Smart Spaces Middleware: A Requirement-Oriented Overview

Sergey Marchenkov, Dmitry Korzun
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
{marchenk, dkorzun}@cs.karelia.ru

***Abstract*—The progress of Internet of Things (IoT) leads to appearance of diverse networked environments where specific class of service-oriented applications—smart spaces—are becoming developed and deployed. The deployment is based on certain software platforms, widely accepted as smart spaces middleware. This paper overviews this type of middleware with focus on the requirements that the middleware supports in IoT environments. We analyze and systemize the middleware requirements. The systemized requirements are mapped to the particular existing middleware solutions, leading to identification and comparison of basic approaches in the smart spaces development problem.**

I. Introduction

Internet of Things (IoT) provides diverse networked environments where specific class of service-oriented applications—smart spaces—can be deployed [1]. An IoT environment typically consists of many various devices. They provide significant computation and information resources, especially related to the "smart property" of participating things [2], even transforming to Ambient Intelligence (AmI) systems [3].

Smart space application enables construction of so called "semantic services"—the concept derived from Semantic Web [4]). Such services are based on interlinked meaning of the involved resources and processes. In contrast to the global view of Semantic Web, smart spaces are focused on localized IoT environments, not on the entire system of web resources. Constructed services have uniquely described semantics (for participants), accessible through the Internet, allow automated discovery, composition, and proactive initiation.

Smart spaces middleware is used to deploy an application in a particular IoT environment [5]). In this paper, we consider the software development problem for smart spaces middleware in the IoT case. Although relatively many middleware solutions are already known (on the technology market or as research pilots), the smart spaces occupy the upper layer from the IoT technology, and straightforward use of IoT middleware is not appropriate.

We show how existing IoT-related approaches and technologies can be used to solve the studied software development problem. We analyze and systemize the smart spaces middleware requirements. The systemized requirements are mapped to the particular existing middleware solutions, leading to identification and comparison of basic approaches in the smart space development problem.

The rest of the paper is organized as follows. Section II defines the smart space development and deployment problem for IoT environments. We introduce our requirements system applicable to smart spaces middleware development. Section III overviews existing middleware solutions to characterize the achieved level of requirements satisfaction and to identify possible design approaches. Section IV summarizes the results of this overview.

II. Deployment of Smart Spaces in IoT Environments

IoT provides diverse networked environments for deploying smart spaces [1]. A smart space supports a shared view on available resources in the IoT environment and creates semantics for the use of these resources in cooperative construction of services by multiple devices [3]. In this section, we consider the deployment problem of smart spaces in respect to the opportunities of emerging IoT, Web, and semantics technologies. To solve the deployment problem, smart spaces middleware can be developed for installing in a particular IoT environment. Based on the presented study of existing technologies, we systemize the requirements to smart spaces middleware.

A. Enabler Technologies

The IoT concept enables the capability of connecting and integrating a wide range of technologies and devices, from home automation and smart cities to any system of sensors, actuators, tags, or physical things in the Internet [6], [7]. The fundamental building blocks are smart objects defined as acting autonomously to make own decisions, sensing the environment, communicating with other objects, accessing resources of the existing Internet, and interacting with human [2], [8].

The next IoT evolution step is connecting smart objects and the Web, leading to the so-called Web of Things (WoT) [9], [10]). The interactions are enabled through the definition of application programming interfaces over HTTP protocol based on Web Services following the RESTful architecture. Accordingly, the services and information provided by the objects can be incorporated in the Web. As a result, IoT smart objects can use the same language as other resources on the Web. One can easily integrate physical devices (things) with web pages (information world) allowing web users and services to experience the physical world and act on its data and services.

The Semantic Web of Things (SWoT) further advances the Web technologies and WoT concept with the Semantic Web technologies [11]. The technologies are focused on enabling

ISSN 2305-7254

wide scale integration and interoperability [4]). Global share and re-use of smart objects enable provision of semantic services based on interlinked meaning of the involved resources and processes. One of the challenges to move towards the SWoT is to define common semantic descriptions (ontologies) that allow data to be universally and understandable. SWoT ensures an extension to the IoT allowing integration of both the physical and information worlds.

The M3 architecture (multidevice, multivendor, multidomain) for smart spaces applies technologies from IoT, WoT, and SWoT to develop service-oriented applications for ubiquitous computing environments [12], [5], [13], where the physical, information, and social worlds are semantically fused. Service construction is implemented by software agents running on various devices in the IoT environment (small or large, local or remote). The focus is on the resources related to a spatial-restricted area: locally produced data, computational power of surrounding devices, expertise of participating users, etc. Semantics of such resources are integrated in a knowledge base, which is collectively created, maintained, and used by agents through semantic information broker (SIB). Such an agent is also called knowledge processor (KP) or smart space participant.

B. Smart Spaces Middleware

Examples of services for IoT environments can be found in [2], [8], [5]. Smart spaces are used in various problem domains such as collaborative work [14], [15], mobile healthcare [16], [17], digital museums and cultural heritage [18], [19]. In general, smart spaces middleware provide support for interaction and integration of various devices, software components, and information resources within a common goal for service construction and delivery [20], [21], [22]). The support is implemented as a separate layer on the top of IoT (the network communication layer).

There are many variants of IoT middleware that implement the network communication layer, e.g., see [23], [24], [25]. Such IoT middleware can be considered from the semantic-oriented viewpoint (among many other viewpoints), reflecting the observable growth of using the semantic methods in the IoT. In this case, IoT middleware become an appropriate base for deploying smart spaces (in some IoT environments), i.e., can be transformed to smart spaces middleware.

Semantic-oriented IoT middleware is based on technologies of the Semantic Web, e.g., RDF, OWL, and SPARQL. The technologies are used for uniform data representation and processing, which enable data exchanging of heterogeneous devices [26]). Various information space solutions are considered. In particular, a point in such a space is represented as an n-tuple. In the basic case, each tuple corresponds to a triple ($n = 3$ components), following the RDF and OWL representation model. The SPARQL query language is used to formulate data update and retrieve queries from the shared information storage. The Smart-M3 platform is one example of this type of middleware [12]. Semantic-oriented IoT middleware integrates Web 3.0 and evolve to Web 4.0, embodying the idea of symbiotic Web—human mind and machines interact in symbiosis [27], [28].

