Infrastructure Multi-Layer Model for Smart Spaces Middleware Development

Sergey A. Marchenkov
Petrozavodsk State University
(PetrSU) Petrozavodsk, Russia
marchenk@cs.karelia.ru

***Abstract*—The most recent advances in Internet of Things (IoT) are aimed at the development of software platforms defined as middleware. A smart space makes fusion of the physical and information worlds enhancing a IoT environment by enabling information sharing for a variety of local digital devices and global resources from the Internet. The growing number of application smart spaces requires a middleware that provides opportunities for the development of context-aware applications based on multi-agent approach. The paper presents a perspective on how to design a software infrastructure of semantic-oriented middleware for smart spaces deployment in IoT environments. This paper introduces the middleware infrastructure multi-layer model. The paper also involves an analysis of middleware requirements for development and deployment of smart spaces in IoT environments and the implementation of the requirements in the proposed model. As a reference case study the CuteSIB semantic information broker implementation of the Smart-M3 platform is considered.**

I. Introduction

The most recent advances in Internet of Things (IoT) are aimed at the development of software platforms defined as middleware [1], [2]. The key features of a middleware are dedicated to the integration of heterogeneous computing devices and software components, as well as to support the interaction between various services and applications. The ever-growing number of IoT applications and the IoT future directions impose additional requirements on a middleware related to intelligent reasoning and simplifying the application development process.

The smart spaces paradigm integrates technologies from IoT and the Semantic Web concepts, creating a certain class of ubiquitous computing environments, typically associated with a physical spatial-restricted place [3]. These environments make fusion of the physical and information worlds enhancing a IoT environment by enabling information sharing for a variety of local digital devices and global resources from the Internet [4]. The growing number of application smart spaces in many different domains, such as collaborative work [5], [6], mobile healthcare [7], museums and cultural heritage [8], [9], requires a middleware that provides opportunities for the development of context-aware applications based on multi-agent approach.

The semantic-oriented perspective is one among the other perspectives from which IoT can be viewed [10]. A semantic-oriented middleware is in general based on the standards and technologies of the Semantic Web, which provide data integration, an easy knowledge exchange, and intelligent reasoning in smart space. Furthermore, the Semantic Web technologies resolve the problems of interoperability and integration within heterogeneous world of ubiquitously interconnected objects and systems.

Most of the middleware requirements cover its software infrastructure as a whole. A software infrastructure is deployed in IoT computing environments including the host devices with middleware, software components of an application, network equipment, and system software, which provide operation and network communication. The design of software infrastructure is the important step in creating and further maintenance of applications for an IoT environment.

The paper presents a perspective on how to design a software infrastructure of semantic-oriented middleware for smart spaces deployment in IoT environments. One of the contributions of this paper is the middleware infrastructure multi-layer model. This model serves as a means for obtaining detailed information about the main layers of semantic-oriented middleware infrastructure. The model is the initial step in the design of such middleware and hence simplifies the middleware development process. The purpose and structural composition of each layer as well as the interconnectivity between them are considered. The paper also involves an analysis of middleware requirements for development and deployment of smart spaces in IoT environments and the implementation of the requirements in the proposed model. As a reference case study the CuteSIB semantic information broker implementation of the Smart-M3 platform is considered. The proposed solutions have a generic character and they can serve for designing such a class of middleware for smart spaces.

The rest of the paper is organized as follows. Section II proposes the infrastructure multi-layer model of semantic-oriented middleware for development and deployment of smart spaces in IoT environments. Section III describes the middleware requirements in accordance with their implementation in the proposed multi-layer model. Section IV considers the CuteSIB semantic information broker as a reference case study. Section V concludes the paper.

II. Infrastructure multi-layer model

The smart spaces paradigm aims at constructing software infrastructure of semantic-oriented middleware that follow the IoT and Semantic Web visions [3]: (1) digital convergence among many surrounding heterogeneous physical/digital objects, (2) services are defined in the context of the interaction of multiple participants (objects and humans) in information sharing environment (smart space), and (3) services are accessible in a uniform semantic way. Semantic-oriented middleware provides opportunities for the development of service-oriented application systems based on multi-agent approach.

As a rule in such systems, the main element of information sharing environment (IoT environment) is a semantic information broker that provides shared knowledge base and organizes information-driven interaction and supports a variety of semantic interoperable access primitives, including ontology-oriented ones [11], [4]. Access primitives provide the opportunity for the interaction of heterogeneous objects based on the processing and analysis of shared information represented by ontologies. Based on ontologically-oriented primitives, semantic reasoning is supported, which allows to transform the integrated low-level data into high-level knowledge to perform the solutions of the arising problems as well as to ensure the self-organization of the participants [12].

