Development of Collaborative Editing Applications through Semantic Publish-Subscribe Platforms

Fabio Viola, Francesco Antoniazzi, Alfredo D'Elia, Giacomo Corsi,
Luca Roffia, Tullio Salmon Cinotti

University of Bologna
Bologna, Italy
{fabio.viola, francesco.antoniazzi, alfredo.delia4, luca.roffia, tullio.salmoncinotti}@unibo.it,
giacomo.corsi@studio.unibo.it

***Abstract*—Cloud computing revolutionized the way resources (i.e. services and data) are accessed and used. Among the many changes in data processing and service provision paradigms, Software as a Service (SaaS) and collaborative editing are two of the most significant. This paper presents an approach based on semantic publish/subscribe paradigm to the development of collaborative editing applications. The main aim of this research work consists in exploiting emerging technologies (i.e. semantic web data representation formalisms, publish-subscribe platforms) in order to (1) overcome limitations coming from the traditional paradigm of single-author workflows, (2) enhance the editing capabilities of the users, (3) ease the development and maintainment of applications and (4) provide a shared high-level abstraction of the documents that fosters information-level interoperability and reusability. The proposed approach is then validated through an ad-hoc vector graphics application, SCEd (Semantic Collaborative Editor) built upon a SPARQL Event Processing Architecture derived from the Smart-M3 semantic publish-subscribe platform.**

I. Introduction

The so-called Web 2.0 brought to a revolution in the interaction of the users with the World Wide Web. The role of the user switched from consumer to prosumer, in fact, as proved by numerous famous examples (e.g. Wikipedia just to name one), nowadays a high number of websites propose a knowledge base made up of the contribution of the community. From a technological point of view, this is obtained through proper software (e.g. Mediawiki for Wikipedia). The importance of collaborative editing software is then continuously growing and the availability of real-time cloud services contributes to the spread of this paradigm. But contributing to large-scale knowledge bases like Wikipedia is not the only advantage brought by the Web 2.0: thanks to Cloud Computing and in particular to Software as a Service (SaaS) it is possible to work together at the same time, on private documents even for users very far from each other. This area, named collaborative editing introduces strict requirements. Above all it is worth mentioning data consistency and timeliness.

The evolution of the Web also led to the birth of the so-called Semantic Web, a new vision of the Web where information represented on the Internet is not only human-readable, but also machine-understandable to foster machine to machine (M2M) communication. To achieve this task, a new hierarchical structure of the Web has been introduced. Thanks to the standards forming the Semantic Web stack, information can be represented in form of univocally identifiable resources that, together with their relationships, produce an oriented and labeled graph that can be browsed, queried and edited using the powerful SPARQL language. This ambitious project is also known in literature as the Web 3.0. But, what is the link between the Web 2.0 and the Semantic Web? Are there pros or drawbacks in applying technologies from the Semantic Web to the development of Web 2.0 applications, like for example collaborative editing?

The main contribution of this paper consists in investigating the use of semantic technologies in collaborative editing applications. Semantic publish-subscribe middelwares in fact, may help in the development of collaborative editing application by:

- Providing an high-level abstraction of a document class and its instances that allows to face the barriers of using heterogeneous devices, operating systems and applications (i.e. achieving a higher level of interoperability).
- Providing a flexible way to follow the development of a specific document (or section) with focused fine-grain subscriptions.
- Allowing an easy development of powerful and fully customizable client applications thanks to the use of a simple architecture and a shared vocabulary.

This work is the result of the research activity carried on at the ARCES center (Advanced Research Center on Electronic Systems) of the University of Bologna and during the course named "Interoperability of Embedded Systems".

The article is organized as follows: after an overview of the state of the art (Section II), the rationale behind the approach is described in Section III. The software architecture of the application developed for the selected use case is described in Section IV. In Section V conclusions are drawn.

II. Related work

The analysis of the related works is structured in two branches: in the first part, collaborative editing research works are presented, while in the second, researches funded on the chosen interoperability platform are discussed.

