Overview of Smart-M3 Principles for Application Development

Dmitry G. Korzun[1,2], Sergey I. Balandin[3,5],
Vesa Luukkala[4], Petri Liuha[4], Andrei V. Gurtov[5,2]

Smart-M3 is an open-source platform that provides the multi-agent distributed application with the shared view of dynamic knowledge and services in ubiquitous computing environments. Smart-M3 semantic information broker maintains smart space in low-level terms of triples, based on resource description framework (RDF). Smart-M3 application consists of agents (also known as Smart-M3 knowledge processors) that consume/produce smart space content according with its ontological representation. This paper overviews the Smart-M3 concept and the principles of programming Smart-M3 knowledge processors.

1. Introduction

Smart spaces provide an environment for a number of devices to share resources and services. Smart spaces can provide better user experience by allowing the user easily integrate new devices into personal information infrastructures and allow seamlessly access all information distributed over the multi-device system from any of the devices. One of the essential features assumed by such environment is the information sub-system that provides permanent robust infrastructure for storing and retrieving the information of different types from the multitude of environment participants.

Smart-M3 is an open-source interoperability platform for information sharing [Honkola et al., 2010], where M3 stands for Multidevice, Multidomain, and Multivendor; the source code of Smart-M3 is available at http://sourceforge.net/projects/smart-m3/. The platform has been developed by a consortium of companies and within research projects: Artemis JU funded

[1] *Petrozavodsk State University, 185009, Lenin st., 33,Petrozavodsk, Russia, dkorzun@cs.karelia.ru*
[2] *Helsinki Institute for Information Technology, P.O. Box 19800, 00076 Aalto, Finland*
[3] *FRUCT Oy, http://www.fruct.org/, sergey.balandin@fruct.org*
[4] *Nokia Research Center, Itamerenkatu 11-13, 00180 Helsinki, Finland, {Vesa.Luukkala, Petri.Liuha}@nokia.com*
[5] *University of Oulu, P.O. Box 4500, 90014 Oulu, Finland, gurtov@ee.oulu.fi*

Sofia project (Smart Objects for Intelligent Applications) and Finnish nationally funded program DIEM (Device Interoperability Ecosystem) [Liuha et al., 2009]. Smart-M3 implements smart space infrastructures for multi-agent distributed applications following the smart space concept [Cook et al., 2007], [Oliver et al., 2009], [Balandin et al., 2009]. The latter is becoming popular in semantic computing.

2. Space-based computing

Space-based (or tuplespace) computing has its roots in parallel and distributed programming. Prior art [Gelernter 1985] defines the generative communication model where common information is shared in a tuplespace. The parallel processes of a distributed application cooperate by publishing/retrieving tuples into/from the space. The tuple is an ordered list of typed fields. Data tuples contain static data. Process tuples represent processes under execution. This asynchronous (publish-based) inter-process communication model allows building programs by gluing together active pieces [Nixon et al., 2008].

Aiming at automated processing in the World Wide Web, the Semantic Web was introduced. It allows describing content in a structured manner, where ontologies become the basic building blocks. The idea of triple space computing [Fensel, 2004] inherits the publication-based communication model from the tuplespace model and extends it with mechanisms of the Semantic Web: tuples are RDF triples (subject, predicate, object). They in turn are organized into the RDF graphs with subjects and objects as nodes and predicates as edges. Hence, semantic-aware queries to the space are possible, utilizing matching algorithms and semantic query languages like SPARQL.

In fact, the triple space computing paradigm states a scalable semantic infrastructure for web applications. It enables integration, communication, and coordination of many autonomous, distributed, and heterogeneous web service providers and consumers. Information stored in the same space can be further processed, providing deduced knowledge that otherwise cannot be available from a single source. Semantic web spaces apply this possibility to a new coordination model: any participant can infer new facts as a reaction to knowledge that has been published by another participant [Nixon et al., 2008]. Semantic web spaces are an extension of tuplespaces: tuples are RDF triples and matching is based on RDF reasoning capabilities.

The conceptual spaces (CSpaces) [Martin-recuerda, 2005] extend triple space computing to be applicable in different scenarios apart from web services. One of the key features of CSpaces is a composition of the tuplespace publish-based model with the publish/subscribe model from the pub/sub

communication paradigm [Eugster et al., 2003]. Transaction support is included to guarantee the successful execution of a group of operations. This advanced coordination model provides flow decoupling from the client side, which is done in addition to time and space decoupling already available in the tuplespace coordination model.

3. Smart Spaces and Smart-M3 infrastructure

The smart spaces form a smart environment, which is “able to acquire and apply knowledge about its environment and to adapt to its inhabitants in order to improve their experience in that environment" [Cook et al., 2007]. In accordance with the ubiquitous computing vision, smart spaces encompass the following information spaces: (i) physical spaces with sensing devices such as homes or cars, (ii) service spaces with information retrieval and processing such as Internet services or surrounding services in tourist place, and (iii) user spaces with personal information such as user profiles or address books. The information is dynamically shared by multiple heterogeneous participants (humans and machines), allowing each user to interact continuously with the surrounding environment, and the services continuously adapt to the current needs of the user [Balandin et al., 2009].