The participants are independent software agents that are run on various computing devices of the system (mobile devices, server computers, desktop computers, single-board computers, wireless network routers, etc). Interaction between such agents is indirect and occurs by a publish/subscribe model through a semantic information broker. The construction of services in smart space is realized as an event-driven and information-driven distributed computing process based on the indirect interaction of software agents [13]. The autonomous activity of software agents in the processes of constructing and delivering services can be performed efficiently through their self-organization.

A smart space service inherits the definition of an IoT service proposed in [14], [15]. An IoT service is defined as a type of service enabling interactions with the physical objects while a smart space service also enables interactions with the digital objects. Thus, a smart space service is an application function of a service-oriented system that is executed between the service and the service provider in order to enable the interaction with physical and digital objects, providing the necessary pieces of information to solve the arising problems and/or providing control actions to such objects.

The software infrastructure of semantic-oriented middleware is responsible for the organization of the processes of constructing and delivering services in the application systems. It is deployed in computing environments including the host devices, network equipment, and system software, which provide operation and network communication. To describe software infrastructure of semantic-oriented middleware the three layers were defined: middleware base, middleware infrastructure, and application systems infrastructure (see Fig. 1).

Middleware base layer contains the main software modules of middleware: shared knowledge base, storage for data or metadata about agents, agent management, agent discovery mechanism and modules that implement protocols. This layer allows agents to register in the smart space and find other agents to communicate with them using protocols supported by the platform. Middleware infrastructure layer contains agents and services to extend functions of middleware base layer and provides a functionality for specific domain or application systems. Application systems infrastructure contains agents

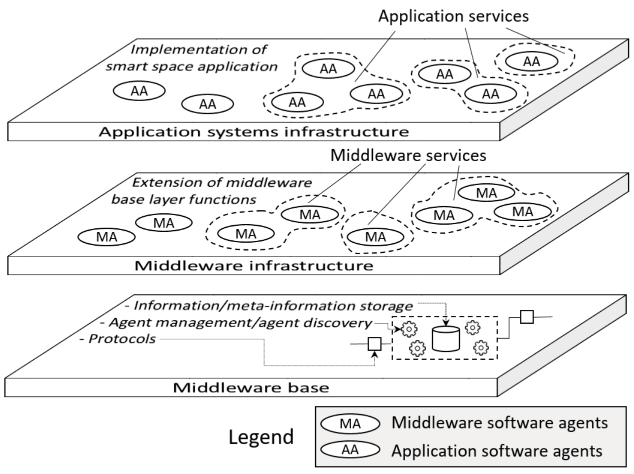
Application services
Implementation of smart space application
AA
Application systems infrastructure
Middleware services
Extension of middleware base layer functions
MA
Middleware infrastructure
- Information/meta-information storage
- Agent management/agent discovery
- Protocols
Middleware base
Legend
MA Middleware software agents
AA Application software agents

Fig. 1. Infrastructure multi-layer model of semantic-oriented middleware

and services that implement smart space applications.

Features of each layer can be used to meet the semantic-oriented middleware requirements. There are several survey works on a middleware for IoT and their requirements such as [16], [10], [1]. Based on these surveys the requirements for the semantic-oriented middleware as well as for other middleware can be categorized as both functional and non-functional. Functional requirements cover the functions provided by semantic platform and non-functional requirements describe the qualities of the platform. The corresponding requirements are summarized in Table I.

III. Middleware requirements

In this section, the above requirements and their implementation with respect to the infrastructure multi-layer model of semantic-oriented middleware are considered. Most of the requirements are described as a result of the relationship of the model layers and are shown by the corresponding schemes, while some of them affect only one of the layers and do not require them.

A. Functional requirements

1) Machine-computable logic: Semantic-oriented middleware for smart spaces make use of the W3C standard for encoding semantics of stored data as class and property axioms in a description logic (machine-computable logic) in RDF/XML. This is done with OWL built upon RDF, RDF Schema, and XML Schema. Machine-computable logic is a logical system specifying instances, their relationships (properties or predicates), and the sets (classes) to which they belong.

2) Inference: Semantic inference can be characterized by discovering new relationships based on the data and some additional information in the form of a set of rules. This process makes use of specific reasoning platform – the resource description framework, or RDF – in order perform the analysis.