The existing semantic-oriented IoT middleware solutions can be divided into the following groups based on their design approaches [24], [29]: event-based, service-oriented, VM-based, agent-based, tuple-spaces, database-oriented, application-specific, and data-driven. Typically, mixed approaches are used in existing middleware implementations. In particular, the database-oriented and agent-based approaches can be combined to support indirect communication of agents with each other. In contrast to the direct communication, when an agent has to discover another agent and explicitly connect to exchange data or commands, the indirect communication uses the shared view model (e.g., blackboard) implemented in some database (e.g., local device storage, nearby storage in the IoT environment, cloud storage).

Commercial IoT middleware follow the three main centralized design approaches to deploying smart spaces [30]: 1) one cloud for all, 2) local cloud, and 3) local cloud with shared global functionality. The first approach is suitable for organizing the cooperation of many distributed devices. The approach is provided by IBM, Microsoft, and Intel, since a large number of software and hardware resources are required to maintain the cloud environment. The second approach deploys a local shared storage for devices in a physical spatial-restricted area. The third approach is a combination of the previous ones. Decentralized design approaches are also possible, e.g., based on peer-to-peer methods [31].

Based on the semantic-oriented properties of existing IoT middleware, we conclude that smart spaces middleware should provide data integration, easy knowledge exchange, and intelligent reasoning over the shared information as well as interoperability and integration within a heterogeneous IoT environment of ubiquitously interconnected objects and systems. This kind of requirements to smart spaces middleware can be classified onto three groups: 1) functional requirements, 2) non-functional requirements, and 3) application development requirements. Particular requirements for each of these groups were considered in the publications above and we discuss the requirements further in this section. The summary view is shown in Fig. 1.

C. Functional Requirements

Functional requirements cover the functions provided by smart spaces middleware. The requirements are related to the Semantic Web standards and technologies and their software implementation constraints. Straightforward transfer of global-scale Semantic Web solutions for localized (edge-centric) IoT environments with resource-constrained devices is inappropriate for practical use. The functional requirements reflect such constraints in smart spaces middleware as follows.

Machine-readable logic: Smart space middleware follows the W3C standard for encoding semantics of stored data as class and property axioms of some description logic in RDF/XML, which is machine-readable. The encoding functions use OWL built upon RDF, RDF Schema, and XML Schema. The machine-readable logic provides a logical system specifying instances, their relationships (properties or predicates), and the sets (classes) to which instances or relations belong.

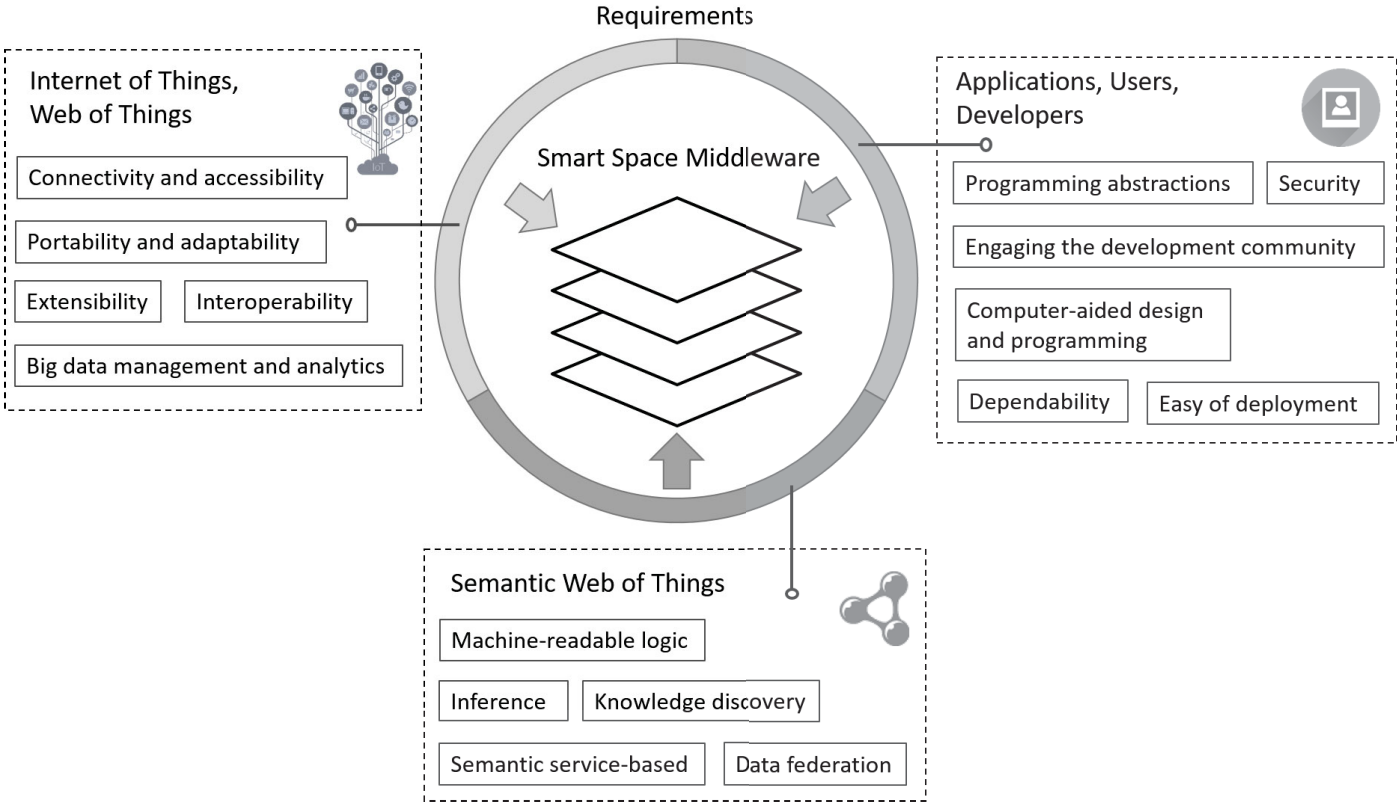

Fig. 1. Smart spaces middleware requirements

Inference: Semantic inference supports discovering new relationships based on the stored semantics and deduction rules. The inference functions use the graph-oriented RDF properties, reducing the problem to connectivity analysis among the basic facts.

