IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 14, 2018, accepted October 29, 2018, date of publication November 12, 2018,
date of current version December 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2880794

Smart Space Concepts, Properties and Architectures

SACHIN BHARDWAJ[1], TANIR OZCELEBI[2], (Member, IEEE), JOHAN J. LUKKIEN[2], (Member, IEEE), AND KEON MYUNG LEE[1], (Member, IEEE)

[1]Department of Computer Science, Chungbuk National University, Cheongju 28644, South Korea
[2]Department of Computer Science and Mathematics, Eindhoven University of Technology, 5612 Eindhoven, The Netherlands

Corresponding author: Keon Myung Lee (kmlee@cbnu.ac.kr)

This work was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea, South Korea, under Grant NRF-2017M3C4A7069432.

ABSTRACT Smart spaces have been actively emerging recently, and researchers are working on developing and testing smart spaces in the real world. They facilitate smart applications that are adaptive to user preferences and contexts. In doing so they must satisfy applications' dynamically changing resource needs. These objectives are achievable by cooperation among connected devices and ubiquitous interaction. Smart space architecture designs in the literature are mostly application specific, their concepts and components defined based on the specific needs of one application. In this paper, we formally define general smart space concepts and architectural models rigorously and discuss related architectural components (both hardware and software) in detail. Based on a literature review we summarize the discriminating properties that a smart space must possess, and its basic components and services to realize these properties. We present a comparative analysis of the architectural designs proposed thus far. A comprehensive smart space architecture is proposed and its semantic interoperability is discussed in detail. In addition, we provide a case study of a smart lighting system, where the properties of smart spaces are analyzed. Finally, we provide a roadmap for future smart space development.

INDEX TERMS Adaptive behavior, smart space, smart space architecture, semantic interoperability, smart lighting.

I. INTRODUCTION

As we enter the era of the Internet of Things (IoT), where devices communicate with one another to support human tasks, the ubiquitous interaction envisioned by Weiser [1] is becoming a reality. 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 receive and display video signals. With internet 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. Similarly, touch-screen-enabled smart phones with 3G (*and now 4G*) connectivity have changed not only the way mobile users view cellular phones, but also their interactions and methods of working, even more dramatically. Smart spaces also represent an 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 experimentation. The technological push in this instance is the increased prevalence of electronic devices around and on people combined with the fact that the devices are networked and produce 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.

Augusto *et al.* [2] introduced *an* intelligent environment and provided the basic conceptual view of a smart space in the scope of intelligent environments. Although smart spaces have not been established very precisely, smart (space) applications have been developed both as project showcases [3]–[6] 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 health care (e.g., patient monitoring, home monitoring), intelligent lighting, media use, and environmental monitoring. Special-purpose systems, however, are costly and do not lead to the commoditization of the system components.

Using the analogy of a smart phone, a smart space can be regarded as a programmable platform for various concurrent applications. This platform concept using an open interface has been recognized by many authors and in numerous recent projects, and it is generally seen as essential for making progress. Among the examples, multimedia has developed the farthest in this direction through the use of Digital Living Network Alliance (DLNA) [7] technology. There are thus more stakeholders than just end users and more criteria than just functionality. Instead, we argue for smart space design considering multiple stakeholders and multiple qualities captured by different metrics.

In this paper, we define the characteristic properties of smart spaces and propose general architectural designs with physical and logical deployment alternatives for providing this set of properties. The proposed architectures were developed based on the iterative process employed in our various experiments related to smart lighting applications and smart spaces in [8]–[12]. We make a comparative analysis of the designs reported in the literature in relation to the presented properties of smart spaces and propose a comprehensive smart space architecture. We argue that semantic interoperability, a smart space property, needs particular attention and we discuss it in detail. Finally, we choose a smart lighting system discussion and provide an analysis of smart space properties.

This paper is organized as follows. Section II provides an overview of smart space concepts, as well as the fundamental properties of smart spaces. Section III outlines common generic architectures for smart space designs. Section IV presents a comparative analysis of various smart space designs in the literature with their distinctive components, as well as a comprehensive smart space architecture. Section V discusses semantic interoperability and adaptation in detail. Section VI provides a discussion on a case study. Finally, Section VII concludes the paper with future directions.

II. SMART SPACE CONCEPTS AND PROPERTIES

In this section, we first define the terminology regarding smart space concepts and then introduce the fundamental properties of a smart space.

A smart space delivers context-aware information services [13], [14], as well as physical services through actuation. It is defined 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 (node) with the software running on top. Such an object is a producer and/or consumer of digital information through its software logic, has a (dynamic) state and can communicate with other objects. Changes in the state of an object can be autonomous, which models a sensing capability. Similarly, some changes of an object state can result in changes in the physical world, which models an actuation capability.

A node or an embedded device is referred to as a smart object in [15]. In this paper for clarity of presentation we distinguish between *smart nodes* (hardware) and the software modules running on top that produce and consume information. We refer to the software modules hosted by smart nodes as information objects (*iO*s). Each *iO* of a smart space has a local state that may change over time and a set of (timed) events in which it can engage. Events are changes of state and include sensing, actuation, user interface events, and communication with other *iO*s.

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 its *behavior*.

We still have not answered the most important 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 and so on. An example is the detection of occupancy in a room by a presence sensor, triggering a state change of the detecting *iO* followed by message transmissions to a light controller *iO*, which in turn sends actuation commands to an actuator *iO* on a light source.

Note that such relations are possible to realize even with very resource-poor smart nodes. In fact, even the so-called passive nodes that require external power sources to become active (*e.g. RFID tags*) are considered to be smart nodes in this sense. 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 its scenarios, which are sequences of timed events (state changes). We refer to these scenarios as the (potential) behaviors of a smart space.

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

Definition (Application Context): The context $c(A)$ of a smart application A is the collective state of all *iO*s constituting A, including external states monitored by these *iO*s that may influence the behavior of A.

Definition (Application Behavior): The collective behavior of *iO*s that form and influence *A* define the application behavior $b(A)$.

Thus, $b(A)$ depends on $c(A)$ and is, in fact, fully defined by it. In interactions between the stakeholders and *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 that are part of $c(A)$. We refer to these as *interface objects*, which form a subset of $c(A)$. Whether an *iO* is an interface object is subjective, but we typically restrict ourselves to *iO*s that affect the function of *A*. If $b(A)$ is fully defined by these interface objects, we say that the application is *context-independent*; otherwise, we call it *context-dependent*.

A *metric m* in this setting is a mapping from smart application behaviors to a non-negative real number. When a lower value is better, the metric is called a *cost function*. 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* is called adequate with respect to a metric *m* if *A* satisfies a certain requirement on *m*, for all behaviors of *A*. We can now define what we mean by a smart space.

Definition (Smart Space): A smart space is a physical space enriched with embedded information and communication technologies running a set of smart applications.

Common properties of smart space designs in the literature are shown in Table 1. Based on our definitions of a smart application and a smart space, the first three properties listed, namely adaptation, communication interoperability and semantic interoperability, are primary properties of all smart spaces. Communication interoperability is mainly solvable by using standardized communication protocol stacks. However, adaptation and semantic interoperability pose challenges as devices that come from multiple vendors and domains differ in their default behaviors, semantics and ontologies. The last three properties, namely openness, extendibility and self-management, are secondary properties that are good to have, but a smart space design may lack these properties. In the following, we discuss smart space properties in more detail.

When an application *A* is context-dependent it may adapt $b(A)$ and trigger new or existing services as required by the scenario in question. Let the context space of *A* in its physical environment be given by C_A such that $c(A) \in C_A$. It is possible to dynamically change $b(A)$ at runtime to cope with performance issues that are stemming from changes in $c(A)$.

Definition (Adaptation): *A* is said to exhibit adaptation with respect to *m* if its behavior $b(A)$ is designed to change dynamically over time to guard *A*'s adequacy with respect to *m* as a response to changes in *A*'s context $c(A)$.

Adaptation may be active or passive. The changes in $b(A)$ are due to the direct interactions of the users with smart nodes when the adaptation is active, and the users are only notified about the variations in $c(A)$ in the case of passive adaptation.

TABLE 1. Properties of smart space solutions in the literature.

Project / Article	Adaptation	Communication Interoperability	Semantic Interoperability	Openness	Extendibility	Self-management
CISE [16]	+	+	+	-	+	-
Goh et al. [17]	+	+	+	+	+	+
PERSIST [18]	+	+	-	+	+	+
GUPSS [19]	+	+	-	+	+	+
Bram et al. [20]	+	+	+	+	+	-
Song et al. [21]	+	+	+	+	+	-
SPITFIRE [22]	+	+	+	+	-	+
INSTANS [23]	+	+	+	+	+	-
Morandi et al. [24]	+	+	+	+	+	+
Natalia et al. [25]	+	+	+	+	+	+
Eila et al. [26]	+	+	+	+	+	+
Jussi et al. [27]	+	+	+	+	+	+
Zeng et al. [28]	+	+	+	+	-	+
Sergey et al. [29]	+	+	+	+	+	-
Andrey [30-31]	+	+	+	+	+	-
Shabir [32]	+	+	+	+	+	+

* A smart space necessarily facilitates adaptation of its applications, communication interoperability and semantic interoperability. It may (or may not) show the properties of openness, extendibility and self-management. Above (+) means that the solution includes a thorough consideration of the respective property and (-) means that it does not.

For example, active 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, passive 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 be directly or indirectly notified about the changes.

*iO*s need to adapt $b(A)$adequately within a limited time interval. One of the main reasons is that long adaptation actions taken by any *iO* may prevent timely adaptations of other *iO*s of *A*, effectively leading to inadequate execution of *A*. Therefore, for adaptive applications, latency of reaction to changes in context is many times an important performance metric, i.e. a cost function to be minimized. Latency is the time interval between a stimulation of an *iO* belonging to *A* and the time at which the corresponding changes take effect in $b(A)$. In computing the total adaptation latency we need to account for the latencies added by all *iO* tasks. *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 changes of $b(A)$.

A may adapt and reconfigure according to variations in $c(A)$ as demanded by the scenarios encountered. Note that learning (as in artificial intelligence) is a specific case of adaptation.

Smart nodes consist of diverse hardware and software platforms and interoperability among these is a requirement. Interoperability between two *iO*s depends on the hardware and software platforms used by the *iO*s. A smart node is only able to exchange information if its hardware and software platforms are harmonious to integrate with other nodes.

Definition (Communication Interoperability): The ability of *iO*s in a smart space to share information over a network, i.e. by following certain (communication) protocols, message formats and syntax, is called communication interoperability.

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

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 S_i. Consider a new iO, iO_{new}, that has just joined SS_i. Let the set of protocols, message formats and syntax required by iO_{new} be given by O_{new}. For communication interoperability O_{new} needs to be a subset of S_i (called full communication interoperability) or the smart space needs to facilitate a translation between S_i and O_i (e.g. by means of a communication gateway). Semantic interoperability is achieved if the additional condition that the ontologies of iO_{new} are a subset of the ontologies of SS_i holds.

Definition (Openness): A smart space architecture is said to be open if its protocols, data formats and syntax are well-described, generally available to public.

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. The smart space architecture must enable programmers to develop applications dynamically 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 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 with ease.

A smart space is typically composed heterogeneous smart nodes and networks. Smart nodes have varying capabilities in terms of resources such as communication bandwidth, computational power, memory and energy. 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 R and a data item d to a data handling property P: "R will only do P with d". A smart space should therefore aim to guarantee and enforce privacy properties while exchanging contextual knowledge with *iO*s. Therefore, self-management of the smart space, including its networks, smart nodes is required for adequate responses in smart spaces.

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

Self-management services, such as energy management or security and privacy management, 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 of opportunities to the associated applications for enhancing their persistence by self-management as failure-free (dependable) operation of smart space applications is key to user satisfaction.

III. SMART SPACE BUILDING BLOCKS AND ARCHITECTURAL DESIGNS

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. This section presents *i*) the fundamental hardware and software building blocks that make up smart spaces and enable their properties (see Section I), and *ii)* an overview of the generic architectural designs employed to realize these properties.

A. CLASSIFICATION OF SMART NODES

The available device classes in networks of resource-constrained devices were defined in [8] and [9] based on an investigation of the commercially available chips and embedded system designs. In this taxonomy, class 0 ($C0$) devices 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 devices are not able to run the Internet protocol stack. We call some devices belonging to this class *passive devices* 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. $C0$ passive devices typically do not facilitate any management or security services other than pairing with other devices over

a trusted proxy. $C0$ devices that are not passive can handle keep-alive messages and provide basic device state information.

$C1$ devices have roughly 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 of the full protocol stack of the Internet. Nevertheless, there are low-resource (lightweight) protocol stacks specifically designed for this device class. For example, these devices can use CoAP (Constrained Application Protocol) [33] over UDP (User Datagram Protocol) and employ 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) as adaptation layers 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 devices perform poorly due to resource optimization techniques such as radio duty cycling.

With around 50 KiB of memory and 250 KiB of storage space, $C2$ devices can run conventional network protocol stacks. However, in practice, $C2$ devices 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 three broad categories of smart nodes with respect to the available resources and communication capabilities: *high-capacity smart node* (HSN), *low-capacity smart node* (LSN), and *passive smart node* (PSN). HSNs are members of $C2$ (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 high-level and complex reasoning. PSNs are resource-poor passive devices (a subclass of $C0$), such as RFID tags and battery-less switches. LSNs are the remaining members of $C0$ plus the members of $C1$.

B. LOGICAL STRUCTURE OF iOs

A smart space allows seamless information sharing among *iO*s, i.e. information such as application contexts and management data. Smart space applications are composed of sets of data and actuation services provided by their *iO*s. The relations between various logical components and *iO*s in a smart space architecture are depicted in Fig. 1. The types of *iO*s shown in Fig. 1 are described in Table 2.

A smart application has application-specific contexts, logic, and ontology. The application contexts are the contexts captured from the environment (*such as user ID, location, activity, and resource attribute value*) and are supplied to the application logic. The application logic can execute application-specific scenarios. The application ontology defines the set of concepts that are relevant, allowing to provide a flexible and up-to-date description of application contexts. An application has an application specific ontology,

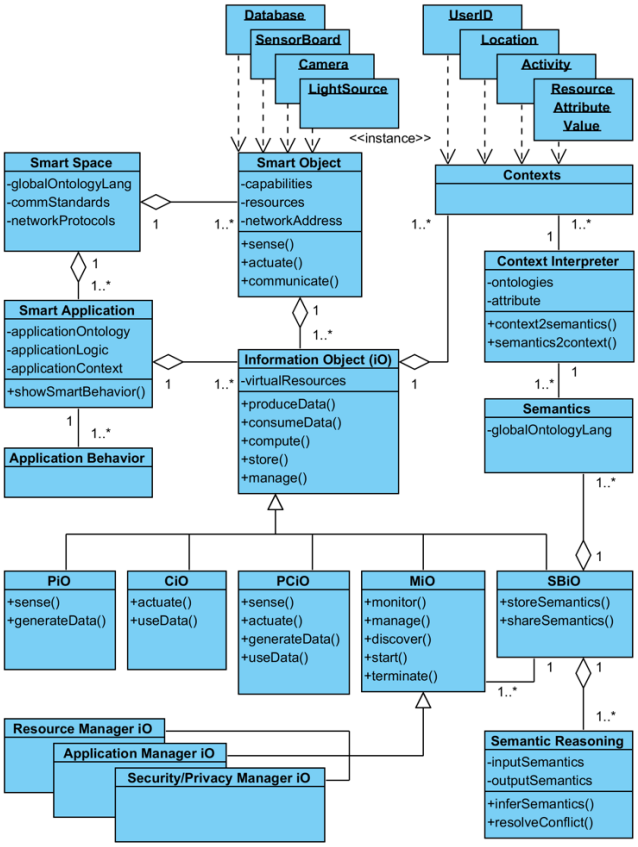


FIGURE 1. A diagram depicting the logical relations of *iO*s in a smart space.

