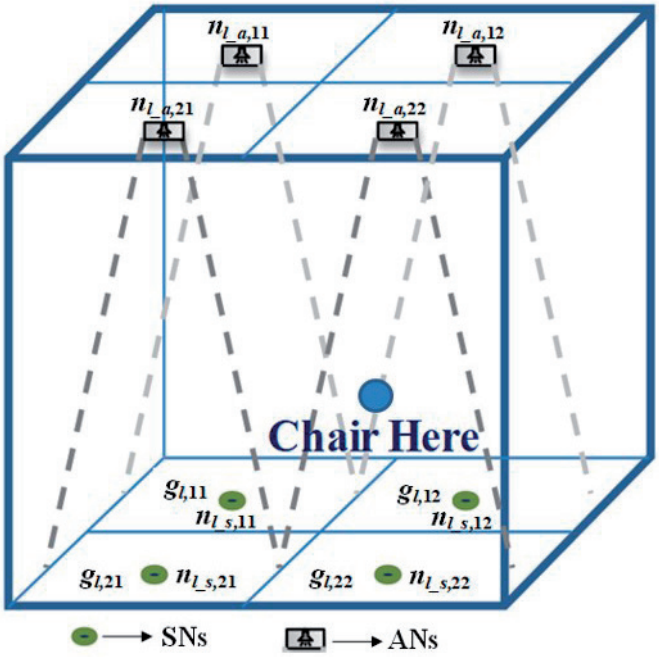


Figure 7.2. Placement of SNs and ANs; bird's eye view of the table surface.

In UC-1, we fixed all SNs on a reading table, while the ANs were positioned above the surface of the reading table on the ceiling, as shown in Fig. 7.2. In a real home lighting environment, the SNs can be integrated into furniture or relevant objects to ensure ac-

curate measurement of illumination at the desired locations. The light sensors measure illumination within the human perceptible light level range of 1-1000 lux. They can generate periodic or event-based illumination values at specific points and update these values to $\boldsymbol{gw_{SL}}$ using Zigbee over IEEE 802.15.4. In our implementation, the frequency of updates is set at 30 updates per second.

Each individual luminary can perform actions to adjust the brightness level from 0% (off) to 100% (fully ON). The luminaries achieve brightness control by utilizing Pulse Width Modulation (PWM), which regulates the current delivered to each LED. It is important to note that all ANs require an external power supply of 6~12V, and they are connected to the gateway node n_{gw_SL} via a USB connector. Therefore, communication between the gateway node n_{gw_SL} and the individual luminaries ANs occurs through USB serial communication. Finally, the gateway node n_{gw_SL} communicates with the node n_{SL} using the SSAP protocol over TCP/IP.

With reference to the smart lighting model, the illumination in individual square grid cells of an activity space is controlled separately. It is important to note that external light sources, such as sunlight, can also influence the sensor's readings. We conducted three sets of experiments for each scenario, considering three different conditions (κ_1, κ_2, and κ_3.

- In κ_1, only the installed luminaries are available, and there are no other sources of light. Note, the illumination from external sources (E_{ext}) has small variation, ranging from 0 to 10 lux, while there is no change by luminaries.
- In κ_2 and κ_3, E_{ext} ranges from 100 to 150 *lux* and 250 to 300 *lux*, respectively.

The execution of scenarios *SC*_1, *SC*_2 and *SC*_3 is shown in Fig. 7.3. The JOIN process remains the same as what was explained in Section 5.3.1, and we will not repeat it here. In Step 1, the scenario *SC*_1 for the reading activity under uniform lighting is updated to $\boldsymbol{gw_{SL}}$. In our implementation, a user updates the activity scenarios manually at $\boldsymbol{sb_{SL}}$, e.g., using a smartphone or any other producer node in a smart space. For example, a *PiO* is deployed on the smartphone, which updates user activities at $\boldsymbol{sb_{SL}}$, and $\boldsymbol{gw_{SL}}$ subscribes to the user activities at $\boldsymbol{sb_{SL}}$. As a result, $\boldsymbol{gw_{SL}}$ receives any updates of user activities. We explain the subscription process in Fig. 7.3, where the user updates the following triple for watching TV: ("Scenario", "hasState", "WatchingTV") and $\boldsymbol{gw_{SL}}$ subscribes for this type of information using the triple: ("Scenario", "hasState", "?").

$\boldsymbol{s_{mn}}$ updates the illumination value event $e_{l,mn}$ at $\boldsymbol{gw_{SL}}$ with a frequency of 30 updates per second, and the latest update of $e_{l,mn}$ is used in the calculation of actuation commands $b_{l,mn}$ at $\boldsymbol{gw_{SL}}$. We conduct three sets of experiments with conditions κ_1, κ_2, and κ_3, where all $b_{l,mn}$ are calculated by $\boldsymbol{gw_{SL}}$ according to the ICA in the smart lighting model. In the experi-

ments, we set ΔB to 1 percent and the conditions κ_1, κ_2 and κ_3 are executed sequentially. $\boldsymbol{gw_{SL}}$ sends the computed actuation command of brightness level $b_{l,mn}$ to $a_{l,mn}$ to activate the luminaries. As a result, the luminaries adapt according to the user preferences based on the scenario SC_1.

Similarly, the scenarios SC_2 and SC_3 are updated to $\boldsymbol{gw_{SL}}$. Based on the most recent update of the illumination value event $e_{l,mn}$ and the conditions (κ_1, κ_2, and κ_3), $\boldsymbol{gw_{SL}}$ recalculates the actuation commands $b_{l,mn}$ for both scenarios. Consequently, the luminaries are adjusted according to the user preferences in scenarios SC_2 and SC_3.

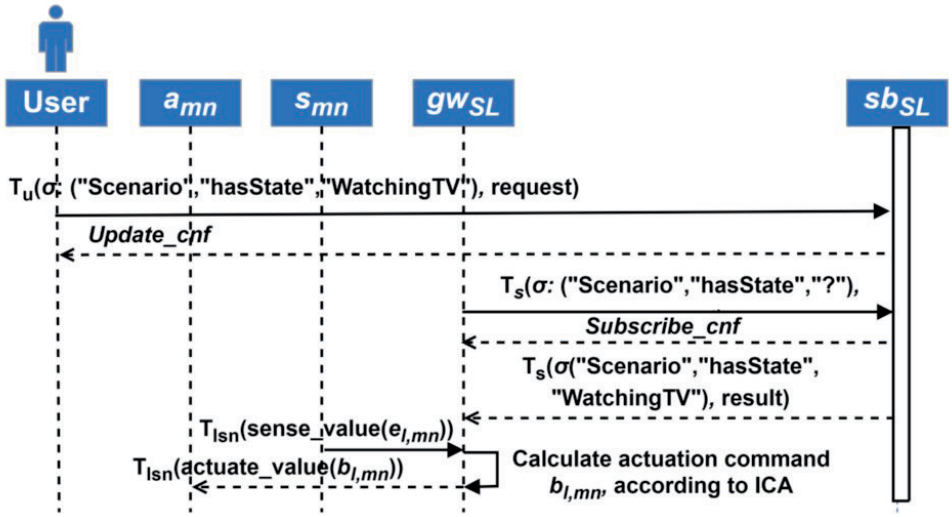


Figure 7.3. Execution of scenarios in UC-1.

The actuation commands calculated by gw_{SL} for scenarios SC_1, SC_2 and SC_3 in UC-1 based on the smart lighting model shown in Table 7.1.

- SC_1 results (*reading under uniform light*): The luminaries' brightness levels (by all $\boldsymbol{a_{mn}}$, adjusted automatically based on the actuation commands received by $\boldsymbol{gw_{SL}}$. This ensures consistent illumination in the activity subspace within the desired range of 250~300 lux. Figure 7.4(a) shows the activity subspace illumination, while Fig 7.4(b) displays the corresponding luminary brightness levels. The instant peaks outside the user preferences are caused by the transitions from κ_1 to κ_2, and from κ_2 to κ_3. For instance, in Fig. 7.4(a), the peaks above 300 *lux* are caused by the abrupt excess of light from external light sources. These peaks are promptly compensated by appropriately dimming the LED luminaries.
- SC_2 results *(reading under non-uniform light)*: When the illumination value due to external light sources exceeds a certain threshold $e_{ext} > e_{l,mn|max}$, $\boldsymbol{gw_{SL}}$ commands the

Table 7.1. Actuation commands $b_{l,mn}$ (in %) calculated by gw_{SL}.

	$b_{l,11}$	$b_{l,12}$	$b_{l,21}$	$b_{l,22}$	
SC_1	100	94	95	90	κ_1
	59	68	71	53	κ_2
	0	18	7	0	κ_3
SC_2	100	98	83	71	κ_1
	61	70	41	34	κ_2
	0	13	0	0	κ_3
SC_3	61	59	62	55	κ_1
	14	13	25	21	κ_2
	0	0	0	0	κ_3

luminary in the corresponding square grid cell $g_{l,mn}$ to 0%, as observed in κ_3 for $g_{l,21}$ and $g_{l,22}$ as shown in Fig. 7.4(d). It is important to note that the external illumination is not uniform across all grid cells. For example, in contrast to other grid cells, the illumination value $e_{l,12}$ of grid cell $g_{l,12}$ becomes equal to the maximum illumination value $e_{l,12|max}$ (300 lux) when the brightness level $b_{l,12}$ reaches 13%, corresponding to 34 lux. This implies that, in κ_3, the external illumination in square grid cell $g_{l,12}$ is approximately 266 lux. Furthermore, the brightness level result for $b_{l,11}$ is same in κ_1 of the scenarios SC_1 and SC_2 due to the identical preference requirement. Similarly, the brightness level result for $b_{l,12}$ is closer due to similar conditions.

- SC_3 results (*watching TV*): In κ_3, the external light source exceeds the maximum illumination value $e_{l,mn|max}$ for all square grid cells, resulting in the luminaries being turned off as shown in Fig. 7.4(f).

The experimental results demonstrate that the desired illumination for individual square grid cells can be achieved and maintained, as shown in Fig. 7.4 (a), (c), and (e), as long as the illumination from external light sources does not exceed the maximum illumination preferred by the user. In cases where the external light exceeds the desired level, the user can manually reduce it, for example, by closing curtains. The smart lighting model effectively operates by providing appropriate brightness levels to all luminaries ($\boldsymbol{\alpha_{mn}}$), enabling them to dynamically adjust the illumination in the activity subspace. This ensures that the illumination remains within the user-specified range. LSNs effectively communicate information in the use case through $\boldsymbol{gw_{SL}}$. Consequently, the context-adaptive smart lighting system successfully executes using the semantic interoperability architecture in conjunction with the smart lighting model.

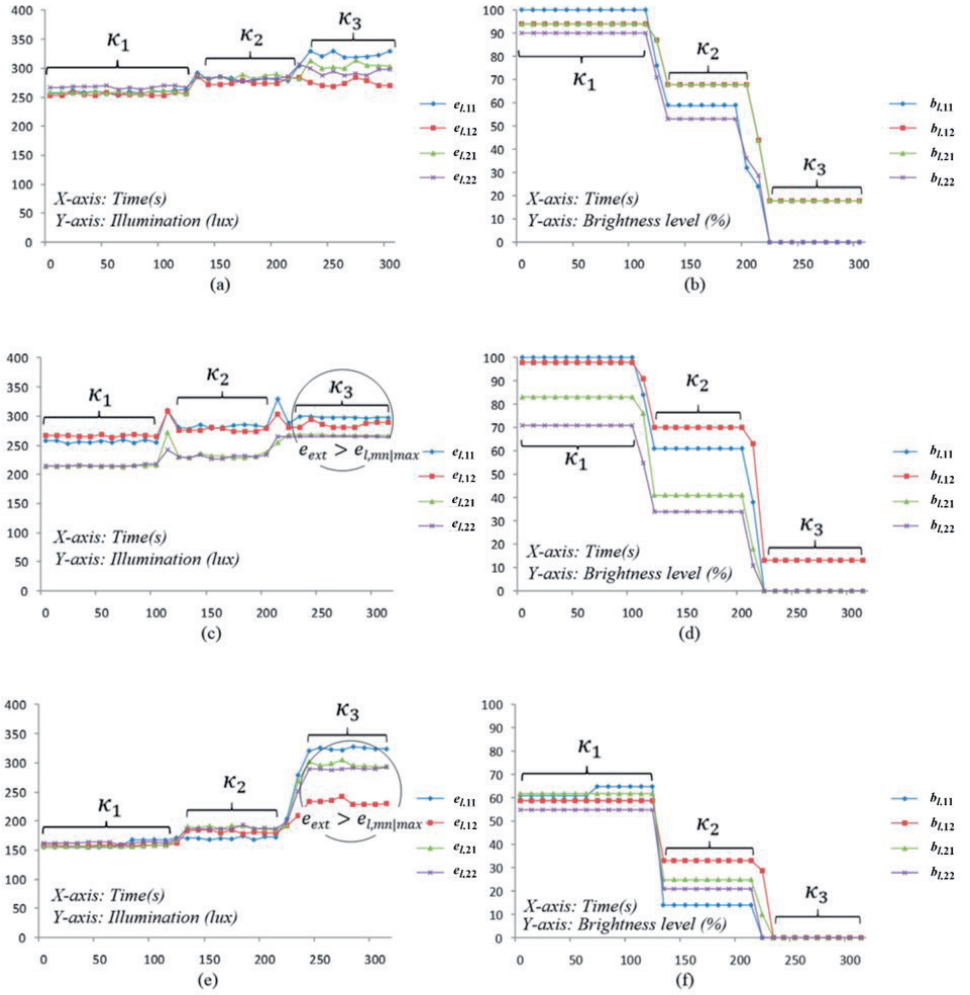


Figure 7.4. Change of illumination and brightness level over time based on the user activities: (a) illumination in SC_1, (b) brightness level in SC_1, (c) illumination in SC_2, (d) brightness level in SC_2, (e) illumination in SC_3 and (f) brightness level in SC_3.

We analyze the stability of the smart lighting model based on the aforementioned experimental results. In the model, stability is defined as the proper functioning of LED luminaries in the system, where the system converges to a state without changing external conditions, resulting in no more brightness level adjustments. In contrast, the instability of the model is flickering of lighting. Flickering of lighting refers to the rapid and repeated changes in light intensity, causing a visible pulsating or flashing effect. This can occur when there is a consecutive rapid change between states. Choosing a small ΔB can prevent flickering in situations with rapid changes in light intensity. Therefore, we selected ΔB to

be 1. Another important factor is coordinating the sensing and actuating frequencies in conjunction with the selection of ΔB.

Let us consider the following sensing and actuating frequencies within the smart lighting model.

> All $\boldsymbol{s_{mn}}$ report the measured illumination $\boldsymbol{e_{l,mn}}$ at their physical location to the $\boldsymbol{gw_{SL}}$ with sensing frequency $\boldsymbol{f_s}$.
>
> The $\boldsymbol{gw_{SL}}$ updates the actuation setting $\boldsymbol{b_{l,mn}}$ at all $\boldsymbol{a_{mn}}$ with actuating frequency of $\boldsymbol{f_a}$.

As mentioned in Table 5.2, the model has two possible states, namely St_1 and St_2, which can result in either an increase of +(ΔB) percent or a decrease of -(ΔB) percent in the luminary brightness level. To discuss stability in the model, we will investigate the following three possible cases: (1) ($f_s = f_a$), (2) ($f_s > f_a$), and (3) ($f_s < f_a$).

Case 1-($f_s = f_a$): This means that updates from all $\boldsymbol{s_{mn}}$ at the $\boldsymbol{gw_{SL}}$ are followed by updates on all $\boldsymbol{a_{mn}}$. As long as the changes in brightness level ΔB are small enough to prevent jumping between St_1 to St_2, flickering is avoided. Therefore, user preferences $\rho_{l,mn}$ must be defined in such a way that the difference between the maximum and minimum illumination values ($e_{l,mn|max} - e_{l,mn|min}$) is greater than the illumination produced by a ΔB percentage change in light output.

Case 2- $f_s > f_a$ **:** In this case, the updates from all $\boldsymbol{s_{mn}}$ at the $\boldsymbol{gw_{SL}}$ occur more frequently than the light adjustment updates to all $\boldsymbol{a_{mn}}$. Consequently, $\boldsymbol{a_{mn}}$ will receive the latest calculated $b_{l,mn}$ from the $\boldsymbol{gw_{SL}}$. If the illumination produced because of the latest calculated $b_{l,mn}$ is not large enough, meaning it is not greater than the difference between the desired illumination range ($|e_{l,mn|max} - e_{l,mn|min}|$), then it will work smoothly without any flickering. Otherwise, there is a possibility of flickering depending on $b_{l,mn}$. For instance, let us define the number of updates by $\boldsymbol{s_{mn}}$ on the $\boldsymbol{gw_{SL}}$ per $\boldsymbol{a_{mn}}$ update as x, where $x = f_s/f_a$. During each of the x updates on the $\boldsymbol{gw_{SL}}$, the target brightness level of $b_{l,mn}$ either changes by $\pm\Delta B$ percent or remains constant.

Let x_1 and x_2 represent the number of updates on the $\boldsymbol{gw_{SL}}$ in between two consecutive actuation commands, for the situations $e_{l,mn} < e_{l,mn|min}$, and $e_{l,mn} > e_{l,mn|max}$, respectively, such that $x = (x_1 + x_2)$. Therefore, the updates in luminary brightness in this case are given by equation (7.1).

$$b_{l,mn} = b_{l,mn} + (\Delta B \times x_1) - (\Delta B \times x_2) = b_{l,mn} + \Delta B(x_1 - x_2) \tag{7.1}$$

In this case, flickering is possible when the brightness level update $\Delta B(x_1 - x_2)$ causes a transition between situations $e_{l,mn} < e_{l,mn|min}$, and $e_{l,mn} > e_{l,mn|max}$. The worst case scenario occurs when $\{x_1,x_2\} = \{x,0\}$, resulting in a percentage change in brightness level of $+\Delta Bx$. Similarly, when $\{x_1,x_2\} = \{0,x\}$ causing the changes $-\Delta Bx$. The necessary and sufficient condition to guarantee LED stability (no flickering) at all times is that the $|e_{l,mn|max} - e_{l,mn|min}|$ difference must be greater than the amount of illumination generated by $\pm\Delta Bx$ percent of LED luminary brightness.

Case 3- $f_s < f_a$**:** In this case, the updates for light adjustment on all $\boldsymbol{a_{mn}}$ are more frequent than the updates for all $\boldsymbol{s_{mn}}$ by the $\boldsymbol{gw_{SL}}$. As a result, the LED brightness values are *unnecessarily* adjusted multiple times based on the $\boldsymbol{s_{mn}}$ readings, leading to increased messaging overhead between the $\boldsymbol{gw_{SL}}$ and $\boldsymbol{a_{mn}}$.