ISSN 2305-7254

Tudorache et al. in [1], [2] propose an interesting technique for the collaborative ontology development applied to Protégé. Some of the requirements highlighted by the authors (e.g. synchronous and asynchronous access to data, support for annotations, scalability and user managements) were considered relevant for the project described in this paper. Still in the field of collaborative ontology development, in [3] a more general approach with a complete and exhaustive ontology is presented. In [4] an approach for collaborative editing of 3D meshes is presented, but, to the best of our knowledge, it is not founded on semantic publish-subscribe platforms, that represent instead the focus of this analysis. The relevance of semantic technologies in collaborative editing has been discussed by Volkel et al. [5] who proposed Semantic MediaWiki (SMW) to semantically annotate wiki pages, enabling enhanced browsing and searching. The approach has been reprised by Skaf-Molly in [6] where, thanks to semantics, a network of SMW servers, named Distributed SMW, is presented.

As it will be proved in the rest of the article, semantic technologies could help the development of collaborative editing software. The project started by Tim Berners-Lee in 1999 and named Semantic Web [7] brought to the birth of a stack of protocols designed to transform the web in a machine-interpretable world. Currently, many of the protocols composing the Semantic Web stack are also used in contexts not related to the web, for example in order to grant interoperability among smart devices in IoT or pervasive computing scenarios. For example Unicode is used to univocally identify resources, RDF (Resource Description Framework) [8] to represent data as graphs composed by triples (subject, predicate and object), OWL (Web Ontology Language) [9] to provide meanings to the represented information and SPARQL UPDATE [10] and QUERY [11] language to respectively modify and retrieve information from the RDF knowledge base. Among the existing interoperability platforms, a suitable choice as a target platform for the development of the article scenario is Smart-M3 [12]. Smart-M3 implements the publish-subscribe paradigm [13], [14] to timely inform entities about changes on the knowledge base. Despite being a relatively young project (it was born in 2008 in the European Project SOFIA), several researches have been conducted, leading to strategies for application development and attempts of integration with other frameworks have been shown [15]. Furthermore, several real-life applications have been presented: a blogging application called SmartScribo was proposed in [16], [17]; then, its integration in a Smart Conference System was performed and described by Korzun et al. in [18]. The following works tried, first, to evolve the latter in a Smart Room application [19] and, then, to combine the E-Tourism services [20], [21] with the Smart Room [22], [23]. The most recent research topics on the Smart-M3 platform include mobile health [24], robots organization [25]. A first attempt of using Smart-M3 for collaborative work can be found in [26]. In the first use case described in the paper, the use of semantic technology is aimed at facilitating the real time participation to a conference: in this paper the focus is instead the production of documents by multiple authors. The second use case presented by [26] seems to be more similar, since the focus is on the production of a shared knowledge base (KB) for tourist information, however we focus on allowing the production not of a single large scale KB, but many small documents represented by independent subgraphs.

III. The approach

In the development of collaborative editing applications we identified some requirements: first of all the need of a real-time communication among clients. The second is about interoperability: users of different devices and operating systems should not be left out.

For each specific application class several file formats exist, and, generally speaking, every software vendor or organization tries to impose its own format. Taking as an example the field of vector graphics, enterprises like Adobe and Corel Corporation (just to name a few) and entities like the W3C promote their own file formats to describe vector graphic artworks. For users it is usually very hard to interoperate if different software and file formats are adopted. To fulfill the requirement of interoperability, we propose to rely on semantics technologies. The approach consists in mapping to a common ontology all the possible elements of an application class and sharing, for each document, only the high-level semantic representation based on the defined document type. In this way interoperability at information level can be granted. Semantic Web technologies play a crucial role in this approach: the common ontology is defined through RDFS and OWL and can be modified and extended to support new features as the selected scenario evolves; documents built according to the shared ontology are represented as RDF graphs and can be retrieved and updated using whatever software capable to perform a query or an update using the SPARQL language (that belongs to the Semantic Web too). This model then intrinsically fosters the maintenance and development of the software thanks to the use of well-known standards and the definition of a vocabulary that can be later on extended to support new advanced features.