TABLE 2. Types of *iO*s shown in Fig. 1.

Item	Description
PiO	**Producer *iO*:** produces information for other *iO*s
CiO	**Consumer *iO*:** consumes information of other *iO*s
PCiO	**Producer–Consumer *iO*:** acts as both a *PiO* and a *CiO*
SBiO	**Semantic Broker *iO*:** a module that stores and shares semantics (data with certain meaning)
MiO	**Manager *iO*:** manages resources, applications, security and privacy.
RMiO	**Resource manager *iO*:** an *MiO* instance that manages resources that *iO*s can access
AMiO	**Application manager *iO*:** an *MiO* instance that manages applications
SPMiO	**Security/privacy manager *iO*:** an *MiO* instance that provides security and privacy services

logic and a notion of its context. It realizes its application behaviors, i.e. *showSmartBehavior*(). A smart node has its own capabilities, resources, and network address to associate with other smart nodes, and to provide *sense*(), *actuate*(), and *communicate*() services.

A *context interpreter* gathers raw data from one or more *iO*s of an application, summarizes them as application contexts and converts the results into the established *semantics* [34] of the smart space. For example, a context

interpreter participating in an application 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) [35]. Contexts are expressed in a standard way 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).

The *semantic reasoning* associated with *SBiO*s refers to inferring logical rules from input semantics in the form of validation (*e.g. to avoid conflicts*) and deduction (*e.g. to deduce complex semantics from simple ones*). When the inference of logical rules (*i.e. inferSemantics*()) is expressed by validations, one instance of validations is employed to check whether a set of semantics S_A is a subset of a second set of semantics S_B, denoted as $S_A \subseteq S_B$. Similarly, the conflicts in the querying semantics need to be resolved (i.e. *resolveConflict*()). Output semantics (*i.e. outputs of semantic reasoning*) that result from validation and deduction may trigger new *iO* behaviors, and thus, new application behaviors. This gives a perception of smartness. In the semantic web [36], new relationships and semantics can be discovered based on the existing semantics. These semantics and inferred logical rules can be shown in a graphical way as in an ontology language. The Web Ontology Language (OWL) [37] is the most commonly used ontology language. There are many semantic reasoners available, such as Pallet [38], Flora-2 [39], Jena [40], and the ELK reasoner [41], to name a few. Ontology graphs and their semantic representations are edited by semantics editors such as Protégé [42], OWLGrEd [43], and VocBench [44].

Smart spaces employ *management* to deal with factors that may cause applications to fail, such as attacks, lack of resources, hardware and software failures, and network topology changes. As trusted systems [45], they must ensure data privacy and enforce security measures by employing *SPMiO*s. The tasks of resource and application management are performed by *RMiO*s and *AMiO*s. The former monitors the available resources and allocates them to *iO*s as necessary. The latter not only orchestrates the *iO*s that participate in smart applications, but also re-orchestrates them 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 smart application failure. "Failure" also includes poor application behavior or, equivalently, failure to adhere to specifications. *Producer iOs* (*PiOs*), e.g., on a smart sensor, produce knowledge in a smart space, whereas *consumer iOs* (*CiOs*), e.g., on a smart light source, utilize such knowledge.

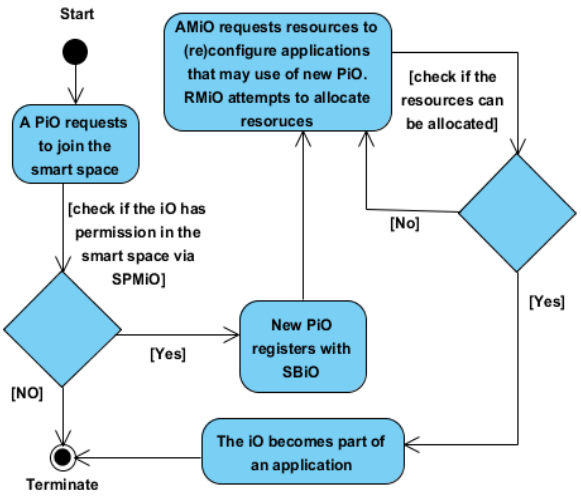


FIGURE 2. Activity diagram illustrating the process of a *PiO* joining a smart space and participating in an application.

Producer–consumer iOs (*PCiOs*), e.g., on an input–output device such as a smart touch display, can both produce and utilize knowledge.

C. PROCESSES AND DEPENDENCIES AMONG iOs

The (simplified) process of an *iO* joining a smart space and participating in a smart application is shown in Fig. 2. A *PiO* sends a request to join a smart space application and the request is verified by an *SPMiO* to obtain permission. The new *PiO* is then registered and can insert data into an *SBiO*. An *AMiO* requests resources to (re)configure the application that may utilize the new *PiO*. If the resources can be allocated, then the new *PiO* becomes part of the smart space application. The process through which a *PiO* inserts data into an *SBiO* is shown in Fig. 3. For example, consider a *PiO* as part of an application that gathers contextual data from the environment through sensing. The context interpreter converts this contextual data into the smart space semantics. These semantics are further stored at the *SBiO* for information sharing. The semantics may be dynamic; thus, the dependencies among *iO*s must be able to modify their processing of ever-changing semantics. The dependencies among the *iO*s in a smart space are depicted in Fig. 4, which explains the various issues to be resolved, such as context interpreter, representation as semantics, semantic reasoning, application performance, smart space management, application management, access control, security and privacy, and their management.

The changes in semantics must follow the conditions defined for semantic interactions.

D. PHYSICAL DEPLOYMENT

The physical part 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 between three types of smart spaces with respect to the physical deployment

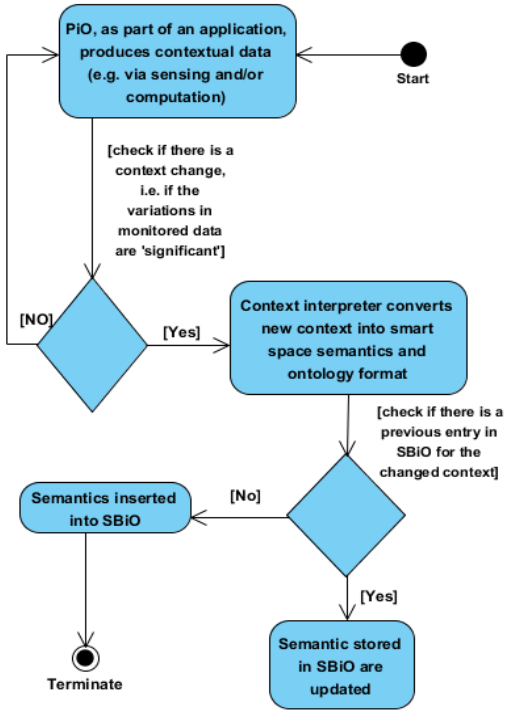


FIGURE 3. Activity diagram illustrating a *PiO* inserting data into the *SBiO*.

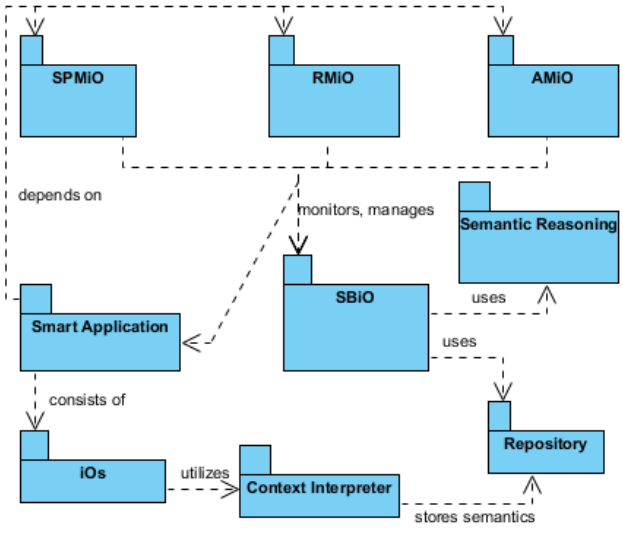


FIGURE 4. Package diagram showing the dependencies among *iO*s in a smart space.

of *iO*s on smart nodes, i.e., centralized, decentralized, and distributed smart spaces.

Table 3 gives a summary of the smart node types used in smart space architectural designs. In a centralized smart space, most of the *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, including management, is performed at a *central iO* (*CTiO*), which is typically placed on the same device as an *SBiO*, as shown in Fig. 5. A *CTiO* must be able

TABLE 3. Types of smart nodes.

Item	Description
LSN	Low-capacity smart node: a low-capacity device in terms of resources and computation, such as a sensor
HSN	High-capacity smart node: a high-capacity device in terms of resources and computation, such as a smart phone
PSN	Passive smart node: a resource-poor passive device, such as an RFID tag
CSN	Central smart node: a smart node running all *iO*s of a single smart space application
PXSN	Proxy smart node: a device 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 via an *SBiO*
GSN	Gateway smart node: a smart node that translates contexts into semantics and vice versa
MSN	Manager smart node: a device that does management tasks via one or more MiOs (i.e. *AMiO*, *RMiO*, *SPMiO*)

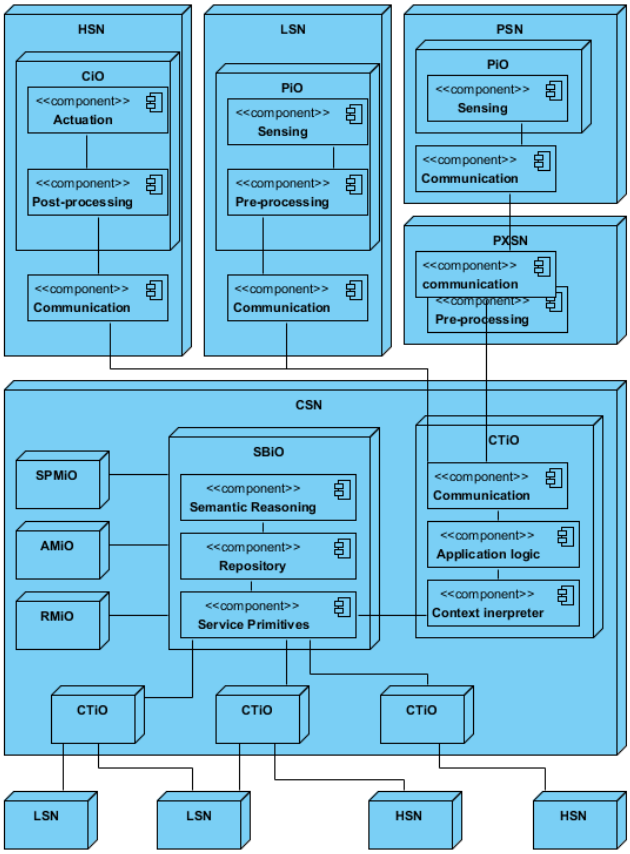


FIGURE 5. Physical deployment of a centralized smart space. Each *CTiO* corresponds to an application in the smart space. A smart node is associated with one or more *CTiO*s. Each *CTiO* running an application is associated with one or more smart nodes.

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*. Adaptive and adequate application behavior is then imposed by the *CTiO*. PSNs do no pre-processing or post-processing. Instead, they are connected to

the central smart node (CSN) over a trusted proxy (PXSN), which also does pre- and post-processing on behalf of the PSN. In practice, the tasks of the proxy can be moved to the CSN. Therefore, the CSN and the 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 for openness and extendibility: deploying an *iO* at the CSN requires either reservation of the needed resources and admission control or compatibility of the *iO* with a dynamic resource management regime. In the latter application adequacy depends on the availability of resources and may be jeopardized when more *iO*s are deployed.

A decentralized smart space is one in which the networked smart nodes do most of the computations necessary to realize user applications in a distributed way. For adequacy, the *iO*s involved must be able not only to communicate, i.e., have communication interoperability, but also to operate with the same meanings (semantics) of data, i.e., have semantic interoperability. To realize this, it is important that *iO*s utilize shared communication standards and have access to sufficient computational resources. Clearly, the resources and communication capabilities depend on the hardware design and the loads on the various smart nodes. A (sub-) network of LSNs in a smart space mostly runs resource efficient communication protocols such as ZigBee [46] and Bluetooth Low Energy [47]. LSNs typically require a gateway smart node (GSN) between themselves and an SBSN for communication interoperability and translation of semantics. Smart spaces typically support wireless communication and must be able to deal with multiple communication standards. A GSN implements the communication standards in all networks that it bridges together. It is also responsible for context interpretation for LSNs [48] into semantics using a gateway *iO* (*GWiO*). A *GWiO* is a specific type of *PCiO* on a GSN that translates contexts into semantics and vice versa between an LSN and an SBSN. Fig. 6 shows the physical deployment of a decentralized heterogeneous smart space with HSNs and LSNs. *SBiO*s play an important role in decision making with the help of MSNs. Each *SBiO* contains a semantic reasoner to produce output semantics. The 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, making it easier to achieve openness, and extendibility.

Services supported by the cloud are becoming increasingly popular and can be utilized in decentralized smart spaces directly or indirectly. In direct utilization [49], the cloud services must have the same ontology format and semantics as the smart space itself and each cloud service acts as an *iO*. Furthermore, the smart devices that communicate directly with the cloud services must be individually reachable (e.g., IP-to-the-leaves using 6LoWPAN [50]). In indirect utilization [51], a gateway between the cloud and smart space runs an *iO* that provides the semantic interface to cloud services, as shown in Fig. 7.

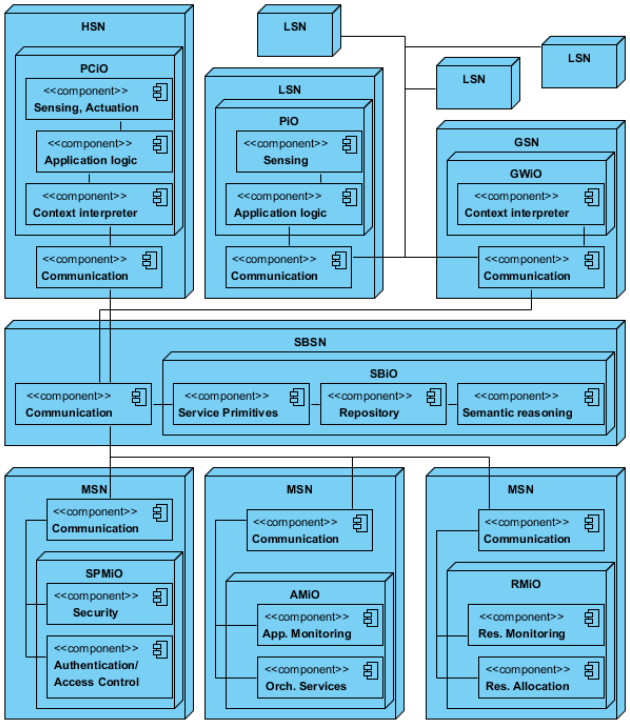


FIGURE 6. Physical deployment of a heterogeneous, decentralized smart space with HSNs and LSNs. Communication interoperability and semantic interoperability with LSNs are achieved by means of GSN.