3) Knowledge discovery: Knowledge discovery is defined as ”the non-trivial extraction of implicit, unknown, and potentially useful information from the data” [17]. This process is the key component of performing enhanced information retrieval in large semantic databases.

TABLE I. SUMMARY OF REQUIREMENTS FOR SEMANTIC-ORIENTED MIDDLEWARE

Requirement	Description and significance
Functional requirements	
Machine-computable logic	W3C standard for encoding semantics of stored data as class and property axioms in a description logic in RDF/XML. OWL built upon RDF, RDF Schema, and XML Schema.
Inference	Discovering new relationships based on the data and some additional information in the form of a set of rules. RDF – in order perform the analysis.
Knowledge discovery	The non-trivial extraction of implicit, unknown, and potentially useful information from the data. Performing enhanced information retrieval in large semantic databases.
Data federation	The integration of data from multiple, disparate sources into a single, concerted view. The interface to access heterogeneous sources, making remote data appear as if it resides in a single local database.
Non-functional requirements	
Extensibility	Plug-ins or modules when new functions need to be added. Inclusion/exclusion of certain plug-ins or modules in compilation phase or in runtime.
Connectivity and accessibility	Any component/participant of information sharing environment can access it anytime. Middleware should enable various components/participants to automatically discover each other.
Dependability	Middleware should remain operational during an application process, even in the presence of failures. It helps in achieving application dependability.
Semantic interoperability	Support simultaneous work with a set of heterogeneous applications/services without additional effort. Information exchange between the ever-growing and changing set of participants.
Security	Three key concepts: integrity, confidentiality and availability. Context-awareness may disclose personal information. Organizing, controlling, and delimiting access to information stored in smart space.
Portability and adaptability	The expansion of multiple computing devices. Independence from network protocol, programming language, and operating system. Correspondence to changes in IoT environment.
End user usability and simplicity	Middleware architecture should be easy to elaborate, evolve and understand by third-party developers. Complicated installation and setup procedures should be avoided.
Engaging the development community	A middleware should be constantly maintained and expanded. Requirement is provided within a community of developers and researchers.

4) Data federation: Data federation is the integration of data from multiple, disparate sources into a single, concerted view. Semantic-oriented middleware provides the ability to implement the interface to access heterogeneous sources, making remote data appear as if it resides in a single local database.

B. Non-functional requirements

1) Extensibility: One of the main requirement is extensibility of a smart space. A semantic-oriented middleware architecture should be plug-ins or modules based to offer high extensibility when new functions need to be added. The architecture allows inclusion/exclusion of certain plug-ins or modules in compilation phase or in runtime. It affords to customize the middleware functionality for given information sharing environment.

Extensibility can be implemented on middleware base and middleware infrastructure layers. On middleware base layer, extensibility is the use of modules and plug-ins. Extensibility on middleware base layer, in most cases, is oriented to the implementation of various application systems. On middleware infrastructure layer, extensibility is the use of special agents or services. This layer extends the functionality of semantic-oriented middleware using features from middleware base layer. A middleware infrastructure agents/services also can make actions specific for some domains and simplify the implementation of the application system.

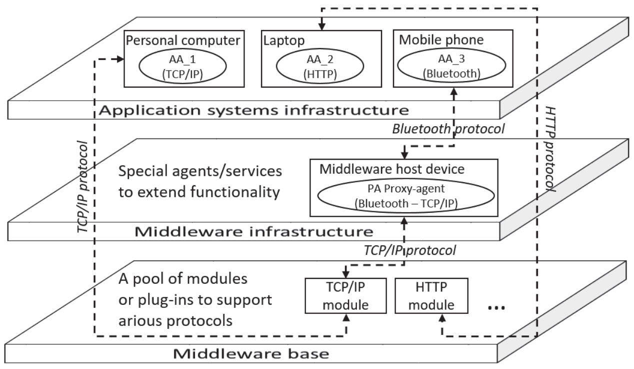

Fig. 2. Implementation of extensibility requirement based on the infrastructure multi-layer model of semantic-oriented middleware

The example of extensibility is shown in Fig. 2. Middleware base layer supports various communication protocols using modules (plug-ins). On middleware infrastructure layer, Proxy-agent can be deployed to provide access to middleware base layer modules based on specific protocol that is not supported on this layer.