But how to fulfill the requirement for real-time communication using a Semantic Web knowledge base? In this paper we exploit our previous experience in the field of semantic publish-subscribe platforms by using a SPARQL Event Processing Architecture [27] (SEPA). SEPA can be described as a central knowledge-base provided with a publish-subscribe mechanism based on SPARQL. The puslish-subscribe mechanism is the key for overcoming the limitations of a traditional single-author workflow.

Specifically, in our approach, we develop a collaborative editing application by:

1) Identifying the application domain;
2) Identifying the main concepts of a document in that application domain;
3) Providing an high-level abstraction of the concepts in form of an ontology;
4) Identifying the SPARQL updates and queries/subscriptions that allows to manipulate and retrieve (parts of) the documents;
5) Providing specific translators that extract the semantic representation of a document from the shared knowledge base and produce an output file in the desired format.

The use case of vector graphics is described in the rest of the paper. An ontology was created to map the concepts of different vector graphics applications into a shared agreed

vocabulary. Then, a new application named Semantic Collaborative Editor was developed. Due to the generality of the approach, authors could as well have extended existing vector graphics applications.

IV. The use case: SCEd

In this Section the approach described in Section III and representing the main contribution of this research work is validated through a proof of concept application. The selected use case is focused on collaborative editing for computer graphics, and in particular vector graphics. The rest of this Section is organized as follows: in Subsection IV-A the ontology designed for this use case is presented and discussed. In Subsection IV-B the reference platform adopted to develop the proof of concept is introduced, while in IV-C the software architecture planned for the development of the collaborative editor (named SCEd, Semantic Collaborative Editor) is depicted. Finally, in Subsection IV-E, the developed software is analyzed to verify the validity of the proposed approach.

A. The ontology

The starting point for the development of a collaborative editor relying on semantic technology consists in the design of a proper ontology. In this proof of concept, the use case is the one of collaborative vector graphics, so the ontology should map the main entities that compose a vector artwork, but also the users and their comments. Starting from the analysis of the graphic elements provided by the Scalable Vector Graphics (SVG) format, the ontology provided in Fig. 1 has been designed. While the leftmost branch represents the class dedicated to the User profile, the central and the rightmost branches contains, on the other hand, the classes needed to represent all the elements of a vector graphics artwork. Vector graphics is only one of the possible use cases. In fact, this approach is general enough to apply to other domains, by simply mapping a different document type in a proper ontology.

The description of the ontology cannot be complete without an exhaustive list of the Datatype and Object properties defined for the domain. They are reported respectively in Table I (with their domain) and II (with domain and range).

B. The reference platform

The development of the Semantic Collaborative Editing Platform has been centered on a SPARQL Event Processing Architecture born as an evolution of the Smart-M3 interoperability platform. This platform provides one or more central nodes aimed at storing the shared knowledge base: this kind of node (formerly named Semantic Information Broker or SIB) is named SEPA. Several implementation of the SIB exists: 1) RedSIB [28] is a C general-purpose implementation, fast and nowadays very diffuse; 2) the OSGi SIB [29], [30] is a more recent work oriented at IoT gateways; 3) pySIB [31] is a lightweight Python implementation developed for low-powered computing nodes as, for example, System on Chips (SoCs) devices; 4) CuteSIB [32] is another recent implementation born as a fork of the old RedSIB. In this research work, we focus on SEPA rather than on the SIB due to the support for standard protocols such as SPARQL 1.1 Protocol to query and update