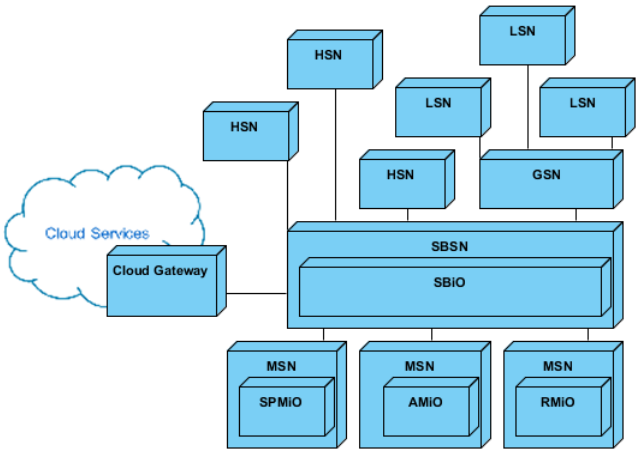


FIGURE 7. Cloud services for smart spaces.

A distributed smart space consists of smart devices without a certain hierarchical structure. Each device contains the software components necessary to realize self-management, semantic reasoning, and distributed application logic as shown in Fig. 8. This architectural design is strongly dependent on the application requirements and is loosely coupled with the components. The components are not dependent for any decision to be made within specific boundaries.

Smart space management must involve a concept of membership, perhaps with different authorization levels. Based on this structure, privacy and security models and policies are built. Service and device discovery are integral parts of

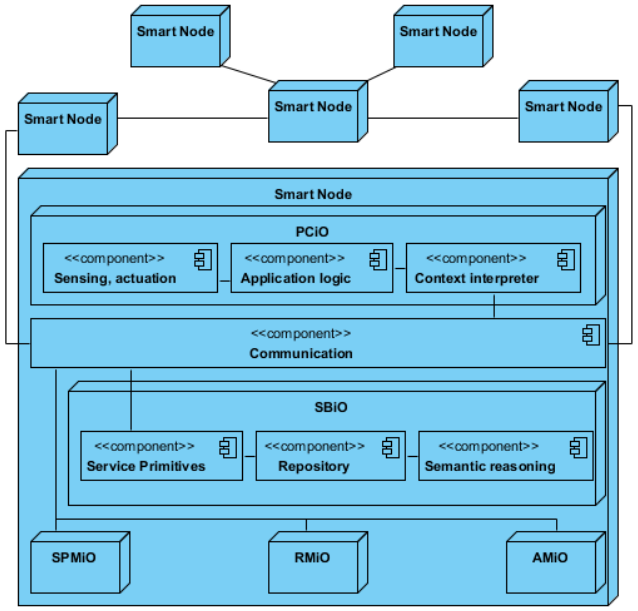


FIGURE 8. Physical deployment of a distributed smart space.

this management system. Numerous software stacks for service discovery exist [8], some of which are communication-standard specific and have various weights. A similar remark can be made about interoperability standards [52]. These must be seen as representing a more abstract concept of device and service discovery defined at the smart-space architectural designs. For example, when a smart node leaves from the application then their services are supposed to handover at another smart node in the application. A suitable system architecture design must be chosen based on the target applications, their goals, and the corresponding performance metrics.

IV. COMPARATIVE ANALYSIS OF SMART SPACES

Many architectural designs for smart spaces have been proposed by many researchers. In this section we compare a selection among them, specifically those that that provide nearly complete solutions with respect to the smart space concepts that we have enlisted. This comparison is given in Table 4. An immediate observation is that most of the smart space architectures in the literature are decentralized by design, while fully centralized or fully distributed architectures are very rare.

An example centralized smart space architecture is called PERSIST [18], 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, which allows them to adapt to new environments automatically in satisfying their users' needs. Interoperability is established based on context sharing using semantic web technologies, where a context management system implemented in an *AMiO* stores and retrieves context information in a distributed manner. PERSIST, which consists of several PSSs, is the only architecture in Table 4 with a distributed design. The facilities it provides to integrate multiple applications, however, are limited.

A centralized smart space design is given in [17]. 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 [17] 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.

Most of the decentralized architectural designs from [19] to [32] in Table 4 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 [22] and CISE [16]. There are two issues that should be noted in this decentralized architecture solution of SPITFIRE. Firstly, it involves a semi-automatic process of creating semantic sensor descriptions for LSNs. Note that our proposed architectural model for decentralized smart spaces allows the semantics to be fetched directly and automatically through the *GWiO*, without requiring manual or semi-automatic creation of semantics. Secondly, SPITFIRE directly connects sensors to the semantics repository, 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 *Context-Based Infrastructure for Smart Environments* (CISE), which focuses on the requirements for dealing with smart space contexts. It 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 and LSNs such as sensors can be replaced and added without affecting the smart application. 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 semantic interoperability for generic infrastructures suitable for any application in smart spaces. The information shared only includes the basic descriptions of devices. A common ontology language based on a semantic web is required for large-scale applications.

TABLE 4. Comparison of architectural designs of smart spaces.

Article	Year	Architectural Style	*PCiO*	*GWiO*	*SBiO*	*AMiO*	*RMiO*	*SPiO*
CISE [16]	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. [17]	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 [18]	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 [19]	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.	-----	-----
Bram et al. [20]	2009	Decentralized	Yes	No	Semantic representations 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. [21]	2010	Decentralized	Yes	No	Semantic representations 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.	-----
SPITFIRE [22]	2011	Decentralized	Yes	Yes	Semantic representations are collected and stored at an 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 [23]	2012	Decentralized	Yes	No	Semantic representations 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. [24]	2012	Decentralized	Yes	No	Semantic representations 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.	-----	-----

TABLE 4. Comparison of architectural designs of smart spaces.

Natalia et al. [25]	2013	Decentralized	Yes	No	Semantic representations 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.	-----	-----
Eila et al. [26]	2014	Decentralized	Yes	No	Semantic representations 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. [27]	2014	Decentralized	Yes	No	IoT based products are used. Semantic representations 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. [28]	2015	Decentralized	Yes	No	A platform library server is introduced for data flow in a smart space.	-----	-----	----
Sergey et al. [29]	2017	Decentralized	Yes	No	Semantic representations 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 [30-31]	2017	Decentralized	Yes	No	Semantic representations 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.	-----	-----
Shabir [32]	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.	-----

In [53], a semantic approach for providing intelligent support into a large IoT-based smart space is introduced where semantically driven information shared among IoT devices and services are provided to the users. Several examples of smart space applications are also discussed such as smart mobile assistant for history-oriented tourists, personalized mobile health system and smart room system for conferences and meetings. An integrated approach of combining smart spaces and IoT is presented in [26]. Although this approach is effective and facilitates the division of the worldwide IoT into manageable smart spaces, abstractions for LSNs are not considered, resulting in poor performance. A common platform for the abstraction of heterogeneous devices, technologies, and protocols is presented. This platform provides

semantic-level interoperability, enabling the creation of applications for pervasive computing and the IoT together.

The main goal of semantic-level interoperability is the integration of devices in the semantic web by using web technologies such as the RDF and OWL. These technologies were originally designed for web resources but were later used also in open-source-based projects such as OpenIoT [54] and *Smart Objects for Intelligent Applications* (SOFIA) [55]. OpenIoT provides a framework for connecting users and IoT devices to establish a global ecosystem for the IoT. The objectives of this framework are twofold. Firstly, it provides facilities for the development of open software architectures and gathering of innovative IoT services. Secondly, these services can be quickly searched by users. Numerous users can benefit from the framework by rapidly searching for results using web technologies for IoT-related services, applications, and products. Meanwhile, SOFIA provides a Smart-M3 platform for interoperability across domains, devices, and vendors. It allows integration of information domains that take part in smart applications using web technologies. The Smart-M3 platform is implemented and its usage is discussed in [56]. The research is focused on service formalism based on Smart-M3 for developing smart space applications, where smart spaces are concluded as an emerging paradigm for future IoT based smart software. Moreover, the researchers and developers are encouraged in [57] to develop smart spaces using the Smart-M3 platform. The research challenges are discussed in detail for the development of smart spaces. A good set of questions is discussed related to the development properties of any smart space. These questionnaires can help to classify the smart space concepts and implementations.

In [20], [23]–[26], and [29]–[31], this interoperability approach employing Smart-M3 was utilized to share and access local semantic information easily. In Smart-M3, the performance metric of the queries and subscriptions at an *SBiO* depends on the reasoning component and network delay. We performed a Smart-M3 experiment involving a smart lighting application in [58] to measure the delay performance with semantic-level interoperability.

Zeng *et al.* [59] presented a coarse-grained computation model called *HyperspaceFlow* where a smart space is modeled using physical flow, data flow, and human flow components. The physical flow specifies the relations between cyberspace and the physical space; the data flow involves the computations and communication related to 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 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 approach allows modeling of a smart space only for a single user. Further, it is enhanced for Cyber-Physical-Social Systems (CPSS) [60], [61], where a system-level design optimization approach is introduced for security, energy consumption and user satisfaction. The approach is examined by the case study of a smart office, which satisfied the design optimization requirements. Ovaska and Kuusijärvi [26] introduced a piecemeal service engineering approach to smart space design and tested it for intelligent applications. The piecemeal service engineering approach enables maximum use of the existing design and technical resources in the development of new smart space applications.

Some researchers have tried to explain smart space architectures using hierarchal models as in MavHome [62]. The smart space architecture in MavHome is realized by providing a complete solution to a smart home and collaborates to meet the goals of the overall home. It contains four layers: a decision layer, an information layer, a communication layer and a physical layer. The decision layer selects the actions for an object 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 between devices. The physical layer corresponds to the devices within the smart home. These layers provide the features necessary for self-managing smart home automation. MavHome was implemented using a Common Object Request Broker Architecture [63] interface between software components and powerline technologies such as X10 [64] and Lon Works [65] as electric devices. Although this architecture enables the integration of several technologies in a smart home, it fails to provide a solution for LSNs and efficient interoperability. It addresses only context-based interoperability and avoids any management of resources and services.

Similarly, in ISHEWS [66], 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, Wi-Fi, and 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. Transport Control Protocol (TCP), User Datagram Protocol (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)

TABLE 5. Smart nodes considered in various smart space designs.

Article	LSN	HSN	SBSN	GSN	MSN
CISE [16]	-	+	+	-	-
Goh et al. [17]	-	+	-	-	-
PERSIST [18]	+	+	-	-	-
GUPSS [19]	+	+	+	+	-
Bram et al. [20]	-	+	+	-	-
Song et al. [21]	+	+	+	-	+
SPITFIRE [22]	+	+	+	-	+
INSTANS [23]	-	+	+	-	-
Morandi et al. [24]	-	+	+	-	-
Natalia et al. [25]	-	+	+	-	+
Eila et al. [26]	-	+	+	-	+
Jussi et al. [27]	+	+	+	+	-
Zeng et al. [28]	+	+	+	-	+
Sergey et al. [29]	+	+	+	+	-
Andrey [30-31]	+	+	+	-	-
Shabir [32]	+	+	+	-	+

*Considered (+) and Not Considered (-)

encoding, Representational State Transfer (REST) [67] 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.

The smart nodes considered in various smart space designs are summarized in Table 5, 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 an area for future development (initial approaches available in [25] and [26]).

It can also be seen from the comparison of various smart space architectures that resource and security/privacy management information objects are rarely ever considered. This is a huge problem for mainstream adoption of smart spaces, considering ethical aspects and gradually introduced laws enforcing privacy of data in many countries, especially in developed countries. Dependability of smart space applications is also a main concern for user experience, i.e. changing an electric circuit (manual) light switch that is almost 100 percent reliable with a smart switch that is 99 percent of the time reliable 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.

Based on the comparative study, we now propose a comprehensive smart space architectural design as shown

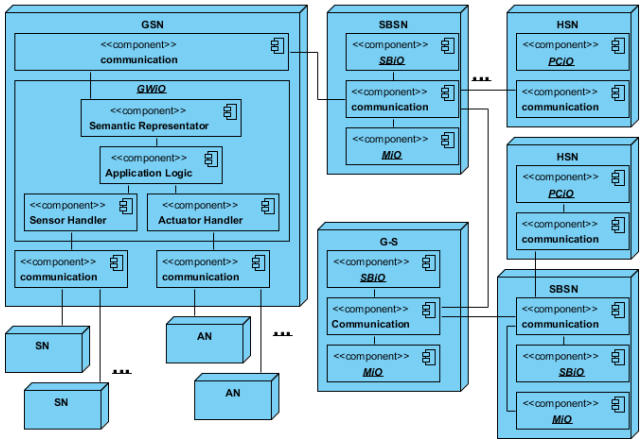


FIGURE 9. Comprehensive smart space architecture.

in Fig. 9. The smart space architectures given in Section III are particular instantiations of this design. In the proposed comprehensive architecture GSNs enable the inclusion of networks of LSNs. We introduce a two-level SBSN hierarchy for load balancing and performance improvement where a special type of SBSN (G-S) is able to share semantics between other SBSNs and delegate its brokering. We consider a single *MiO* component where developers can implement the tasks of *AMiOs*, *SPMiOs* and *RMiOs* as needed based on the application's requirements.

The *GWiO* component at a GSN has two handlers for LSNs, i.e. a sensor handler for sensor nodes (SNs) and an actuator handler for actuator nodes (ANs). These handlers are controlling the input and output information from sensors and actuators respectively. The contexts received from SNs are transferred to the application logic and ANs receive the actuation commands from the application logic. The application logic infers the actuation commands based on the SNs contexts and semantics received from *SBiO*, where the semantics received from *SBiO* is translated into contexts before being used by the application logic.

Each SBSN contains an *SBiO* and an *MiO*. Similarly, as discussed in the previous sections an *SBiO* does semantic reasoning on the input semantics to produce output semantics. The *MiO* monitors and manages the resources and services of the respective SBSN. The G-S helps to collaborate with other SBSNs in a smart space with the help of an *MiO*. The *MiO* makes subscriptions and updates for semantics at the G-S. Therefore, by means of the G-S any HSN in an SBSN can communicate with other HSNs at other SBSNs.

Our survey of the literature resulted in a list of properties that are specified in fair detail in *all* smart space implementations. We call these three properties, namely adaptation, communication interoperability and semantic interoperability, the *primary properties* of a smart space. *Secondary smart space properties*, namely openness, extendibility and self-management can be implemented as needed based on application requirements.