Finally, according to the frequency coordination cases, the convergence time of the smart lighting model depends on the desired illumination level, the updating LED luminary frequency (i.e. f_a), and the factor (i.e.$\pm\Delta B$). The convergence time is calculated based on the number of increments or decrements to meet the desired level of illumination, i.e., (f_a × *number of* $\pm\Delta B$). When ($f_s \neq f_a$), we either risk flickering or unnecessary communication for multiple LED luminary adjustments based on the $\boldsymbol{s_{mn}}$ readings. Therefore, choosing the ideal cases of ($f_s = f_a$) for both use cases is recommended.

In some use case scenarios, the sensing frequencies can be event based (i.e. trigger-adaptive rather than periodic-adaptive), where all $\boldsymbol{s_{mn}}$ send information to the $\boldsymbol{gw_{SL}}$ only when there is a state transition in readings. Upon receiving a state transition, the $\boldsymbol{gw_{SL}}$ will start calculating $\boldsymbol{b_{mn}}$ to increment or decrement the LED luminary brightness in the corresponding square grid cells until a new state transition occurs. However, this approach reduces network traffic and allows for an increased number of $\boldsymbol{s_{mn}}$ associated with a single $\boldsymbol{gw_{SL}}$, but it is highly dependent on specific scenarios. The choice between trigger-adaptive and periodic-adaptive is determined by the use case scenario, presenting a decision that needs careful consideration. In our analysis, we opted for the periodic-adaptive approach. This choice was motivated by our desire to examine the behavior of luminaries in a systematic manner. Utilizing periodic-adaptive allowed us to monitor changes in luminaries sequentially, providing valuable insights into their dynamics.

7.2 UC-2 Implementation and Experimental Results

The power-managed smart lighting use case assumes that different types of rooms (HPRs and LPRs) in a building aim to consume power according to assigned quotas. HPRs have priority in power allocation, meaning LPRs can only use the remaining power budget after

subtracting the consumption in HPRs. To facilitate this, we employ the proposed smart space SS_{SL} with the mapping of the smart lighting model, which requires establishing semantic interoperability within and between two rooms: one LPR and one HPR that contain smart nodes from various suppliers.

The application A_l in the smart space SS_{SL} belongs to LPR, while the application A_h belongs to HPR. The choices of LPR for A_l and HPR for A_h are based on their respective priorities and node capacities. Furthermore, we consider an equal number of smart node types in both applications, with M=2 and N=2, satisfying the conditions $1 \leq m \leq 2$ && $1 \leq n \leq 2$. As per the insights from the smart space SS_{SL}, the physical setup and deployment view of the use case UC-2 are shown in Fig 7.5 and Fig 7.6, respectively, where the specifications of the nodes are presented in Table 7.2.

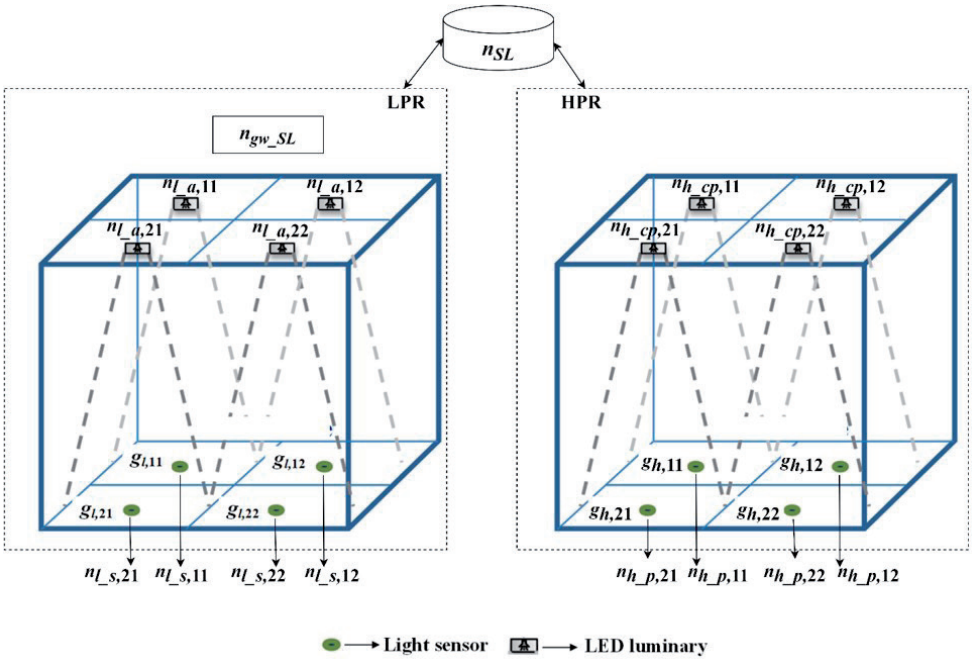


Figure 7.5. Physical setup of the UC-2.

We recall the following nodes and *iO*s in the smart space SS_{SL} for the use case UC-2 as follows:

- The application A_l is equipped with four SNs of light sensors, denoted as $n_{l\_s,mn}$, four ANs of LED luminaries, denoted as $n_{l\_a,mn}$, along with the gateway node n_{gw_SL} that is connected to all SNs and ANs in the smart space SS_{SL}. Similarly, the application A_h is equipped with four HSNs of light sensors, denoted as $n_{h\_p,mn}$ and four HSNs of LED luminaries, denoted as $n_{h\_cp,mn}$ in the smart space SS_{SL}. It is important to note that the

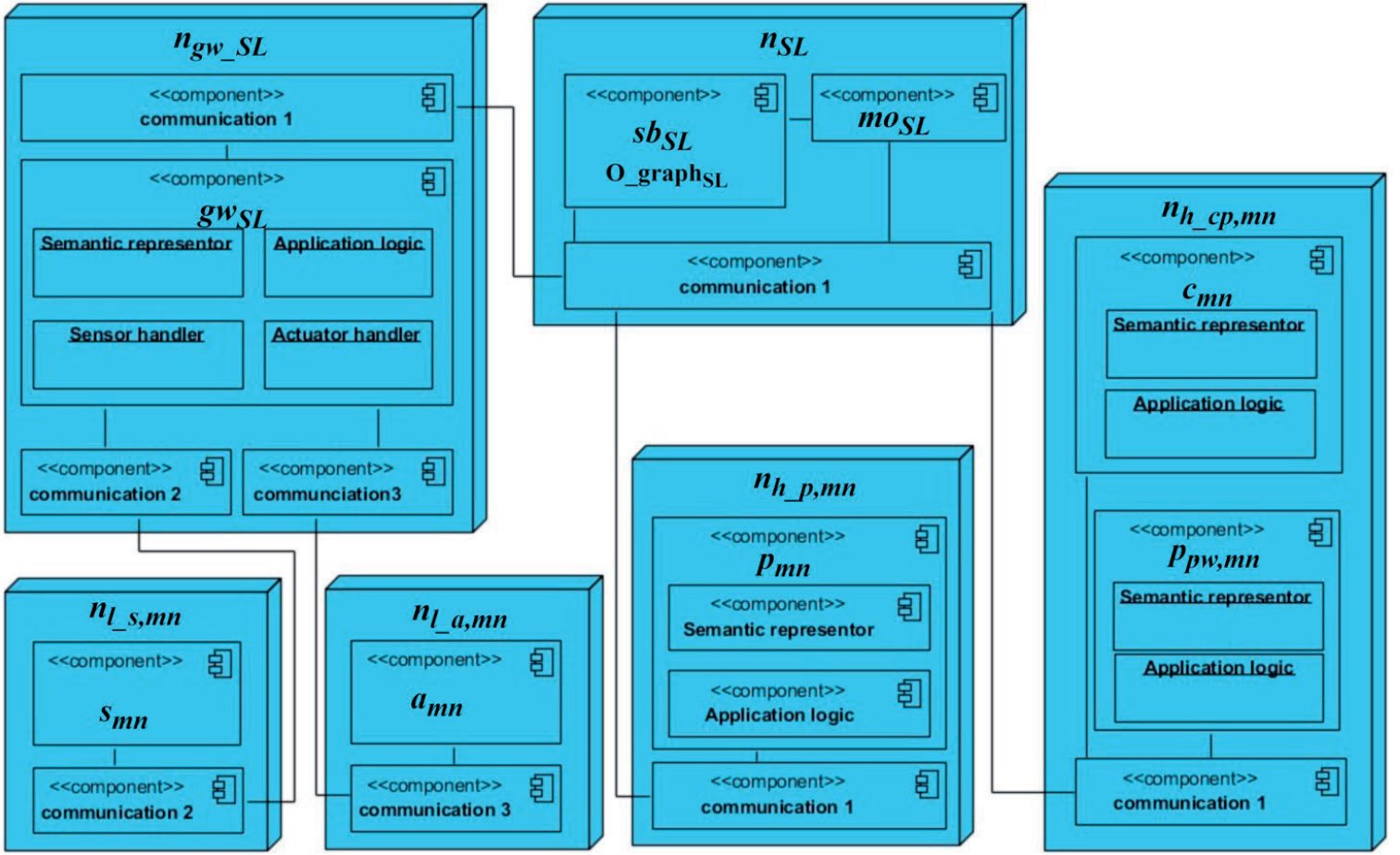


Figure 7.6. Deployment view of SS_{SL} for UC-2, where the communication 1 is SSAP over TCP/IP, communication 2 is Zigbee over IEEE 802.15.4 and communication 3 is USB serial communication.

only difference between the smart space SS_{SL} in Fig. 5.4 and the current setup is the inclusion of the node $n_{h\_cp,mn}$ instead of $n_{h\_c,mn}$, as we need to measure power consumption by individual luminaries in HPR.

- The nodes n_{gw_SL}, $n_{h\_p,mn}$ and $n_{h\_cp,mn}$ are also connected to the node n_{SL}, where the node n_{SL} has two *iO*s: $\boldsymbol{sb_{SL}}$ and $\boldsymbol{mo_{SL}}$.
- The node n_{gw_SL} is deployed with $\boldsymbol{gw_{SL}}$, while $\boldsymbol{s_{mn}}$ and $\boldsymbol{a_{mn}}$ are deployed on $n_{l\_s,mn}$ and $n_{l\_a,mn}$ nodes, respectively. $\boldsymbol{s_{mn}}$ generates events of the illumination $e_{s,mn}$ and $\boldsymbol{a_{mn}}$ executes actions of the brightness level $b_{a,mn}$ in LPR.
- The nodes $n_{h\_p,mn}$ and $n_{h\_cp,mn}$ are equipped with $\boldsymbol{p_{mn}}$ and $\boldsymbol{c_{mn}}$, respectively. $\boldsymbol{p_{mn}}$ generates events of the illumination value $e_{p,mn}$ and $\boldsymbol{c_{mn}}$ executes actions of the brightness level $b_{c,mn}$ in HPR. In addition to $\boldsymbol{c_{mn}}$ on the node $n_{h\_cp,mn}$, we deployed another *PiO* ($p_{pw,mn}$) to calculate the power consumption by the individual luminary in HPR. The interaction state of $p_{pw,mn}$ is denoted by $POW_{h,mn}$.

Table 7.2. Nodes with associated *iO*s and their communication technologies in the smart space SS_{SL} for UC-2.

Nodes	*iO*s	Communication
$n_{l\_s,mn}$	s_{mn}	Zigbee over IEEE 802.15.4
$n_{l\_a,mn}$	a_{mn}	USB serial communication
$n_{h\_p,mn}$	p_{mn}	SSAP over TCP/IP
$n_{h\_cp,mn}$	c_{mn} and $p_{pw,mn}$	SSAP over TCP/IP
n_{SL}	sb_{SL} *and* mo_{SL}	SSAP over TCP/IP
n_{gw_SL}	gw_{SL}	Zigbee over IEEE 802.15.4, USB serial communication, and SSAP over TCP/IP

In addition to the application ontology graph O_graph_{SL}, we add $p_{pw,mn}$ and its interaction state $POW_{h,mn}$ with the help of $\boldsymbol{mo_{SL}}$ as shown in Fig. 7.7(a). In addition, we also added the triple of the total power consumption in HPR (Total_pow_HPR) value and having the state POW_h (the power consumption due to lighting in A_h) as shown in Fig. 7.7(b).

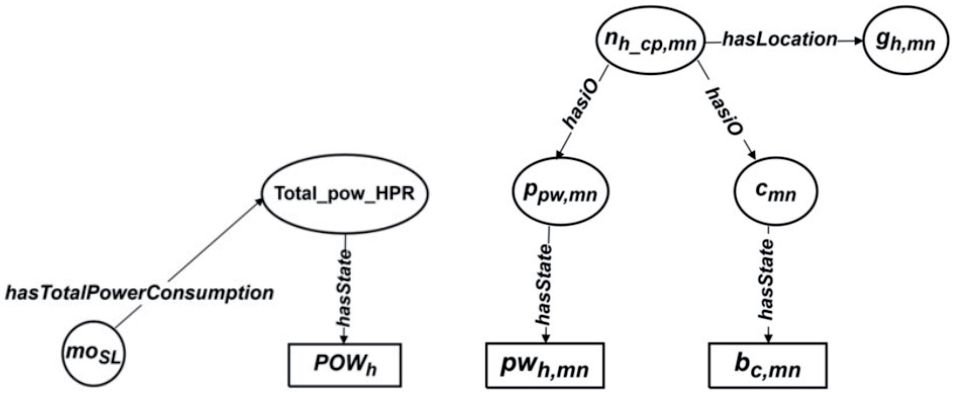


Figure 7.7. The added RDF triples to the application ontology graph *O_graph*$_{SL}$ (Fig 5.3) in *SS*$_{SL}$ for UC-2.

The applications A_l and A_h can independently execute the smart lighting model, similar to UC-1, where A_l and A_h implement the model according to Section 5.3.1 and Section 5.3.2, respectively. For UC-2, we do not repeat our explanation of the implementation steps of the smart lighting model, such as the placement of sensors and actuators in square grid cells and the calculation of actuation commands based on user preferences and activities. In this section, we focus on a priority mechanism to manage power using semantic interoperability between two applications A_l and A_h in the smart space *SS*$_{SL}$. The priority mechanism aims to regulate power consumption and manage the power quota based on power usage in A_h, while also maintaining the desired illumination in A_l. We will show the priority mechanism through the following two scenarios: *SC_A* and *SC_B*. (Note: the UC-2 prototype was also constructed on the sixth floor of the Meta Forum building at the Technical University of Eindhoven in The Netherlands.)

Scenarios:

***SC_A*: When the power quota is large enough to support all activities in both applications A_l and A_h,** the sensor readings in these rooms are used to set and maintain the illumination based on user preferences for the current activity.

***SC_B*: When the power quota is not sufficient to support both applications A_l and A_h simultaneously,** the priority mechanism dims the light sources in A_l. Therefore, ***gw*$_{SL}$** needs to calculate actuation commands based on the remaining power budget for A_l after the power quota is utilized by A_h. This ensures that the power consumption in A_l remains just below the remaining power budget, denoted as Q_l. Therefore, the joint execution of both applications never exceeds the total power quota for the building, denoted as *Q*, if the external light source is under control.

Note that the actuation commands are calculated at ***gw*$_{SL}$** in A_l while the individual c_mn in A_h calculates the actuation commands for light sources in HPR. According to the smart lighting model, if the measured illumination differs from the desired illumination, the light outputs of the luminaries in each application are automatically adjusted until the desired illumination is achieved.

We assumed the power consumption due to lighting in A_h is $POW_h < Q$. Therefore, there is always some leftover power budget Q_l for illuminating the LPR.

$$Q_l = Q - POW_h \tag{6.1}$$

Let POW_{lowreq} denote the power required for the illumination of a specific user activity in A_l. The requested power can be provided to rooms within the application A_l only if $Q_l \geq POW_{lowreq}$. Otherwise, the smart lighting model in these rooms must limit their power usage to the remaining power quota, Q_l.

We explain the execution of scenarios *SC_A* and *SC_B* in Fig. 7.8. An individual $\boldsymbol{p_{pw,mn}}$ updates the power consumption $POW_{h,mn}$ at $\boldsymbol{sb_{SL}}$. We need to calculate the total power consumption POW_h in A_h. For this purpose, we establish a subscription by $\boldsymbol{mo_{SL}}$ to receive the power consumption updates from $\boldsymbol{p_{pw,mn}}$ at $\boldsymbol{sb_{SL}}$ to receive the updates of $POW_{h,mn}$ and calculates the total power consumption value POW_h in A_h. Finally, $\boldsymbol{mo_{SL}}$ updates the total power consumption value POW_h at $\boldsymbol{sb_{SL}}$.

Further, $\boldsymbol{gw_{SL}}$ subscribes at $\boldsymbol{sb_{SL}}$ to receive the power consumption in the application A_h. Subsequently, $\boldsymbol{gw_{SL}}$ automatically receives the value POW_h as a response to the established subscription. It then calculates the remaining quota for A_l, denoted as $Q_l = Q - POW_h$. The smart lighting model in A_l takes Q_l into consideration and computes the actuation commands for $\boldsymbol{a_{mn}}$ accordingly. Finally, the luminaries in A_l adjust their settings based on the remaining quota.