2) Connectivity and accessibility: Next requirement is connectivity and accessibility. A semantic-oriented middleware needs connectivity and accessibility, which means that any component/participant of information sharing environment can access it anytime. Moreover, a middleware should enable various components/participants to automatically discover each other.

This requirement is necessary to ensure communication between agents. A semantic-oriented middleware can support agent discovery. One agent can find some other agent and communicate with it using direct or indirect communication mechanism. Indirect communication provides by middleware base layer or by middleware infrastructure layer. Direct communication is carried by agents. In this way, Sa middleware can be used for storing information about agents protocols and other parameters needed for communication.

This requirement is also the main requirement for users who does not know what smart space is. A middleware should support connection of different devices. A middleware and personal software agents should support auto connection and disconnection to/from smart space when user change locations. Some devices (low performance, with different protocol etc.) cannot be connected directly. In this case proxy-agents running on middleware infrastructure (by analogy with extensibility requirement) layer could be used to represent devices as agent in the smart space

3) Dependability: The important requirement is dependability of semantic-oriented middleware. A middleware should remain operational during an application process, even in the presence of failures. Every component in a semantic-oriented middleware should be dependable to achieve overall dependability, which includes devices, communication, technologies, data, and implementation of middleware layers.

Application systems are constructed due to indirect/direct interaction of agents via information storage of middleware base layer. Volatile nature of information sharing environment devices and communications is the cause of failures between agents during processing requests or failures in middleware base layer. The way the operation processing is performed and supported in middleware base layer affects on operation efficiency during agents interaction. If some operations are not properly processed it results in interaction problems and errors in applications.

Dependable interactions between agents and dependable deploy of infrastructure layers in information sharing environment can be ensured: by improving operations processing scheme of middleware base layer, by enforcing the layers of semantic-oriented middleware with fault tolerance mechanisms. Plug-in based architecture of middleware base layer allows to configure operation processing flow by activating/deactivating different operation handlers and extends operations of a protocol to access smart space according to actual needs of an environment. Fault tolerance mechanisms can extend software infrastructure of middleware layers with restart/reconnect and operation control functionality and introduce special services for persistent storage of critical data.

4) Semantic interoperability: A middleware for smart spaces should support simultaneous work with a set of heterogeneous applications/services without additional effort from the application or service developer. Semantic interoperability should allow for information exchange between the ever-growing and changing set of participants (application and services) in IoT environment. To enable semantic interoperability, a semantic-oriented middleware can use ontologies to represent data of multiple domains and their relationships.

The requirement of semantic interoperability can be based on the use of ontologies, see Fig. 3. Agents in smart spaces can use different ontologies to represent data and different rules for generating new knowledge based on stored data. It is not possible for all agents to have special software tools and resources to work with ontologies. In other words, reasoning, mapping, or other actions on stored data can be performed on middleware base layer or middleware infrastructure layer. For example, in order to represent data, application agents AA_1 and AA_2 use ontologies ont_1 and ont_2, respectively. Mapping service on middleware infrastructure layer tracks updates in the storage and translates data based on mappings (ont_1 and ont_2). As a result, these agents can communicate without additional actions to translate data according to their ontologies.

On middleware base layer, a special module (e.g., reasoning/analyzing module) can be added to analyze data storage and infer new knowledge or translate data from one representation to another based on rules. Rules on middleware base layer is applicable for various application systems. On middleware infrastructure layer, special services or agents can perform rule-based actions for a specific application system. For instance, Reasoning service, acting on a middleware information storage, can be utilized to derive higher level abstractions from the sensor data published by Sensor-agent.

5) Security: Security is important to the operation of smart spaces. A security mechanism in semantic-oriented middleware should ensure three key concepts: integrity, confidentiality and availability. Context-awareness in middleware may disclose personal information (e.g., private phone number, home address, the current location of an object or a person).

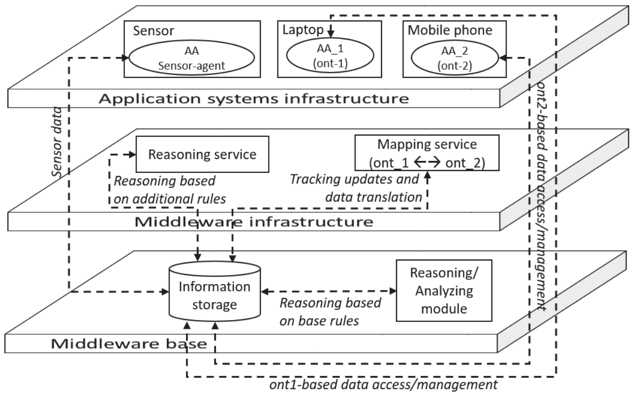

Fig. 3. Implementation of semantic interoperability requirement based on the infrastructure multi-layer model of semantic-oriented middleware

In order to preserve the owner's privacy, a middleware needs to ensure security requirement, thereby organizing, controlling, and delimiting access to information stored in smart space.