Knowledge discovery: In general, knowledge discovery is defined as "the non-trivial extraction of implicit, unknown, and potentially useful information from the data" [32]. This process is the key component of enhanced information retrieval in semantic databases. A smart space is an information collection about the IoT environment. The knowledge discovery functions are performed over this dynamic and heterogeneous collection in order to construct advanced services.

Data federation: Integration of data from multiple, disparate sources to form a single, concerted view. Data federation functions enable an interface to access heterogeneous sources, making the distributed data sources appear as if they are in a single local database. The key challenge is high data dynamicity and heterogeneity in IoT environments, even in localized (edge-centric) environments.

Big data management and analytics: Many Semantic Web solutions have the exponential time complexity, i.e. of poor scalability and low performance when size parameters of data are growing. Functions for big data management and analytics provide the ability to effectively filter, aggregate, and collect data in or close to real-time mode. The data are coming from a wide variety of devices and other sources. Furthermore, the functions support real-time data mining and assistance in decision-making.

D. Non-Functional Requirements

A non-functional requirement describes constraints on the smart spaces middleware as a whole system or for a particular function. The non-functional requirements are mainly inherited from the corresponding requirements for IoT middleware.

Extensibility: Middleware architecture is composite, e.g., based on plug-ins or modules. This property offers high extensibility when new functions are added. The composite architecture supports inclusion/exclusion of certain plug-ins or modules in compilation time or in runtime. n particular, the middleware functionality can be customized for a given IoT environment and other prerequisites.

Connectivity and accessibility: The property of ubiquitous computing, when the middleware keeps regular connectivity for any participant to access appropriate resources in the IoT environment. Moreover, the middleware supports participants with ability to discover resources and each other.

Dependability: Smart space middleware should remain operational during an application process, even in the presence of failures. The dependability of a middleware helps in achieving application dependability. Every component in a middleware should be dependable to achieve overall dependability, which includes devices, communication, technologies, data, and implementation of middleware layers.

Interoperability: Middleware should support simultaneous operation of heterogeneous applications (each constructs services) when information is exchanged between applica-

tions and their services. The semantic interoperability means transparent and easy information exchange even if the set of participants is dynamically changed (e.g., new devices join the IoT environment). To enable the semantic interoperability, smart spaces middleware can use ontologies to represent shared data such that the information is interpreted similarly by any participant.

Security: A security mechanism is needed to ensure the three key concepts: integrity, confidentiality and availability. Use of context-awareness may disclose some personal-like information (e.g., phone number, home address, current location). In order to preserve the owner's privacy, the smart spaces middleware support organizing, controlling, and delimiting the access to information shared in the smart space.

Portability and adaptability: The requirement is important specially due to the expansion of various computing devices (PC, tablets, smartphones, routers, etc.), including Linux and Windows based systems, as well as embedded systems. Such devices are used to host some middleware components. The middleware should provide certain independence level from network protocol, programming language, and operating system. The adaptability supports evolution of hosted middleware component in respect to changes in the IoT environment.

Easy of deployment: Middleware architecture, which defines the components hosted on various devices, should support easy elaboration, evolution, and understanding by third-party developers. The multi-device components deployment is not oriented to expert knowledge. A common user can deploy the components, with no complicated installation and setup procedures.

Engaging the development community: Community engaging is important since no single company or developer can create and support all smart space middleware components. The new functionality can be implemented by third-party developers and preferably as open source.

E. Application Development Requirements

The use of smart spaces middleware for application development needs appropriate programming models and software development tools On the one hand, the ever-growing number of smart spaces applications impose requirements on simplifying the application development and software maintenance. On the other hand, the Semantic Web technologies impose unified standards for application development.

Programming abstractions: Providing an API for application developers is an important requirement for any smart space middleware. The required support is implemented as high-level programming interfaces for application developers. The programming abstraction (e.g., the publish/subscribe model) and the interface type that defines the style of the programming need to be considered when defining an API.

Semantic service-based system: The Semantic Web introduced the concept of semantic web services, which can be used in smart spaces. A smart space application aims at constructing services with uniquely described semantics, accessible through the Internet, and suitable for automated discovery, composition, and proactive initiation.

Computer-aided design and programming: Smart spaces middleware should provide application development approaches with heavy developer involvement and extensive use of tools of computer-aided design and computer-aided programming that support application prototyping. Computer-aided design tools are being used to automate the work of creating and maintaining the various ontological and graphic representations of smart space application systems design. Computer-aided programming tools are being used to simplify the task of programming of smart space applications providing an integrated developed environment and automated program-code generation tools.

Summary

The discussed smart spaces middleware requirements were summarized in Figure 1 above. The requirements systemize the properties that a particular smart spaces middleware can have. Middleware development uses this requirements system to map particular properties to generic-form requirements. The system also serves as a basis for determining to which extend an existing IoT middleware can be used as smart spaces middleware.

III. Middleware Overview

The requirements discussed in the previous section are essentially inherited from the rich area of IoT middleware. This section presents an overview of particular existing middleware solutions that can be used (directly or by enhancing) for the case of smart spaces. For each considered solution the overview focuses on the achieved level of requirements satisfaction. As a result, we identify and systemize possible design approaches and characteristics for smart spaces middleware.

A. Semantic Middleware for Networked Embedded Systems

LinkSmart [33]) is an open source platform for developing IoT applications in various domain. The platform provides the framework and the service infrastructure for creation of distributed IoT applications, combining the service-oriented architecture, peer-to-peer networking, and semantic web services technologies.