Both scenarios *SC_A* and *SC_B* follow the aforementioned interaction steps. The only distinction lies in the derived value of Q_l in each scenario. In *SC_A*, the application A_l executes the smart lighting model based on the desired illumination specified by the user, as the remaining quota Q_l is sufficient, $Q_l \geq POW_{lowreq}$. However, in *SC_B*, the quota is insufficient to fulfill the requirements in A_l, i.e., $Q_l \leq POW_{lowreq}$, prompting $\boldsymbol{gw_{SL}}$ to enforce the utilization of only Q_l. Moreover, it is possible to regulate power in A_l in a way that the illumination in the activity subspace does not directly diminish. Initially, the model will reduce the brightness of the luminaries in those square grid cells of A_l that are located outside the activity subspace but still within the same room. In practice, most user activities are confined to a specific area within a room. Therefore, a user's activity may span multiple square grid cells depending on the size of each grid cell, and the luminaries in these grid cells are automatically adjusted. For instance, a reading activity may be performed in a designated reading table area, which encompasses multiple square grid cells in A_l. Consequently, the following steps can be executed in case $Q_l \leq POW_{lowreq}$:

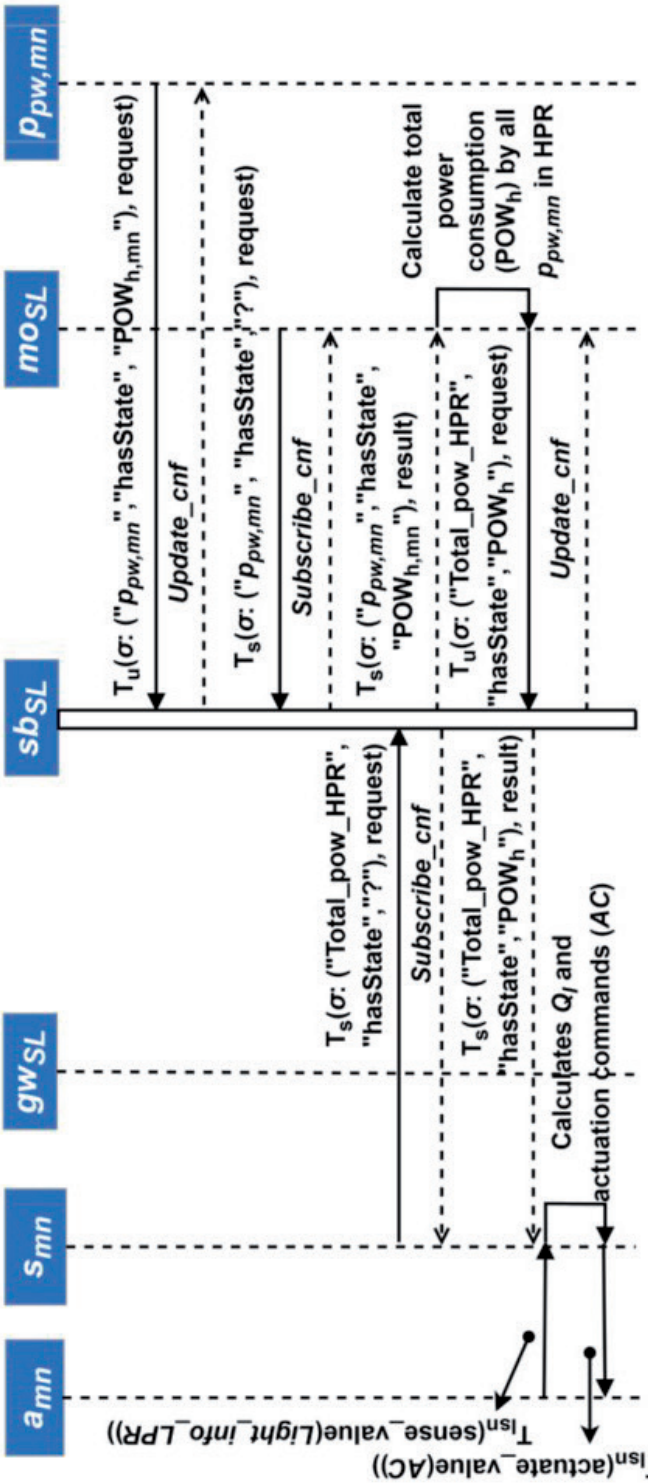


Figure 7.8. Execution steps of the priority mechanism in UC-2.

- **Step 1**: Dim the brightness levels of luminaries until the power consumption in A_l is lower or equal to Q_l in all square grid cells of the application A_l that are outside the activity subspace.
- **Step 2:** If the brightness levels of luminaries in the first step is equal to 0% and the remaining power quota is still less than the required power ($Q_l \leq POW_{lowreq}$), dim the brightness levels of luminaries in the activity square grid cells till the power consumption in A_l is less than the remaining budget Q_l or equal to 0.

In our implementation, the quota Q is set to 12 watts, which is slightly higher than the maximum total power that can be consumed by LED luminaries in A_h. This quota is implemented to govern the regulation of luminaries in A_l, allowing the power consumption to be in the range from 0 to Q.

In the evaluation of application A_h, we conducted four distinct tests with $\boldsymbol{mo_{SL}}$ responsible for calculating power consumptions POW_h, resulting in values of (8437, 2476, 5632, 9200) milliwatts. Subsequently, $\boldsymbol{mo_{SL}}$ updated the total power consumption POW_h in HPR at $\boldsymbol{sb_{SL}}$, and $\boldsymbol{sb_{SL}}$ relayed this information to $\boldsymbol{gw_{SL}}$. Following this communication, $\boldsymbol{gw_{SL}}$ determined the remaining power quota Q_l, with values (3563, 9524, 6368, 2800) milliwatts. The corresponding values of POW_h and Q_l are depicted in Fig. 7.9. The lighting requirements for both room types are easily satisfied in Tests 1 and 4. This implies that the power required in the application A_l is less than the leftover quota, i.e., $SC\_A{:}POW_{lowreq} < Q_l$. Conversely, in Tests 2 and 3, the power required in A_l is higher than its leftover quota, i.e., $SC\_B{:}POW_{lowreq} > Q_l$, resulting in the luminaries in A_l being constrained to use less power than required to fulfill user preferences. The comparison of required and regulated power in A_l is presented in Fig. 7.10. The important point to note here is that A_l sacrifices to adjust the brightness level of luminaries to meet user preferences in Test 2 and 3.

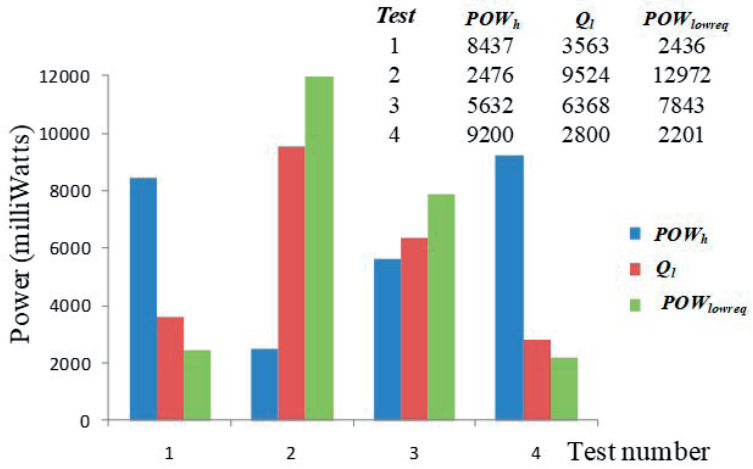


Figure 7.9. Power consumptions in the applications A_l and A_h over four tests.

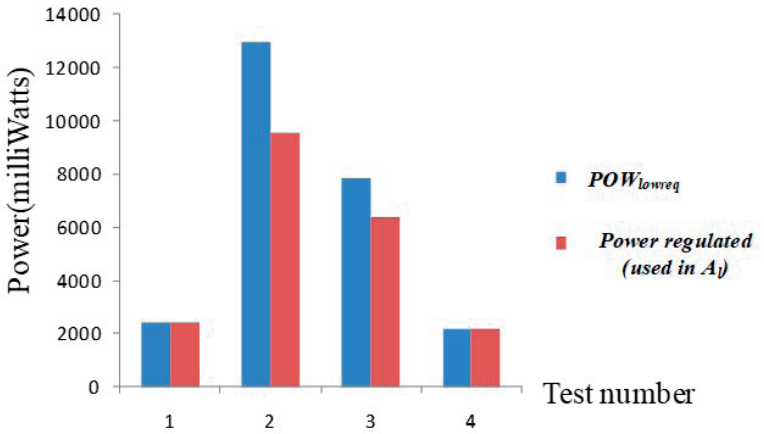


Figure 7.10. Comparison of required and regulated power in the application A_l.

The priority mechanism successfully upheld the designated power quota for the use case. We observed that in certain instances, the application A_l compromised its power requirements in order to remain within the limit of Q. This observation indicates that we can maintain the assigned power quota in the smart space. Overall, the power-managed smart lighting use case effectively controlled the total power consumption across various test sets, adhering to the power quota regime. The implementation of the proposed semantic interoperability architecture facilitated information exchange between applications in LPR and HPR. The use of different types of smart nodes in the use case demonstrated the compatibility and effectiveness of the proposed solution in achieving semantic interoperability. Furthermore, the gateway node enabled the operation on low-capacity nodes once it received information from the semantic broker in the smart space.

7.3 Discussion on Smart Space Properties

In this section, we review the primary properties of a smart space, namely adaptation, communication interoperability, and semantic interoperability, in the context of the experiments conducted in Section 7.2. Additionally, we discuss the potential benefits of secondary properties such as openness, extendibility, and self-management. To illustrate these concepts, we specifically analyze UC-2, which provides a comprehensive depiction of the semantic interoperability architecture involving both LSNs and HSNs. In contrast, UC-1 primarily emphasizes the gateway approach for LSNs.

7.3.1 Discussion on primary properties of smart spaces

In this section, we delve into the potentials of primary properties: adaptation, communication interoperability, and semantic interoperability, within the context of smart spaces.

Adaptation: The luminaries in scenarios *SC_A* and *SC_B* adapted to fulfill the defined objective of maintaining the power quota in the smart space SS_{SL}. The behaviors of *iO*s in both rooms were adjusted according to the specific requirements of each scenario. Changes in lighting conditions in A_h consequently affected both applications, A_l and A_h. This indicates that state changes in the *iO*s of A_h influenced the behavior of *iO*s in both A_l and A_h. As a result, the predetermined quotas for both rooms were effectively managed and controlled. Additionally, any new test set that consumes power can be easily adapted, reflecting the adaptability demonstrated in the four conducted tests.

Communication Interoperability: In scenarios *SC_A* and *SC_B*, HSNs established connections using TCP/IP, while LSNs connected to the gateway node using Zigbee over IEEE 802.15.4. The gateway node, in turn, was connected to the smart space using TCP/IP. As a result, all nodes in the smart space were able to communicate directly with each other or through the gateway node.

Semantic Interoperability: In scenarios *SC_A* and *SC_B*, meaningful information (semantics) was exchanged with the help of ontologies irrespective of the hardware and software specifications of the nodes. We employed heterogeneous nodes from different vendors and communication technologies in the use case, and semantic interoperability using semantic interactions through SSAP allowed them to share information in a meaningful way. Furthermore, the gateway node resolved the complexity of translating information from LSNs into semantics.

7.3.2 Discussion of secondary properties of smart spaces

In this section, we delve into the potentials of secondary properties: openness, extendibility, and self-management, within the context of smart spaces.

Openness: The incorporation of new nodes mandates adherence to a set of protocols, message formats, and syntax that align with the semantic interoperability architecture. In A_l and A_h, the inclusion of a new node is dependent upon its compatibility with both communication interoperability and semantic interoperability. As an illustration, within A_l, the integration of a new node (LSN) necessitates a communication technology that aligns with Zigbee over IEEE 802.15.4 for establishing seamless communication with $\boldsymbol{gw_{SL}}$. This ensures a harmonious integration of new nodes into the existing setup while maintaining effective communication standards. This means that the new node can be added to the application A_l once it meets the necessary communication technology requirements. Therefore, adding nodes to the smart space SS_{SL} is independent of other installed nodes in the applications. Similarly, in A_h, a new node (HSN) needs to support the underlying communication technology, TCP/IP and the SSAP protocol.

The joining and leaving processes for the new node are independent and follow the same procedures as those explained in the smart node life cycle. Once the new node joins, it can subscribe to available services in the smart space. If it is a sensing node, it can update sensor information at the semantic broker (SBSN), which other nodes (e.g., actuators) in the smart space can access after subscribing.

Extendibility: Extendibility can be discussed by considering the addition of a new application in the smart space SS_{SL}. For instance, we can integrate a temperature measurement application into the smart space SS_{SL}. To accomplish this, a node capable of measuring temperature in each room of both LPR and HPR needs to be added. Similar to the openness property, the node must support the required communication technology and be able to utilize the SSAP protocol. In LPR, temperature information is initially updated at $\boldsymbol{gw_{SL}}$ and then transmitted to $\boldsymbol{sb_{SL}}$. In HPR, temperature information is directly updated at $\boldsymbol{sb_{SL}}$. Furthermore, users can easily access temperature updates through a *CiO* installed on their smartphones, either via queries or subscriptions. Therefore, new applications can seamlessly integrate into the smart space SS_{SL} alongside existing applications in the power-managed smart lighting system. Additionally, new applications have the capability to utilize nodes from the current application through subscriptions at the semantic broker (SBSN). For instance, a new application can utilize the installed light sensors for various purposes by subscribing to the broker. Finally, the system offers a decoupled programming interface integrated with application development. Combined with the openness property, this enables the smooth insertion of new nodes and applications into a smart space.

Self-management: We can manage the services of a failed node by transferring them to other nodes or newly joined nodes in the smart space SS_{SL}. For instance, if a node in A_l stops working due to network failure or decommissioning, $\boldsymbol{gw_{SL}}$ can transfer the services provided by the failed node to a new node that takes its place. It is assumed that the new

node has the required communication technology, such as IEEE.802.15.4, to communicate with $\boldsymbol{gw_{SL}}$. Similarly, in A_h, if a node fails, the services of the failed node are transferred to a newly joined node that replaces it at $\boldsymbol{sb_{SL}}$. The newly joined node must support the necessary communication technology, such as TCP/IP, and the SSAP protocol. In both applications A_l and A_h, the new nodes will adhere to the smart node and service life cycles to become integrated and execute services within the smart space. As a result, the smart lighting models in both A_l and A_h will be managed by integrating the newly joined nodes.

We have discussed the contributions of primary smart space properties and the potential benefits of secondary properties within the use case. Through this discussion, we have highlighted the dynamic development of smart applications within a smart space. As a result, we can establish semantic interoperability, providing flexibility and compatibility across smart nodes, irrespective of their hardware and software specifications.

7.4 Performance Evaluation

The semantic interoperability architecture provides a solution for adaptations in smart spaces, as explained in Section 7.3. The adaptive behaviors of *iO*s refer to the decisions made by *iO*s based on received interaction state inputs, resulting in changes to interaction states that trigger new behaviors in other *iO*s. For instance, in both use cases UC-1 and UC-2, the interaction state of a sensor *iO* triggers an action that changes the interaction state of an actuator *iO*. The cumulative adaptive behaviors of *iO*s ultimately contribute to the adaptive behavior of an application, exemplifying semantic interoperability in smart spaces. Therefore, the overall performance of an application depends on the cumulative adaptive behaviors of *iO*s. As discussed in Section 3.2 regarding the adaptation property, if one *iO* takes a long time to adapt, it may hinder the timely adaptation of other *iO*s within an application, leading to inadequate execution of cumulative behaviors. Thus, minimizing delay measurements between events generated by *iO*s and corresponding actions at other *iO*s is crucial for optimal performance in a smart space. In this section, we evaluate application performance based on delay measurements between events and associated actions. To analyze performance based on delay measurements in UC-2, we consider following two example scenarios (ES-1 and ES-2) that illustrate the complete process of sensing (event) at one *iO* and actuating (action) at another *iO*. The delay between an event generated by an *iO* and the associated action at another *iO* or *iO*s is represented by Δt. The events and actions of *iO*s in ES-1 and ES-2 are as follows:

In ES-1: Preserve quota in the smart space SS_{SL} when the illumination changes in HPR.
Event: When the illumination changes in HPR, the event of changed illumination is generated by $\boldsymbol{p_{mn}}$, which updates the information at $\boldsymbol{sb_{SL}}$. As a result, the interaction states of

luminaries in HPR also change based on the smart lighting model. Consequently, $p_{pw,mn}$ individually updates the power consumption $POW_{h,mn}$ at sb_{SL}.
Action: In LPR, a_{mn} adjusts the brightness level according to the actuation commands provided by gw_{SL}.
Note: In this example scenario, the event is generated in one room (i.e., HPR) while the associated action is executed in another room (i.e., LPR).

In ES-2: Preserve quota of the smart space SS_{SL} when the illumination changes in LPR.
Event: When the illumination changes in LPR, s_{mn} generates an event of the changed illumination and further updates it at gw_{SL}.
Action: In LPR, a_{mn} adjusts the brightness level according to the actuation commands by gw_{SL}.
Note: In this example scenario, both the event and action occur in the same room (LPR).

We consider the following links between *iO*s to measure delay, where the time taken between *iO*s in the scenarios ES-1 and ES-2 are depicted in Fig. 7.11

ES-1:

i) p_{mn} generates the event of changed illumination HPR and updates at sb_{SL}, where the time taken from p_{mn} to sb_{SL}, is denoted by t_{1a}.
ii) Since c_{mn} is subscribed to receive events of illumination in HPR, sb_{SL} notifies the changed illumination to c_{mn}, where the time taken from sb_{SL} to c_{mn}, is denoted by t_{1b}.
iii) $p_{pw,mn}$ is installed on the same node of c_{mn} and updates the power consumption by the luminary at sb_{SL}, is denoted by t_{1c}.
iv) Since mo_{SL} is subscribed to receive the power consumption of the individual luminaries in HPR, sb_{SL} notifies the power consumption by the luminary to mo_{SL}, is denoted by t_{1d}.
v) mo_{SL} calculates the total power consumption in HPR and updates it at sb_{SL}, is denoted by t_{1e}.
vi) Since gw_{SL} is subscribed to receive the information of the total power consumption in HPR, sb_{SL} notifies the total power consumption to gw_{SL}, is denoted by t_{1f}.
vii) gw_{SL} calculates the actuation commands and updates a_{mn}, where the action takes place to adjust the brightness of the luminary in LPR, is denoted by t_{1g}.

ES-2:

viii) s_{mn} generates the event of changed illumination in LPR and updates gw_{SL}, is denoted by t_{2a}.
ix) gw_{SL} calculates the actuation commands and updates a_{mn}, where the action takes place to adjust the brightness of the luminary in LPR, is denoted by t_{2b}.

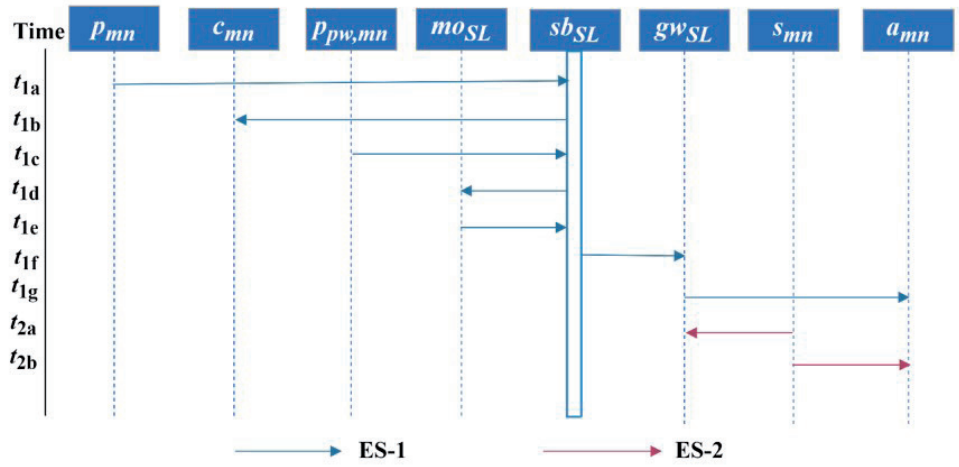


Figure 7.11. Delay calculation links between *iO*s for ES-1 and ES-2.