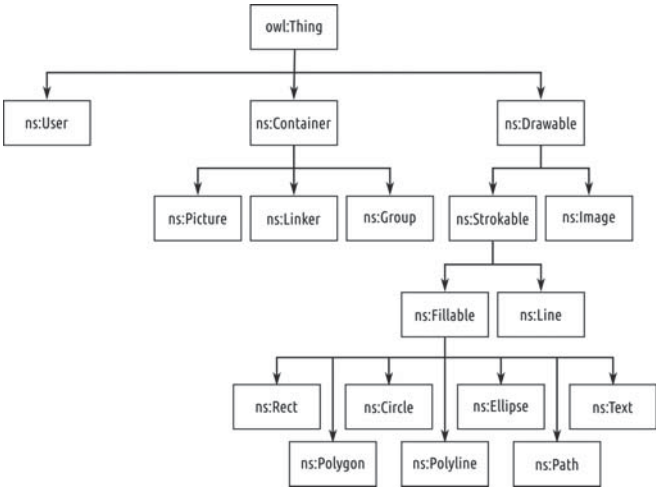

Fig. 1. The ontology for the vector graphics application

TABLE I. Datatype Properties

Property	Domain
hasWidth	Picture, Rect, Image
hasHeight	Picture, Rect, Image
hasReferenceWidth	Picture
hasReferenceHeight	Picture
hsaLink	Linker
hasGroupStroke	Group
hasGroupStroleWidth	Group
hasGroupFill	Group
startsAtX	Rect, Text, Image
startsAtY	Rect, Text, Image
hasRoundingX	Rect
hasRoundingY	Rect
hasStroke	Strokable
hasStrokeWidth	Strokable
isFilledBy	Fillable
hasX1	Line
hasX2	Line
hasY1	Line
hasY2	Line
hasCenterX	Circle, Ellipse
hasCenterY	Circle, Ellipse
hasRadius	Circle
hasRadiusX	Ellipse
hasRadiusY	Ellipse
hasPoints	Polygon, Polyline
hasFont	Text
hasFontSize	Text
hasDescription	Path

TABLE II. Object Properties

Property	Domain	Range
hasPicture	User	Picture
hasFriend	User	User
isUpon	Drawable	Drawable
contains	Container	Drawable, Linker, Group

the KB or the SPARQL 1.1 Subscribe Protocol defined by the authors and introduced in this platform to provide subscriptions over WebSockets. Agents interacting with the SIB/SEPA are called Knowledge Processors (KPs) and can be developed exploiting one of the many existing APIs (currently available for Java, Python, C, Ruby, Javascript). The architecture of the SEPA platiform is summarized in Fig. 2.

The SEPA Platform is founded on a SPARQL endpoint (i.e. Blazegraph in our case) holding an RDF graph. The

SEPA Engine wraps up the endpoint and provides support for the publish-subscribe mechanism providing a Websocket interface and a novel protocol named SPARQL Subscribe Protocol. Applications may interact with SEPA by producing (i.e. through the SPARQL Update Language) or consuming information (i.e. in real time with the SPARQL 1.1 Subscribe Language or by polling the KB with a SPARQL query). In the first case we talk about Producers, while in the latter about we talk about Consumers. Both write and read information according to a shared ontology that defines the content and the meaning of each triple. For an in-depth introduction and analysis of the reference platform we invite the reader to refer to [33], [27].

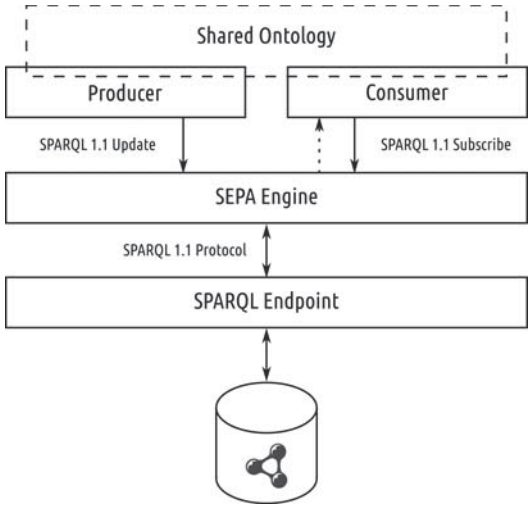

Fig. 2. The architecture of the SEPA platform

C. The Software architecture

The software architecture of SCEd is represented in Fig. 3. The broker of the SEPA platform hosts in its RDF graph the high-level semantic representation of the shared documents. The client-side of the application is composed by several interoperating modules described in the following subsections.