The security requirement defines critical features of semantic-oriented middleware that should be implemented to secure smart spaces resources. The security requirement is imposed upon the following smart space security components proposed in [3]: (1) shared resources, (2) smart space access, and (3) communication.

Security of shared resources is ensured by agents of application systems, each of which makes own decisions on its resources, determining which locally available knowledge to place in information storage for sharing with others agents.

Security of smart space access is based on mechanism for controlling access to storage, providing agents of application systems with exclusive access to information [18]. Such access control depends on security meta-information placed in storage. Exclusive access can be on RDF triples level. In this case, meta-information is additional RDF triples that specify which data are protected and which agent is their owner. This mechanism can be implemented on middleware base layer as a separate security module. Security module can restrict agents in access to a given set of triplets by performing reasoning over meta-information.

In order to establish secure communication between agents and middleware, agent identity and cryptographic keys can be implemented with the Host Identity Protocol (HIP) [19], [20]. The HIP exchange authenticates a agent-to-middleware communication session based on cryptographic identifiers and algorithms, thereby ensuring confidentiality and integrity of messages.

6) Portability and adaptability: The portability and adaptability is an important requirement of semantic-oriented middleware, specially due to the expansion of multiple computing devices (PC, tablets, smartphones, routers, etc.). Various computing devices can be hosts for components of middleware, including Linux and Windows based systems, as well as embedded systems. A middleware for smart spaces should provide independence from network protocol, programming language, and operating system. A middleware needs to be adaptive so that it can evolve to correspond to changes in information sharing environment.

On the one hand, operating systems and hardware heterogeneity implies that a middleware itself must be portable across different hardware and operating systems technologies. On the other hand, various dynamic changes in information sharing environment imply that a middleware itself must adapt to altered circumstances, including varying computational and network loads, to keep applications/services operating.

To ensure portability of application systems infrastructure, a desirable middleware must provide developers with an abstraction layer that masks the complex underlying mechanisms and instead exposes a set of useful APIs to middleware layers. This is possible through the use of the standard protocol to access smart space (Smart Space Access Protocol, SSAP), that have been already implemented on different programming languages [21].

Portability of middleware base layer can be achieved on implementation stage by using C/C++ cross-platform languages and Qt framework. In turn, in order to ensure adaptability, middleware base layer can provide inclusion/exclusion of certain modules in compilation phase or at runtime to customize the middleware functionality for given host device and a certain information sharing environment. For example, a wireless router as middleware host device have limited resources and communications to store/access large amounts of data.

7) End user usability and simplicity: A middleware architecture should be easy to elaborate, evolve and understand by third-party developers. Deployment of middleware components should not require expert knowledge or support, since an middleware is typically deployed by an ordinary user. Complicated installation and setup procedures should be avoided.

8) Engaging the development community: A middleware for smart spaces should be constantly maintained and expanded. This requirement is provided within a community of developers and researchers.

Engaging the development community is an important requirement, since it is not always possible to create and maintain all elements of semantic-oriented middleware. This requirement is associated with extensibility and portability/adaptability requirements. New functionality can be implemented by third-party developers as open source software on middleware base layer and on infrastructure layers. It is also possible to implement pre-configured infrastructures (a set of agents/services) for a well-known application system. This infrastructures can be supported by the community and used to deploy the required applications.

IV. Reference case study

CuteSIB is the software implementation for such a central element of an M3 smart space as Semantic Information Broker (SIB) [4]. CuteSIB follow the M3 architecture (multi-device, multi-vendor, multi-domain) for creating smart spaces, which integrates technologies from IoT and the Semantic Web concepts. CuteSIB is adopted as the reference case study for the infrastructure multi-layer model of semantic-oriented middleware proposed in Section II.

The CuteSIB project aims to implement middleware base layer with features to create middleware infrastructure and application system infrastructure layers. SIB stores information using RDF-model (RDF-triples) and provides a blackboard service for agents interaction. There are several SIB implementations [22], [23]. The CuteSIB project aims to implementation extensibility, connectivity and accessibility, dependability, portability and adaptability, end user usability and simplicity, and engaging the development community non-functional requirements with supporting semantic interoperability. The security mechanism is developing in other project and can be included into CuteSIB later as a security modules. 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.