Finally, all luminaries in LPR adjust their light outputs based on the remaining quota in both ES-1 and ES-2. Therefore, we measure the delay between the event and action in ES-1 given by $\Delta t_{ES\text{-}1}$:

$$\Delta t_{ES-1} = t_{1a} + t_{1b} + t_{1c} + t_{1d} + t_{1e} + t_{1f} + t_{1g} \tag{6.2}$$

In ES-2, we measure the delay between the event and action in ES-2 given by $\Delta t_{ES\text{-}2}$:

$$\Delta t_{ES-2} = t_{2a} + t_{2b} \tag{6.3}$$

The results of delay measurements between the events and associated actions in ES-1 and ES-2 are presented in Table 7.3. Based on the measurements, the maximum delays for $\Delta t_{ES\text{-}1}$ *a*nd $\Delta t_{ES\text{-}2}$ are 326.42 and 52.05 milliseconds, respectively. The average delay measurements between the event and action relationships are 219.25 and 32.65 milliseconds for both example scenarios ES-1 and ES-2, respectively. These delay measurements indicate that there is a relatively large and varying delay between *iO*s of HSNs, but it falls within acceptable limits. This level of delay is perfectly acceptable because the scenario itself is not an interactive scenario that involves human users. Furthermore, it is a very complex scenario including many components that are participating and communicating among themselves. Most interactive scenarios are meant to be simpler, e.g., consider a scenario where entering a room results in lights turning on. On the other hand, the delay measurements for communication between *iO*s of LSNs through the GSN exhibit a small variance, which can be attributed to the ease of accessing the radio channel. It is important to note that these measurements were obtained with the SN located at a one hop distance from the GSN and without employing duty cycling for the radio channel. With a

more extensive wireless network configuration and the implementation of duty-cycled radios, the delay characteristics would become more complex.

Table 7.3. Delay measurements (in milliseconds) for ES-1 and ES-2.

Δt	$\Delta t_{ES\text{-}1}$	$\Delta t_{ES\text{-}2}$
Average	219.25	32.65
Standard Deviation	33.98	11.15
Minimum	158.83	13.28
Maximum	326.42	55.05

Therefore, the delay measurements in the use case are deemed sufficient since they fulfill the objective of controlling quotas in both rooms in both scenarios. This indicates that the adaptive behaviors of *iO*s are adequately executed within a defined timeframe to seamlessly handle the collective behaviors of *iO*s in the application. As a result, the semantic interoperability architecture successfully meets the performance metric requirement. The smart lighting applications prove to be feasible within the framework of the semantic interoperability architecture.

7.5 Conclusions

In this chapter, we implemented two use cases of smart lighting applications using our proposed approaches. We analyzed the feasibility of the proposed smart lighting model through the implementations. Furthermore, the experiments conducted in the proposed system demonstrated that semantic interoperability enabled the integration of lighting products from different smart node suppliers. The utilization of *iO*s with an ontology approach enhanced semantic interoperability, enabling the seamless sharing of information in a smart space with diverse hardware and software specifications. Moreover, the example scenarios showcased the compatibility of primary and secondary smart space properties. Additionally, we conducted an experiment to measure the delays between events and actions performed by *iO*s as a performance metric. The analysis concluded that the semantic interoperability architecture is a suitable choice for the lighting application scenarios under consideration.

The scenarios presented in this chapter provide generalization across a wide range of smart space implementations found in the literature. The semantic interoperability architecture emerged as a competitive solution for the domain of lighting applications and holds the potential to be advantageous for other smart space domains.

Chapter 8

Conclusion

In the preceding chapters, we delved into the theoretical and practical aspects of semantic interoperability in smart spaces. In this final chapter, we conclude this thesis by providing a comprehensive summary of our contributions and answers to research questions in Section 8.1. Additionally, we offer insights into potential future research directions in Section 8.2.

8.1 Contributions and Answers to Research Questions

Smart spaces have a wide range of applications in sectors such as homes, offices, and transportation, aiming to assist human beings with their daily activities and enhance their overall experience. However, efficiently integrating the diverse smart nodes within these spaces to collaborate and achieve specific goals poses a significant challenge. In Chapter 1, we delved into this challenge by introducing the concept of a smart space in a formal manner. One particular issue that arises in smart spaces is the need for consistent information understanding, also known as semantic interoperability, among different smart nodes. Despite various proposals and implementations of smart spaces in the literature, they often tend to be showcase-specific or application-specific, lacking a generic smart space architectural design based on semantic interoperability. To address this gap, we formulated three research questions (RQ_1, RQ_2, and RQ_3) in Chapter 1, aimed at tackling this challenge. In conclusion, this thesis summarizes our contributions to these research questions and provides insights into potential solutions.

> ***RQ_1:*** *What are the concepts, properties, and architectural design alternatives of smart spaces?*
> ***RQ_{1A}:*** *What are the fundamental concepts, building blocks and properties of a smart space?*
> ***RQ_{1B}:*** *How can we compare potential smart space architectural design alternatives?*
> ***RQ_{1C}:*** *What is an appropriate architectural solution for realizing these concepts? Which smart space properties are the most essential for this architectural solution?*

In Chapter 2, we provided a formal definition of smart space concepts and architectural designs. Additionally, we delved into the detailed discussion of related architectural components, encompassing both hardware and software aspects. Building upon a comprehensive literature review, we proceeded to summarize and propose the discriminating properties that are essential for a smart space in Chapter 3.

In response to RQ_{1A}, we presented an in-depth exploration of the fundamental concepts of hardware and software within a smart space. We introduced the notion of an information object as a software unit and a smart node as a hardware unit. Furthermore, we elucidated the behaviors exhibited by information objects and smart space applications. To ensure a clear understanding, we provided a distinct classification of smart nodes, outlining the various types of information objects and their respective functions within a smart space.

To address RQ_{1B}, we conducted a comprehensive comparative analysis of the architectural designs proposed in the literature. Furthermore, we presented an example that illustrates how to choose a suitable smart space architectural design based on a multiple-objective optimization. Our findings revealed that the selection of a smart space architecture depends on the specific requirements of the application, considering performance metrics that are context-dependent.

Additionally, we compared the identified architectural designs to state-of-the-art smart space designs, focusing on the availability of general smart space components. As a result, we concluded that decentralized smart spaces and high-capacity smart nodes are predominantly utilized in these architectural designs. Some examples of these smart spaces include Apple Home, Google Home, and Philips HUE. These solutions utilize more proprietary protocols with limited openness, except for development kits. There is no specific software cooperation protocol, and there are no explicit roles for semantic brokers and ontologies. However, we were in search of a software solution that can offer semantic interoperability. Thus, in this thesis, we have proposed the semantic interoperability architecture and discussed it in the context of the next sub-research question RQ_{1C}. The researchers may also explore advanced data behavior that incorporates machine learning concepts within smart spaces in the future. We will discuss machine learning aspects in the future works in Section 8.2.

To address RQ_{1C}, we introduced the semantic interoperability architecture as a solution. Additionally, we proposed five essential properties for smart spaces, namely adaptation, communication interoperability, semantic interoperability, openness, extendibility, and self-management. Based on our comparative analysis of smart space designs, we identified three primary properties—adaptation, communication interoperability, and semantic interoperability—as crucial, while the remaining properties are considered to be dependent on the specific requirements of the application. Furthermore, we delved into the logical structure of information objects within the proposed semantic interoperability architecture, providing a detailed explanation of the processes and dependencies among these information objects.

RQ2: *How can we establish semantic interoperability of heterogeneous embedded devices (nodes) in smart spaces?*

RQ_{2A}*: What are the processes and dependencies for semantic interactions among components that will enable semantic interoperability?*

RQ_{2B}*: What are the additional mechanisms and components needed for semantic interoperability in smart spaces? How can we achieve semantic interoperability with resource-poor (low-capacity) devices in a smart space?*

RQ_{2C}*: How can we achieve semantic interoperability with multiple applications in a smart space?*

In Chapter 4, we focused on establishing communication interoperability using standard network protocols, while highlighting the challenges associated with achieving semantic interoperability. To address this challenge, we elucidated the mechanism for establishing semantic interoperability among smart nodes within the proposed semantic interoperability architecture.

To address ***RQ_{2A}***, we presented the procedure of converting contexts into semantics within the proposed architecture. In this process, the RDF format was utilized to represent semantics, and relationships were governed by ontologies in smart spaces. Additionally, semantic reasoning was employed at the semantic broker to apply RDF matching operations and inference rules, facilitating the outcomes of semantic interactions. Within the proposed semantic interoperability architecture, SSAP transactions were utilized for semantic interactions among information objects in a smart space. These interactions included operations: JOIN, LEAVE, INSERT, UPDATE, QUERY, REMOVE, SUBSCRIBE, and UNSUBSCRIBE. Furthermore, we elucidated the mechanism of semantic interactions between information objects through several example scenarios in a smart space.

To address ***RQ_{2B}***, we proposed a gateway approach for integrating low-capacity devices into a smart space. The gateway node assumes the responsibility of performing computations and knowledge representations on behalf of low-capacity nodes, such as sensors and actuators. We illustrated the gateway approach through example scenarios in a smart space, showcasing the interaction mechanisms and explaining the interoperability between low and high-capacity nodes facilitated by the gateway node.

To address ***RQ_{2C}***, we proposed the integration concept of multiple applications in a smart space. To achieve this, we proposed a mechanism for semantic interactions between information objects from different applications. This mechanism is integrated into the semantic interoperability architecture, enabling information objects to effectively collaborate and share information across multiple applications within a smart space.

***RQ3**: How can our proposed solutions be applied in real systems?*
- ***RQ$_{3A}$**: How can we map the concepts and properties of smart spaces in real systems (including an illustration of how to measure the performance of a smart space architecture)?*
- ***RQ$_{3B}$**: What are the smart space life cycles?*
- ***RQ$_{3C}$**: How can semantic interoperability be established in real systems?*

In Chapter 5, we focused on smart lighting applications and provided an overview of related work within the context of smart spaces. To address ***RQ$_{3A}$***, a model is required to map the concepts and properties of smart spaces onto smart lighting applications. Consequently, we introduced a novel smart lighting model designed for controlling lighting applications in indoor spaces. This model facilitates the regulation of illumination based on user preferences and activities. We integrated the smart lighting model into the proposed semantic interoperability architecture, enabling the sharing of lighting information across multiple networks within a smart space. Additionally, we explored two use cases: context-adaptive smart lighting and power-managed smart lighting.

To address ***RQ$_{3B}$***, and ***RQ$_{3C}$*** in Chapters 6 and 7, we designed and explained the life cycles of smart spaces, which encompass the smart node life cycle, smart service life cycle, and smart application life cycle. These life cycles were demonstrated using the smart lighting model and the semantic interoperability architecture. Through implementation, we achieved semantic interoperability among smart nodes from different manufacturers and technologies. The behaviors of information objects were adapted through semantic interactions, enabling successful integration. The results of our implementations were discussed, highlighting the smart space properties observed during the conducted experiments. To support our analysis, we selected example scenarios that demonstrated the appropriate behaviors of information objects in lighting applications. Additionally, we evaluated the smart space by using measuring delays in transmission links as the performance metric. The events and associated actions of information objects were found to be crucial factors in ensuring their adequate behaviors within smart spaces. Through our analysis, we determined that the selected lighting use cases demonstrated robust adaptability, affirming the effectiveness of the semantic interoperability architecture for smart lighting applications.

In this thesis, we have presented smart space concepts, properties, and architectures, along with their semantic interoperability implementations. The semantic interoperability architecture offered a publish-subscribe system, wherein publishers produce information, and subscribers express interest in specific types of information. This architecture's key feature is the decoupling of publishers and subscribers, eliminating the need for information objects to be aware of one another. The shared information undergoes translation into semantics comprehensible by all information objects, facilitated by the smart space access protocol through the semantic broker. Furthermore, this architecture incorporates

a gateway solution tailored for low-capacity devices, ensuring a seamless connection between the gateway and the broker. As a result, the semantic interoperability architecture fosters smooth collaboration across a spectrum of communication and hardware technologies. This holds true whether for a single application or multiple applications with a shared objective.

A well-designed smart space, with semantic interoperability and smart space properties at its core, has the potential to deliver highly effective solutions for application implementations. Our research findings have played a pivotal role in seamlessly integrating these properties into the semantic interoperability architecture.

8.2 Options for Future Works

Recent research has started addressing the integration of machine learning techniques in smart spaces. Several studies [7.1-7.3] discuss the concept of incorporating machine learning models into smart spaces, such as the application of a machine learning model for smart home activities in different environmental settings [7.4]. However, while research has demonstrated the capability of analyzing big data in various applications, there has been less emphasis on machine learning techniques for semantics in smart spaces. The state-of-the-art describes the concepts of semantic relations with machine learning approaches but lacks a formalized approach for smart spaces. Therefore, we propose future work in this direction, aiming to enhance the semantic interoperability architecture's predictive capabilities by integrating semantic learning alongside semantic reasoning at the semantic broker. Semantic learning refers to applying machine learning algorithms to the semantics stored in a smart space, generating knowledge to guide application behaviors. For instance, in a power-managed smart lighting system, we can record peak hours of power consumption in high-priority rooms and notify low-priority rooms based on semantic learning analysis. As a result, the low-priority rooms may reschedule activities during off-peak hours in high-priority rooms, allowing them to utilize sufficient power to control luminaries and optimize the remaining power quota. However, there are two immediate challenges for deploying semantic learning within the existing semantic broker of the semantic interoperability architecture. Firstly, generating accurate knowledge at the semantic broker requires a sufficient amount of data. Insufficient data can lead to inaccuracies in predictions or classifications. Secondly, the computational cost of executing the machine learning algorithm at the semantic broker should be optimized to ensure efficient processing within a reasonable timeframe. Lengthy processing times are not preferable in many use cases, especially for the adoption of timely behaviors by information objects, such as smart lighting applications.

Next, the subscriptions to information at the semantic brokers play a crucial role in evaluating the performance of a smart space. These subscriptions involve implementing persistent queries, allowing semantic brokers to receive regular notifications about updates to the subscribed information. However, this passive query module exhibits lower performance in large-scale smart spaces. To address this limitation, [7.5] has begun analyzing the integration of multiple semantic brokers in parallel smart spaces by introducing an active subscription approach. The active subscription approach enables runtime changes to subscriptions, offering a more dynamic and efficient solution. This approach is currently being tested in a large-scale IoT environment, facilitating interactions among numerous sensors and users. Preliminary results indicate that the active subscription approach significantly improves the efficiency of notification delivery and overall performance. Although the research has provided preliminary results, further experimentation in various parallel smart spaces is needed. This direction presents an avenue for future work, exploring the potential benefits of the active subscription approach across different scenarios.

Next, the tradeoff between performance and scalability within the context of the proposed semantic interoperability architecture for smart spaces is an aspect that we have yet to investigate. This investigation will be part of our future work. The objective is to comprehend the factors influencing system performance as the number of nodes connected to a semantic broker increases, and to identify methods for optimizing performance in large-scale smart space installations, such as smart cities.

Finally, in the implementation conducted in this thesis, we have not addressed factors such as security and privacy. We recommend further research to develop a secure and private platform for smart spaces in the future.

Appendix A: SOFIA Smart Home Pilot Case Study

The concept of smart spaces is significantly evolving to enhance daily life improvements. One notable example is presented in [A.1], where a magic room is proposed for children with neurodevelopmental disorders. This room incorporates various physical objects, such as soap bubbles, aromas, ambient sound, visual projections, lights, and toys, which are controlled and interacted with through a sensory system. Over a period of four months, experiments were conducted at two magic rooms in Milan and Rome, Italy, involving eight caregivers and nineteen children with severe neurodevelopmental disorders. The analysis and observations revealed that the learning improvement in these children was much faster compared to traditional classroom settings. Although the results are still preliminary, these experiments serve as noticeable real-world examples of smart space development for the betterment of our society.

Similarly, in this appendix, we describe a real scenario of smart homes called the "Smart Home Pilot"[16] for the advancement of smart space development. The smart home pilot was a collaborative effort involving TU/e (Eindhoven University of Technology) and companies such as Philips, NXP, and Conante. It was based on the SOFIA project and resulted in a joint demonstrator showcased and evaluated at the Experience Lab in the High-Tech Campus in Eindhoven, The Netherlands. In this appendix, our focus primarily lies on providing an abstract description of the implementation and emphasizing the contribution made by TU/e. The objective of the smart home pilot demo was to demonstrate information sharing across applications in a smart space, utilizing smart nodes with specific hardware and software specifications. Furthermore, we highlight the feasibility of the proposed semantic interoperability architecture for implementation.

The following scenario is considered for the smart home pilot:

Scenario
Mark and Dries enter their home. The smart lighting system detects their presence, switches the lights on, and notifies the smart space about their presence. Subsequently, the decorative wall-wash lights receive a notification from the smart space and illuminate. Mark and Dries decide to synchronize the music with visual effects on a lighting device. After querying the smart space, they discover that the lighting device can achieve this effect. They establish a connection between the music player and the lighting device using the interaction tile. The light begins to synchronize with the music on the lighting device. To draw attention to the lighting device, the decorative wall-wash lights in the room automatically dim. Simultaneously, the light pattern is also replicated on the remote lighting device, allowing Mark's sister Sofia to experience the same visual effects in her own house at a different location. At a different location, after a while, Sofia becomes curious and wants to listen to the music that Mark and Dries are enjoying. She uses the spotlight navigation device to establish a connection from the bonding device to the stereo. As she starts using the spotlight navigator, the lights in the room dim to enhance the visibility of the spotlight. Once the connection is established, the lights return to their original brightness.

16 SOFIA project pilot - YouTube

Note that, in this scenario, media content is shared among multiple devices within a smart home environment. Specifically, music is distributed between a mobile device, a stereo speaker set, and a lighting device that enhances the atmosphere with colored lighting. These music experiences, encompassing both light and sound elements, can also be shared remotely among friends residing in separate homes through their respective lighting devices. Additionally, other lighting sources, such as smart functional and wall wash lights, are responsive to user presence and the use of other lighting sources within the smart homes.

The smart home pilot demo consists of a single smart space, which is divided into two applications deployed across separate homes using multiple smart nodes. Semantic interoperability has been established between these smart home applications, enabling the exchange of information at a semantic level. The system architecture of the smart home pilot implementation is depicted in Fig A.1. To connect the two homes, two wireless routers were placed in different rooms, representing smart home A and smart home B, and bridged using an Ethernet network cable. One router was configured to operate as a Dynamic Host Configuration Protocol (DHCP) server, while the other served as a network bridge.