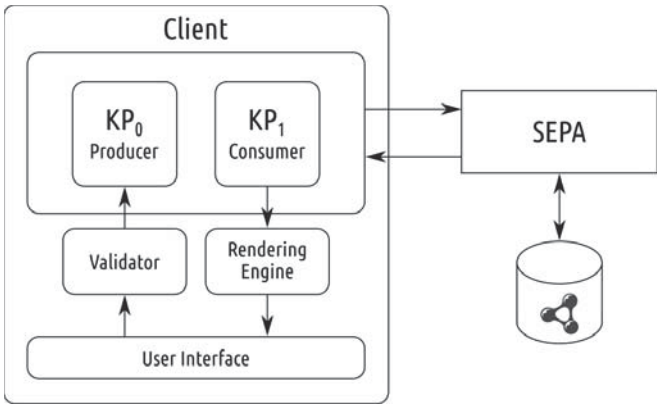

Fig. 3. Software architecture of SCEd

In Fig. 4, an example of information flow is depicted. Two instances of SCEd load the same document, composed by a triangle. The document is retrieved using a SPARQL subscription that, as a first confirm message, sends the reply of the corresponding query. Then the client on the left creates a circle. This shape is translated into a proper SPARQL update issued to the SEPA. The SPARQL Event Processing Architecture detects a change in the subgraph of the image and sends a notification to the client on the right that, in a few milliseconds, updates the drawing on the User Interface.

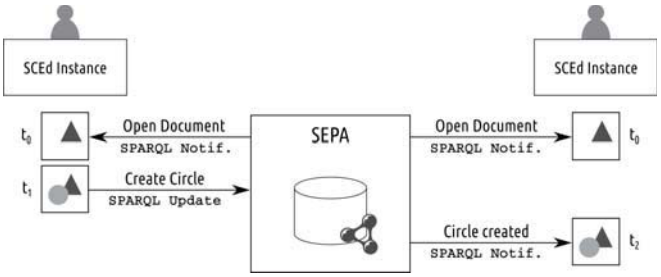

Fig. 4. SCEd: example information flow

1) The Input Validator: It is responsible of translating user inputs into semantic requests (i.e. SPARQL SUBSCRIBE and UPDATE requests). It also assesses the validity of the user input and, if needed, filters out wrong or unauthorized requests.

2) The User Interface: The User Interface (UI) constitutes the highest layer of the application. A screenshot of the first implementation made using Javascript's popular framework JQuery is shown in Fig. 5.

Through the UI users visualize and edit documents. The UI allows to select the operating mode (i.e. online or offline) and if online mode is selected, the Semantic Information Broker to be used can be specified by providing the URL for update requests and the one for subscriptions.

While the uppermost part of the UI is dedicated to the connection setup and to the choice of the document, the bottom area of the UI presents the canvas where the current state of the image is visualized in real-time (exploting the subscription mechanism). The canvas also presents a toolbar that contains the main interaction buttons classified among creation, editing, removal and export buttons.

3) The SEPA Interaction Module: The SEPA interaction module represents the interface between the client application and the SEPA platform. It is composed by two conceptually different knowledge processors: one (i.e. the Producer) responsible for transferring local changes to the SIB; the other (i.e. the Consumer) to retrieve all the changes made by the other users. The simultaneous actions performed by the two KPs allows to keep the local and remote documents synchronized. The SEPA Interaction module is configured through the Configuration Manager: the URL for updating and subscribing to the knowledge base are provided to the SEPA Interaction Module by the Configuration Manager. Exploiting the subscription mechanism, the SEPA Interaction Module is timely notified about changes on a documents and such changes are propagated to the Rendering Engine to be visualized. The publish-subscribe paradigm allows the application to be fast and responsive.

4) The Configuration Manager: As mentioned in the previous susbsection, the Configuration module is used to set

Fig. 5. A screenshot of SCEd

the connection parameters needed to connect to the SEPA. Through this module it is also possible for the user to customize the look and feel of the application.