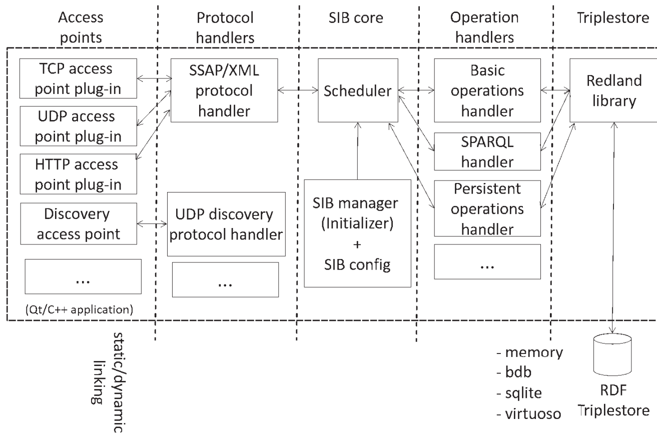

Fig. 4. The plug-ins based architecture of CuteSIB

The implementation of CuteSIB is based on the Qt framework in order to support a wide spectrum of Qt-based devices. The plug-ins based architecture achieves higher extensibility due to the modular approach, see Fig. 4. In the CuteSIB, the middleware base layer consists of five modules: (1) access points, (2) protocol handlers, (3) SIB core, (4) operation handlers, and (5) triplestore.

The access points bind to particular network (transport) protocol, receive agents requests and send responses. Protocol handlers implement application protocols. Protocol handlers work in combination with access points. Different combination allows to satisfy requirements (end user usability and simplicity, extensibility, connectivity and accessibility). Operation handlers implement operations that are needed for protocol logic. For example, the main protocol in the Smart-M3 platform is SSAP. Base implementation of SSAP is based on TCP and XML. In this case, the access point implements TCP/IP protocol and processes all routine with clients. The protocol handler implements the SSAP protocol. The access point sends to protocol handler notifications (client connected or disconnected) and requests. The protocol handler processing requests creates responses and sends responses to access point.

The access points for various network protocols (e.g., TCP, HTTP or UDP) are plug-ins. In current implementation of CuteSIB there are four access point plug-ins. They can be loaded/unloaded from the main CuteSIB program as dynamic libraries. TCP access point provides TCP access and is used as main access point to communicate with CuteSIB. UDP access point provides UDP access. HTTP access point handles HTTP SPARQL request (GET/POST). With HTTP Access Point CuteSIB can be accessed as a SPARQL-endpoint (e.g. DBPedia) with query request support (update and subscription are not supported). There is no need to implement a special Protocol Handler to process HTTP SPARQL request. HTTP access point translates HTTP to/from SSAP format and communicates with SSAP/XML Protocol Handler. Discovery access point is used to discovery CuteSIB with broadcast requests. UDP discovery protocol handler prepares information about CuteSIB and sends it back for request.

Operation handlers implement logic of CuteSIB operations. CuteSIB supports operations from SSAP protocol. Operations are implemented with tree operation handlers. Basic operation handler is for basic operations under triples (insert, remove, update, and query), persistent operations handler is for persistent operations (such as subscription), and SPARQL handler is for advanced search queries. These operations are mapped with SSAP operation and the SSAP operations are forwarded to appropriate handler for processing. Supported operations are not limited with SSAP operations, new operations (for example to support other protocol) can be added with implementation of new operation handler. For example, there is a binary protocol that similar with SSAP [21]. The one of the advantages of binary protocol is in smaller messages size. CuteSIB can support this binary protocol with implementation of the new protocol handler. To receive/send messages TCP access point can be used. Additional operations that are supported by binary protocol will be implemented with the new operation handlers. Operation handlers work with triplestore. Triplestore is represented by Redland libraries and wrapped with Qt classes to simplify access.

The CuteSIB core contains some auxiliary submodules and implements commands processing. Command is an action that should be done by module. CuteSIB modules use commands to send actions to other modules. Modules are independent from each other and send command to scheduler from core module. Scheduler sends command to destination module (if it set) or to the module that responsible for processing such command. Responsible modules are set through a mapping, used by scheduler in runtime to find modules for commands. It is possible to add some new module and set the mapping for this module.