The semantic layer of the platform is based on the RDF/OWL knowledge representation formalisms and related reasoning mechanisms. In order to create the basis for decision making and data federation at the layers of the middleware architecture, LinkSmart includes data fusion, situation patterns recognition, and complex event processing mechanisms.

The modular architecture provides software components for building local smart environments consisting of a number of devices, applications and services, which can be discovered and communicated with using the publish/subscribe or request/response messaging. The middleware also focused on connecting remote environments over the Internet. LinkSmart provides device/protocol specific components that integrate various low-level (e.g., ZigBee, Modbus) and high-level protocols (e.g., like HTTP, MQTT) to enable interoperability at the physical and application layers respectively. The semantic interoperability is accomplished by combining the use of ontologies with semantic web services. Distributed security and social trust components at the security layer offer secure

and trustworthy communication within devices, applications and services. Regarding the easy of deployment requirement, the application layer contains customizable user applications that may include modules for workflow management, user interface, custom logic and configuration details.

LinkSmart provides a mechanism for wrapping standard API interfaces of services, sensors, and various physical devices with a defined web service extension. LinkSmart services uses a semantic model for service composition, which covers the service execution preconditions and post-conditions, the models for orchestration of services into processes or grounding the services to a concrete implementation.

B. Open Source IoT in Cloud

The OpenIoT project [34] provides an open source cloud-based middleware platform enabling the semantic unification of diverse IoT applications.

OpenIoT exploits the Linked Data concept realized through the Linked Stream Middleware (LSM), which has been re-designed with cloud interfaces. The RDF triple store is deployed by LSM for encoding semantics of sensors data and supporting a description logic. LSM provides a Linked Data query processor that supports the SPARQL 1.1 standard. SPARQL queries, which are continuously executed as new data arrive from different types of sensors, output the data in a unified format, providing data federation. However, besides continuous queries, there is no real semantic reasoning possible.

The architecture of LSM allows inclusion/exclusion of a wide range of wrappers at runtime. Wrappers can collect data from sensors through serial port communication, UDP connections, HTTP requests, and etc. Each sensor is available for discovery and querying from any layer of the OpenIoT architecture, a sensor is registered and corresponding RDF triples are stored in LSM. OpenIoT enables he semantic interoperability of IoT services through a software infrastructure for collecting and semantically annotating data from virtually any sensor available. The privacy & security module is used to perform user management, authentication, and authorization.

LSM provides a wide range of interfaces for accessing sensor readings such as physical connections, APIs, and database connections. OpenIoT provides an integrated environment for building/deploying and managing IoT applications that essentially accelerates the process of developing IoT applications.

Ali et al. ([35]) present a conceptual architecture of IoT-enabled communication systems, which are built upon existing frameworks for semantic data acquisition, and tools to enable continuous processing, discovery and federation of dynamic data sources based on Linked Data. The proposed middleware extends the functionalities of the OpenIoT platform by introducing HTTP Listener wrapper for capturing streaming data, and semantic querying and reasoning layer, which allows IoT-enabled communication systems to include semantically annotated data streams produced by sensors as an additional source information. The main advantage of the middleware over the OpenIoT project is the introduction of Stream Processing and Reasoning Layer. The Stream Processing component enables to continuously query sensor data streams and detect events in realtime. The Stream Reasoning component contains application logic to make smart decisions customised to the particular requirements and context of the user.

C. Integrated Semantics Service Platform

Ryu et al. ([36]) proposed an integrated semantic service platform (ISSP) to support ontological models in various IoT-based service domains.

In order to express the explicit metadata and machine logic, the platform uses linguistic techniques of the Semantic Web, such as XML, RDF, and OWL. The web-based tool of the ISSP provides support for the semantic web rule language, which is used for reasoning based on added ontologies in the IoT-based service integration ontology. Through SPARQL, the semantic discoverer, one of the five main components of the ISSP, performs semantic discoveries over an ontology for a specific service domain. The other component, the service connector, generates commands for collaborating with external IoT platforms using the individuals received from the semantic discoverer enabling the data federation. The service connector uses HTTP verbs according to the open APIs.

The ISSP provides open APIs to monitor and control IoT devices through RESTful interfaces. The ISSP handles and stores various service domain knowledge in a smart city using ontologies and then provides semantic interoperability between different service domains (e.g., heating, ventilation, and air conditioning) based on the integrated knowledge. The ISSP is developed using Java on the web application server. The main components run on a web browser, so as to support various devices, such as a tablet, smartphone and personal computer. Developers or administrators in each service domain input values using the web-based tool to create an ontology based on their service domain knowledge.

ISSP resolves three main problems for providing semantic services via applying the semantic technologies: (1) integrated semantic discovery; (2) dynamic semantic representation; (3) semantic data repository.

D. Semantic Middleware with Big Data Storage and Analytics

SOFIA2 [37] is a semantic middleware with Big Data storage and analytics capacities, which allows the exchange of information from the real world between smart applications to build composed services. The middleware is based on mixed design approach combining such approaches as event-based, service-oriented, agent-based, and database-oriented.

SOFIA2 proposes using JSON o exchange information and to define the ontologies of middleware application systems. There is a lightweight syntax to serialize Linked Data in JSON called JSON-LD. In this way developers can generate or consume Linked Data, an RDF graph, or an RDF Dataset in a JSON syntax specifying instances, their relationships (properties or predicates), and the sets (classes). This middleware also offers advanced functionality like real-time events subscription and rule definition and execution. However, there is no real semantic reasoning possible, besides the ontology governance rules ensuring that all the information in the middleware has a homogenous structure among the different applications.

As for non-functional requirements, SOFIA2 meets most of them. Actually, SOFIA2 is a multi-language and multi-protocol middleware that provides a semantic interoperability of multiple heterogeneous devices and systems. Extensibility requirement is implemented in a middleware architecture through a mechanism of plug-ins. The middleware uses several security mechanisms that ensure authorization, authentication, consistency and as a whole integral protection of the data. It also has a third-party API for many languages, such as Javascript, Arduino, Android, C and Python, among others.