5) The File Loader: Feeding SCEd with an existing file, a semantic representation of the document content is built and shared through the SIB. The SCEd module responsible of such operation is the File Loader. The development process of the File Loader must take into account the file formats to be supported; in the use case described in this article, the File Loader currently provides support only for the SVG file format.

6) The Rendering Engine: The Rendering Engine draws on the users device the shared document in its current state. This particular software module is also responsible for rendering the document into file with the file format specified by the user. In this case the Rendering Engine plays a complementary role with respect to the File Loader. The application developed for the scenario supports rendering into SVG files or PNG files. The semantic representation of the document is general enough to make possible for every developer to implement a custom rendering function to translate the high-level document to a specific format, as in the case of SVG and PNG.

D. SCEd and the web of things

SCEd is not only a simple collaborative editing application founded on Semantic Web technologies and the publish-subscribe paradigm. SCEd is a Web of Things application. The Web of Things (WoT) is a novel research area born in 2009 but becoming popular only in these latest years [34]. The aim of the WoT is to fight the fragmentation of the Internet of Things through the use of standard and well-known protocols adopted in the Web. Every device can be discovered and controlled by means of its Thing Description (TD) that contains an exhaustive description of all the Thing's properties, events and actions.

That said, SCEd also includes a module that enables the automatic generation of a Web Thing for every illustration. This allows to create a virtual device discoverable and controllable through SEPA. The Thing Description of the virtual device includes its properties, but first of all the events (i.e. about new modifications to the sketch) and the available actions (i.e. draw a new shape).

E. Discussion

The objective of this study was the analysis of the advantages of using semantic publish-subscribe platforms to develop collaborative editing applications.

As stated in the Introduction, one of the main advantages, is the higher degree of information-level interoperability. In fact people with different operating systems and requirements in terms of file formats can work together using SCEd and only manipulating the high-level representation of the artwork: this abstract entity can be translated by the proper exporter in the desired file at the end of the work. The ability to extend SCEd with a custom exporter provides the power to support potentially each destination file format.

A second advantage is the ability to follow the development in a flexible way: in fact, being this approach centered on a semantic publish-subscribe platform, the user may subscribe to the whole document he is working on or issuing a more

specific subscription to follow the evolution of only a part of the graph.

Third, the development model is easy and powerful: in fact, after designing an ontology mapping the main concepts of the domain of interest, the development of two simple Knowledge Processors is enough to provide the basic features of a collaborative editor. Furthermore, being each artwork only a graph in the main knowledge base, it is quite easy to extend an application providing custom SPARQL queries/subscriptions (e.g. for custom selection mechanisms) or updates (e.g. to simultaneously update multiple elements).

Drawbacks of the approach may be represented by the poor performance that may affect the scalability and responsiveness [27]. The performance of the application are influenced by the size of the document (i.e. the number of triples composing a document), the number of clients connected to the system (i.e. due to the number of subscriptions to be processed), by the size of the KB (i.e. high density means higher time to detect changes in the graph). Further investigations will be carried on to assess the performance of the designed system and to quantify the maximum amount of contemporary users, drafts (and their complexity) that can grant a fluid behaviour.

V. Conclusions and future work

In this paper a domain-agnostic approach for the development of collaborative editing applications has been presented. The main idea consisted of relying on the powerful duo made by semantic technologies and the publish-subscribe paradigm. The selection of a use case allowed to validate the approach through the development of a proper application centered on the domain of vector graphics.

The approach proved to be general enough to be extended to different application domains (e.g. word processing, spread sheet manipulation and so on). Furthermore, the flexibility granted by Smart-M3 follower, SEPA, allowed to easily build an application starting by the design of a proper ontology and the development of the two main components, the Knowledge Processors, responsible for the interaction with the Semantic Information Broker and then for manipulating each document.

Further investigations will be carried on in the next months to analyze the possibility to merge several different application domains by proper extensions of SCEd (and, of course, its ontology).

Acknowledgements

Authors would like to thank Mirco Gurioli and Valerio Carpani for the preliminary development of the presented application.