Example of using command with processing of SSAP message is shown in Fig. 5. Agent sends bytes of SSAP/XML message to TCP access point. TCP access point receives a full message and sends it to SSAP/XML protocol handler. SSAP/XML protocol handler creates a command-request based on received SSAP message for inserting data and send it to scheduler. SSAP/XML protocol handler does not know about how data will be processed and stored. Scheduler based on command-request properties and mapping selects operation handler and sends command-request to it. Operation handler performs all needed actions with data to store it (insert to smart space) and sends a command-response.

In CuteSIB core, the data is represented in the unified form of triples. Each protocol handler translates received data from agent request to this format and sends commands to CuteSIB core. Commands are sent to the appropriate operation handler. The response for command is sent when the command is processed. Based on response, protocol handler creates a response for agent in the appropriate format and sends response to an agent through an access point.

As described above, CuteSIB can be extended with additional modules to support new protocols and internal operations. Middleware base layer does not fully cover all proposed requirements. The ”end user usability and simplicity” and ”connectivity and accessibility” can be covered by implementing services on middleware infrastructure layer and these requirements depend from application system. The security is important, but also depend from application system. Secure access to CuteSIB can be done by using an access point with secure socket (SSL). For private smart spaces the secure access can be not so important but access rights to get/insert data should be supported.

CuteSIB covers semantic interoperability in following way. CuteSIB stores data with RDF model. Data can be easily integrated with data from public SPARQL-endpoints (such as DBpedia) and other RDF storages. These data can be provided by services on middleware infrastructure level. Development libraries for Smart-M3 agent provide such functionality (CKPI, SmartSlog DPI). Internal module of CuteSIB can provide inferencing, knowledge discovery, and data federation by watching for triplestore and using general rules. Instead of internal module, middleware infrastructure services can use specific rules for application domain. In this case, changing middleware infrastructure services gives a new way for inferencing knowledge. In addition, there is no binding to particular inferencing mechanism and rule sets. Finally, the coverage of the requirements for the implementation of CuteSIB middleware is presented in Table II.

There are several works with performance evaluation of CuteSIB. In [22], the set of experiments showing the performance of basic operation (insert, update, query), persistent operations (subscription), and SPARQL queries was carried out. The update test was related to the insertion of a block of n triples (with n ranging from 100 to 2000). In this test, CuteSIB showed the result of about 80 ms to insert 2000 triples. The query test was related to retrieve a block of n triples with (with n ranging from 100 to 1000) an RDF query. In this test, In this test, CuteSIB showed the result of about 0.25 s to retrieve 1000 triples. The subscription test took into account time to receive the notification for the update of a triple with a total storage of 5000 triples. In this case, CuteSIB showed the result of about 0.6 ms. However, the results of SPARQL query test showed that CuteSIB performance, especially with the larger dataset composed by 50k triples, are not acceptable (more than 10 s or even network timeout). These experiments partially satisfy the non-functional requirements for CuteSIB in some way.

In [24], the CuteSIB performance for operation in an information sharing environment with many heterogeneous

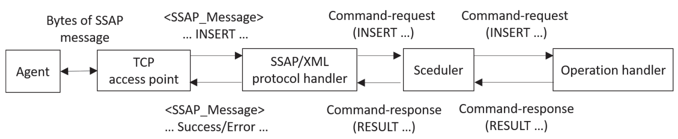
Bytes of SSAP message
<SSAP_Message> ... INSERT ...
Command-request (INSERT ...)
Command-request (INSERT ...)
Agent
TCP access point
SSAP/XML protocol handler
Sceduler
Operation handler
<SSAP_Message> ... Success/Error ...
Command-response (RESULT ...)
Command-response (RESULT ...)

Fig. 5. SSAP message processing

TABLE II. SUMMARY OF REQUIREMENTS AND THEIR SOLUTIONS IN CUTESIB MIDDLEWARE

Requirement	Provided solutions
Functional requirements	
Machine-computable logic	Description logic in RDF/XML. OWL built upon RDF, RDF Schema, and XML Schema.
Inference	Reasoning module in middleware base layer based on rules. RDF model.
Knowledge discovery	SPARQL-endpoint. SPARQL CONSTRUCT and SELECT queries. RDF model.
Data federation	SPARQL-endpoint. SPARQL CONSTRUCT and SELECT queries. RDF model.
Non-functional requirements	
Extensibility	The plug-ins based architecture.
Connectivity and accessibility	Discovery access point is used to discovery CuteSIB with broadcast requests. Services on middleware infrastructure layer and depend from application system.
Dependability	Restart/reconnect and operation control functionality and introduce special services for persistent storage of critical data.
Semantic interoperability	SSAP protocol. RDF model.
Security	Secure access can be done by using an access point with secure socket (SSL).
Portability and adaptability	The implementation is based on the Qt framework. The plug-ins based architecture.
End user usability and simplicity	CKPI, SmartSlog DPI.
Engaging the development community	Open source software and pre-configured infrastructures (a set of agents/services) for a well-known application system.