E. Semantically Interoperable Federation of IoT Experimentation Facilities

FIESTA-IoT project [38]) provides IoT middleware infrastructure that adapts and federates existing IoT platforms and testbeds.

The data of platforms and testbeds are adapted to a common FIESTA-IoT ontology (i.e. compliance to common semantics). Different RDF representation formats (i.e., RDF/XML, JSON-LD, Turtle) are supported. Furthermore, the proposed semantic-based platform enables the management of triple store databases and the execution of inference and reasoning engines.' Regarding the data federation requirement, the common FIESTA-IoT ontology enables makes it possible to seamlessly deal with data from different sources. Moreover, the platform provides a single experimentation-as-a-service application program interface for accessing IoT data resources independently of their source IoT platform/testbed.

The central component of the FIESTA-IoT meta-platform is a directory service. This directory will enable the dynamic discovery and use of resources from all the interconnected testbeds. The FIESTA-IoT ontology is the baseline for the semantic interoperability of the heterogeneous testbeds and IoT platforms. The FIESTA-IoT architecture ensures secure access to testbed resources by authenticated and authorized users. Although the middleware provides the API for registering, managing and querying resources, the deployment of platform components requires expert knowledge and support, making it complicated to install, setup, and evolve by third-party developers.

However, the involvement of third-parties plays an important instrumental role for the large scale validation of the FIESTA-IoT experimental infrastructure.

F. Reactive Middleware for Sensor Data

MASSIF [29] is a reactive data-driven middleware for the semantic annotation of and reasoning on raw sensor data. The middleware allows the development and deployment of services that can operate on a subset of data, improving reasoning efficiency. Each of these services operates with its own context model to retrieve high-level knowledge. It allows semantic annotation of IoT data and the high-level coordination between semantic IoT services.

A context model is internally represented as an ontology using the OWL API. Data are represented as a set of OWL axioms, describing the data semantically. Through the use of semantic reasoning and description logics over this data representation, inference requirement is met by the extraction of intelligent high-level conclusions and the execution of intelligent decisions.

MASSIF has a plug-in architecture that allows loosely-coupled modular services, which enable extensibility and scalability. Reactive and real-time data processing ensures connectivity and accessibility through detection and immediate reaction to events. In order to meet the dependability requirement, the middleware uses a specialized backup system to provide to minimize the data loss upon failure and intelligent caching to match similar events without the need to reason. The middleware also supports communication and collaboration between the different components. The different components publish their data on the Semantic Communication Bus (SCB) in the form of OWL axioms. Each component can subscribe to the SCB by passing a filter rule in the form of an OWL class expression.

MASSIF includes the API-components, which can semantically annotate the data and services, which process the semantic data to retrieve high-level knowledge.

G. Adaptive IoT and WoT Convergence Platform

The concept platform [39] provides global intercompatibility to help users to easily communicate with things by connecting through the webs.

The platform provides semantic-based thing storage (based on RDF) and inference module, things retrieval module, and Ubiquitous Process Management (UPM)-based thing dynamic collaboration modules. Services and applications may be provided with a semantic search result through the API to handle the SPARQL query.

The plug-in-based architecture of the Web and networking layer includes the HTTP communication manager, things and devices resource manager, internal message networking and monitor, and the security functionality and things metadata repository. UPM-based module module supports collaborative monitoring thing resource identification, services mashup with smart devices, thing-to-thing communication, and thing monitoring capabilities for collaboration. The middleware enables semantic interoperability between separate thing data and improves analysis by optimizing the situational awareness. The users have access to things through the Web and perform a search for the thing resource information acquisition and control functions.

Services and applications may be provided with a useful semantic search result through the API to handle the SPARQL query.

H. SPARQL Event Processing Architecture

Work [22] proposes a decentralized Web-based software architecture, named SEPA (SPARQL Event Processing Architecture), underlying the open interoperability platform for smart space applications.

SEPA is used the Semantic Web technologies enabling machines to generate, publish and consume new information autonomously based on computable logic described by RDF data model. SEPA is built on top of the SPARQL 1.1 Protocol and provides general semantic reasoning and data integration

techniques based on this protocol. SEPA stores information in a Big Data RDF store to be later analyzed.

The SEPA Framework provides a development environment and offers developers a modular and extensible solution. Connectivity and accessibility are achieved by publish-subscribe mechanism. Connectivity and accessibility are achieved by publish-subscribe mechanism where publishers and subscribers use SPARQL 1.1 Queries communicating through HTTP/HTTPS and WS/WSS protocols. SEPA aims to provide a minimum level of dependability through the Dependability Manager that implements the client credentials grant type and uses JSON Web Tokens. Semantic interoperability is granted by RDF/RDFS/OWL ontologies offering an API in several programming languages and at different levels of abstraction. The SPARQL 1.1 Secure Event Protocol is proposed to support subscriptions and secure communications. Secure requests are authorized through JSON Web Token and sent over TLS connections.

I. Knowledge-Aware and Service-Oriented Middleware

KASOM [40] is a knowledge-aware and service-oriented middleware for pervasive embedded networks.

The knowledge bases of KASOM are intended to be structured as ontology over a RDF/OWL 2. Such ontology is parsed in order to create a WSDL 2.0 document. KASOM provides the reasoning and inferencing engine, which gets the contextual information from the environment using the approach based on three phases in charge of managing the contextual information: discovery, acquisition, and reasoning.

The middleware provides the Knowledge Management services enabling an effective way of managing the amount of information generated in a so heterogeneous network as the WSAN. The Knowledge Management services are founded on complex reasoning mechanisms and protocols based on the WSAN's Contextual Model. KASOM showed a good reliability-delay balance during the reliability and performance tests when dealing with event-based and on-demand services. Security Service provides procedures to manage various major security issues (e.g. permissions for accessing to services). KASOM enables an execution platform over which a number of different in-network Perceptual Reasoning Agents (PRAs) can be run. When developing PRAs the developer has to keep in mind the two major features which characterize PRA: independence and reusability. PRA was designed following a lightweight agent philosophy, which facilitates a rapid design, development and deployment.