References

[1] T. Tudorache, N. F. Noy, and M. A. Musen, "Collaborative protege: Enabling community-based authoring of ontologies," in *Proceedings of the 2007 International Conference on Posters and Demonstrations-Volume 401*. CEUR-WS. org, 2008, pp. 151–152.

[2] T. Tudorache, N. F. Noy, S. Tu, and M. A. Musen, "Supporting collaborative ontology development in protégé," in *International Semantic Web Conference*. Springer, 2008, pp. 17–32.

[3] N. F. Noy, A. Chugh, W. Liu, and M. A. Musen, *A Framework for Ontology Evolution in Collaborative Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 544–558. [Online]. Available: http://dx.doi.org/10.1007/11926078_39

[4] G. Salvati, C. Santoni, V. Tibaldo, and F. Pellacini, "Meshhisto: Collaborative modeling by sharing and retargeting editing histories," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 205:1–205:10, Oct. 2015. [Online]. Available: http://doi.acm.org/10.1145/2816795.2818110

[5] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic wikipedia," in *Proceedings of the 15th International Conference on World Wide Web*, ser. WWW '06. New York, NY, USA: ACM, 2006, pp. 585–594. [Online]. Available: http://doi.acm.org/10.1145/1135777.1135863

[6] H. Skaf-Molli, G. Canals, and P. Molli, "Dsmw: Distributed semantic mediawiki," in *Extended Semantic Web Conference*. Springer, 2010, pp. 426–430.

[7] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[8] O. Lassila, R. R. Swick *et al.*, "Resource description framework (rdf) model and syntax specification," 1998.

[9] P. F. Patel-Schneider, P. Hayes, I. Horrocks *et al.*, "Owl web ontology language semantics and abstract syntax," *W3C recommendation*, vol. 10, 2004.

[10] A. Seaborne, G. Manjunath, C. Bizer, J. Breslin, S. Das, I. Davis, S. Harris, K. Idehen, O. Corby, K. Kjernsmo *et al.*, "Sparql/update: A language for updating rdf graphs," *W3c member submission*, vol. 15, 2008.

[11] E. PrudHommeaux, A. Seaborne *et al.*, "Sparql query language for rdf," *W3C recommendation*, vol. 15, 2008.

[12] J. Honkola, H. Laine, R. Brown, and O. Tyrkko, "Smart-m3 information sharing platform." in *ISCC*, 2010, pp. 1041–1046.

[13] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, Jun. 2003. [Online]. Available: http://doi.acm.org/10.1145/857076.857078

[14] R. Baldoni, M. Contenti, and A. Virgillito, *The Evolution of Publish/Subscribe Communication Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 137–141. [Online]. Available: http://dx.doi.org/10.1007/3-540-37795-6_25

[15] D. G. Korzun, S. I. Balandin, V. Luukkala, P. Liuha, and A. V. Gurtov, "Overview of smart-m3 principles for application development," in *Proc. Congress on Information Systems and Technologies (IS&IT11), Conf. Artificial Intelligence and Systems (AIS11)*, vol. 4, 2011, pp. 64–71.

[16] D. Zaiceva, I. Galov, and D. Korzun, "A blogging application for smart spaces," in *Proc. 9th Conf. of Open Innovations Framework Program FRUCT and 1st Regional MeeGo Summit Russia–Finland*, 2011, pp. 154–163.

[17] D. G. Korzun, I. V. Galov, and S. I. Balandin, "Proactive personalized mobile multi-blogging service on smart-m3," *CIT. Journal of Computing and Information Technology*, vol. 20, no. 3, pp. 175–182, 2012.

[18] D. G. Korzun, I. V. Galov, A. M. Kashevnik, N. G. Shilov, K. Krinkin, and Y. Korolev, "Integration of smart-m3 applications: Blogging in smart conference," in *Smart Spaces and Next Generation Wired/Wireless Networking*. Springer, 2011, pp. 51–62.