All HSNs utilized in the smart home pilot communicate with a common SBSN (n_{sh}), with the nodes defined in Table A.1. As a result, we have a single smart space comprising two smart applications: smart home A and smart home B. For this pilot demo, our focus was

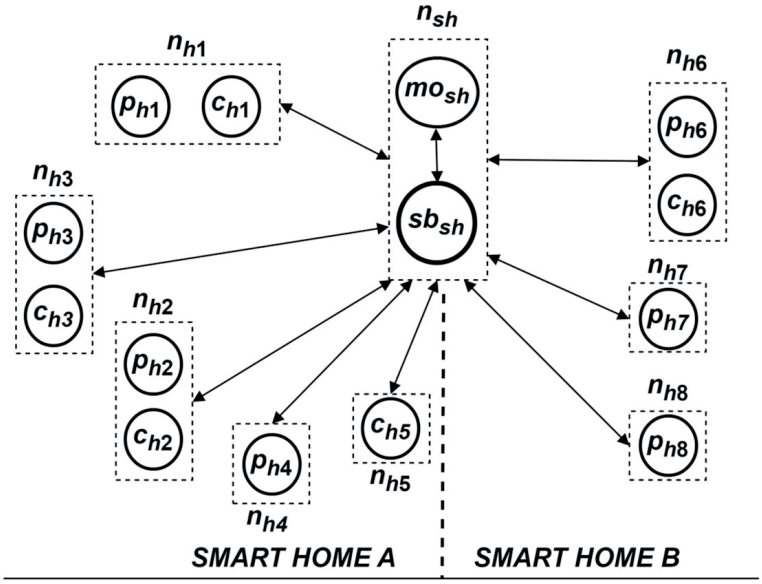


Figure A.1. System architecture of the smart home pilot.

exclusively on HSNs. In smart home A, we considered the following HSNs connected to the router: n_{h1}, n_{h2}, n_{h3}, n_{h4}, and n_{h5}. Similarly, in smart home B, we had the following HSNs connected to the router: n_{h6}, n_{h7}, and n_{h8}. Given that the communication protocol in this scenario is SSAP over the internet, all network components of the smart nodes were connected to the network using IEEE 802.11g wireless technology.

Table A.1. Description of all smart nodes used in the smart home pilot demo.

Node	Description	*iO*s
n_{sh}	The SBSN node is used to share and store semantics for the smart space in the smart home pilot.	sb_{sh}
n_{h1}	The sound-light transformer smart node operates on the music and transforms it into various light display settings in Smart Home A. This smart node serves as both a producer and consumer of *iO*s.	p_{h1}, c_{h1}
n_{h2}	The music player smart node is responsible for playing music in Smart Home A. It functions as both a producer and consumer of *iO*s.	p_{h2}, c_{h2}
n_{h3}	The connector smart node enables the exploration and manipulation of semantic connections between different devices in Smart Home A. It serves as both a producer and consumer of *iO*s.	p_{h3}, c_{h3}
n_{h4}	The presence smart node detects the presence of a user within the activity area of the room in Smart Home A and updates this presence information at the SBSN. It functions solely as a producer *iO*.	p_{h4}
n_{h5}	The lamp smart node is connected to four LED lamps, also known as decorative wall-wash lights, which are used to illuminate colored lights on the wall in Smart Home A. This smart node functions solely as a consumer *iO*.	c_{h5}
n_{h6}	The remote music player smart node is an additional music player used for playing music in Smart Home B. This smart node serves as both a producer and consumer of *iO*s.	p_{h6}, c_{h6}
n_{h7}	The spotlight navigator smart node is a navigation device that establishes connections between two devices in Smart Home B by navigating commands. This smart node functions solely as a producer *iO*.	p_{h7}
n_{h8}	The functional light smart node refers to the lamp located in Smart Home B. This smart node functions solely as a consumer *iO*.	c_{h8}

All *iO*s in the smart home pilot demo are connected to sb_{sh} using the JOIN transaction. Additionally, the smart space of the smart home pilot (sb_{sh}) comprises the following collections of smart nodes and *iO*s.

$$SS_{sh}.N = \{n_{sh}, n_{h1}, n_{h2}, n_{h3}, n_{h4}, n_{h5}, n_{h6}, n_{h7}, n_{h8}\} \tag{A.1}$$

$$SS_{sh}.iO = \{sb_{sh}, p_{h1}, c_{h1}, p_{h2}, c_{h2}, p_{h3}, c_{h3}, p_{h4}, c_{h5}, p_{h6}, c_{h6}, p_{h7}, c_{h8}\} \tag{A.2}$$

The specifications of the hardware and software capabilities of nodes in the smart home pilot are shown in Table A.2.

Table A.2. Specification of the hardware and software capabilities of nodes in the smart home pilot.

Node	CPU	Operating System	Programming Language
n_{sh}	Core 2 Duo 2.8GHz	Ubuntu Linux	Java
n_{h1}	Core 2 Duo 2.2GHz	Ubuntu Linux	Java
n_{h2}	ARM Cortex-A8	Maemo 5	Python
n_{h3}	Core 2 Duo 2.6GHz	Mac OS X	Python
n_{h4}	Pentium M	Ubuntu Linux	Python
n_{h5}	Pentium M	Ubuntu Linux	Python
n_{h6}	ARM Cortex-A8	Maemo 5	Python
n_{h7}	OMAP 3530	Ubuntu Linux	Prolog, C++
n_{h8}	Arm cortex M0	µCLinux	C

We contributed to design and integrated the wall-wash lights (smart node n_{h5}) in the pilot demo. The smart node n_{h5} is connected with four lamps and shown in Fig. A.2 with the description of its components. The functioning of lamps is based on the information of user presence received at c_{h5} (consumer of the node n_{h5}) from p_{h4} (producer of the node n_{h4}) and the information of music from p_{h1} (producer of the node n_{h1}) via sb_{sh} The semantic interactions among p_{h1}, p_{h4}, c_{h5}, the lamps and sb_{sh} are shown in Fig A.3. p_{h4} determines the presence of users (Mark and Dries) in an activity area of a room and updates the presence at sb_{sh}. c_{h5} is subscribed for the presence of a user and gets an update when the presence is updated by p_{h4} at sb_{sh}. There are two interaction states to be updated by p_{h4} on sb_{sh}: 'AWAY' and 'PRESENT'. According to these interaction states, the lamps turn 'on' or 'off'. For example, when the 'PRESENT' state is updated by p_{h4} at sb_{sh} then c_{h5} turns 'on' all lamps while 'off' when the 'AWAY' state is updated. The c_{h5} is also subscribed for the states of p_{h1}. When music starts rendering c_{h5} turns to 'DIM' the lamps.

We contributed to the design and integration of the wall-wash lights (smart node n_{h5}) in the pilot demo. Smart node n_{h5} is connected to four lamps, as shown in Fig. A.2, along with a description of its components. The functioning of the lamps is based on information received from various sources. Specifically, consumer c_{h5} receives information about user presence from producer p_{h4}, and information about music from producer p_{h1} via sb_{sh}.

Semantic interactions among p_{h1}, p_{h4}, c_{h5}, the lamps, and sb_{sh} are illustrated in Fig A.3. The producer p_{h4} determines the presence of users (Mark and Dries) in the activity area of a room and updates this information at sb_{sh}. Consumer c_{h5} is subscribed to receive presence updates from p_{h4} at sb_{sh}. There are two interaction states that can be updated by p_{h4} on sb_{sh}: 'AWAY' and 'PRESENT'. Based on these interaction states, the lamps either turn 'on' or 'off'. For instance, when the 'PRESENT' state is updated by p_{h4} at sb_{sh}, c_{h5} turns on all lamps, while it turns them off when the 'AWAY' state is updated.

Consumer c_{h5} is also subscribed to receive updates from producer p_{h1}. When the music starts playing, c_{h5} adjusts the lamps to a 'dim' setting.

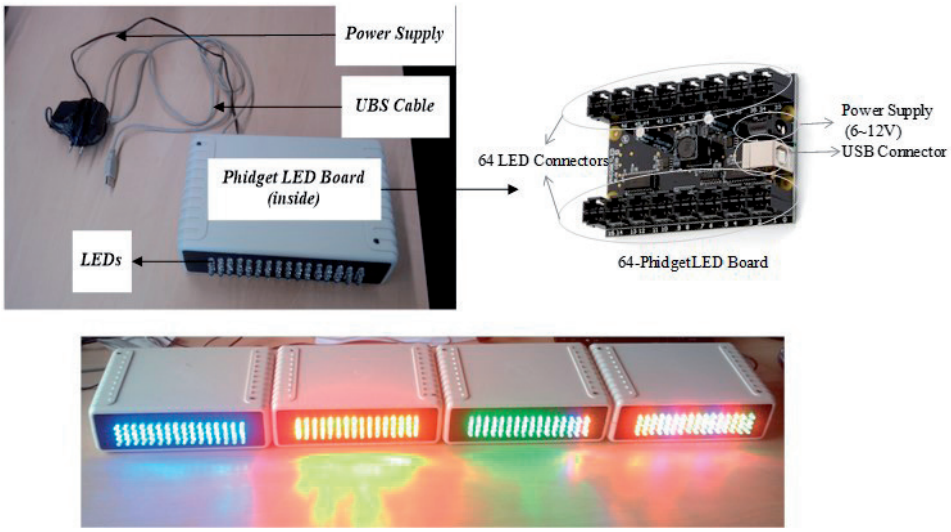


Figure A.2. The smart node n_{h5} with lamp designs and used for wall wash lighting.

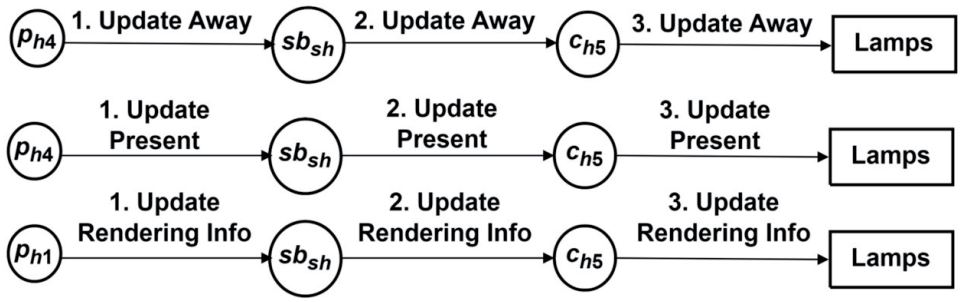


Figure A.3. Sequence of commands from the c_{h5} to Lamps in smart home A.

The smart node n_{h3} (connector smart node) is shown in Fig. A.4 and is used to explore and manipulate semantic connections between different nodes in the smart home A. It is a handheld device that identifies nodes, by scanning RFID tags that are located on the nodes themselves. The users can explore the connection possibilities that are visualized with lights on top of the connector node holding the connector on top of the tag. After holding the node in the RFID field for a moment, the node-ID is locked and the other node to be connected can be selected in a similar fashion.

The connector smart node (n_{h3}) is depicted in Fig. A.4 and serves the purpose of exploring and manipulating semantic connections between different nodes in Smart Home A. It is

a handheld device equipped with RFID scanning capabilities, allowing it to identify nodes by scanning RFID tags affixed to them. Users can utilize the connector node to explore the available connection options, which are visualized through lights located on top of the connector node. By placing the connector node on top of a tag, the user can lock the node-ID and subsequently select another node for connection using a similar approach.

Figure A.4. Connector for the smart node n_{h3} . 11. This node is developed by Gerrit Niezen and Bram J.J. van der Vlist from the industrial design department at TU/e [A.2].

Consumer c_{h1}, which is deployed to node n_{h1}, accepts a music stream as input and generates a stream of RGB values by analyzing the music stream. To transmit the stream of RGB values, a separate TCP/IP connection is used. Therefore, it is important for the lighting node to determine whether the source node is capable of communication via TCP/IP. The sequence of queries from consumers c_{h1} and c_{h3} to sb_{sh} is depicted in Fig A.5.

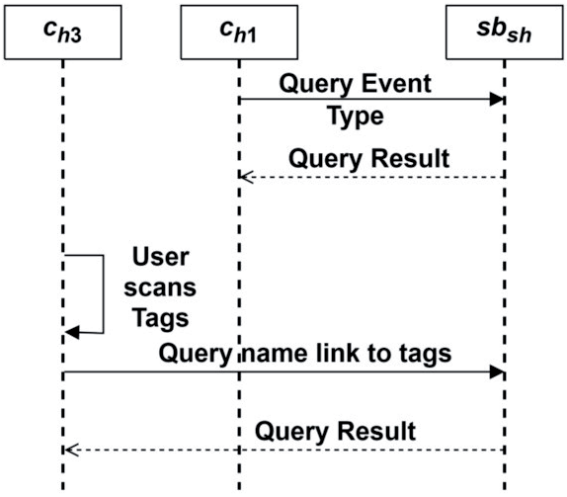


Figure A.5. A sequence diagram of queries by c_{h1} and c_{h3}.

During the smart home pilot study, we conducted 86 measurements using the smart node n_{h1}, which occurred each time an event was received. Additionally, we conducted 961 measurements using the smart node n_{h3}, which occurred each time a user scanned a tag. As for the smart node n_{h2}, we measured the time between inserting a new event and receiving an update from sb_{sh} indicating that the specific event had occurred.

For instance, consumer c_{h2}, connected to node n_{h2}, subscribes to the PlayEvent type, as shown in Fig A.6. When the node is notified of a PlayEvent by sb_{sh}, producer p_{h2} generates a new PlayEvent. Consumer c_{h2} then queries sb_{sh} to confirm the notification of the PlayEvent, particularly for the event it generated itself.

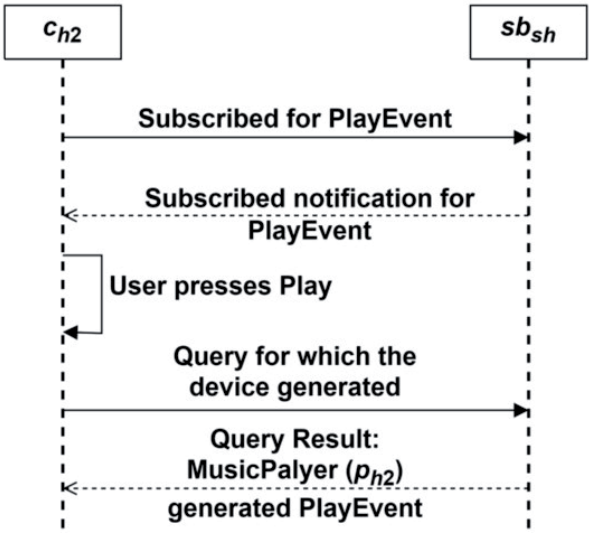


Figure A.6. A sequence diagram of query and subscription by c_{h2}.

Smart node n_{h7}, which acts as the spotlight navigator, is identified based on its physical location, relying on a natural mapping approach. The projection of the spotlight navigation, performed by producer p_{h7} (associated with node n_{h7}), for connecting lighting and music nodes is illustrated in Fig A.7. Connections between nodes are established simply by drawing lines, and an erasing gesture with p_{h7}, pointed at an existing connection, can break the connection. This operation is achieved by continuously measuring the orientation, and optionally the position, of the node.

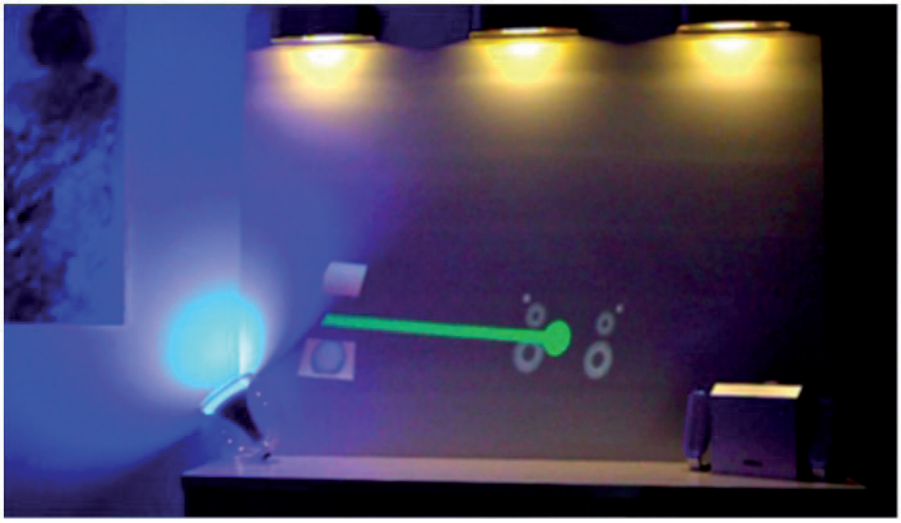


Figure A.7. Projection of the spotlight navigation for connecting lighting and music nodes.

Simultaneously, p_{h7} shares information with consumer c_{h8} (connected to node n_{h8}) to adjust the illumination in Smart Home B. Consumer c_{h8} possesses the ability to adjust the illumination within Smart Home B and is also integrated with a presence sensor to detect the presence of a user in the smart home. When p_{h7} is active in connecting, erasing, or breaking connections between nodes, c_{h8} dims down to enhance the visibility of the projected spotlight.

In the smart home pilot, sb_{sh} is utilized with the reasoning support of OWL2. Reasoning on the information contained within sb_{sh} is performed using OWL2. The OWL engine iterates until no new triples are constructed, which we refer to as one reasoning cycle. Before each new reasoning cycle, the existing inferences from the previous cycle are cleared. Semantics are inferred, using matching operations, when new semantics are inserted by nodes connected to a semantic store.

The detailed analysis is presented in [A.3]. For c_{h1}, the majority of queries completed within 100ms, with very few queries taking longer than 500ms to complete. In contrast, subscriptions were completed in an average of 860ms for c_{h2}. This is due to the addition of the new PlayEvent and the performance of inferencing on the triple store before the subscribe notification is generated.

In terms of communication from p_{h4} to sb_{sh}, it involves an update request and the corresponding confirmation response. The measurements for subscriptions in our setup are also noticeably slower. While sb_{sh} measurements took only 140ms, c_{h2} measurements averaged around 860ms. This delay is primarily attributed to the additional time required

for reasoning, which typically takes about 275ms on average. Other contributing factors include the number of nodes used, the number of semantic triples exchanged, and the network environment.Our analysis has demonstrated that delay measurements can have a significant impact on the stability of collaborative services provided by the smart space. Additionally, we have derived a statistical upper bound on the worst-case delay performance for the specific experimental setup. One important finding from our analysis is that transmission delays between high-capacity nodes and the semantic broker can have a dominant influence on the delay measurements.

Finally, this appendix has presented the implementation and feasibility of semantic interoperability for the smart home pilot. The integration of smart nodes from different manufacturers, with their varied technologies, was made possible through semantic interoperability. This has contributed to the successful implementation of smart spaces, where nodes can seamlessly collaborate and operate together.