KASOM provides to developers a common API to register PRAs in the system hiding the details of underlying software as well as the allocation and management of resources.

J. Semantic-Aware Policy Framework

SeCoMan (Semantic Web-based Context Management) is a framework for developing context-aware smart applications preserving the users' privacy in a semantic-oriented IoT vision [41].

The middleware makes use of OWL 2 for encoding semantics of stored data. In order to infer new knowledge, the Reasoner module of SeCoMan receives ontological models generated by the Interpreter module and returns inferred models with new knowledge by using semantic rules.

The Plug-in layer provides extensibility to SeCoMan. This layer is composed of different plug-ins that interact with the Middleware module, which communicate with sensors or other devices to receive context information, and with the Location Systems module to obtain information about the environment. SeCoMan defines a collection of ontologies to shape the space and context information to provide the semantic interoperability. The framework dynamically controls users' privacy, their authorization to stay in certain environments, and generates new knowledge by using semantic rules, which form policies (Operational policies, Authorization policies, and Location policies).

SeCoMan makes only SPARQL queries in order to obtain the space and context information desired by the users.

K. Semantic Agent-Based Service Middleware

Liu et al. [42] proposed an agent-based, service-oriented middleware towards semantic service enablement in IoT applications.

In order to describe the semantics of stored data the middleware provides a general service ontology based on the XSD design. This ontology is consisted of four properties with range classes: output, capability, deployment and resource. This representation enables the standardized, structured data to instantiate the service ontology. Each agent in the middleware abides by a Belief-Desire-Intention (BDI) model. Agents activate the decision model by executing the plan and actions to reach its goals using such opportunities of semantic representation as inference and knowledge discovery.

This middleware provides a semantic service representation model to support interoperability between heterogeneous M2M services, applications, and devices.

The M2M applications layer of the middleware provides an interface with various M2M applications. As part of the middleware design an efficient semantic service discovery and matching approach for the service combination process are presented. The service model is based on such service design patterns as OWL-S, SAWSDL, and SWSO. This model also follows the bottom-up data flow and the REST operation style.

L. Semantic Information Broker

CuteSIB [20] is the open-source software implementation for such a central element of an M3 smart space as Semantic Information Broker (SIB). CuteSIB is follows the mixed design approach based on event-based, service-oriented, agent-based, and tuple-spaces approaches. The implementation of CuteSIB is based on the Qt framework in order to support a wide spectrum of Qt-based devices.

The functional requirements are implemented through the use of W3C standard for encoding semantics of stored data and OWL built upon RDF, RDF Schema, and XML Schema.

The plug-ins based architecture achieves higher extensibility due to the modular approach. The CuteSIB implementation consists of five modules: (1) access points, (2) protocol

TABLE I. COMPARISON OF THE EXISTING MIDDLEWARE SOLUTIONS FOR SMART SPACES

Middleware	Requirements															
	Functional					Non-functional								App. development		
	MRL	Inf	KD	DF	BDM-A	Ext	C-A	Dep	Int	Sec	P-A	ED	EDC	PA	SSB	CAD-P
LinkSmart	✓	✓	✓	✓	n/a	✓	✓	n/a	✓	✓	n/a	✓	✓	✓	✓	n/a
OpenIoT	✓	n/a	n/a	✓	n/a	✓	✓	n/a	✓	✓	n/a	✓	✓	✓	n/a	✓
Ali et al.	✓	✓	✓	✓	n/a	✓	✓	n/a	✓	✓	n/a	✓	✓	✓	n/a	✓
Ryu et al.	✓	✓	✓	✓	n/a	n/a	✓	n/a	✓	n/a	✓	✓	n/a	✓	✓	n/a
SOFIA2	✓	✓	✓	✓	✓	✓	✓	n/a	✓	✓	✓	n/a	✓	✓	n/a	n/a
FIESTA	✓	✓	n/a	✓	n/a	n/a	✓	n/a	✓	✓	n/a	n/a	✓	✓	n/a	n/a
MASSIF	✓	✓	✓	n/a	n/a	✓	✓	✓	✓	n/a	n/a	n/a	n/a	✓	✓	n/a
Yu et al.	✓	✓	✓	✓	n/a	✓	✓	n/a	✓	✓	n/a	✓	n/a	✓	n/a	n/a
SEPA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	n/a	✓	✓	✓	n/a	n/a
KASOM	✓	✓	✓	✓	n/a	n/a	✓	✓	✓	✓	n/a	✓	n/a	✓	n/a	n/a
SeCoMan	✓	✓	✓	✓	n/a	✓	✓	n/a	✓	✓	n/a	n/a	n/a	n/a	n/a	n/a
Liu et al.	✓	✓	✓	n/a	n/a	n/a	✓	n/a	✓	n/a	n/a	n/a	n/a	✓	✓	n/a
CuteSIB	✓	n/a	✓	✓	n/a	✓	✓	✓	✓	n/a	✓	n/a	✓	✓	n/a	n/a
Legend: (n/a) No info was found (✓) Req is met	(MRL) Machine-readable logic (Inf) Inference (KD) Knowledge discovery (DF) Data federation (BDM-A) Big data management and analytics					(C-A) Connectivity and accessibility (P-A) Portability and adaptability (ED) Easy of deployment (EDC) Engaging the development community (Ext) Extensibility (Dep) Dependability (Int) Interoperability (Sec) Security									(P-A) Programming abstractions (SSB) Semantic service-based (CAD-P) Computer--aided design and programming		

handlers, (3) SIB core, (4) operation handlers, and (5) triple-store. The access points bind to particular network (transport) protocol, receive agents requests and send responses. Operation handlers implement operations that are needed for protocol logic. For example, the main M3 protocol is SSAP. Base implementation of SSAP is based on TCP and XML. CuteSIB provides the fault tolerance mechanisms with restart/reconnect and operation control functionality as well as uses special services for persistent storage of critical data. CuteSIB stores data with RDF model. Data can be easily integrated with data from public SPARQL-endpoints (e.g., such as DBpedia) and other RDF storages. Portability and adaptability of the CuteSIB base layer are achieved on implementation stage by using C/C++ cross-platform languages and Qt framework.