The primary properties are essential for the proposed comprehensive smart space architecture because of the following reasons:

i.) The behavior of each *iO* needs to adapt according to the state changes of smart applications. This requirement comes from our definition of a smart space.

ii.) Communication is needed to exchange information among smart nodes that produce and consume it.

iii.) Semantic interoperability is needed to establish a shared meaning of the information that is exchanged.

Communication interoperability can easily be established by using standard network protocols or by using a gateway to connect with LSNs. Example standards are powerline technologies such as X10 and LonWorks, wireless technologies such as IEEE 802.15.4 for wireless sensor networks, and CAT5 for audio, video or data communication. Furthermore, middleware such as Jini [68], HAVi [69] and UPnP [70] may be employed to connect to smart nodes. Therefore, we focus on adaptation and semantic interoperability properties for the proposed comprehensive architecture in the next section and give a detailed discussion.

V. SEMANTIC INTEROPERABILITY AND ADAPTATION

In this section we first discuss how semantic interoperability is realized by the semantic interactions among *iO*s and smart objects. Then, we discuss how adaptation of smart applications is achieved by the adaptive behaviors of *iO*s that constitute these smart space applications.

A. SEMANTIC INTERACTIONS AMONG iOs

The format and syntax of transactions among *iO*s are dependent on smart node hardware and software specifications, which are heterogeneous by nature. The main challenge in the presence of such heterogeneity is achieving semantic interoperability such that multiple *iO*s can cooperatively realize execution of smart space applications.

A source *iO* and a destination *iO* can interact meaningfully through semantics of their smart space when there is an onto mapping between contexts and semantics. Semantics are typically represented using ontology graphs that facilitate complex queries. A smart space architecture must enable mechanisms capable of representing, modifying, and updating the semantics to produce meaningful and valid results. Our earlier experiments for semantic interoperability in [9]–[12], [48], and [56] were in the context of the SOFIA project, where we used Smart Space Access Protocol (SSAP) to establish semantic interoperability. In this work, we take SSAP as the baseline for semantic interoperability for transactions in the proposed architecture. SSAP transactions are used in information exchanges between an *SBiO* and *iO*s. The SSAP transactions are: JOIN, INSERT, REMOVE, UPDATE, SUBSCRIBE, UNSUBSCRIBE and QUERY. The JOIN and LEAVE transactions are used for joining and leaving a smart space respectively, where *SBiO* associates the connection with the identity and authority verified by the credentials through an *MiO* at the SBSN. An *iO* starts a session with an *SBiO* with a JOIN transaction and ends the session by using a LEAVE transaction. By using an INSERT transaction, an *iO* inserts semantics σ (a tuple) at an *SBiO*, causing the *SBiO* to generate a blank node in the RDF graph with a specified URI. Furthermore, an *iO* can remove and update the RDF graph at the *SBiO* by REMOVE and UPDATE transactions, respectively. An *iO* can make a query for σ at the *SBiO* by using a QUERY transaction and get results from the *SBiO* in reply. Any *iO* in a smart space may subscribe for σ, i.e. a persistent query for σ stored at the *SBiO* by using the SUBSCRIPTION transaction. In that case, the *iO* is notified of changes in σ. Similarly, an UNSUBSCRIPTION transaction terminates the persistent query from the *SBiO*. All transactions must be executed in an atomic fashion, i.e. the information content of the smart space is not changed by any other transaction during the execution of the transaction. This is especially important for the queries, which may require several accesses to the underlying data structures within the *SBiO*. Only two entities, *iO* and *SBiO*, are involved in any single transaction. Any *iO* in a smart space can join and enjoy the facilities provided by the *SBiO* through the given SSAP transactions, depending on their requirements. JOIN and LEAVE are transactions (eventually) employed by all *iO*s.

SSAP semantic interactions among smart nodes consist of *i*) *PiO*, *CiO*, and *PCiO* semantic interactions at the *SBiO*, *ii*) *GWiO* semantic interactions at the *SBiO*, *iii*) semantic interactions between the *MiO* and the *SBiO* at the SBSN and *iv*) the semantic interaction steps between two *SBiO*s.

1) *PiO*, *CiO*, AND *PCiO* SEMANTIC INTERACTIONS

The semantic interactions for *PiO*, *CiO* and *PCiO* at the *SBiO* are described in Table 6 and Fig. 10. The JOIN and LEAVE transactions for semantic interactions are performed by all *PiO*s, *CiO*s and *PCiO*s. The INSERT, UPDATE and REMOVE transactions are performed by *PiO*s and *PCiO*s, while QUERY SUBSCRIPTION and UNSUBSCRIPTION transactions are performed by *CiO*s and *PCiO*s.

*iO*s communicate input semantics σ_i to the *SBiO* where semantic reasoning is performed, resulting in output semantics σ_j, which is communicated to the *iO*s in reply. When a new σ is inserted and updated at the *SBiO* then an ontology graph is extended with the new entry. Subsequently, if any *iO* is subscribed to insertions and updates of σ then it will be notified with the extended entry in the ontology graph.

A smart space application is equipped with heterogeneous smart nodes. These nodes can exchange information by using the *iO*'s semantic interactions. Any smart node in an application can share semantics with other nodes by using these transactions in an appropriate method and the requirements in the application. To achieve the objectives in an application, these semantic interoperability interactions are the primary and basic things to be established first.

TABLE 6. ***PiO*, *CiO*, and *PCiO* semantic interactions at the *SBiO*.**

No.	Semantic interaction steps
1	For semantic input tuple σ_i, and semantic output tuple σ_j:
2	**JOIN Request (R^J) and LEAVE Request (R^L)**
3	R^J begin
4	R^J sent by *PiO* / *CiO* / *PCiO* to *SBiO*: $R^J(\sigma_i)$
5	R^J received by *SBiO*: $R^J(\sigma_i)$
6	R^J processed at *SBiO*: $R^J(\sigma_i) \rightarrow R^J(\sigma_j)$
7	R^J confirmation received at *PiO* / *CiO* / *PCiO* : $R^J(\sigma_j)$
8	R^J end
9	R^L begin
10	R^L send by *PiO*, *CiO*, and *PCiO* to *SBiO*: $R^L(\sigma_i)$
11	R^L received by *SBiO*: $R^L(\sigma_i)$
12	R^L processed by *SBiO*: $R^L(\sigma_i) \rightarrow R^L(\sigma_j)$
13	R^L confirmation received at *PiO*, *CiO*, and *PCiO* : $R^L(\sigma_j)$
14	R^L end
15	**INSERT tuple σ_I**
16	*PiO* / *PCiO* sends tuple σ_I to *SBiO*: $INSERT(\sigma_I)$
17	*SBiO* receives and saves σ_I
18	**SUBSCRIBE Request (R^S) and UNSUBSCRIBE Request (R^U) and QUERY Request (R^Q)**
19	R^S begin
20	R^S sent by *CiO* / *PCiO* to *SBiO*: $R^S(\sigma_i)$
21	R^S received by *SBiO*: $R^S(\sigma_i)$
22	R^S processed by *SBiO*: $R^S(\sigma_i) \rightarrow R^S(\sigma_j)$
23	R^S confirmation received at *CiO* / *PCiO* : $R^S(\sigma_j)$
24	R^S end
25	R^U begin
26	R^U sent by *CiO* / *PCiO* to *SBiO*: $R^U(\sigma_i)$
27	R^U receive by *SBiO*: $R^U(\sigma_i)$
28	R^U process by *SBiO*: $R^U(\sigma_i) \rightarrow R^U(\sigma_j)$
29	R^U confirmation received at *CiO* / *PCiO*: $R^U(\sigma_j)$
30	R^U end
31	R^Q begin
32	R^Q sent by *CiO* / *PCiO* to *SBiO*: $R^Q(\sigma_i)$
33	R^Q received by *SBiO*: $R^Q(\sigma_i)$
34	R^Q processed by *SBiO*: $R^Q(\sigma_i) \rightarrow R^Q(\sigma_j)$
35	R^Q result received at *CiO* / *PCiO*: $R^Q(\sigma_j)$
36	R^Q end
37	**UPDATE Request (R^D) and REMOVE Request (R^R)**
38	R^D begin
39	R^D sent by *PiO* / *PCiO* to *SBiO*: $R^D(\sigma_i)$
40	R^D received by *SBiO*: $R^D(\sigma_i)$
41	R^D processed by *SBiO*: $R^D(\sigma_i) \rightarrow R^D(\sigma_j)$
42	R^D confirmation received at *PiO* / *PCiO* : $R^D(\sigma_j)$
43	R^D end /* *CiO*'s subscribed at *SBiO* receive $R^D(\sigma_j)$ */
44	R^R begin
45	R^R send by *PiO* and *PCiO* to *SBiO*: $R^R(\sigma_i)$
46	R^R receive by *SBiO*: $R^R(\sigma_i)$
47	R^R process by *SBiO*: $R^R(\sigma_i) \rightarrow R^R(\sigma_j)$
48	R^R confirmation received at *PiO* and *PCiO* : $R^R(\sigma_j)$
49	R^R end /* *CiO*'s subscribed at *SBiO* receive $R^R(\sigma_j)$ */

2) *GWiO* SEMANTIC INTERACTIONS AT THE *SBiO*

An SN gets contextual information from the environment and produces data for further analysis, typically for decision making (e.g. related to user notification or actuation). In order for SNs to integrate with smart spaces, they need to produce meaningful information to exchange with other nodes in a smart space application. A big challenge is that SNs are typically not capable of doing complex computations due to power and memory constraints and there is a long list of potential applications such as those in smart homes [71], smart healthcare [72], transport and logistics management [73], inventory and product management [74], firefighting systems [75], social networks [76], smart cities [77], and smart lighting systems [78], just to name

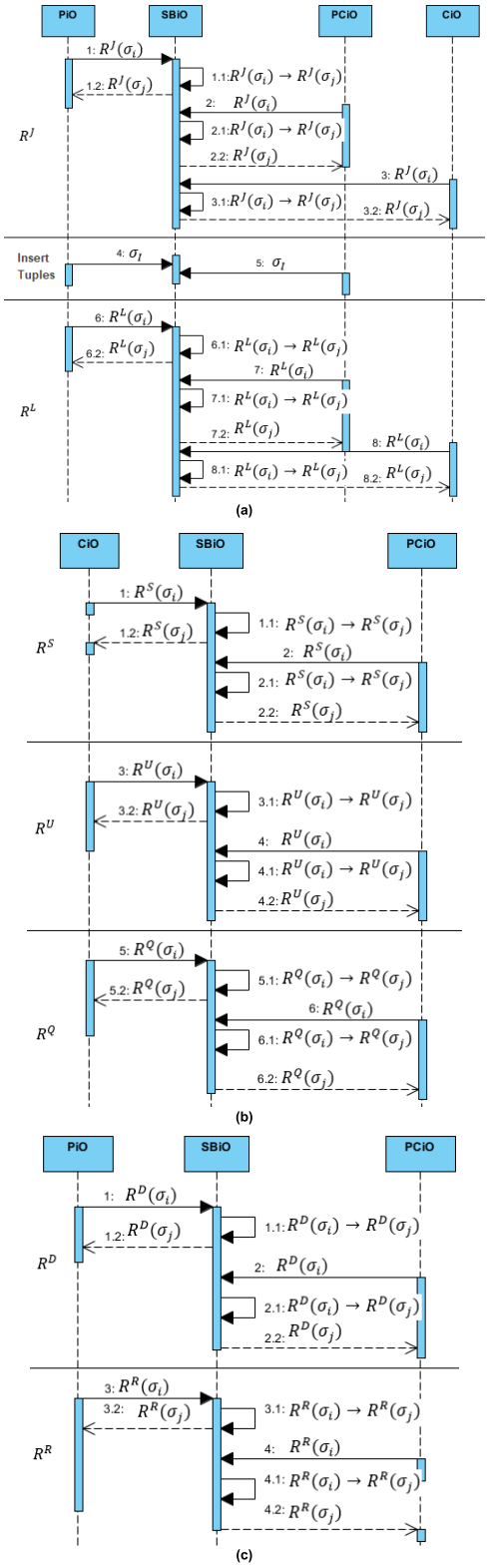


FIGURE 10. **Sequence diagram of semantic interactions by *iO*s at SBiO for the following transactions: (a) R^J, R^L and insert tuples, (b) R^S, R^U and R^Q, and (c) R^D and R^R.**

a few. Thus, information representation in semantics appears as a bottleneck to SNs.

In some infrastructures [79]–[84], SNs are capable of sharing information with semantic web technologies, e.g. by using the Sensor Web Enablement (SWE) specification. SWE specifies sensor data semantics in Sensor Model Language (SensorML), which uses the 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 deal with sensor data in heterogeneous sensor network applications. It tries to improve the practicability of producing semantics for smart nodes and to establish interoperability with the semantic web. There are similar approaches for SNs to comply with the semantic web such as Sensorpedia [85], SensorWare [86], and SensorMap [87]. However, while these approaches are good at integrating SNs with the semantic web, their results are unfavorable when semantics are represented by sensor nodes themselves, as this requires double the power consumption of SNs. In addition, memory constraints mean that SNs are unable to process multiple parallel transactions. This results in a short battery lifespan of SNs in the network. Such a tradeoff between the lifespans and computations of sensor nodes provides a scope for integrating an external unit—a gateway node—which can provide the capability to execute the complex computations and semantic representations necessary to share information with other smart nodes. The GSN provides a good solution to resolve the computation bottleneck of SNs. It can compute the semantic representations for SNs and ANs and process them for further improved results. Therefore, the GSN is a powerful smart node that can perform computations and semantic representations on behalf of resource poor SNs and ANs.

Consider a service for actuating ANs based on the commands given by the GSN, for which the possible semantic interactions are explained in Table 7 and the message sequence diagram is shown in Fig. 11. Firstly, the services related to initialization are installed by the GSN to SNs and ANs. Then the GSN is subscribed for the information of SNs and ANs (i.e. I^S and I^A respectively). I^S and I^A are processed by the *GWiO* at the GSN to transform them into semantics i.e. σ_i^S and σ_i^A, stored at the *SBiO*. The *GWiO* is also subscribed for any update related to actuating the AN, for example, a preference tuple (σ_p). There are three possibilities when ANs can receive actuation commands from *GWiO*: *i*) when the actuation command is based only on SN tuples, *ii*) when the actuation command is based on tuples received from the *SBiO*, i.e. σ_p only, and *iii*) when the actuation command is based on SN tuples and σ_p both.

3) *MiO* AND *SBiO* SEMANTIC INTERACTIONS AT SBSN

The semantic interactions between the *SBiO* and the *MiO* at the SBSN are explained in Table 8. Any *iO* residing on an LSN, an HSN and a GSN interacting with the SBSN needs permission to access the SBSN through the *MiO*. Their credentials are verified at the SBSN and the confirmation messages are given in return. For example, a *PCiO* sends their credentials of joining to the *SBiO* and then the *SBiO* sends a request to *MiO* for the credential verification of the *PCiO*. The *MiO* gives confirmation to the *SBiO* for valid credentials and a confirmation is sent to the *PCiO* subsequently. In case of invalid credentials, the *PCiO* cannot join the *SBiO* and receives exit confirmation. The joining transaction is only completed for all *iO*s from LSNs, HSNs, and GSNs once the *MiO* gives confirmation to the *SBiO* for their valid credentials. Integrating the *MiO* with the *SBiO* at the SBSN is an optional feature of any smart space because there are several applications where strict security and privacy policies may not be required. The *MiO* at the SBSN manages resources in a smart space. We have proposed and implemented resources

TABLE 7. Execution of a service from SNs to ANs.