Publications by Author

The author has contributed the following publications within the scope of Ph.D. research:

Journals

1. **Sachin Bhardwaj**, Keon Myung Lee, Jee-Hyong Lee, "An adaptive framework for applying machine learning in smart spaces", SAC 19, Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp.1263-1270, April 2019.
2. **Sachin Bhardwaj**, Tanir Ozcelebi, Johan J. Lukkien, Kyung M. Lee, "Smart Space Concepts, Properties and Architectures", IEEE Access, Vol. 6, 70088-70112, Nov 2018.
3. **Sachin Bhardwaj**, Tanir Ozcelebi, Johan J. Lukkien, Kyung M. Lee, "Semantic Interoperability Architecture for Smart Spaces", International Journal of Fuzzy Logic and Intelligent Systems, Vol. 18, Issue 1, pp. 50-57, 2018.
4. **Sachin Bhardwaj**, Tanir Ozcelebi, Ozgur Ozunlu, Johan J. Lukkien, "Increasing Reliability and Availability in Smart Spaces: A Novel Architecture for Resource and Service Management", IEEE Transactions on Consumer Electronics, (TCE), Vol.58, Issue 3, August, 2012.
5. **Sachin Bhardwaj**, Tanir Ozcelebi, Cagri Uysal and Johan Lukkien, "Resource and Service Management Architecture of a Low Capacity Network for Smart Spaces", IEEE Transactions on Consumer Electronics (TCE), Vol.58, Issue 2, May, 2012.
6. **Sachin Bhardwaj**, Tanir Ozcelebi, Richard Verhoeven, Johan J. Lukkien, "Smart Indoor Solid State Lighting Based on a Novel Illumination Model and Implementation", IEEE Transactions on Consumer Electronics (TCE), Vol.57, Issue 4, November, 2011.
7. **Sachin Bhardwaj**, Aly A. Syed, Tanir Ozcelebi, Johan J. Lukkien, "Power-managed Smart Lighting Using a Semantic Interoperability Architecture", IEEE Transactions on Consumer Electronics, Vol.57, Issue 2, 2011.

Conferences/Workshops

8. Prince U.C Songwa, Aaqib Saeed, **Sachin Bhardwaj**, Thijs W. Kruisselbrink and Tanir Ozcelebi. 'LumNet: Learning to Estimate Vertical Visual Field Luminance for Adaptive Lighting Control.' In: Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2021.
9. **Sachin Bhardwaj**, Keon Myung Lee, Jee-Hyong Lee, "An adaptive framework for applying machine learning in smart spaces", SAC 19, Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp.1263-1270, New York, USA, April 2019.
10. **Sachin Bhardwaj**, Keon Myung Lee, "A Smart Space Design using Deep Learning Approaches", Proceedings of the Korean Content Society ICCC, pp. 39-40, South Korea, December, 2018.
11. Gerrit Niezen, Bram J.J. van der Vlist, **Sachin Bhardwaj** and Tanir Ozcelebi, "Performance Evaluation of a Semantic Smart Space Deployment", IEEE International Conference on Pervasive Computing and Communications Workshops, Lugano, Switzerland, 2012, pp. 835-841.
12. **Sachin Bhardwaj**, Tanir Ozcelebi, Cagri Uysal, Johan Lukkien, "Resource and Service Management Architecture of a Low Capacity Network for Smart Spaces", IEEE ICCE 2012.
13. **Sachin Bhardwaj**, Tanir Ozcelebi, Ozgur Ozunlu, Johan Lukkien, "Increasing Reliability and Availability in Smart Spaces: A Novel Architecture for Resource and Service Management", IEEE ICCE 2012.

14. **Sachin Bhardwaj**, Tanir Ozcelebi, Richard Verhoeven, Johan Lukkien, "Delay Performance in a Semantic Interoperability Architecture" IEEE/IPSJ International Symposium on Applications and the Internet, Munich, Bavaria, 2011, pp. 280-285.
15. **Sachin Bhardwaj**, Aly A. Syed, Tanir Ozcelebi, J.J. Lukkien, "Power-managed Smart Lighting Using a Semantic Interoperability Architecture", IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, USA, January 2011.
16. **Sachin Bhardwaj**, Tanir Ozcelebi and Johan Lukkien, "Failure Detection and Recovery in a Semantic Interoperability Architecture" ICT.Open. Veldhoven, The Netherlands, Nov., 2011.
17. Aly A. Syed, **Sachin Bhardwaj**, Tanir Ozcelebi and Johan Lukkien, "Smart LED Lighting for Power Management in a Building", Artemis Technology Conference, Bologna, Italy, September, 2011.
18. **Sachin Bhardwaj**, Tanir Ozcelebi, Johan Lukkien and Richard Verhoeven, "Semantic Interoperability in a Heterogeneous Smart Lighting System" The IEEE symposium on Computers and Communications, Riccione, Italy, 2010, pp. 1035-1040.
19. **Sachin Bhardwaj**, Tanir Ozcelebi and Johan Lukkien, "Smart Lighting Using LED Luminaries", 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Mannheim, 2010, pp. 654-659.

Posters

20. **Sachin Bhardwaj**, Tanir Ozcelebi and Johan Lukkien, "Failure detection and recovery in a semantic interoperability architecture", ICT.Open, November, 2011.
21. Aly A. Syed, **Sachin Bhardwaj**, Tanir Ozcelebi, Johan Lukkien, "Smart LED Lighting for Power Management in a Building", Artemis Technology Conference, Bologna, Italy, September, 2011.

The author has also contributed to the following publications that go beyond the scope of this thesis:

22. **Sachin Bhardwaj**, Kwan Il Kim, and Keon Myung Lee, "Smart Space Architecture Using a Semantic Learning Approach", The 6th International Conference on Big Data Applications and Services (BIGDAS), August 19-22, 2018.
23. **Sachin Bhardwaj** and Keon Myung Lee, "Smart Healthcare Monitoring Model for Elderly Care using a Semantic Approach", International Conference on Innovation Convergence Technology (ICICT), 2018.
24. Mangal Sain, **Sachin Bhardwaj**, Hoon Jae Lee and Wan-Young Chung, "Architecture of Personal Healthcare Information System in Ubiquitous Healthcare", In Communication and Networking, Vol. 56, Springer, December 2009.
25. **Sachin Bhardwaj**, Dae-Seok Lee, S.C. Mukhopadhyay and Wan-Young Chung "Ubiquitous Healthcare Data Analysis and Monitoring Using Multiple Wireless Sensors for Elderly Person" Sensor and Transducer Journal, Vol. 90, Special Issue, pp. 87-99, April 2008.
26. **Sachin Bhardwaj**, Dae-Seok lee and Wan Young Chung, "Ubiquitous Computing Environment for Healthcare of Elderly Person at Home/Hospital", The Journal of Computer Science and Information Technology, 0973-4872, Vol.5, No.1, Jan-Jun 2007.
27. **Sachin Bhardwaj**, Dae-Seok Lee and Wan-Young Chung, "A Combined QRS-complex and P-wave Detection in ECG Signal for Ubiquitous Healthcare System", International Journal of KIMICS, Vol.5, No.2, June 2007.

28. **Sachin Bhardwaj**, Dae-Seok Lee and Wan-Young Chung, "An ECG Monitoring and Analysis Method for Ubiquitous Healthcare System in WSN", International Journal of KIMICS, Vol. 5, No.1, March 2007.
29. Dae-Seok Lee, **Sachin Bhardwaj** and Wan-Young Chung, "Evaluation of functional sensor node for Ubiquitous Healthcare System" The Second International Symposium on Medical Information and Communication Technology, Dec. 11-13, Oulu, Finland, 2007.
30. **Sachin Bhardwaj**, Dae-Seok Lee, S.C. Mukhopadhyay and Wan-Young Chung, "A Fusion Data Monitoring of Multiple Wireless Sensors for Ubiquitous Healthcare System", 2nd International Conference on Sensing Technology, Nov.26-28, Palmerston North, New Zealand, 2007
31. Dae-Seok Lee, **Sachin Bhardwaj** and Wan-Young Chung "A New Concept of Healthcare Parameter Analysis on Sensor Node for Ubiquitous Healthcare System" International Conference of Convergence Information Technology, IEEE proceeding, pp.1775-1780, Dec 21-23, Gyeongju, South Korea, 2007.
32. Dae-Seok Lee, **Sachin Bhardwaj**, Esko Alasaarela and Wan-Young Chung "An ECG Analysis on Sensor Node for Reducing Traffic Overload in U-Healthcare with Wireless Sensor Network" IEEE Sensors 2007, pp. 256-259, Oct. 28-31, Atlanta, Georgia, USA, 2007.
33. **Sachin Bhardwaj**, Dae-Seok Lee and Wan-Young Chung "An Advance ECG Signal Processing For Ubiquitous Healthcare System", International Conference on Control, Automation and Systems, IEEE proceeding, pp. 2433-2436, Oct. 17-20, Seoul, South Korea, 2007.
34. Wan-Young Chung, **Sachin Bhardwaj**, Amit Purwar, Dae-Seok Lee and Risto Myllylae "A Fusion Health Monitoring and Analysis with ECG and Accelerometer Sensors for Elderly Person at Home" 29th Annual International Conference of the IEEE EMBS (Engineering in Medicine and Biology Society), pp.3818-3821, Aug. 23-26, Lyon, France, 2007.
35. Dae-Seok Lee, **Sachin Bhardwaj**, Wan-Young Chung "U-Healthcare System Using Wireless Sensor Network to Detect QRS Complex", Processing of KISPS Summer Conference 2007, pp.156-159, 2007.6.23, Ulsan, South Korea.
36. **Sachin Bhardwaj**, Dae-Seok Lee, Wan-Young Chung "A ECG Analysis with Activity Monitoring for Healthcare of Elderly Person", Proceeding of KIMICS Conference, 2007.06.01-02,Jinchu, South Korea.
37. Dae-Seok Lee, **Sachin Bhardwaj**, Wan-Young Chung "A Study of Wireless Sensor Node to Detect QRS- Complex with Restrictive Resource" Proceeding of The Korea Society of Medical and Biological Engineering, pp.258-260, 2006.11.03-04, Wonju, South Korea.
38. Amit Purwar, **Sachin Bhardwaj**, Kwang-Sik Shin, Wan-Young Chung "Activity Monitoring Using Triaxial Accelerometer and Wireless Sensor Node" Proceeding of The KISPS autumn conference, pp.281-284, 2006.11.17-18, Daegu, South Korea.
39. **Sachin Bhardwaj**, Dae-Seok Lee, Wan-Young Chung "A Monitoring and Analysis Method of ECG Signal in Wireless Sensor Network" Proceedings of KIMICS Conference, pp491-494, 2006.10.27-28, Kwangju, South Korea.
40. Dae-Seok Lee, **Sachin Bhardwaj**, Wan-Young Chung "WSN Based ECG and Body Temperature Monitoring System" In proceeding of The Korean Institute of Signal Processing and Systems, pp.113-116, 2006.06.16-17, Suncheon, South Korea.
41. Dae-Seok Lee, **Sachin Bhardwaj**, Wan-Young Chung "WSN Real-Time Vital Signal Monitoring System with Patient Diagnosis and Dangerous Alarm Function" Proceeding of The Institute of Control, Automation, and System Engineering, pp.122-126,2006.06.01-03, KINTEX, Kyungi-do, South Korea.

42. **Sachin Bhardwaj**, Dae-Seok Lee, Wan-Young Chung "A Wireless ECG Monitoring System for Application in Life Emergency Event Detection and Analysis" Proceedings of KIMICS Conference, pp.421-425, 2006.07.27-28, Busan, South Korea.
43. **Sachin Bhardwaj**, Dae-Seok Lee, Sung-Ju Oh, Wan-Young Chung "Wireless Sensor Module for Remote Room Environment Monitoring at Home" Proceeding of The Combined Conference of The Institute of Electronic Engineers of Korean, Korean Institute of Communication Society and The Institute of Control, Automation and System Engineering, pp.187-193, 2005.12.3, Busan, South Korea.
44. **Sachin Bhardwaj**, Gaurav Walia, Risto Myllylae, Wan Young Chung "Query Based ECG Monitoring and Analyzing System via Wireless Sensor Network" 2nd International Conference on Wireless Communication and Sensor Networks, pp.146-153, Dec 17-19, India, 2006.
45. Dae-Seok Lee, **Sachin Bhardwaj**, Risto Myllylae, Wan Young Chung" A Wireless ECG Monitoring System for Continuous Event Detection and Analysis" 11th International Meeting on Chemical Sensors, July 16-19, Brescia, Italy, 2006.

List of Figures

List of Tables

Acronyms

AC	Alternating Current
ADC	Analog to Digital Converter
AN	Actuator node
ARP	Address Resolution Protocol
BP	Blood Pressure
BSN	Body Sensor Network
CoAP	Constrained Application Protocol
CHIP	Connected Home over IP
CPU	Central Processing Unit
CSN	Central Smart Node
DaaS	Data as a Service
DHCP	Dynamic Host Configuration Protocol
DLNA	Digital Living Network Alliance
DNS	Domain Name System
ECG	Electrocardiogram
IEEE	Institute of Electrical and Electronics Engineers
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMG	Electromyography
ES-1	Example Scenario 1
ES-2	Example Scenario 2
FTP	File Transfer Protocol
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GSN	Gateway Smart Node
HPR	High-priority Rooms
HSN	High-capacity Smart Node
HTTP	HyperText Transfer Protocol
HVAC	Heating, Ventilation, and Air Conditioning
IaaS	Infrastructure as a Service
ICA	Illumination Control Algorithm
IEC	International Electrotechnical Commission
IoT	Internet of Things
IOP	Interoperability Platform
IP	Internet Protocol
ISO	International Organization for Standardization
KB	Kilo Byte
KiB	Kibi Byte
KP	Knowledge Processor

LED	Light Emitting Diode
LPR	Low-priority Rooms
LSN	Low-capacity Smart Node
MAVHome	Managing Adaptive, Versatile Home
MAC	Medium Access Control
MOP	Multi-objective Optimization Problem
MSN	Manager Smart Node
M3	Multi-vendor, Multi-device, Multi-domain
OSI	Open Systems Interconnection
OWL	Ontology Web Language
PaaS	Platform as a Service
PDA	Personal Digital Assistant
PERSIST	Personal Self-Improving Smart Spaces
PPP	Point-to-Point Protocol
PSN	Passive Smart Node
PSS	Personal Smart Space
PWM	Pulse Width Modulation
PXSN	Proxy Smart Node
REST	Representational State Transfer
RDF	Resource Description Framework
RDF-S	RDF Schema
RFID	Radio Frequency Identification
RGB	Red, Green, Blue
RQ	Research Question
SAN	Semantic Actuator Network
SaaS	Software as a Service
SBSN	Semantic Broker Smart Node
SensorML	Sensor Model Language
SIB	Semantic Information Broker
SMTP	Simple Mail Transfer Protocol
SN	Sensor node
SOFIA	Smart Objects For Intelligent Applications
SOAP	Simple Object Access Protocol
SPIN	SPARQL Inferencing Notation
SSAP	Smart Space Access Protocol
SS	Smart Space
SSN	Semantic Sensor Network
SWE	Sensor Web Enablement
TCP	Transport Control Protocol
TU/e	Eindhoven University of Technology

UC-1	Use Case 1
UC-2	Use Case 2
UDP	User Datagram Protocol
UPS	User Profile Server
UPnP	Universal Plug and Play
USB	Universal Serial Bus
VPN	Virtual Private Network
WQL	Wilbur Query Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks

Bibliography

[1.1] M. Mühlhäuser and I. Gurevych, "Introduction to ubiquitous computing", In Ubiquitous and pervasive computing: Concepts, methodologies, tools, and applications, IGI Global, pp. 1 – 19, 2007.

[1.2] D. A. Patterson, "IRAM: a microprocessor for the post-PC era", International Symposium on VLSI Technology, Systems, and Applications, pp. 30 – 41, 1999.

[1.3] D. Saha, A. Mukherjee, "Pervasive Computing: A Paradigm for the 21st Century", IEEE Computer, March, pp.25 – 31, 2003.

[1.4] E. Aarts, "Ambient Intelligence: a multimedia perspective", IEEE MultiMedia, Vol.11 (1), pp.12 – 19, 2004.

[1.5] R. Want, T. Pering, G. Borriello, K.I. Farkas, "Disappearing hardware (ubiquitous computing)", IEEE Pervasive Computing, Vol. 1(1), pp 36 – 47, 2002.

[1.6] K. N. Biyani, S. S. Kulkarni, "Mixed-Mode Adaptation in Distributed Systems: A Case Study, International Workshop on Software Engineering for Adaptive and Self-Managing Systems", pp. 14, 20 – 26 may, 2007.

[1.7] H. Ishii, "Tangible bits: designing the boundary between people, bits, and atoms, International Symposium on Mixed and Augmented Reality", pp. 199, September 30 – October 1, Darmstadt, Germany, 2002.

[1.8] T. El Kiki, E. Lawrence, "E. Government as a Mobile Enterprise: Real-time, Ubiquitous Government", Third International Conference on Information Technology: New Generations, pp. 320 – 327, 2006.

[1.9] S. Poslad "Ubiquitous Computing Smart Devices, Smart Environments and Smart Interaction", Wiley, ISBN 978-0-470-03560-3, 2009.

[1.10] M. Weiser, "The world is not a desktop", Magazine interactions, Volume 1 Issue 1, pp. 7 – 8, ACM, Jan. 1994.

[1.11] M. Mühlhäuser and I. Gurevych, "Chapter 1.1 Introduction to Ubiquitous Computing", Handbook of Research: Ubiquitous Computing Technology for Real Time Enterprises" edited by Max Mühlhäuser and Iryna Gurevych, , IGI Global, www.igi-pub.com, 2007.

[1.12] S. Bhardwaj, D. S. Lee, S.C. Mukhopadhyay and W. Y. Chung "Ubiquitous Healthcare Data Analysis and Monitoring Using Multiple Wireless Sensors for Elderly Person", Sensors and Transducer Journal, Vol. 90, Special Issue, ISSN: 1726-5479, pp. 87 – 99, April 2008.

[1.13] H. Liu, H. Ning, Q. Mu, Y. Zheng, J. Zeng, L. T. Yang, R. Huang, and J. Ma, "A review of the smart world", Future Generation Computer Systems, pp. 678 – 691, Vol. 96, 2019.

[1.14] J. C. Augusto, V. Callaghan, D. Cook, A. Kameas, and I Satoh, Ichiro. "Intelligent Environments: a manifesto", Human-centric Computing and Information Sciences, Vol. 3, pp. 1 – 18, 2013.

[1.15] A. R. Yuliantoputri, W. Muhamad and S. Suhardi, "Smart Classroom Services System Design Based on Services Computing System", 2019 International Conference on ICT for Smart Society (ICISS), Bandung, Indonesia, pp. 1 – 6, 2019.

[1.16] P. Mtshali and F. Khubisa, "A Smart Home Appliance Control System for Physically Disabled People", 2019 Conference on Information Communications Technology and Society (ICTAS), Durban, South Africa, pp. 1 – 5, 2019.

[1.17] T. Dahoumane, M. Haddadi and Z. Amokrane, "Web Services and GSM based Smart Home Control System", 2018 International Conference on Applied Smart Systems (ICASS), Medea, Algeria, pp. 1 – 4, 2018.