The CuteSIB middleware provides the development libraries (e.g., SmartSlog DPI, CKPI) for M3 agents and services.

Summary

Table I provides a summary of the overviewed middleware solutions in respect to the functional, non-functional, and application development requirements. The "n/a" wildcard indicates that no particular information was found about the ability of the middleware to meet the requirement or the requirement is not supported.

In general, service-oriented, agent-based, and data-based design approaches address more smart space middleware requirements than others. Middleware based on these design approaches are LinkSmart, SOFIA2, and SEPA. Furthermore, these approaches support such middleware characteristics as indirect interaction of agents, multi-layer infrastructure-based, and resource-constrained that stand out from the others in the meeting requirements. These approaches and characteristics are related to providing the multi-layer model of the middleware software infrastructure in next section.

Although the existing semantic-oriented middleware solutions address many requirements associated with middleware in smart spaces, some requirements and related research issues remain relatively unexplored, such as big data management and analytics, semantic services, and computer-aided design and programming. Two of them, semantic services and computer-aided design and programming, are associated with the application developments requirements, which, with the ever-increasing number of smart space applications, are becoming more and more relevant compared to other requirements.

IV. CONCLUSION

This paper considered the development problem for smart spaces middleware. The middleware is used for smart space deployment in IoT environment to support construction of advanced services using available resources. We systemized the requirements applicable to development of smart spaces middleware for IoT environments. The requirement system was then mapped to the particular existing middleware solutions. The presented overview of the middleware solution was focused on characterization to which extent the requirements are satisfied. The existing solutions are compared and basic approaches are identified to this type of middleware development.

ACKNOWLEDGMENT

This research is implemented in Petrozavodsk State University (PetrSU) with financial support by the Ministry of Science and Higher Education of Russia within Agreement no. 075-11-2019-088 of 20.12.2019 on the topic "Creating the high-tech production of mobile microprocessor computing modules based on SiP and PoP technology for smart data collection, mining, and interaction with surrounding sources".

REFERENCES

[1] D. Korzun, S. Balandin, and A. Gurtov, "Deployment of Smart Spaces in Internet of Things: Overview of the design challenges," in *Internet of Things, Smart Spaces, and Next Generation Networking*, ser. Lecture Notes in Computer Science, S. Balandin, S. Andreev, and Y. Koucheryavy, Eds., vol. 8121. Springer, Aug. 2013, pp. 48–59.

[2] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, Jan. 2010.

[3] D. Korzun, E. Balandina, A. Kashevnik, S. Balandin, and F. Viola, *Ambient Intelligence Services in IoT Environments: Emerging Research and Opportunities*. IGI Global, 2019.

[4] A. Gyrard, M. Serrano, and G. A. Atemezing, "Semantic web methodologies, best practices and ontology engineering applied to internet of things," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 412–417.

[5] D. G. Korzun, I. V. Galov, and A. A. Lomov, "Smart space deployment in wireless and mobile settings of the Internet of Things," in *Proc. IEEE 3rd International Symposium on Wireless Systems IDAACS:SWS*, 2016, pp. 86–91.

[6] E. T. Chen, "The internet of things: Opportunities, issues, and challenges," in *The Internet of Things in the Modern Business Environment*, I. Lee, Ed. IGI Global, 2017, pp. 167–187.

[7] V. Gazis, "A survey of standards for machine-to-machine and the internet of things," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 482–511, 2017.

[8] L. Atzori, A. Iera, and G. Morabito, "From "smart objects" to "social objects": The next evolutionary step of the internet of things," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 97–105, Jan. 2014.

[9] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, *From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices*. Springer Berlin Heidelberg, 2011, pp. 97–129.

[10] J. Heuer, J. Hund, and O. Pfaff, "Toward the web of things: Applying web technologies to the physical world," *Computer*, vol. 48, no. 5, pp. 34–42, 2015.

[11] A. J. Jara, A. C. Olivieri, Y. Bocchi, M. Jung, W. Kastner, and A. F. Skarmeta, "Semantic web of things: An analysis of the application semantics for the iot moving towards the iot convergence," *Int. J. Web Grid Serv.*, vol. 10, no. 2/3, pp. 244–272, Apr. 2014.

[12] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *Proc. IEEE Symp. Computers and Communications (ISCC'10)*. IEEE Computer Society, Jun. 2010, pp. 1041–1046.

[13] A. D'Elia, F. Viola, L. Roffia, P. Azzoni, and T. S. Cinotti, "Enabling interoperability in the internet of things: A osgi semantic information broker implementation," *Int. J. Semant. Web Inf. Syst.*, vol. 13, no. 1, pp. 134–154, Jan. 2017.

[14] S. A. Marchenkov, A. S. Vdovenko, and D. G. Korzun, "Enhancing the opportunities of collaborative work in an intelligent room using e-tourism services," *Trudy SPIIRAN*, vol. 50, pp. 165–189, 2017.

[15] D. Korzun, I. Galov, A. Kashevnik, and S. Balandin, "Virtual shared workspace for smart spaces and M3-based case study," in *Proc. 15th Conf. of Open Innovations Association FRUCT*, S. Balandin and U. Trifonova, Eds. ITMO University, Apr 2014, pp. 60–68.

[16] Y. V. Zavyalova, D. G. Korzun, A. Y. Meigal, and A. V. Borodin, "Towards the development of smart spaces-based socio-cyber-medicine systems," *International Journal of Embedded and Real-Time Communication Systems (IJERTCS). Special Issue on Big Data Analytics and Intelligent Environments in Internet of Things*, vol. 8, no. 1, pp. 45–63, 2017.