No.	Interaction steps
1	Service installation (G^I) begin
2	$GWiO$ sends $G^I$to SNs and ANs
3	G^Ireceived and installed on SNs and ANs
4	G^Iend
5	$GWiO$ subscribed for SNs and ANs information
6	SN sends I^S to $GWiO$: $SN(I^S) \rightarrow GWiO(I^S)$
7	AN sends I^A to $GWiO$: $AN(I^A) \rightarrow GWiO(I^A)$
8	$GWiO$ processing for semantics : $I^S \rightarrow \sigma_i^S$ & $I^A \rightarrow \sigma_i^A$
9	$SBiO$ communication begin
10	R^J executed by $GWiO$ to join at $SBiO$
11	R^D executed by $GWiO$ to update σ_i^S & σ_i^A at $SBiO$
12	R^S executed by $GWiO$ to subscribe σ_p at $SBiO$
13	When $SBiO$ updated with σ_p then $GWiO$ receives: σ_p
14	$SBiO$ communication end
15	AN communication begin
16	**If** case 1 exists
17	then $GWiO(I^S) \longrightarrow AN(I^A)$
18	**Else** If case 2 exists
19	$GWiO(\sigma_p) \longrightarrow AN(I^A)$
20	**Else** If case 3 exists
21	$GWiO(I^S + \sigma_p) \longrightarrow AN(I^A)$
22	**Else**
23	No commands from $GWiO$ to AN
24	AN communication end

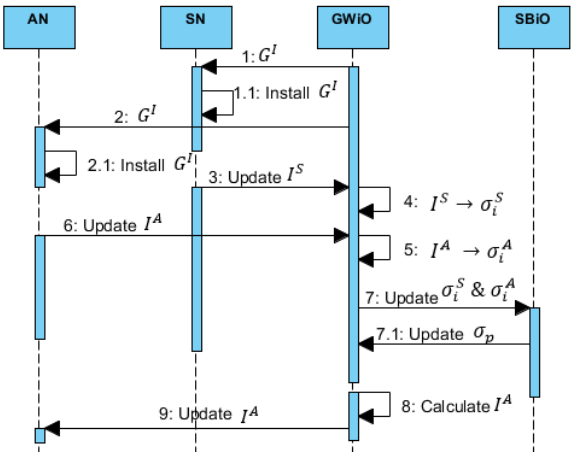


FIGURE 11. Sequence diagram of service execution from sensing to actuation.

TABLE 8. Semantic interactions between *MiO* and *SBiO*.

No.	Semantic interactions steps
A	**Monitoring of HSN, LSN and GSN**
1	**If** (smart node == HSN)
2	then HSN updates the PS at SBSN
3	**Else** If (smart node == LSN)
4	then LSN updates the PS at GSN
5	GSN updates the PS at SBSN
6	**If** (update time > user specified updating time) for HSN, GSN and LSN
7	then declare the smart node: stop working or fail
B	**Joining of new HSN and new GSN**
8	**If** new HSN and new GSN joined SBSN
9	then available services update at *MiO*
10	**If** new LSN joined GSN
11	then update first at GSN and then SBSN
12	*MiO* transmits available services to all subscribed *iO*s
C	**Allocation of services for HSN and LSN**
13	**If** HSN and LSN leave from SBSN and GSN respectively
14	then allocate services of leaving smart nodes to existing smart nodes
15	**If** SN <25% of memory or battery level
16	then classify the critical node
17	GSN transfers services to other SNs
D	**SBSN resources management**
18	**If** CPU or Memory or Network usage >75%
19	then classify the critical node
20	*MiO* transfers services to another SBSN

and services management mechanisms in [9] for LSN and in [11] for HSN. In this paper, we combine these mechanisms for LSNs and HSNs into a single management mechanism realized by the *MiO*. The *MiO* performs two tasks as described in Table 8: *i*) services management of LSN, HSN, GSN and SBSN, (A, B, and C in the table) and *ii*) resource management of the SBSN (D in the table).

A Presence Signal (PS) is introduced to monitor the state of existence of nodes in the smart space. The HSNs and LSNs update by sending PS messages periodically (e.g. a user specified period) to the SBSN and the GSN, respectively. Furthermore, the GSN also updates the presence state at the SBSN after receiving a PS as the *MiO* at the SBSN has subscription for PS updates. In case the updates are not delivered within a defined period of time the particular node is classified by the *MiO* as ''failed''. Moreover, a new HSN and GSN joining the SBSN can be provided suggestions for other SBSNs based on their service requirements through the *MiO*. When a new LSN joins at the GSN, the services of the new LSN are updated at the SBSN. The *MiO* at the SBSN offers the newly available service instances of a new smart node to other HSNs or LSNs (service subscribers) that need them. The *MiO* also replaces the services of a leaving smart node (e.g. upon failure or decommissioning) with those of remaining smart nodes still connected to the SBSN. LSNs are more prone to failure due to resource constraints, e.g. memory and battery. We can evaluate the states of memory and battery, for example, such that if any one of them is below 25% level then it is classified as a critical-node. If any LSN is classified as a critical-node then the *MiO* transfers its services to another LSN that is not in a critical condition.

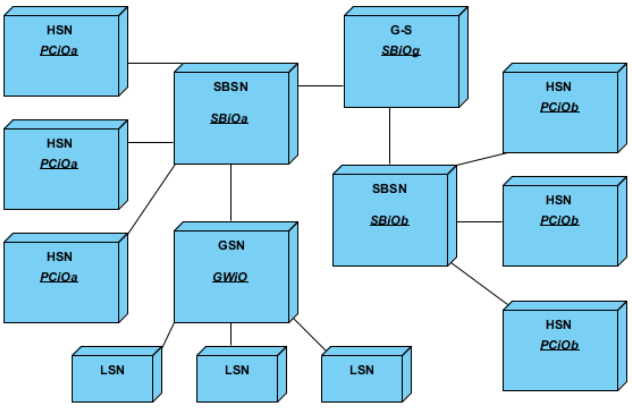


FIGURE 12. Physical deployment of two SBSNs in the smart space architecture.

The performance of a single SBSN is dependent on the number of *iO*s connected to it, representing either a ''normal'' or an ''overloaded'' state of the SBSN. If the CPU usage, already allocated memory space and network capacity usage of the SBSN are all below 75% then the SBSN is categorized as being in the ''normal'' state, and otherwise it is in the ''overloaded'' state. In case of the overloaded situation, there is a possibility of delay in transactions and increased chances of service failure. A *MiO* monitors its associated SBSN's resource availability such as CPU usage, free memory space and network usage, and updates this information at the G-S. The *MiO* also has access to the G-S for other SBSNs resource availability information. Therefore, the *MiO* can provide suggestions to HSNs and GSN for another SBSN in the case of an overload situation.

4) SEMANTIC INTERACTIONS BETWEEN TWO *SBiOs*

The semantic interactions between two *SBiO*s (i.e. $SBiO_a$ and $SBiO_b$, residing at two different SBSNs) are possible via the *MiO*s of the respective SBSNs and a $SBiO_g$ at G-S as shown in Fig. 12. The $SBiO_g$ at the G-S acts as a bridge between $SBiO_a$ and $SBiO_b$ via their corresponding *MiO*s.

$SBiO_a$ is connected with HSNs that host *PCiO*s such as $\{PCiO_{a1}, \ldots, PCiO_{an}\}$ and a *GWiO* for several LSNs. The set $\{PCiO_{a1}, \ldots, PCiO_{an}\}$ of *PCiO*s joined to $SBiO_a$ make up a local subdomain of a smart space. Similarly, $SBiO_b$ serves a set of $\{PCiO_{b1}, \ldots, PCiO_{bn}\}$ and makes up a local subdomain of the smart space. If any *PCiO* wants to share semantics from one subdomain to another then it needs to interact with the $SBiO_g$. This interaction is possible via the *MiO*s of the respective subdomains, making it possible to exchange semantics between $PCiO_{a1}$ with $PCiO_{b1}$ through the $SBiO_g$. Let us take an example of semantic interactions between two *SBiO*s. Consider the event that $PCiO_{a1}$ updates σ_x, where σ_x is an arbitrary piece of semantics; and $PCiO_{b1}$ is subscribed for the updates by $PCiO_{a1}$ (Sub_$PCiO_{a1}$). Note, it is assumed that the necessary transactions such as *JOIN* and *LEAVE* according to Table 6 have already taken place for $PCiO_{a1}$ and $PCiO_{b1}$ at their respective *SBiO*s. The semantic interactions between $PCiO_{a1}$ and $PCiO_{b1}$ are explained in Table 9

TABLE 9. Semantic interactions from $PCiO_{a1}$ to $PCiO_{b1}$.

S.No.	Semantic interactions steps
1	$PCiO_{b1}$ subscribed for $PCiO_{a1}$ updates at $SBiO_b$
2	MiO_b with $SBiO_b$ subscribe for $PCiO_{a1}$ updates at $SBiO_g$
3	MiO_g with $SBiO_g$ subscribed for $PCiO_{a1}$ at $SBiO_a$
4	When $PCiO_{a1}$ updates σ_x at $SBiO_b$
5	then the $PCiO_{b1}$ receive σ_x by the following sequence : $PCiO_{a1} \xrightarrow{\sigma_x} MiO_a \text{ with } SBiO_a \xrightarrow{\sigma_x} MiO_g \text{ with } SBiO_g \xrightarrow{\sigma_x} MiO_b \text{ with } SBiO_b \xrightarrow{\sigma_x} PCiO_{b1}$

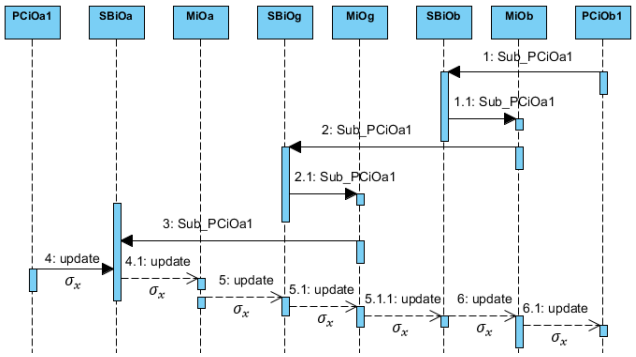


FIGURE 13. Sequence diagram for subscription between $PCiO_{a1}$ and $PCiO_{b1}$.

and the corresponding message sequence diagram is shown in Fig. 13. Once σ_x is updated at $SBiO_{a1}$ by $PCiO_{a1}$ then $PCiO_{b1}$ receives $\sigma_x$after several steps followed by MiOs in the sequence, where MiO_a, MiO_b and MiO_g are associated with $SBiO_a$, $SBiO_b$ and $SBiO_g$ respectively.

B. ADAPTATION OF iO AND APPLICATION BEHAVIORS

In a smart space many *iO*s collaborate to achieve certain goals of an application and there are adaptation requirements for different *iO*s, derived from these goals. These *iO*s adapt their behaviors at run time without the need of prior configuration. We classify the adaptation of *iO* behavior into the following three types:

1.) **Periodic Adaptive *iO* Behavior:** Gathering contextual information from an environment and periodically updating it to the *iO*, potentially changing its state and behavior.

2.) **Triggered Adaptive *iO* Behavior:** Triggering a certain event based on context information received from another *iO*, the result of which may change the receiving *iO*'s state and behavior.

3.) **Controlled Adaptive *iO* Behavior:** Controlling the *iO* state explicitly for achieving a certain application goal.

A set of joint adaptive behaviors of *iO*s of an application for realizing a common goal is referred to as adaptive application behavior. This is shown in Fig.14, following an intelligent adaptive lighting example. The lighting example has the assumptions and requirements as follows:

*Assumptions: iO*s at SN and AN are connected to the *GWiO* at GSN, where SN is a light sensor and AN is a light source.

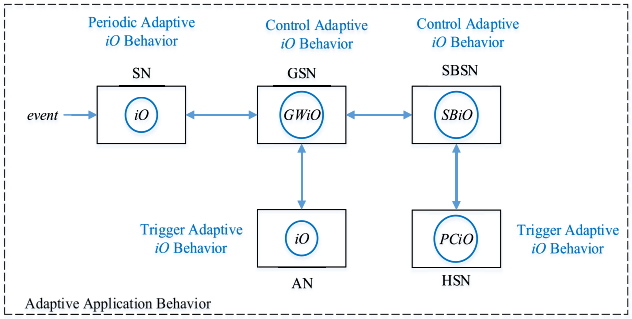


FIGURE 14. Adaptive application behavior by cumulative adaptive behaviors of *iO*s.

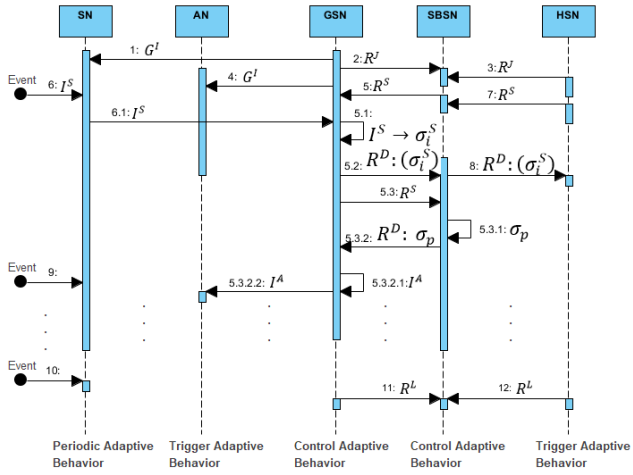


FIGURE 15. Semantic interactions for adaptive behaviors of *iO*s in a lighting example.

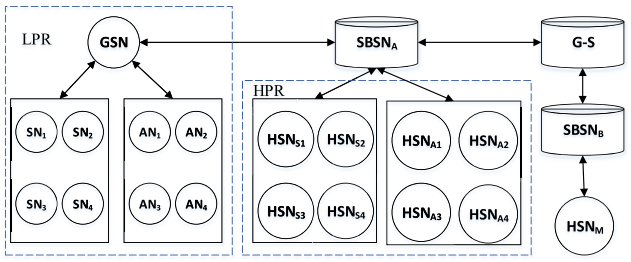


FIGURE 16. A case study scenario architecture.

The light sensor updates current light intensity values periodically every 10 seconds. The light source needs to control its light output based on the commands given by the GSN, where the preference of light intensity is set by a user at the GSN.

Requirements: The example system needs to achieve the goal of maintaining (measured) light intensity according to the user preferences. The HSN, which is subscribed at the SBSN, needs to get informed about any updates to the current light intensity state.