participating devices was evaluated. This evaluation was aimed at quantitative determination of the scalability bounds when the number of devices is growing and reaching the stress workload of the CueSIB host device. This experiments showed the high level of satisfaction of connectivity, interoperability and dependability requirements, when CuteSIB resists stress workload and continues the operation after the workload reduction.

In [25], the applicability of a wireless router for being a CuteSIB host device was examined experimentally. The experiments carried out in this work confirmed that capacity of a wireless router is satisfactory for deployment of smart spaces to support collaborative activity of participants. This experiments are related to the portability requirement.

V. CONCLUSION

The paper presented a perspective on how to design a software infrastructure of semantic-oriented middleware for smart spaces deployment in IoT environments. This paper introduced the middleware infrastructure multi-layer model. The paper analysed of the middleware requirements for development and deployment of smart spaces in IoT environments and the implementation of the requirements in the proposed model. As a reference case study, the CuteSIB semantic information broker implementation of the Smart-M3 platform was considered. The proposed model have a generic character and it can serve for designing such a class of middleware for smart spaces.

ACKNOWLEDGMENT

The research study is supported by the Ministry of Education and Science of Russia within project # 2.5124.2017/8.9 of the basic part of state research assignment for 2017–2019. The work is implemented within the Government Program of Flagship University Development for Petrozavodsk State University in 2017–2021.

REFERENCES

[1] 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, Feb 2017.

[2] 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.

[3] 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.

[4] D. G. Korzun, S. I. Balandin, A. M. Kashevnik, A. V. Smirnov, and A. V. Gurtov, "Smart spaces-based application development: M3 architecture, design principles, use cases, and evaluation," *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 8, no. 2, pp. 66–100, 2017.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the Internet of Things: early progress and back to the future," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 8, no. 1, pp. 1–121, 2012.

[13] D. Korzun, "Service formalism and architectural abstractions for smart space applications," in *Proc. 10th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR 2014)*. ACM, Oct. 2014, pp. 19:1–19:7.

[14] F. H. Mohammed and R. Esmail, "Survey on IoT services: Classifications and applications," *International Journal of Science and Research (IJSR)*, vol. 4, no. 1, pp. 2124–2127, 2015.

[15] S. De, P. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for the Internet of Things," in *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, Sept 2011, pp. 949–955.

[16] 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.

[17] 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.

[18] A. D'Elia, D. Manzaroli, J. Honkola, and T. S. Cinotti, "Access control at triple level: Specification and enforcement of a simple RDF model to support concurrent applications in smart environments," in *Proc. 11th Int'l Conf. Next Generation Wired/Wireless Networking (NEW2AN'11) and 4th Conf. Smart Spaces (ruSMART'11)*. Springer-Verlag, 2011, pp. 63–74.

[19] D. Korzun, I. Nikolaevskiy, and A. Gurtov, "Service intelligence and communication security for ambient assisted living," *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 6, no. 1, pp. 76–99, 2015.

[20] A. Gurtov, M. Komu, and R. Moskowitz, "Host Identity Protocol (HIP): Identifier/locator split for host mobility and multihoming," *Internet Protocol Journal*, vol. 12, no. 1, pp. 27–32, Mar. 2009.

[21] J. Kiljander, F. Morandi, and J.-P. Soininen, "Knowledge sharing protocol for smart spaces," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 3, pp. 100–110, 2012.

[22] 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.

[23] 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.

[24] A. S. Vdovenko, D. G. Korzun, and I. V. Galov, "Simulation performance evaluation of Smart-M3 applications for Internet of Things environments," in *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017 9th IEEE International Conference on*, vol. 2. IEEE, 2017, pp. 994–999.

[25] S. Marchenkov, D. Korzun, A. Shabaev, and A. Voronin, "On applicability of wireless routers to deployment of smart spaces in Internet of Things environments," in *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2. IEEE, Sep 2017, pp. 1000–1005.