[17] D. Korzun, A. Borodin, I. Timofeev, I. Paramonov, and S. Balandin, "Digital assistance services for emergency situations in personalized mobile healthcare: Smart space based approach," in *Proc. 2015 Int'l Conf. on Biomedical Engineering and Computational Technologies (SIBIRCON/SibMedInfo)*. IEEE, Oct. 2015, pp. 62–67.

[18] A. Smirnov, A. Kashevnik, A. Ponomarev, N. Teslya, M. Shchekotov, and S. Balandin, "Smart space-based tourist recommendation system," in *Proc. 14th Int'l Conf. Next Generation Wired/Wireless Networking and 7th Conf. on Internet of Things and Smart Spaces (NEW2AN/ruSMART 2014), LNCS 8638*, S. Balandin, S. Andreev, and Y. Koucheryavy, Eds. Springer, Aug. 2014, pp. 40–51.

[19] D. Korzun, S. Marchenkov, A. Vdovenko, and O. Petrina, "A semantic approach to designing information services for smart museums," *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 7, no. 2, pp. 15–34, 2016.

[20] I. Galov, A. Lomov, and D. Korzun, "Design of semantic information broker for localized computing environments in the Internet of Things," in *Proc. 17th Conf. of Open Innovations Association FRUCT*. IEEE, Apr. 2015, pp. 36–43.

[21] F. Viola, A. D'Elia, D. Korzun, I. Galov, A. Kashevnik, and S. Balandin, "The M3 architecture for smart spaces: Overview of semantic information broker implementations," in *Proc. of the 19th Conference of Open Innovations Association FRUCT*, S. Balandin and T. Tyutina, Eds. IEEE, Nov. 2016, pp. 264–272.

[22] L. Roffia, P. Azzoni, C. Aguzzi, F. Viola, F. Antoniazzi, and T. Salmon Cinotti, "Dynamic linked data: A sparql event processing architecture," *Future Internet*, vol. 10, no. 4, p. 36, Apr 2018.

[23] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta, "Role of middleware for Internet of Things: A study," *International Journal of Computer Science & Engineering Survey (IJCSES)*, vol. 2, no. 3, pp. 94–105, August 2011.

[24] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for Internet of Things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, Feb. 2016.

[25] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[26] M. Murth and E. Kühn, "Knowledge-based coordination with a reliable semantic subscription mechanism," in *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 2009, pp. 1374–1380.

[27] P. Kujur and B. Chhetri, "Evolution of world wide web: Journey from web 1.0 to web 4.0," *International Journal of Computer Science and Technology*, vol. 6, Jan. 2015. [Online]. Available: http://www.ijcst.com/vol61/1/30-Pranay%20Kujur.pdf

[28] A. Y. Meigal, D. G. Korzun, L. I. Gerasimova-Meigal, A. V. Borodin, and Y. V. Zavyalova, "Ambient intelligence at-home laboratory for human everyday life support," *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 10, no. 2, 2019, in Press.

[29] P. Bonte, F. Ongenae, F. De Backere, J. Schaballie, D. Arndt, S. Verstichel, E. Mannens, R. Van de Walle, and F. De Turck, "The MASSIF platform: a modular and semantic platform for the development of flexible IoT services," *Knowledge and Information Systems*, vol. 51, no. 1, pp. 89–126, Apr 2017.

[30] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the Internet of Things," in *Proc. IEEE 20th Conf. on Emerging Technologies & Factory Automation (ETFA 2015)*. IEEE, Sep. 2015, pp. 1–8.

[31] E. Meshkova, J. Riihijärvi, M. Petrova, and P. Mähönen, "A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks," *Comput. Netw.*, vol. 52, no. 11, pp. 2097–2128, Aug. 2008.

[32] S. Kalarani and G. V. Uma, "Integration of semantic web and knowledge discovery for enhanced information retrieval," *International Journal of Computer Applications*, vol. 1, no. 1, pp. 99–103, 2010.

[33] P. Kostelnik, M. Sarnovsky, and K. Furdík, "The semantic middleware for networked embedded systems applied in the internet of things and services domain," *Scalable Computing: Practice and Experience*, vol. 12, no. 3, pp. 307–316, 2011.

[34] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko, L. Skorin-Kapov, and R. Herzog, "Openiot: Open source internet-of-things in the cloud," in *Interoperability and Open-Source Solutions for the Internet of Things*, I. Podnar Žarko, K. Pripužić, and M. Serrano, Eds. Springer International Publishing, 2015, pp. 13–25.

[35] M. I. Ali, N. Ono, M. Kaysar, K. Griffin, and A. Mileo, "A semantic processing framework for iot-enabled communication systems," in *The Semantic Web — ISWC 2015*, M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab, Eds. Springer International Publishing, 2015, pp. 241–258.

[36] M. Ryu, J. Kim, and J. Yun, "Integrated semantics service platform for the internet of things: A case study of a smart office," *Sensors*, vol. 15, pp. 2137–2160, 2015.

[37] "Sofia2," 2018. [Online]. Available: http://sofia2.com

[38] J. Lanza, L. Sanchez, D. Gomez, T. Elsaleh, R. Steinke, and F. Cirillo,

"A proof-of-concept for semantically interoperable federation of iot experimentation facilities," *Sensors*, vol. 16, p. 1060, 2016.

[39] J. Yu, H.-C. Bang, H. Lee, and Y. S. Lee, "Adaptive internet of things and web of things convergence platform for internet of reality services," *The Journal of Supercomputing*, vol. 72, pp. 84–102, Jan. 2016.

[40] I. Corredor, J. F. Martinez, M. S. Familiar, and L. Lopez, "Knowledge-aware and service-oriented middleware for deploying pervasive services," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 562–576, 2012, simulation and Testbeds.

[41] A. Huertas Celdran, F. J. Garcia Clemente, M. Gil Perez, and G. Martinez Perez, "SeCoMan: A semantic-aware policy framework for developing privacy-preserving and context-aware smart applications," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1111–1124, 2016.

[42] M. Liu, Y. Xu, H. Hu, and A.-W. Mohammed, "Semantic agent-based service middleware and simulation for smart cities," *Sensors*, vol. 16, p. 2200, 2016.