[19] D. Korzun, I. Galov, and S. Balandin, "Development of smart room services on top of smart-m3," in *Open Innovations Association (FRUCT), 2013 14th Conference of*. IEEE, 2013, pp. 37–44.

[20] A. Smirnov, A. Kashevnik, S. I. Balandin, and S. Laizane, *Intelligent Mobile Tourist Guide*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 94–106. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40316-3_9

[21] A. Smirnov, A. Kashevnik, A. Ponomarev, N. Shilov, M. Schekotov, and N. Teslya, "Recommendation system for tourist attraction information service," in *14th Conference of Open Innovation Association FRUCT*, Nov 2013, pp. 148–155.

[22] D. Korzun, I. Galov, A. Kashevnik, and S. Balandin, "Virtual shared workspace for smart spaces and m3-based case study," in *Proceedings of 15th Conference of Open Innovations Association FRUCT*, April 2014, pp. 60–68.

[23] A. S. Vdovenko, S. A. Marchenkov, and D. G. Korzun, "Enhancing the smartroom system with e-tourism services," in *2015 17th Conference of Open Innovations Association (FRUCT)*, April 2015, pp. 237–246.

[24] D. G. Korzun, I. Nikolaevskiy, and A. Gurtov, "Service intelligence support for medical sensor networks in personalized mobile health systems," in *Conference on Smart Spaces*. Springer, 2015, pp. 116–127.

[25] A. Smirnov, A. Kashevnik, N. Teslya, S. Mikhailov, and A. Shabaev, "Smart-m3-based robots self-organization in pick-and-place system," in *Open Innovations Association (FRUCT), 2015 17TH Conference of*. IEEE, 2015, pp. 210–215.

[26] D. G. Korzun, A. M. Kashevnik, S. I. Balandin, and A. V. Smirnov, "The smart-m3 platform: experience of smart space application development for internet of things," in *Conference on Smart Spaces*. Springer, 2015, pp. 56–67.

[27] L. Roffia, F. Morandi, J. Kiljander, A. DElia, F. Vergari, F. Viola, L. Bononi, and T. S. Cinotti, "A semantic publish-subscribe architecture for the internet of things," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1274–1296, 2016.

[28] F. Morandi, L. Roffia, A. DElia, F. Vergari, and T. S. Cinotti, "Redsib: a smart-m3 semantic information broker implementation," in *Proc. 12th Conf. of Open Innovations Association FRUCT and Seminar on e-Tourism*. SUAI, 2012, pp. 86–98.

[29] D. Manzaroli, L. Roffia, T. S. Cinotti, E. Ovaska, P. Azzoni, V. Nannini, and S. Mattarozzi, "Smart-m3 and osgi: The interoperability platform," in *Computers and Communications (ISCC), 2010 IEEE Symposium on*. IEEE, 2010, pp. 1053–1058.

[30] A. D'Elia, F. Viola, L. Roffia, P. Azzoni, and T. S. Cinotti, "Enabling Interoperability in the Internet of Things:," *International Journal on Semantic Web and Information Systems*, vol. 13, no. 1, pp. 147–167, jan 2017. [Online]. Available: http://www.igi-global.com/article/enabling-interoperability-in-the-internet-of-things/172427

[31] F. Viola, A. D'Elia, L. Roffia, and T. S. Cinotti, "A modular lightweight implementation of the smart-m3 semantic information broker," in *2016 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT)*, April 2016, pp. 370–377.

[32] I. V. Galov, A. A. Lomov, and D. G. Korzun, "Design of semantic information broker for localized computing environments in the internet of things," in *2015 17th Conference of Open Innovations Association (FRUCT)*, April 2015, pp. 36–43.

[33] F. Viola, A. DElia, D. Korzun, I. Galov, A. Kashevnik, and S. Balandin, "The m3 architecture for smart spaces: Overview of semantic information broker implementations."

[34] D. Guinard, V. Trifa, T. Pham, and O. Liechti, "Towards physical mashups in the web of things," in *Networked Sensing Systems (INSS), 2009 Sixth International Conference on*. IEEE, 2009, pp. 1–4.