The semantic interactions executed for adaptive behaviors are shown in Fig.15. A piece of contextual information I^S is generated by the SN based on sensing the environment. The SN updates I^S periodically (e.g. every 10 seconds) at the GSN, allowing the *iO* at the GSN to show a periodic

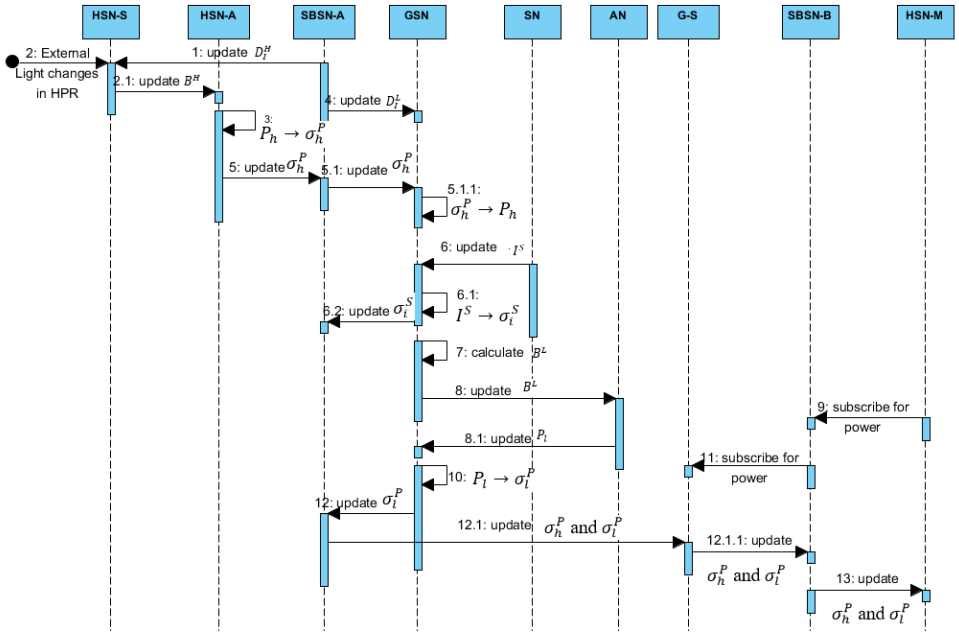


FIGURE 17. Case 1: Semantic interactions in case of external light changes in HPR.

TABLE 10. Smart nodes description for Figure 16.

Types	Description	Communication
SN_1 SN_2, SN_3 and SN_4	Light sensor nodes sense the illumination value in LPR and update to the GSN.	IEEE 802.15.4
AN_1, AN_2, AN_3 and AN_4	LED luminaries adjust the brightness levels of luminaries in LPR based on the GSN brightness commands.	USB serial communication
HSN_{S1},HSN_{S2}, HSN_{S2} and HSN_{S4}	Sensors placed in HPR, which produce illumination readings from the room and update these at $SBSN_1$.	SSAP over TCP/IP
HSN_{A1}, HSN_{A2}, HSN_{A3} and HSN_{A4}	LED luminaries with actuators placed in LPR, which received brightness level updates from $SBSN_1$.	SSAP over TCP/IP
$SBSN_A$	SBSN for smart nodes in both HPR and LPR (connected to G-S).	SSAP over TCP/IP
$SBSN_B$	SBSN placed outside of the building.	SSAP over TCP/IP
GSN	Gateway for communicating between LSNs and HSNs; placed in LPR.	IEEE 802.15.4
HSN_M	Smart phone connected to $SBSN_B$ for receiving any updates for power consumption from HPR and LPR.	SSAP over TCP/IP

TABLE 11. Interactions steps for Power managed smart lighting system.

No.	Steps for the interactions
1	GSN installs services and subscribes for information at SN_1 SN_2, SN_3, SN_4 , AN_1, AN_2, AN_3 and AN_4 (according to Table 7).
2	GSN, HSN_{S1},HSN_{S2}, HSN_{S3}, HSN_{S4}, HSN_{A1}, HSN_{A2}, HSN_{A3} and HSN_{A4} join, insert, update, subscribe at $SBSN_A$ (according to Table 6).
3	HSN_M joins for queries and subscriptions at $SBSN_B$ to get power consumption details from LPR and HPR (according to Table 6).
5	$SBSN_A$ and $SBSN_B$ follow the query and subscription (according to Table 9).

adaptive *iO* behavior. This information is further processed at the GSN and translated into semantics, i.e. $I^S \rightarrow \sigma_i^S$. The GSN further updates σ_i^S at the SBSN upon an R^D transaction. Consequently, σ_i^S is also updated at the HSN because the HSN is subscribed to updates made to σ_i^S semantics at the SBSN. Therefore, the *PCiO* at the HSN shows the triggered adaptive *iO* behavior. The GSN is also subscribed for the preference semantics σ_p at the SBSN and accordingly receives σ_p updates from the SBSN. The command for steering an actuator i.e. I^A is calculated by the GSN and is issued to the AN to actuate accordingly. The *GWiO* at the GSN and *SBiO* at the SBSN show the control adaptive *iO* behavior. Therefore, the goal of the lighting example is achieved by maintaining light intensity at the desired level, defined by the user preferences.

The above example explains the realization of adaptive behaviors for a lighting application. All three adaptive *iO* behaviors work together to make an adaptive smart space application behavior.

VI. CASE STUDY FOR SMART SPACES

There are many case studies for smart spaces in the literature, one of which is given in [88]. In this case study,

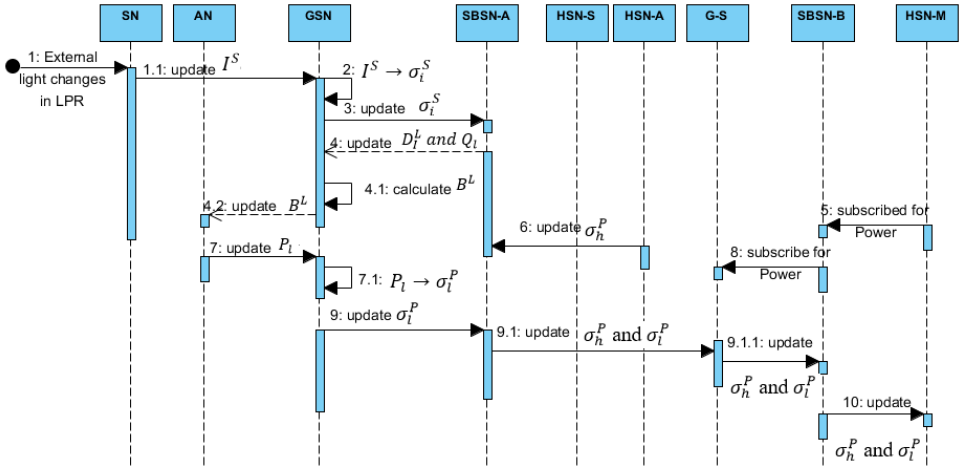


FIGURE 18. Case 2: Semantic interactions in case of external light changes in LPR.

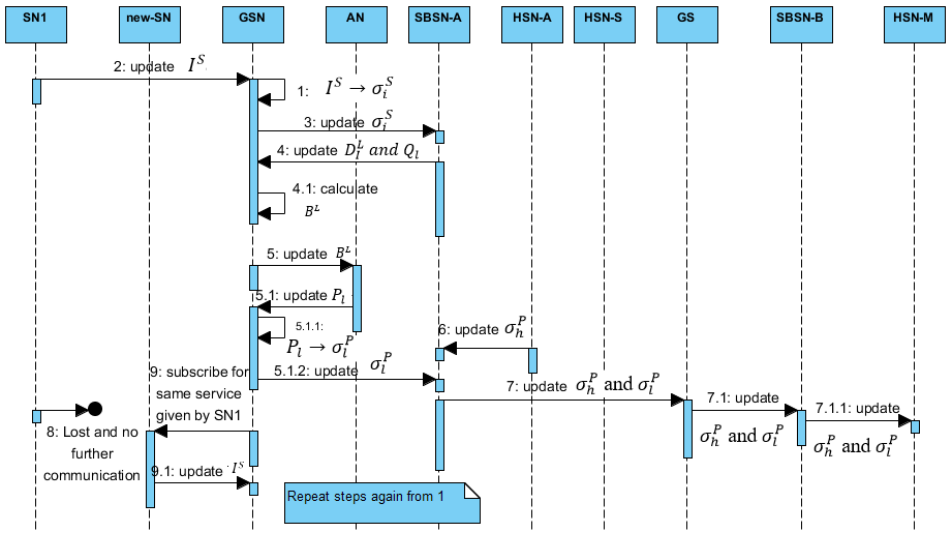


FIGURE 19. Case 3: Semantic interactions of a new SN joins and SN_1 fails in LPR.

a successful cultural heritage e-tourism scheme is discussed, which is based on the smart services provided by a human interactive device i.e. a mobile device. A system design solution for developing smart services in a smart space is provided.

In this paper, we considered a case study of a smart lighting application whose architecture is shown in Fig. 16. The smart nodes included are described in Table 10. The case study is based on the application scenario of a power-managed smart lighting system introduced and experimented in [48]. We consider two classes of rooms: high priority room (HPR) and low priority room (LPR).

Scenario: The rooms (HPR and LPR) in a building aim to consume power according to a quota (Q) assigned. HPRs get priority and, therefore, LPRs can use only the leftover power budget from the quota after subtracting the consumption in HPR. The scenario requires establishing semantic interoperability in and between LPR and HPR that contain smart nodes from various suppliers.

$SBSN_A$, $SBSN_B$ and G-S perform semantic reasoning using Pellet semantic reasoner and OWL semantic editor. The semantic interactions between the nodes in the scenario are explained in Table 11.

For this case study, we discuss the smart space properties, i.e., adaptation, interoperability, openness, extendibility and self-management, based on the following possible cases.

Case 1 (External Light Changes in HPR): When the external light changes in HPR the changed illumination is sensed by HSN_{Sn}. The possible interactions followed by the external light changes are shown in Fig. 17.

The commands of the brightness level for luminaries in HPR (i.e. B^H) are sent to HSN_{An} ($n \in \{1, 2, 3, 4\}$) based on the desired illumination in HPR (i.e. D_I^H) as defined by the subscription results from $SBiO_A$. Information on the

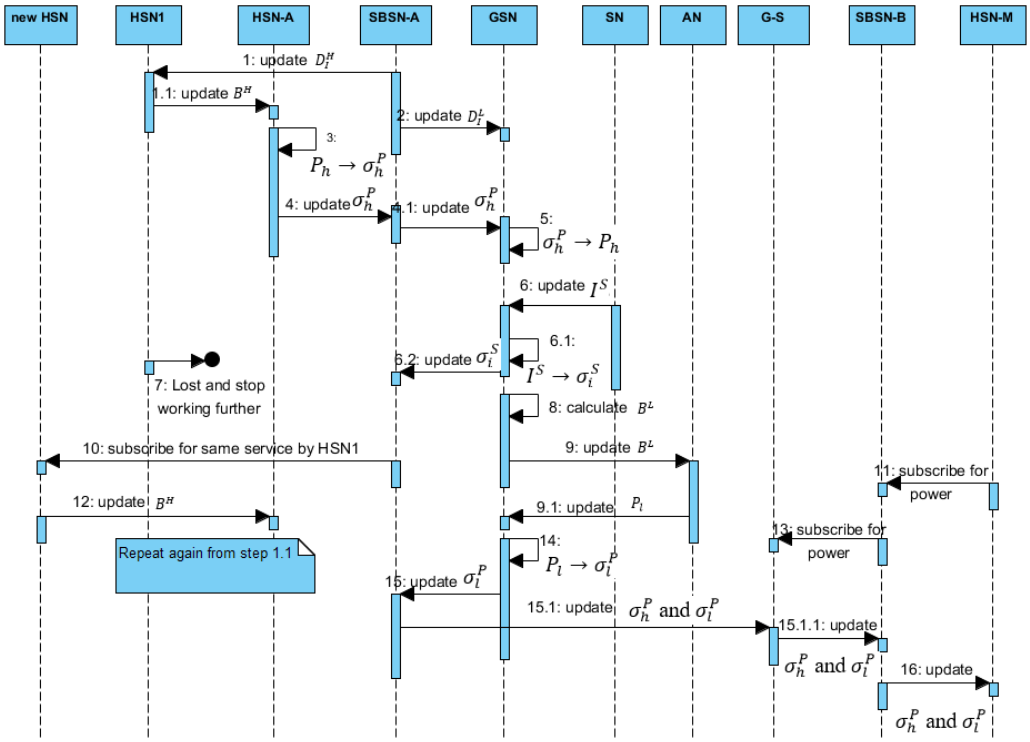


FIGURE 20. Case 4: Semantic interactions for a new HSN joins and HSN_1 fails in HPR.

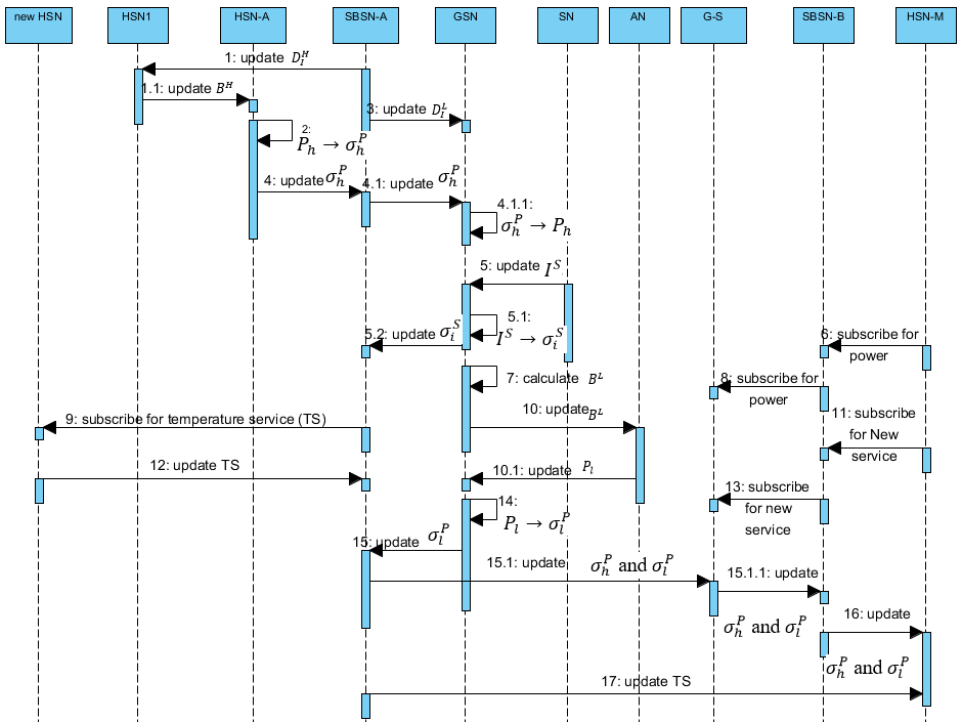


FIGURE 21. Case 5: Semantic interactions in case of a temperature service to be extended in the scenario.

power consumption in HPR (P_h) as a semantic representation (i.e. σ_h^P) is updated at $SBiO_A$. In LPR, SNs regularly sense the illumination (I^S) and update this at the *GWiO* of the GSN. Furthermore, I^S is translated into semantics (i.e. $I^S \rightarrow \sigma_i^S$) by the *GWiO* and is stored at the $SBiO_A$ of $SBSN_A$. The *GWiO* is also subscribed for σ_h^P and the

TABLE 12. Smart space properties and the scenario cases.