[1.18] S. Sharma, A. Sharma, T. Goel, R. Deoli and S. Mohan, "Smart Home Gardening Management System: A Cloud-Based Internet-of-Things (IoT) Application in VANET", 2020 11th Interna-

tional Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, pp. 1 – 5, 2020.
[1.19] B. G. Mohammed and D. S. Hasan, "Smart Healthcare Monitoring System Using IoT", International Journal of Interactive Mobile Technologies, (iJIM), 17(01), pp. 141 – 152, 2023.
[1.20] C. Li and S. Xu, "Interaction Design for Smart Healthcare System Considering Older Adults' Healthy and Wellbeing Lifestyles", 2020 IEEE 2nd Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), Tainan, Taiwan, pp. 151 – 153, 2020.
[1.21] W. Antoun, A. Abdo, S. Al-Yaman, A. Kassem, M. Hamad and C. El-Moucary, "Smart Medicine Dispenser (SMD)", 2018 IEEE 4th Middle East Conference on Biomedical Engineering (MECBME), Tunis, Tunisia, pp. 20 – 23, 2018.
[1.22] O. Ayan and B. Turkay, "IoT-Based Energy Efficiency in Smart Homes by Smart Lighting Solutions", 2020 21st International Symposium on Electrical Apparatus & Technologies (SIELA), Bourgas, Bulgaria, pp. 1 – 5, 2020.
[1.23] G. S. Ramachandran, R. Radhakrishnan and B. Krishnamachari, "Towards a Decentralized Data Marketplace for Smart Cities", 2018 IEEE International Smart Cities Conference (ISC2), Kansas City, MO, USA, pp. 1 – 8, 2018.
[1.24] M. Karaduman and H. Eren, "Smart driving in smart city", 2017 5th International Istanbul Smart Grid and Cities Congress and Fair (ICSG), Istanbul, Turkey, pp. 115 – 119, 2017.
[1.25] S. Antonov, "Smart Solution for Fire Safety in a Large Garage", 2019 International Conference on Creative Business for Smart and Sustainable Growth (CREBUS), Sandanski, Bulgaria, pp. 1 – 4, 2019.
[1.26] N. Ouerhani, N. Pazos, M. Aeberli and M. Muller, "IoT-based dynamic street light control for smart cities use cases", 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, pp. 1 – 5, 2016.
[1.27] S. S. Arumugam et al., "IOT Enabled Smart Logistics Using Smart Contracts", 2018 8th International Conference on Logistics, Informatics and Service Sciences (LISS), Toronto, ON, Canada, pp. 1 – 6, 2018.
[1.28] M. Botticelli, A. Monteriù, A. Zanela and S. Romano, "Smart Homes and Assisted Living as an Additional Service Offered to the Users", 2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin), Berlin, Germany, pp. 42 – 45, 2019.
[1.29] I. A. Gufron, O. Fathurohman, M. Roifah, M. Wildan, P. Supendi and E. A. Z. Hamidi, "Prototype Design of Smart Office at Institut Agama Islam Bunga Bangsa Cirebon (IAI-BBC) Base on LoRa", 2020 6th International Conference on Wireless and Telematics (ICWT), Yogyakarta, Indonesia, pp. 1 – 6, 2020.
[1.30] IEEE Computer Society, IEEE Std 802.15.4-2006: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), 323 pages, Sept. 2006.
[1.31] V. D. K. Mai and Y. Kim, "Using DLNA cloud for sharing multimedia contents beyond home networks", 16th International Conference on Advanced Communication Technology, Pyeongchang, pp. 54 – 57, 2014.
[1.32] P. Kafka, "Spotify relies on the big labels for most of its music. It thinks that will change". [online] Recode. Available at: https://www.recode.net/2018/4/3/17191390/spotify-plan-big-music-labels-platform-marketplace-ipo [Accessed 27 Oct. 2023].
[1.33] "Google Cloud for media and entertainment" [online], https://cloud.google.com/solutions/media-entertainment/. [Accessed 27 Oct. 2023].

[1.34] J. Kiljander, A. D' Elia, F. Morandi, P. Hyttinen, J. T. Mattila, A. Ylisaukko-oja, J. Soininen, and T. S. Cinotti, "Semantic Interoperability Architecture for Pervasive Computing and Internet of Things", IEEE Access, vol. 2, pp. 856 – 873, 2014.

[1.35] J. Fernández, G. Pimpollo, and R. Otaolea, "Smart Objects for Intelligent Applications – ADK", IEEE Symposium on Visual Languages and Human-Centric Computing, Leganes, Spain, pp.267 – 268, 21-25 Sept. 2010.

[2.1] S. B. Baker, W. Xiang and I. Atkinson, "Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities", in IEEE Access, vol. 5, pp. 26521 – 26544, 2017.

[2.2] H. Jiang, C. Cai, X. Ma, Y. Yang and J. Liu, "Smart Home Based on WiFi Sensing: A Survey", in IEEE Access, vol. 6, pp. 13317 – 13325, 2018.

[2.3] L. Qiu, Q. Lei and Z. Zhang, "Advanced Sentiment Classification of Tibetan Microblogs on Smart Campuses Based on Multi-Feature Fusion", in IEEE Access, vol. 6, pp. 17896 – 17904, 2018.

[2.4] E. Mathews; S. S. Guclu, Q. Liu, T. Ozcelebi, and J.J. Lukkien, "The Internet of Lights: An Open Reference Architecture and Implementation for Intelligent Solid State Lighting Systems", Energies 10(8), 1187, pp. 1 – 27, 2017.

[2.5] K. Scott and R. Benlamri, "Context-Aware Services for Smart Learning Spaces", IEEE Transactions on Learning Technologies, vol. 3, no. 3, pp. 214 – 227, 2010.

[2.6] N. D. Rodriguez, "A Framework for Context-Aware Applications for Smart Spaces", IEEE/IPSJ International Symposium on Applications and the Internet, Munich, Bavaria, pp. 218 – 221, 2011.

[2.7] T. Ozcelebi, J. J. Lukkien, R. Bosman, and O. Uzun, "Discovery, monitoring and management in smart spaces composed of low capacity nodes", IEEE Transactions on Consumer Electronics (TCE), Vol. 56, no. 2, pp. 570 – 578, May 2010.

[2.8] S. Bhardwaj, T. Ozcelebi, J. Lukkien and C. Uysal, "Resource and service management architecture of a low capacity network for smart spaces", IEEE Transactions on Consumer Electronics, vol. 58, no. 2, pp. 389–396, May 2012.

[2.9] Z. Shelby, B. Frank and D. Sturek, "Constrained Application Protocol (CoAP) (CoRE Working Group)", http://www.ietf.org/ id/draft-ietf-core-coap-06.txt (07/07/2011), 2011.

[2.10] H. He, T. Watson, C. Maple, J. Mehnen and A. Tiwari, "A new semantic attribute deep learning with a linguistic attribute hierarchy for spam detection", International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, Alaska, pp. 3862 – 3869, 2017.

[2.11] P. Maillot, T. Raimbault, D. Genest and S. Loiseau, "Consistency Evaluation of RDF Data: How Data and Updates are Relevant", Tenth International Conference on Signal-Image Technology and Internet-Based Systems, Marrakech, Morocco, pp. 187 – 193, 2014.

[2.12] K. A. Taipale, "The Trusted Systems Problem: Security Envelopes, Statistical Threat Analysis, and the Presumption of Innocence," Homeland Security - Trends and Controversies, IEEE Intelligent Systems, Vol. 20 No. 5, pp. 80 – 83, Sept-Oct, 2005.

[2.13] R. M. Lee, M. J. Assante and T. Conway, "Analysis of the cyber-attack on the Ukrainian power grid", 2016.

[2.14] F. Alrimawi, L. Pasquale and B. Nuseibeh, "On the Automated Management of Security Incidents in Smart Spaces", in IEEE Access, vol. 7, pp. 111513 – 111527, 2019.

[2.15] K. K. Jung and Y. J. Kim, "Design of smart monitoring system based on bluetooth low energy", International Conference on Electronics, Information, and Communication (ICEIC), Honolulu, HI, USA, pp. 1 – 3, 2018.

[2.16] D. M. Han and J. H. Lim, "Smart home energy management system using IEEE 802.15.4 and ZigBee", IEEE Transactions on Consumer Electronics, vol.56, issue 3, pp.1403 – 1410, August 2010.

[2.17] S. Bhardwaj, A. A. Syed, T. Ozcelebi and J. J. Lukkien, “Power-managed smart lighting using a semantic interoperability architecture”, IEEE Transactions on Consumer Electronics, vol. 57, no. 2, pp. 420 – 427, May 2011.

[2.18] E. Carlini, M. Coppola, P. Dazzi, M. Mordacchini and A. Passarella, “Self-Optimising Decentralised Service Placement in Heterogeneous Cloud Federation”, IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), Augsburg, Germany, pp. 110 – 119, 2016.

[2.19] X. Wang and Y. Mu, “Addressing and Privacy Support for 6LoWPAN”, in IEEE Sensors Journal, vol. 15, no. 9, pp. 5193 – 5201, Sept. 2015.

[2.20] F. Wu, C. Rüdiger, J. M. Redouté and M. R. Yuce, “WE-Safe: A wearable IoT sensor node for safety applications via LoRa”, IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, pp. 144 – 148, 2018.

[2.21] A. K. Dey, G. D. Abowd, and D. Salber, “A Context-Based Infrastructure for Smart Environments,” GVU Technical Report;GIT-GVU-99-39, Georgia Institute of Technology, http://hdl.handle.net/ 1853/3406, 1999.

[2.22] E. Goh, D. Chieng, A. K. Mustapha, Y. C. Ngeow and H. K. Low, “A Context-Aware Architecture for Smart Space Environment”, International Conference on Multimedia and Ubiquitous Engineering (MUE’07), Seoul, South Korea, pp. 908 – 913, 2007.

[2.23] I. G. Roussaki, N. K. Kalatzis, K. J. Doolin, N. K. Taylor, G. P. Spadotto, N. D. Liampotis, and M. H. Williams, “Self-improving personal smart spaces for pervasive service provision,” IOS Press, E-book: Towards the Future Internet, pp. 193 – 203, 2010.

[2.24] T. Kawashima, M. Jianhua, H. Runhe, and B. O. Apduhan, “GUPSS: A Gateway-Based Ubiquitous Platform for Smart spaces,” International Conference on Computational Science and Engineering, Vancouver, BC, Canada, pp. 213 – 220, 2009.

[2.25] B. J. J. Vlist, G. Niezen, J. Hu and L. M. G. Feijs, “Semantic connections: Exploring and manipulating connections in smart spaces”, The IEEE symposium on Computers and Communications, Riccione, Italy, pp. 1 – 4, 2010.

[2.26] Z. Song, A. A. Cárdenas and R. Masuoka, “Semantic Middleware for the Internet of Things,” Proc IEEE Internet of Things (IOT), Tokyo, Japan, pp. 1 – 8, 2010.

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

[2.28] H. Abdullah, M. Rinne, S. Törmä and E. Nuutila “Efficient Matching of SPARQL Subscriptions using Rete”, Proc 27th Annual ACM Symposium on Applied Computing, New York, NY, USA, pp. 372 – 377, 2012.

[2.29] F. Morandi, L. Roffia, A. D’Elia, F. Vergari and T. S. Cinotti, “RedSib: A smart-M3 semantic information broker implementation”, 12th Conference of Open Innovations Association (FRUCT), Oulu, Finland, pp. 1 – 13, 2012.

[2.30] N. Díaz Rodríguez, J. Lilius, M. P. Cuéllar and M. Delgado Calvo-Flores, “An approach to improve semantics in Smart Spaces using reactive fuzzy rules”, Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), Edmonton, AB, Canada, pp. 436 – 441, 2013.

[2.31] E. Ovaska and J. Kuusijärvi, “Piecemeal Development of Intelligent Applications for Smart Spaces”, IEEE Access, vol. 2, pp. 199 – 214, 2014.

[2.32] J. Kiljander, A. Delia, F. Morandi, P. Hyttinen, J. T. Mattila, A. Ylisaukko-oja, J. Soininen, and T. S. Cinotti, “Semantic Interoperability Architecture for Pervasive Computing and Internet of Things”, IEEE Access, vol. 2, pp. 856 – 873, 2014.

[2.33] J. Zeng, L. T. Yang, H. Ning and J. Ma, “A systematic methodology for augmenting quality of experience in smart space design,” IEEE Wireless Communications, vol. 22, no. 4, pp. 81 – 87, 2015.

[2.34] S. A. Marchenkov, D. G. Korzun, A. I. Shabaev and A. V. Voronin, “On applicability of wireless routers to deployment of smart spaces in Internet of Things environments,” 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest, Romania, pp. 1000 – 1005, 2017.

[2.35] A. S. Vdovenko, D. G. Korzun and I. V. Galov, “Simulation performance evaluation of SmartM3 applications for Internet of Things environments,” 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest, Romania, pp. 994 – 999, 2017.

[2.36] A. S. Vdovenko, O. I. Bogoiavlenskaia and D. G. Korzun, “Study of active subscription control parameters in large-scale smart spaces,” 2017 21st Conference of Open Innovations Association (FRUCT), Helsinki, Finland, pp. 344 – 350, 2017.

[2.37] S. Ahmad, L. Hang, and D.H. Kim, “Design and Implementation of Cloud-Centric Configuration Repository for DIY IoT Applications,” Sensors, 18, 474, 2018.

[2.38] J. Zeng, L. T. Yang, J. Ma and M. Guo, “HyperspaceFlow: A System-Level Design Methodology for Smart Space,” IEEE Transactions on Emerging Topics in Computing, vol. 4, no. 4, pp. 568 – 583, Oct.-Dec. 2016.

[2.39] D. J. Cook, M. Youngblood, E.O. III Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, “MavHome: An Agent-Based Smart Home”, Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, (PerCom 2003), Fort Worth, Texas, USA, pp 521 – 524, 2003.

[2.40] Sukalikar, Shriya, S. Kumar, and N. Baliyan. “Analysing Cohesion and Coupling for Modular Ontologies.” International Conference on Advances in Computing, Communications and Informatics (ICACCI), New Delhi, India, pp. 2063-2066, 2014.

[2.41] W. Lumpkins, “Home automation: Insteon (X10 meets powerline)” IEEE Consumer Electronics Magazine,” vol.4, pp.140 – 144, 2015.

[2.42] M. Neugebauer, J. Plonnigs, K. Kabitzsch, and P. Buchholz, “Automated modeling of LonWorks building automation net-works,” Proceedings of IEEE International Workshop on Factory Communication Systems, Vienna, Austria, Array, pp.113 – 118, 2004.

[2.43] T. Perumal, A. R. Ramli, C. Y. Leong, and S. Mansor, “Interoperability for Smart Home Environment Using Web Services”, International Journal of Smart Home , vol. 2, no. 4, October, 2008.

[2.44] R. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Ph.D. dissertation, University of California, Irvine, [Online]. Available: https://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm, 2000.

[2.45] W. Villegas-Ch, X. Palacios-Pacheco, and S. Luján-Mora, “Application of a smart city model to a traditional university campus with a big data architecture: A sustainable smart campus”, Sustainability, 11, 2857, 2019.

[2.46] Z. D. Tekler, R. Low, B. Gunay, R. K. Andersen and L. Blessing, “A scalable Bluetooth Low Energy approach to identify occupancy patterns and profiles in office spaces”, Building and Environment, Vol. 171, 2020.

[2.47] Z. D. Tekler and A. Chong, “Occupancy prediction using deep learning approaches across multiple space types: A minimum sensing strategy”, Build. Environ., 226, 109689, 2022.

[2.48] B. Alsamani, S. Chatterjee, A. Anjomshoae, and P. Ractham, “Smart Space Design–A Framework and an IoT Prototype Implementation”, Sustainability, 15, 111, 2023.

[3.1] Y. Guozheng, W. Hongjie, D. Guoqing and L. Liangming, “Augmented Lagrange multiplier based fuzzy evolutionary algorithm and application for constrained optimization,” Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No.02EX527), Shanghai, China, pp. 1774 – 1778 vol.3, 2002.

[3.2] D. A. G. Vieira, R. L. S. Adriano, J. A. Vasconcelos and L. Krahenbuhl, “Treating constraints as objectives in multiobjective optimization problems using niched Pareto genetic algorithm,” in IEEE Transactions on Magnetics, vol. 40, no. 2, pp. 1188 – 1191, March 2004.

[3.3] I. Paweloszek and J. Korczak, “From data exploration to semantic model of customer,” 2017 Intelligent Systems Conference (IntelliSys), United Kingdom, pp. 382 – 388, 2017.

[3.4] P. Warren, J. Davies; D. Brown, “The Semantic Web – From Vision to Reality,” in ICT Futures:Delivering Pervasive, Real-time and Secure Services , 1, Wiley Telecom, pp.268, 2007.

[3.5] Y. Wang, “Formal rules for concept and semantics manipulations in cognitive linguistics and machine learning,” 2017 IEEE 16th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), Oxford, pp. 43 – 50, 2017.

[3.6] A. L. Garrido, M. S. Pera and S. Ilarri, “SOLE-R: A Semantic and Linguistic Approach for Book Recommendations,” 2014 IEEE 14th International Conference on Advanced Learning Technologies, Athens, pp. 524 – 528, 2014.

[3.7] O. Kovalenko, J. Euzenat, “Semantic Matching of Engineering Data Structures”, In: Biffl S., Sabou M. (eds) Semantic Web Technologies for Intelligent Engineering Applications. Springer, Cham, 2016.

[3.8] J. Nilsson and F. Sandin, “Semantic Interoperability in Industry 4.0: Survey of Recent Developments and Outlook,” 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, Portugal, pp. 127 – 132, 2018.

[3.9] O. Novo and M. D. Francesco. “Semantic Interoperability in the IoT: Extending the Web of Things Architecture”. ACM Transaction of Internet Things 1, 1, Article 6, 25 pages, February, 2020.

[3.10] K. Kadowaki, T. Koita, K. Sato and H. Hayakawa, “Design and Implementation of Adaptive Jini System to Support Undefined Services,” 6th Annual Communication Networks and Services Research Conference (cnsr), Halifax, NS, 2008, pp. 577 – 583, 2008.

[3.11] R. Lea, S. Gibbs, A. Dara-Abrams and E. Eytchison, “Networking home entertainment devices with HAVi,” in Computer, vol. 33, no. 9, pp. 35 – 43, Sep 2000.

[3.12] L. Yiqin, F. Fang and L. Wei, “Home Networking and Control Based on UPnP: An Implementation,” 2009 Second International Workshop on Computer Science and Engineering, Qingdao, pp. 385 – 389, 2009.

[3.13] J. Cardoso, C. Pereira, A. Aguiar and R. Morla, “Benchmarking IoT middleware platforms,” 2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), Macau, pp. 1 – 7, China, 2017.

[3.14] M. Khan, S. Din, S. Jabbar, M. Gohar, H. Ghayvat, and S. C. Mukhopadhyay, “Context-aware low power intelligent SmartHome based on the Internet of things,” Computers & Electrical Engineering, vol. 52, pp. 208 – 222, 2016.