Smart space properties	Description with the scenario cases
Adaptation	Cases 1 and 2 showed that the fixed quotas for both rooms remained under control. It was observed that sometimes LPR sacrifices the maintenance of its power requirements to stay within Q. Finally in all cases, the system adapted according to the set objectives in the scenario. The behaviors of *iO*s in both rooms were adaptive according to the scenario requirements.
Interoperability	In all cases, 'meaningful' information is exchanged regardless of the node hardware and software specifications. Heterogeneous nodes from various vendors and communication technologies showed both communication and semantic interoperability.
Openness	Cases 3 and 4 explained the openness in the scenario and demonstrated the successful joining of a new smart node in both HPR and LPR. It was also observed that the joining of a node was possible once basic communication was established according to the application requirements. The new SN could join the smart space once connected to the GSN via IEEE 802.15.4 and a new HSN could easily join any of SBSNs once the communication was established by SSAP over TCP/IP. The set of protocols, message formats and syntax required by the new SN and HSN were easily compatible with the smart space architecture. Therefore, new smart nodes are also compatible with the ontology graph in the application scenario.
Extendibility	In Case 5, extendibility was shown with the addition of a new temperature application. The architecture was able to develop the application dynamically without having to interact with many smart nodes in the power managed smart lighting system. The system provided a decoupled programming interface and integrated with the application development. Combined with the openness property this allows new nodes and applications to easily be inserted into a smart space.
Self-management	In cases 3 and 4, the *MiO* self-managed the node's failure automatically by transferring the services of the failing (or normally leaving) nodes to other nodes in LPR and HPR, respectively.

quota Q. The commands for the brightness level of luminaries in LPR (B^L) are computed at the *GWiO* based on Q, P_h, I^S and the desired illumination in LPR (D_I^L). If a change is required in lighting of LPR as per the computation then the B^L command is sent to ANs in LPR accordingly. Furthermore, the power consumption in LPR (P_l) is sent to the *GWiO*, where P_l is converted into semantics σ_l^P and are stored at $SBiO_A$. The semantics σ_h^P and σ_l^P are also further stored at G-S by SBSN_A, allowing to share them with SBSN_B. Finally, HSN_M is subscribed to any update about power consumption in both LPR and HPR; and consequently receives σ_h^P and σ_l^P as subscription results once they are updated at SBSN_B.

In this case, we see the adaptive behavior of the scenario is achieved based on the objective set while the external light changes in HPR which accordingly results in the changes in both LPR and HPR accordingly. This means that the behaviors of *iO*s in HPR trigger changes in the behaviors of *iO*s in both LPR and HPR.

Case 2 (External Light Changes in LPR): When the external light changes in LPR the changed illumination is sensed by the corresponding SNs. Figure 18 explains the semantic interactions followed by an external light change in LPR. The B^L command is computed based on the subscription results of D_I^L from $SBiO_A$ and the leftover quota for LPR from HPR (Q_l) and it is sent to LPR ANs. Meanwhile, the σ_l^P consumption semantics are updated at $SBiO_A$. Any change of illumination in HPR is not required until a change is detected in external light, Q, and D_I^H. Moreover, the semantics σ_h^P and σ_l^P are updated at SBSN_A. Similar to Case 1, HSN_M receives σ_h^P and σ_l^P as subscription results once they are updated at SBSN_B.

In this case, the adaptive behavior of *iO*s in the scenario is achieved according to the desired illumination in LPR. This means that the behaviors of *iO*s in LPR are triggering the changes in other *iO*s' behaviors in LPR without any effect in the behaviors of *iO*s in HPR.

Case 3 (Joining of a New SN and SN_1 Fails in LPR): Consider a new SN, where the communication technology is compatible with IEEE.802.15.4 and the services are installed by the GSN. The new SN updates I^S as before, similar to other SNs, and executes the interactions as in Case 1 and 2.

If SN_1 stops working (e.g. due to hardware or network failure, or decommissioning from the network) then the *GWiO* transfers services provided by SN_1 to the new SN or to any other SN. The interaction steps for transferring the services from SN_1 to another SN are performed according to Table 8 and are shown in Fig. 19.

In this case, the joining or failure of SNs in LPR are performed without affecting the nodes and services in HPR. Therefore, the joining or leaving of smart nodes in the scenario is independent from the installed nodes in another priority rooms.

Case 4 (Joining of a New HSN and HSN_{S1} Leaves in HPR): For a new HSN to join, it needs to support the underlying communication technology, i.e. SSAP over TCP/IP. Once the joining of a new HSN is established in the network it can work and perform any task like other HSNs in HPR.

If HSN_{S1} leaves the network (e.g. due to hardware or network failure, or decommissioning from the network) then the *MiO* manages to transfer the services of HSN_{S1} to the new HSN or any other node that is capable of providing the same services. The semantic interactions to manage services are performed according to Table 8 and are shown in Fig. 20.

As in Case 3, the joining or leaving of a HSN in HPR is performed without affecting nodes in LPR. Therefore, the joining or leaving of smart nodes is independent from the installed nodes in another priority rooms.

Case 5 (Adding a New Application With a New Smart Node): In the existing scenario, consider an extension to integrate a temperature measurement application in both LPR and HPR. For this a temperature sensor needs to be added in each room. The two temperature sensors provide the service

of measuring temperatures from both rooms and updating the corresponding semantics at $SBSN_A$. The joining of the temperature sensors in LPR and HPR is established as in the cases 3 and 4, respectively.

The GSN introduces the new application information at $SBSN_A$ and updates the temperature information periodically which is able to be used by other *iO*s if required. Similar for HSNs, they need to first introduce their temperature services in HPR. Thereafter the temperatures of the rooms are measured and updated periodically at $SBSN_A$. The semantics of the temperature sensors from both LPR and HPR are shared with $SBSN_B$ via G-S. If HSN_M is subscribed to the temperature information then it will get the notification once the temperature information is updated at $SBSN_B$. The possible semantic interactions are shown in Fig. 21.

In this case, we showed the possibility of adding a new application by introducing a new smart node.

These cases were discussed to cover all of the smart space properties of the proposed solutions. We can relate and discuss the properties in detail in Table 12.

The study and discussion of several cases of the power-managed smart lighting scenario explained the use of our comprehensive smart space architecture in a real scenario. The smart space properties of adaptation, interoperability, openness, extendibility and self-management performed well and contributed to the comprehensive smart space architecture. The power-managed smart lighting system based on the comprehensive smart space architecture shows that it is possible to develop a smart space that establishes interoperability using various types of smart nodes and their technologies.

VII. CONCLUSIONS

In this paper, fundamental smart space concepts, components making up a smart space and the smart space itself were *formally* defined. We introduced the role and importance of information objects in the development of smart space designs. Five general properties of smart space were proposed (*adaptation*, *interoperability*, *openness*, *extendibility* and *self-management*) based on a literature survey. A comparison of state-of-the-art smart space designs with respect to the availability of general smart space components and the smart space properties was provided. Furthermore, generic architectural models for smart spaces were presented and discussed in relation to the other designs described in the literature. Finally, we proposed a comprehensive smart space architecture as a contemporary solution for smart spaces and instantiated it on a power managed smart lighting system.

As future work, a scalability, reliability and availability analysis and optimizations considering these aspects are needed for better smart space design. Scalability refers to the capability of a smart space to accommodate numerous information objects, smart nodes, and applications. Reliability refers to the mean time between failures of a smart space application. Availability refers to the probability of failure-free smart application operation at any given time. In addition, the tradeoff between these concerns and the size of a smart space (number of smart nodes) is also worth investigating.

REFERENCES

[1] M. Weiser, "The computer for the 21 st century," *Sci. Amer.*, vol. 265, no. 3, pp. 94–105, Sep. 1991.
[2] J. C. Augusto, V. Callaghan, D. Cook, A. Kameas, and I. Satoh, "Intelligent environments: A manifesto," in *Human-Centric Computing and Information Sciences*. Berlin, Germany: Springer, 2013.
[3] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of Things for smart healthcare: Technologies, challenges, and opportunities," *IEEE Access*, vol. 5, pp. 26521–26544, 2017.
[4] H. Jiang, C. Cai, X. Ma, Y. Yang, and J. Liu, "Smart home based on WiFi sensing: A survey," *IEEE Access*, vol. 6, pp. 13317–13325, 2018.
[5] L. Qiu, Q. Lei, and Z. Zhang, "Advanced sentiment classification of tibetan microblogs on smart campuses based on multi-feature fusion," *IEEE Access*, vol. 6, pp. 17896–17904, 2018.
[6] E. Mathews, S. S. Guclu, Q. Liu, T. Özçelebi, and J. J. Lukkien, "The Internet of lights: An open reference architecture and implementation for intelligent solid state lighting systems," *Energies*, vol. 10, no. 8, p. 1187, 2017.
[7] (Mar. 2014). *DLNA Guidelines*. [Online]. Available: http://www.dlna.org/guidelines/
[8] T. Özçelebi, J. Lukkien, R. Bosman, and Ö. Uzun, "Discovery, monitoring and management in smart spaces composed of low capacity nodes," *IEEE Trans. Consum. Electron.*, vol. 56, no. 2, pp. 570–578, May 2010.
[9] S. Bhardwaj, T. Ozcelebi, J. Lukkien, and C. Uysal, "Resource and service management architecture of a low capacity network for smart spaces," *IEEE Trans. Consum. Electron.*, vol. 58, no. 2, pp. 389–396, May 2012.
[10] S. Bhardwaj, T. Ozcelebi, J. J. Lukkien, and K. M. Lee, "Semantic interoperability architecture for smart spaces," *Int. J. Fuzzy Logic Intell. Syst.*, vol. 18, no. 1, pp. 50–57, Mar. 2018.
[11] S. Bhardwaj, T. Ozcelebi, A. A. Syed, J. J. Lukkien, and O. Ozunlu, "Increasing reliability and availability in smart spaces: A novel architecture for resource and service management," *IEEE Trans. Consum. Electron.*, vol. 58, no. 3, pp. 787–793, Aug. 2012.
[12] S. Bhardwaj, T. Ozcelebi, R. Verhoeven, and J. Lukkien, "Smart indoor solid state lighting based on a novel illumination model and implementation," *IEEE Trans. Consum. Electron.*, vol. 57, no. 4, pp. 1612–1621, Nov. 2011.
[13] K. Scott and R. Benlamri, "Context-aware services for smart learning spaces," *IEEE Trans. Learn. Technol.*, vol. 3, no. 3, pp. 214–227, Jul./Sep. 2010.
[14] N. D. Rodriguez, "A framework for context-aware applications for smart spaces," in *Proc. IEEE/IPSJ Int. Symp. Appl. Internet, Munich, Bavaria*, Jul. 2011, pp. 218–221.
[15] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Comput.*, vol. 14, no. 1, pp. 44–51, Jan./Feb. 2010.
[16] A. K. Dey, G. D. Abowd, and G. D. Salber, "A context-based infrastructure for smart environments," Georgia Inst. Technol., Atlanta, GA, USA, GVU Tech. Rep. GIT-GVU-99-39, 1999. [Online]. Available: http://hdl.handle.net/1853/3406
[17] E. Goh, D. Chieng, A. K. Mustapha, Y. C. Ngeow, and H. K. Low, "A context-aware architecture for smart space environment," in *Proc. Int. Conf. Multimedia Ubiquitous Eng. (MUE)*, Seoul, South Korea, 2007, pp. 908–913.
[18] I. G. Roussaki *et al.*, "Self-improving personal smart spaces for pervasive service provision," in *Future Internet Assembly*. Amsterdam, The Netherlands: IOS Press, 2010, pp. 193–203, doi: 10.3233/978-1-60750-539-6-193.
[19] T. Kawashima, J. Ma, R. Huang, and B. O. Apduhan, "GUPSS: A gateway-based ubiquitous platform for smart spaces," in *Proc. Int. Conf. Comput. Sci. Eng.*, Vancouver, BC, Canada, Aug. 2009, pp. 213–220.
[20] B. J. J. van der Vlist, G. Niezen, J. Hu, and L. M. G. Feijs, "Semantic connections: Exploring and manipulating connections in smart spaces," in *Proc. IEEE Symp. Comput. Commun.*, Riccione, Italy, Jun. 2010, pp. 1–4.
[21] Z. Song, A. A. Cárdenas, and R. Masuoka, "Semantic middleware for the Internet of Things," in *Proc. IEEE Internet Things (IOT)*, Tokyo, Japan, Nov./Dec. 2010, pp. 1–8.
[22] D. Pfisterer *et al.*, "SPITFIRE: Toward a semantic Web of things," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 40–48, Nov. 2011.

[23] H. Abdullah, M. Rinne, S. Törmä, and E. Nuutila, "Efficient matching of SPARQL subscriptions using rete," in *Proc 27th Annu. ACM Symp. Appl. Comput.*, New York, NY, USA, 2012, pp. 372–377.

[24] F. Morandi, L. Roffia, A. D'Elia, F. Vergari, and T. S. Cinotti, "RedSib: A smart-M3 semantic information broker implementation," in *Proc. 12th Conf. Open Innov. Assoc. (FRUCT)*, Oulu, Finland, 2012, pp. 1–13.

[25] N. D. Rodríguez, J. Lilius, M. P. Cuéllar, and M. D. Calvo-Flores, "An approach to improve semantics in Smart Spaces using reactive fuzzy rules," in *Proc. Joint IFSA World Congr. NAFIPS Annu. Meeting (IFSA/NAFIPS)*, Edmonton, AB, Canada, 2013, pp. 436–441.

[26] E. Ovaska and J. Kuusijarvi, "Piecemeal development of intelligent applications for smart spaces," *IEEE Access*, vol. 2, pp. 199–214, 2014.

[27] J. Kiljander *et al.*, "Semantic interoperability architecture for pervasive computing and Internet of Things," *IEEE Access*, vol. 2, pp. 856–873, 2014.

[28] J. Zeng, L. T. Yang, H. Ning, and J. Ma, "A systematic methodology for augmenting quality of experience in smart space design," *IEEE Wireless Commun.*, vol. 22, no. 4, pp. 81–87, Aug. 2015.

[29] 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," in *Proc. 9th IEEE Int. Conf. Intell. Data Acquisition Adv. Comput. Syst., Technol. Appl. (IDAACS)*, Bucharest, Romania, Sep. 2017, pp. 1000–1005.

[30] A. S. Vdovenko, D. G. Korzun, and I. V. Galov, "Simulation performance evaluation of Smart-M3 applications for Internet of Things environments," in *Proc. 9th IEEE Int. Conf. Intell. Data Acquisition Adv. Comput. Syst., Technol. Appl. (IDAACS)*, Bucharest, Romania, Sep. 2017, pp. 994–999.

[31] A. S. Vdovenko, O. I. Bogoiavlenskaia, and D. G. Korzun, "Study of active subscription control parameters in large-scale smart spaces," in *Proc. 21st Conf. Open Innov. Assoc. (FRUCT)*, Helsinki, Finland, 2017, pp. 344–350.

[32] S. Ahmad, L. Hang, and D. H. Kim, "Design and implementation of cloud-centric configuration repository for DIY IoT applications," *Sensors*, vol. 18, no. 2, p. 474, 2018.

[33] Z. Shelby, B. Frank, and D. Sturek. (2011). *Constrained Application Protocol (CoAP) (CoRE Working Group)*. Accessed: Jul. 7, 2011. [Online]. Available: http://www.ietf.org/id/draft-ietf-core-coap-06.txt

[34] 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," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, AK, USA, 2017, pp. 3862–3869.

[35] P. Maillot, T. Raimbault, D. Genest, and S. Loiseau, "Consistency evaluation of RDF data: How data and updates are relevant," *Proc. 10th Int. Conf. Signal-Image Technol. Internet-Based Syst.*, Marrakech, Morocco, 2014, pp. 187–193.

[36] B. Cheng, S. Zhao, C. Li, and J. Chen, "A Web services discovery approach based on mining underlying interface semantics," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 950–962, May 2017.

[37] *OWL (Web Ontology Language)*. Accessed: Nov. 13, 2018. [Online]. Available: https://www.w3.org/OWL/

[38] B. Parsia and E. Sirin, "Pellet: An OWL-DL reasoner," in *Proc. 3rd Int. Semantic Web Conf. (ISWC)*, Hiroshima, Japan, Nov. 2004.

[39] G. Yang, M. Kifer, and C. Zhao. (Jun. 2002). *Flora-2: User's Manual*. [Online]. Available: http://flora.sourceforge.net/

[40] *Jena 2.7.3—A Semantic Web Framework*. Accessed: Nov. 13, 2018. [Online]. Available: http://downloads.sourceforge.net/jena/jena-2.7.3

[41] Y. Kazakov, M. Krötzsch, and F. Simancík, "ELK reasoner: Architecture and evaluation," in *Proc. 1st Int. Workshop OWL Reasoner Eval. (ORE), CEUR Workshop*, Manchester, U.K., 2012, pp. 1–12.

[42] J. H. Gennari *et al.*, "The evolution of Protégé: An environment for knowledge-based systems development," *Int. J. Hum.-Comput. Stud.*, vol. 58, no. 1, pp. 89–123, 2003.

[43] J. Bārzdiņš, G. Bārzdiņš, K. Čerāns, R. Liepiņš, and A. Sproģis, "UML style graphical notation and editor for OWL 2," in *Perspectives in Business Informatics Research* (Lecture Notes in Business Information Processing), vol. 64, no. 2. Berlin, Germany: Springer, 2010, pp. 102–114.

[44] A. Stellato *et al.*, "VocBench: A Web application for collaborative development of multilingual thesauri," in *The Semantic Web. Latest Advances and New Domains* (Lecture Notes in Computer Science), vol. 9088. Cham, Switzerland: Springer, 2015, pp. 38–53.

[45] K. A. Taipale, "The trusted systems problem: Security envelopes, statistical threat analysis, and the presumption of innocence," in *Proc. IEEE Intell. Syst. Homeland Secur.-Trends Controversies*, Sep./Oct. 2005, vol. 20, no. 5, pp. 80–83.

[46] D.-M. Han and J.-H. Lim, "Smart home energy management system using IEEE 802.15.4 and ZigBee," *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, pp. 1403–1410, Aug. 2010.

[47] M. Honkanen, A. Lappetelainen, and K. Kivekas, "Low end extension for Bluetooth," in *Proc. IEEE Radio Wireless Conf.*, Atlanta, GA, USA, 2004, pp. 199–202.

[48] S. Bhardwaj, A. A. Syed, T. Ozcelebi, and J. J. Lukkien, "Power-managed smart lighting using a semantic interoperability architecture," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 420–427, May 2011.

[49] E. Carlini, M. Coppola, P. Dazzi, M. Mordacchini, and A. Passarella, "Self-optimising decentralised service placement in heterogeneous cloud federation," in *Proc. IEEE 10th Int. Conf. Self-Adapt. Self-Organizing Syst. (SASO)*, Augsburg, Germany, Sep. 2016, pp. 110–119.

[50] X. Wang and Y. Mu, "Addressing and privacy support for 6LoWPAN," in *IEEE Sensors J.*, vol. 15, no. 9, pp. 5193–5201, Sep. 2015.

[51] F. Wu, C. Rüdiger, J.-M. Redouté, and M. R. Yuce, "WE-Safe: A wearable IoT sensor node for safety applications via LoRa," in *Proc. IEEE 4th World Forum Internet of Things (WF-IoT)*, Singapore, Feb. 2018, pp. 144–148.

[52] *IEEE Standard for Service Composition Protocols of Next Generation Service Overlay Network*, IEEE Standard 1903.2-2017, May 2018, pp. 1–54.

[53] D. Korzun, "On the smart spaces approach to semantic-driven design of service-oriented information systems," in *Communications in Computer and Information Science*, vol. 615. Cham, Switzerland: Springer, 2016, pp. 181–195.

[54] J. Kim and J.-W. Lee, "OpenIoT: An open service framework for the Internet of Things," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Seoul, South Korea, Mar. 2014, pp. 89–93.

[55] J. Fernández, G. Pimpollo, and R. Otaolea, "Smart objects for intelligent applications—ADK," in *Proc. IEEE Symp. Vis. Lang. Hum.-Centric Comput.*, Leganes, Spain, Sep. 2010, pp. 267–268.

[56] D. G. Korzun, "Service formalism and architectural abstractions for smart space applications," in *Proc. 10th Central Eastern Eur. Softw. Eng. Conf. (CEE-SECR)*, Moscow, Russia, Oct. 2014.

[57] S. Balandin and H. Waris, "Key properties in the development of smart spaces," in *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments* (Lecture Notes in Computer Science), vol. 5615. Berlin, Germany: Springer, 2009, pp. 3–12.

[58] S. Bhardwaj, T. Özcelebi, R. Verhoeven, and J. J. Lukkien, "Delay performance in a semantic interoperability architecture," in *Proc. IEEE/IPSJ Int. Symp. Appl. Internet*, Munich, Germany, Jul. 2011, pp. 280–285.

[59] J. Zeng, L. T. Yang, J. Ma, and M. Guo, "HyperspaceFlow: A system-level design methodology for smart space," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 4, pp. 568–583, Oct./Dec. 2016.

[60] J. Zeng, L. T. Yang, and J. Ma, "A system-level modeling and design for cyber-physical-social systems," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 2, May 2016, Art. no. 35.

[61] J. Zeng, L. T. Yang, M. Lin, Z. Shao, and D. Zhu, "System-level design optimization for security-critical cyber-physical-social systems," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 2, Apr. 2017, Art. no. 39.

[62] D. J. Cook *et al.*, "MavHome: An agent-based smart home," in *Proc. 1st IEEE Int. Conf. Pervasive Comput. Commun., (PerCom)*, Fort Worth, TX, USA, Mar. 2003, pp. 521–524.

[63] S. Sukalikar, S. Kumar, and N. Baliyan, "Analysing cohesion and coupling for modular ontologies," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, New Delhi, India, 2014, pp. 2063–2066.

[64] W. Lumpkins, "Home Automation: Insteon (X10 meets powerline) [product reviews]," *IEEE Consum. Electron. Mag.*, , vol. 4, no. 4, pp. 140–144, Oct. 2015.

[65] M. Neugebauer, J. Plonnigs, K. Kabitzsch, and P. Buchholz, "Automated modeling of LonWorks building automation networks," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, Vienna, Austria, Sep. 2004, pp. 113–118.

[66] T. Perumal, A. R. Ramli, C. Y. Leong, and S. Mansor, "Interoperability for smart home environment using Web services," *Int. J. Smart Home*, vol. 2, no. 4, pp. 1–16, Oct. 2008.

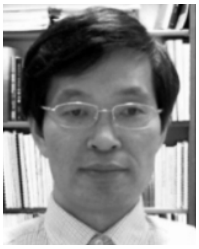
[67] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, Newport Beach, CA, USA, 2000. [Online]. Available: https://www.ics.uci.edu/?fielding/pubs/dissertation/top.htm

[68] S. Lee, Y. Lee, and H. Lee, "Jini-based ubiquitous computing middleware supporting event and context management services," in *Ubiquitous Intelligence and Computing* (Lecture Notes in Computer Science), vol. 4159, J. Ma, H. Jin, L. T. Yang, and J. J. P. Tsai, Eds. Berlin, Germany: Springer, 2006.
[69] R. Lea, S. Gibbs, A. Dara-Abrams, and E. Eytchison, "Networking home entertainment devices with HAVi," *Computer*, vol. 33, no. 9, pp. 35–43, Sep. 2000.
[70] L. Yiqin, F. Fang, and L. Wei, "Home networking and control based on UPnP: An implementation," in *Proc. 2nd Int. Workshop Comput. Sci. Eng.*, Qingdao, China, 2009, pp. 385–389.
[71] 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," *Comput. Elect. Eng.*, vol. 52, pp. 208–222, May 2016.
[72] S. R. Moosavi *et al.*, "SEA: A secure and efficient authentication and authorization architecture for IoT-based healthcare using smart gateways," *Procedia Comput. Sci.*, vol. 52, pp. 452–459, 2015.
[73] S. Jabbar, M. Khan, B. Nathali Silva, and K. Han, "A REST-based industrial Web of things' framework for smart warehousing," *J. Supercomput.*, vol. 74, no. 9, pp. 4419–4433, 2016.
[74] A. Agra, M. Christiansen, K. S. Ivarsøy, I. E. Solhaug, and A. Tomasgard, "Combined ship routing and inventory management in the salmon farming industry," *Ann. Oper. Res.*, vol. 253, no. 2, pp. 799–823, 2016.
[75] 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, Jan. 2015.
[76] 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 Commun.*, vol. 23, no. 5, pp. 68–74, May 2016.
[77] R. Zhang, S. Newman, M. Ortolani, and S. Silvestri, "A network tomography approach for traffic monitoring in smart cities," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2268–2278, Jul. 2018.
[78] D. M. L. Escriva, J. Torres-Sospedra, and R. Berlanga-Llavori, "Smart outdoor light desktop central management system," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 2, pp. 58–68, Apr. 2018.
[79] X. Su, H. Zhang, J. Riekki, A. Keränen, J. K. Nurminen, and L. Du, "Connecting IoT sensors to knowledge-based systems by transforming SenML to RDF," *Procedia Comput. Sci.*, vol. 32, pp. 215–222, Jun. 2014.
[80] J. J. Jung, "Semantic preprocessing for mining sensor streams from heterogeneous environments," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 6107–6111, May 2011.
[81] F. Amato, V. Casola, A. Gaglione, and A. Mazzeo, "A semantic enriched data model for sensor network interoperability," *Simul. Model. Pract. Theory*, vol. 19, no. 8, pp. 1745–1757, Sep. 2011.
[82] S. Jabbar, F. Ullah, S. Khalid, M. Khan, and K. Han, "Semantic interoperability in heterogeneous IoT infrastructure for healthcare," *Wireless Commun. Mobile Comput.*, vol. 2017, Mar. 2017, Art. no. 9731806.
[83] C. Malewski, A. Bröring, P. Maué, and K. Janowicz, "Semantic matchmaking & mediation for sensors on the sensor Web," in *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 3, pp. 929–934, Mar. 2014.
[84] H. Dibowski, J. Ploennigs, and M. Wollschlaeger, "Semantic device and system modeling for automation systems and sensor networks," in *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1298–1311, Apr. 2018.
[85] B. L. Gorman, D. R. Resseguie, and C. Tomkins-Tinch, "Sensorpedia: Information sharing across incompatible sensor systems," in *Proc. Int. Symp. Collaborative Technol. Syst.*, Baltimore, MD, USA, 2009, pp. 448–454.
[86] K. A. Delin and E. Small, "The sensor Web: Advanced technology for situational awareness," in *Wiley Handbook of Science and Technology for Homeland Security*. Hoboken, NJ, USA: Wiley, 2009. [Online]. Available: http://www.sensorwaresystems.com/historical/resources/briefings.shtml
[87] S. Nath, J. Liu, and F. Zhao, "SensorMap for wide-area sensor Webs," *Computer*, vol. 40, no. 7, pp. 90–93, Jul. 2007.
[88] K. A. Kulakov, O. B. Petřina, D. G. Korzun, and A. G. Varfolomeyev, "Towards an understanding of smart service: The case study for cultural heritage e-Tourism," in *Proc. 18th Conf. Open Innov. Assoc. Seminar Inf. Secur. Protection Inf. Technol. (FRUCT-ISPIT)*, St. Petersburg, Russia, Apr. 2016, pp. 145–152.

SACHIN BHARDWAJ received the B.Tech. degree in computer science and engineering from Dr. A. P. J. Abdul Kalam Technical University, India, in 2004, and the M.S. degree in ubiquitous and network engineering from Dongseo University, Pusan, South Korea, in 2007. He is currently a Researcher with the Artificial Intelligence Lab, Department of Computer Science, Chungbuk National University, South Korea. His research interest lies in smart spaces, artificial intelligence, semantic interoperability, and deep learning.

TANIR OZCELEBI received the Ph.D. degree in electrical engineering from Koç University, Istanbul, in 2006. During his Ph.D. studies, he developed multiple objective optimization models for content adaptive online streaming of below entertainment quality videos. In 2006, he joined the research group System Architecture and Networking (SAN), Eindhoven University of Technology (TU/e), The Netherlands, as a Post-Doctoral Researcher. For two years he did research on enhancing the Quality of Service for streamed multimedia in Next Generation Networks and IP Multimedia Subsystem. He is currently an Assistant Professor with TU/e SAN and also the Program Manager for the Bright Environments, a Research Program of the TU/e Intelligent Lighting Institute. His main research interests are life-cycle management for resource-constrained embedded devices, architecture development for smart spaces and Internet of Things, and resource and QoS management, and data analytics for networked services.

JOHAN J. LUKKIEN received the M.Sc. and Ph.D. degrees from Groningen University, The Netherlands. In 1991, he joined Eindhoven University after a two years leave at the California Institute of Technology. He has been the Head of the System Architecture and Networking Research Group, Eindhoven University of Technology (TU/e), since 2002, and also has been the Dean of the TU/e Department of Mathematics and Computer Science since 2017. His research interests include the design, architecture, and performance analysis of parallel and distributed systems. Until 2000, he was involved in large-scale simulations in physics and chemistry. Since 2000, his research focus has shifted to the application domain of networked resource-constrained embedded systems which includes the field of IoT Contributions of the SAN group are in the area of component-based middleware for resource-constrained devices, distributed coordination, quality of service in networked systems, and schedulability analysis in real-time systems.



KEON MYUNG LEE received the B.S., M.S., and Ph.D. degrees in computer science from KAIST, South Korea. He was a Post-Doctoral Fellow with INSA de Lyon, France. He was a Visiting Professor with the University of Colorado at Denver and a Visiting Scholar with Indiana University, USA. He is currently a Professor and the Head with the Department of Computer Science, Chungbuk National University, South Korea. His principal research interests are data mining, machine learning, soft computing, big data processing, and intelligent service systems.