[3.15] B. R. Nugroho, “The architecture of an IoT-based healthcare monitoring system using smart e-health gateways in home/hospital domain,” Buletin Inovasi ICT & Ilmu Komputer, vol. 2, no. 1, 2015.

[3.16] S. Jabbar, M. Khan, B. Nathali Silva, and K. Han, “A REST-based industrial web of things’ framework for smart warehousing,” The Journal of Supercomputing, 2016.

[3.17] A. Agra, M. Christiansen, K. S. Ivarsøy, I. E. Solhaug, and A. Tomasgard, “Combined ship routing and inventory management in the salmon farming industry,” Annals of Operations Research, pp. 1 – 25, 2016.
[3.18] C. C. Grant, A. Jones, A. Hamins, and N. Bryner, “Realizing the vision of smart fire fighting,” IEEE Potentials, vol. 34, no. 1, pp. 35 – 40, 2015.
[3.19] A. Paul, A. Ahmad, M. M. Rathore, and S. Jabbar, “Smartbuddy: defining human behaviors using big data analytics in social internet of things,” IEEE Wireless Communications, vol. 23, no. 5, pp. 68 – 74, 2016.
[3.20] R. Zhang, S. Newman, M. Ortolani and S. Silvestri, “A Network Tomography Approach for Traffic Monitoring in Smart Cities,” in IEEE Transactions on Intelligent Transportation Systems, 2018.
[3.21] D. M. Llido Escriva, J. Torres-Sospedra and R. Berlanga-Llavori, “Smart Outdoor Light Desktop Central Management System,” in IEEE Intelligent Transportation Systems Magazine, vol. 10, no. 2, pp. 58 – 68, Summer 2018.
[3.22] X. Su, H. Zhang, J. Riekki, A. Keränen, J. K. Nurminen, L. Du, “Connecting IoT Sensors to Knowledge-based Systems by Transforming SenML to RDF”, Procedia Computer Science, Volume 32, pp. 215 – 222, 2014.
[3.23] J. J. Jung, “Semantic preprocessing for mining sensor streams from heterogeneous environments” Expert Systems with Applications, Volume 38, Issue 5, pp. 6107 – 6111, May 2011.
[3.24] A. Flora, C. Valentina, G. Andrea, M. Antonino, “A semantic enriched data model for sensor network interoperability” Simulation Modelling Practice and Theory, Volume 19, Issue 8, pp. 1745 – 1757, September, 2011.
[3.25] X. Su; H. Zhang; J. Riekki; A. Keränen; J. K. Nurminen; L. Du, “Connecting IoT Sensors to Knowledge-based Systems by Transforming SenML to RDF”, Procedia Computer Science, Volume 32, pp. 215 – 222, 2014.
[3.26] S. Jabbar, F. Ullah, S. Khalid, Murad Khan, and Kijun Han, “Semantic Interoperability in Heterogeneous IoT Infrastructure for Healthcare,” Wireless Communications and Mobile Computing, vol. 2017, Article ID 9731806, 10 pages, 2017.
[3.27] C. Malewski, A. Bröring, P. Maué and K. Janowicz, “Semantic Matchmaking & Mediation for Sensors on the Sensor Web,” in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 3, pp. 929 – 934, March 2014.
[3.28] H. Dibowski, J. Ploennigs and M. Wollschlaeger, “Semantic Device and System Modeling for Automation Systems and Sensor Networks,” in IEEE Transactions on Industrial Informatics, vol. 14, no. 4, pp. 1298 – 1311, April 2018.
[3.29] R. Edwards, L. Parker, D. Resseguie, “Robopedia: Leveraging Sensorpedia for web-enabled robot control”. 183-188. 10.1109/PERCOMW.2010.5470670, 2010.
[3.30] K. A. Delin and E. Small “The Sensor Web: Advanced Technology for Situational Awareness”, Wiley Handbook of Science and Technology for Homeland Security, John Wiley & Sons, 2009.
[3.31] S. Nath, J. Liu, and F. Zhao “SensorMap for Wide-Area Sensor Webs”, Computers, Vol 40, No. 7, pp 90 – 93, July 2007.
[3.32] S. Bhardwaj, T. Ozcelebi, O. Ozunlu, J. J. Lukkien, “Increasing Reliability and Availability in Smart Spaces: A Novel Architecture for Resource and Service Management”, IEEE Transactions on Consumer Electronics, (TCE), Vol.58, Issue 3, August, 2012.
[4.1] A. Tolk, ‘‘Composable mission spaces and M&S repositories— Applicability of open standards,’’ in Proc. Simulat. Interoperability Workshop, Washington, DC, USA, pp. 1 – 14, 2004.

[4.2] S. Pantsar-Syväniemi, A. Purhonen, E. Ovaska, J. Kuusijärvi, and A. Evesti, ‘‘Situation-based and self-adaptive applications for the smart environment,’’ J. Ambient Intell. Smart Environment, vol. 4, no. 6, pp. 491 – 516, 2012.

[4.3] Barbara Hayes-Roth, “A blackboard architecture for control”, Artificial Intelligence, vol. 26, Issue 3, Pages 251 – 321, 1985.

[5.1] R. Brown, World On the Edge: How to Prevent Environmental and Economic Collapse (New York: W.W. Norton & Company, 2010.

[5.2] S. H. A. Begemann, G. J. van den Beld, A. D.Tenner, “Daylight, artificial light and people in an office environment”, overview of visual and biological responses. International Journal of Industrial Ergonomics; 20: pp. 231 – 239, 1997.

[5.3] Y.A.W. de Kort, K.C.H.J. Smolders, “Effects of dynamic lighting on office workers: first results of a field study with monthly alternating settings”, Lighting Research and Technology, 42(3), pp. 345 – 360, 2010.

[5.4] R. Meier, “The Light. Global User Study on Perceived Lighting Quality in Offices (2015)” http://www. zumtobel.web-erhebung.de/english/, last accessed: October 2023.

[5.5] S. Lee, I. Shin and N. Lee, “Development of Intelligent Space Lighting System,” 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea (South), pp. 1030 – 1032, 2019.

[5.6] J. Kolo, U. Dauda, “Development of a Simple Programmable Control Timer”, Leonardo Journal of Sciences, 2008.

[5.7] D. Hossain, H. Alam, A. A. Rafi, K. Rakib and U. Hany, “Development of Microcontroller based Light Following Emergency Evacuation Chair,” 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, pp. 1 – 5, 2019.

[5.8] J. Gao, F. Yang and X. Ma, “Indoor positioning system based on visible light communication with gray-coded identification,” 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, pp. 899-903, 2017.

[5.9] K.C.H.J. Smolders, Y.A.W. de Kort, P.J.M. Cluitmans, “A higher illuminance induces alertness even during office hours : findings on subjective measures, task performance and heart rate measures”, Physiology & Behavior, 107(1), pp. 7 – 16, 2012.

[5.10] K. Kaida, M. Takahashi, T. Haratani, Y. Otsuka, K. Fukasawa, A. Nakata A, “Indoor exposure to natural bright light prevents afternoon sleepiness”, Sleep, 29(4), pp. 462 – 469, 2006.

[5.11] J. Uttley, S. Fotios and C. Cheal, “Using lighting to make pavements safer for pedestrians” Experiencing Light, 2014.

[5.12] K. Kaida, M. Takahashi, Y. Otsuka, “A short nap and natural bright light exposure improve positive mood status”, Industrial Health; 45(2), pp. 301 – 308, 2007.

[5.13] T. Morita, H. Tokura, “The influence of different wavelengths of light on human biological rhythms”, Applied Human Science, 17, pp. 91 – 96, 1998.

[5.14] G. Chinazzo, J. Wienold, M. Andersen, “Daylight affects human thermal perception”, Sci Rep 9, 13690, https://doi.org/10.1038/s41598-019-48963-y, 2019.

[5.15] M. Ruger, M. C. M. Gordijn, D. G. M. Beersma, B. de Vries, S. Daan S, “Time-of-day-dependent effects of bright light exposure on human psychophysiology”, Comparison of daytime, night time exposure. American Journal of Physiology – Regulatory, Integrative and Comparative Physiology; 290(5), pp. 1413 – 420, 2006.

[5.16] F. A. Scheer, R. M. Buijs, “Light affects morning salivary cortisol in humans”, Journal of Clinical Endocrinical Metabolism; 84(9), pp. 3395 – 3398, 1999.

[5.17] B. Wansink, K. Van Ittersum, “Fast Food restaurant Lighting and Music can Reduce Calorie Intake and Increase Satisfaction. Psychological Reports”, Human Resources & Marketing, 111(1), pp. 1 – 5, 2012.

[5.18] Y. de Kort, W. IJsselsteijn, A. Haans, D. Lakens, I. Kalinauskaite and A. Schietecat, “De-escalate: Defusing escalating behaviour through the use of interactive light scenarios”, International Conference on the Effects of Light on Wellbeing, Experiencing Light, 2014.

[5.19] J. Martínez-Patiño, J. M. Lozano-García, I. A. Hernández-Robles, P. Sánchez-Razo and F. F. Macias-Aguilera, “Cost for Decorative Lighting in Homes, Case of Study: Seasonal Lighting,” 2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), Valparaiso, Chile, pp. 1 – 4, 2019.

[5.20] J. Higuera, W. Hertog, M. Perálvarez, J. Polo and J. Carreras, “Smart Lighting System ISO/IEC/IEEE 21451 Compatible,” in IEEE Sensors Journal, vol. 15, no. 5, pp. 2595 – 2602, May 2015.

[5.21] S. Kim, W. Kang and H. Ku, “Networked smart LED lighting system and its application using Bluetooth beacon communication,” 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Seoul, pp. 1 – 4, 2016.

[5.22] A. Prasetio, S. R. Akbar and B. Priyambadha, “Implementation of semantic system in the smart home lights device based on agent,” 2017 International Conference on Sustainable Information Engineering and Technology (SIET), Malang, pp. 93 – 99, 2017.

[5.23] J. Liu, W. Zhang, X. Chu, Y. Liu “Fuzzy logic controller for energy savings in a smart LED lighting system considering lighting comfort and daylight” Energy and Buildings, Elsevier Science Direct, vol. 127 pp. 95 – 104, 2016.

[5.24] T. M. Lwin, A. Kumar, N. Xavier and S. K. Panda, “A smart lighting system using wireless sensor actuator network,” 2017 Intelligent Systems Conference (IntelliSys), United Kingdom, pp. 217 – 220, 2017.

[5.25] F. Viani, A. Polo, P. Garofalo, N. Anselmi, M. Salucci and E. Giarola, “Evolutionary Optimization Applied to Wireless Smart Lighting in Energy-Efficient Museums,” IEEE Sensors Journal, vol. 17, no. 5, pp. 1213 – 1214, 2017.

[5.26] V. Barve, “Smart lighting for smart cities,” 2017 IEEE Region 10 Symposium (TENSYMP), Cochin, pp. 1 – 5, 2017.

[5.27] N. Khatavkar, A. A. Naik and B. Kadam, “Energy efficient street light controller for smart cities,” 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS), Vellore, pp. 1 – 6, 2017.

[5.28] A. Kumar, A. Kajale, P. Kar, A. Shareef and S. K. Panda, “Location-aware smart lighting system for individual visual comfort in buildings,” 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE), Nagoya, pp. 1 – 2, 2017.

[5.29] E. Mathews, S. S. Guclu, Q. Liu, T. Ozcelebi, J. J. Lukkien, “ The Internet of Lights: An Open Reference Architecture and Implementation for Intelligent Solid State Lighting Systems” Energies, 10, 1187, 2017.

[5.30] A. K. Sikder, A. Acar, H. Aksu, A. S. Uluagac, K. Akkaya and M. Conti, “IoT-enabled smart lighting systems for smart cities,” 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, pp. 639 – 645, 2018.

[5.31] C. A. G. Amarillo, C. L. C. García, J. A. C. Muñoz, and M. A. M. Moreno, “Smart Lumini: A Smart Lighting System for Academic Environments Using IOT-Based Open-Source Hardware,” Revista Facultad de Ingeniería, vol. 29 (54), e11060, 2020.

[5.32] M. Soheilian, G. Fischl and M. Aries, “Smart Lighting Application for Energy Saving and User Well-Being in the Residential Environment”, Sustainability, 13(11), May, 2021.

[5.33] D. Gowda, A. Annepu, Ramesha, P. Kumar, and P. Singh, “IoT Enabled Smart Lighting System for Smart Cities”, Journal of Physics: Conference Series, 2021.

[5.34] A. K. Putri, A. Pramono, D. M. Yasmin, B. A. Safitri, Y. S. Zaharani and F. F. Zebua, “The Smart Lighting System in the Coworking Space’s Meeting Room,” 2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), Jakarta, Indonesia, pp. 534 – 538, 2022.

[5.35] M. Miki, T. Hiroyasu and K. Imazato, “Proposal for an intelligent lighting system and verification of control method effectiveness” IEEE Cybernetics and Intelligent Systems, vol.1, pp.520 – 525, Dec. 2004.

[5.36] Y.-J. Wen and A.M. Agogino, “Wireless networked lighting systems for optimizing energy savings and user satisfaction” in Proc. of IEEE Wireless Hive Networks Conf., Austin, Texas, USA, , pp.1 – 7, Aug. 2008.

[5.37] M.-S. Pan, L.-W. Yeh, Y.-A. Chen, Y.-H. Lin and Y.-C. Tseng, “A WSN-based intelligent light control system considering user activities and profiles,” IEEE Sensors Journal, vol. 8, issue 10, pp. 1710 – 1721, Oct. 2008.

[5.38] M. Richards and D. Carter, “Good lighting with less energy: Where next?”, Lighting Research Technology, vol.41, 285 – 286, 2009.

[5.39] T. M. Goodman, “Measurement and specification of lighting: A look at the future,” Lighting Research and Technology, vol.41, no.3, pp.229 – 243, 2009.

[5.40] H. Yang, J. W. M. Bergmans and T. C.W. Schenk, “Illumination sensing in LED lighting systems based on frequency-division multiplexing” IEEE Tans. On Signal Processing, vol.57, no.11, pp 4269 – 4281, Nov. 2009.

[5.41] H. Yang, J. W.M. Bergmans, T.C.W. Schenk and J.M.G. Linnartz, “Uniform: illumination rendering using an array of LEDs: A signal processing perspective”, IEEE Transactions on Signal Processing, vol.57,no.3, pp.1044 – 1057, Mar., 2009.

[5.42] D. M. Han and J. H. Lim, “Smart home energy management system using IEEE 802.15.4 and ZigBee”, IEEE Trans. on Consumer Electronics, vol.56, issue 3, pp.1403 – 1410, Aug. 2010.

[5.43] Y. J. Wen and A.M. Agogino, “Control of wireless-networked lighting in open-plan offices”, Lighting Research and Technology, DOI:10.1177/1477153510382954, Nov. 2010.

[5.44] E. M. Guttsait, “Analysis of LED modules for local illumination,” Journal of Communications Technology and Electronics, vol. 52, no. 12, pp. 1377 – 1395, 2007.

[5.45] A. Ryer, “Light measurement handbook.” Int. Lights Inc, pp.25, 1998.

[5.46] J. Nipkow and E. Bush, “Stand-by consumption of household appliances,” Swiss Agency for Efficient Energy Use S.A.F.E. on behalf of the Swiss Federal Office of Energy SFOE, Berne 2003.

[6.1] L. F. Rahman, T. Ozcelebi, and J. J. Lukkien, “Understanding IoT Systems: A Life Cycle Approach”, Procedia Computer Science, Volume 130, pp. 1057 – 1062, 2018.

[7.1] S. Belaidouni and M. Miraoui, “Machine Learning Technologies in Smart Spaces”, The Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies”, UBICOMM, Venice, Italy, pp. 52 – 55, 2016.

[7.2] M. Babar and F. Arif, “Smart urban planning using big data analytics based internet of things”, UbiComp’17, ACM International Joint Conference on Pervasive and Ubiquitous Computing and International Symposium on Wearable Computers, pages 397 – 402, Hawaii, USA, September 11 - 15, 2017.

[7.3] S. H. Kaisler, W. Money and S. Cohen, “Smart Objects: An Active Big Data Approach”, 51st Hawaii International Conference on System Sciences, pp. 809 – 818, USA, 2018.

[7.4] D. Cook, “Learning Setting-Generalized Activity Models for Smart Spaces,” in IEEE Intelligent Systems, vol. 27, no. 1, pp. 32 – 38, Jan.-Feb. 2012.

[7.5] A. S. Vdovenko, O. I. Bogoiavlenskaia and D. G. Korzun, “Study of active subscription control parameters in large-scale smart spaces,” 2017 21st Conference of Open Innovations Association (FRUCT), Helsinki, pp. 344 – 350, 2017.

[A.1] F. Garzotto and M. Gelsomini, “Magic Room: A Smart Space for Children with Neurodevelopmental Disorder,” in IEEE Pervasive Computing, vol. 17, no. 1, pp. 38 – 48, Jan.-Mar. 2018.

[A.2] B. J. J. van der Vlist, G. Niezen, J. Hu and L. M. G. Feijs, “Interaction primitives: Describing interaction capabilities of Smart Objects in ubiquitous computing environments,” IEEE Africon’11, Victoria Falls, Zambia, pp. 1 – 6, 2011.

[A.3] G. Niezen, B. J.J. van der Vlist, S. Bhardwaj and T. Ozcelebi, “Performance Evaluation of a Semantic Smart Space Deployment”, IEEE International Conference on Pervasive Computing and Communications Workshops, Lugano, Switzerland, pp. 835 – 841, 2012.

Curriculum Vitae

Sachin Bhardwaj, a native of Delhi, India, laid the foundation for his academic journey by obtaining a bachelor's degree in computer science and engineering from Dr. A.P.J. Abdul Kalam Technical University (formerly Uttar Pradesh Technical University), Uttar Pradesh, India in August 2004. Building on his educational background, Sachin pursued a master's degree in ubiquitous and network engineering at Dongseo University, South Korea, completing the program in 2007. His master's thesis delved into the topic of "A Fusion of Ubiquitous Healthcare Monitoring Using ECG and Accelerometer Sensors for Elderly Persons at Home."

In 2009, Sachin embarked on a Ph.D. program within the Department of Mathematics and Computer Science at Eindhoven University of Technology, The Netherlands, under the guidance of Prof. dr. J.J. Lukkien and dr. T. Ozcelebi. The comprehensive Ph.D. journey unfolded in two distinct phases. The initial phase, from April 2009 to March 2013, was funded by the SOFIA project. The subsequent phase, spanning from March 2019 to August 2021, received support from the SCOTT and OPTILIGHT projects. The outcomes of his extensive research efforts are presented in this dissertation.

Presently, Sachin contributes his expertise as a lecturer at Fontys University of Applied Sciences, specifically at Fontys Hogeschool ICT in Eindhoven, The Netherlands. His academic and research pursuits continue to enrich the field of computer science and technology.