Smart spaces require a software infrastructure that turns the constituting spaces into programmable distributed entities. Smart-M3 provides such an infrastructure to use a shared view of dynamic knowledge and services within a distributed application. To the best of our knowledge the Smart-M3 platform is the only general-purpose open-source smart spaces platform currently available.

In addition to the normal range of personal computers and embedded devices, mobile devices with various means of connectivity become the primary gateway to the service space and the major storage point in the user space [Balandin et al., 2009], [Oliver et al., 2009]. Smart-M3 follows the space-agent approach. Each device, service, or storage point is programmable as an agent. In this ***m****ultidevice* system, agents place, share, and manipulate with local and global information using their own locally agreed semantics.

Information sharing in Smart-M3 is based on the space-based models using the same mechanisms as in the Semantic Web, thus allowing ***m****ultidomain* applications, where the RDF representation allows easy exchange and linkage of data between different ontologies, making cross-domain interoperability straightforward [Honkola et al., 2010]. Smart-M3 currently supports only limited reasoning, e.g., queries with subclass relations; see the following reference [Luukkala et al., 2010] for more details and possible extensions.

The core component of a Smart-M3 space infrastructure is semantic information broker (SIB) – an access point to the smart space. Each SIB maintains a part of information represented as an RDF triplestore. It provides simple reasoning, e.g., understanding of the owl:sameAs concept. The current Smart-M3 implementation supports WilburQL as a basic query language. Migration to the SPARQL is in progress.

Devices participate in the space by using software agents called knowledge processor (KP). KP connects to SIBs via some network and can modify and query the information by insert, remove and update queries, and (un)subscribe operations using the Smart Space Access Protocol (SSAP). Smart-M3 provides a number of solutions for network connectivity (e.g., HTTP, plain TCP/IP, NoTA, Bluetooth), yielding ***m****ultivendor* device interoperability. Accessing the space is session-based by join/leave operations, thus providing the base for mechanisms of access control and secure information sharing.

From the KP point of view, information in the space is organized into an RDF graph, usually according to some ontology defined in OWL. Use of a specific ontology is not demanded, i.e. a group of KPs can locally agreed ontology to use for interpretation of a certain part of the space. It is also important to note that the consistency of stored information is not guaranteed. KPs are free to interpret information in whatever way they want. Any KP sees the same information content regardless via which of the SIBs it is connected to the smart space.

4. Notion of the Smart-M3 application

Smart-M3 application can be considered a composition of possible scenarios enabled by a certain group of KPs. Execution of scenarios in such a composition targets the current needs of a user. For instance, an email application consists of the following scenarios: sending, receiving, composing, and reading email. Each scenario can be implemented by a dedicated KP. The same KP can be used in different applications. For instance, a KP for composing email can be a part of application for browsing social networks.

From this point of view, the basic principle is that the user has a collection of KPs. Together they are capable to execute a certain use scenario. If the provided collection does not support the required scenario then an additional KPs shall be found or developed. Each KP understands and makes own interpretation of a non-exclusive set of information. The set is typically described with the ontology of the KP, at least implicitly. Overlap of the sets of different KPs is needed for interoperability. KPs can see actions of each other.

Applications are constructed as the ad-hoc assemblies of KPs. Each scenario emerges from the observable actions taken by KPs based on smart

space content or from use of the available services. Some scenarios can be transient: the execution is changed as the participating KPs join and leave the smart space as well as some services become available or unavailable.

The aim of this Smart-M3 approach is at the ease of combining multiple scenarios into various applications. The key point is the loose coupling between the participating KPs. They use the space-based and pub/sub communication models modifying and querying the information in the common smart space. Thus, the impact of each participating KP to others is limited by the information the KP provides into the space. Note that Smart-M3 does not prevent direct communication between KPs, even though use of such approach should be minimized. Thus some actions can be activated using the traditional inter-process communication models. Adding elements of this traditional approach, however, impedes the easy use of affected KPs in other applications, set critical interdependencies between the modules and so reducing the key modularity benefit of the Smart-M3 platform.

5. Ontology Libraries for Programming Knowledge Processors in Smart-M3

Developers of the KP application logic use KP interface (KPI) to access information in smart space. The content conform the ontological description. Low-level access requires the user code to operate with RDF triples directly following the SSAP operations, where triples are the basic exchange elements. In contrast, the high-level KP development is based on an ontology library. It allows the user code to be written using high-level ontology entities (classes, relations, individuals). They are implemented in the code with predefined data structures and methods. Table 1 shows available low-level KPIs and ontology library generators for several popular programming languages.

Table 1. KP Interfaces to Smart-M3 Smart Space

Library	Description
Low-level KP programming: RDF triples	
Whiteboard, Whiteboard-Qt + QML	Language: C/Glib, C/Dbus, C++/Qt. Network: TCP/IP, NoTA. BSD license. A part of the Smart-M3 distribution, http://sourceforge.net/projects/smart-m3/
KPI_Low	Language: ANSI C. Network: TCP/IP, NoTA. GPLv2. Primarily oriented to low-performance devices. VTT-Oulu Technical Research Centre (Finland), http://sourceforge.net/projects/kpilow/
Smart-M3 Java KPI library	Language: Java. Network: TCP/IP. University of Bologna (Italy) and VTT-Oulu Technical Research Centre (Finland), http://sourceforge.net/projects/smartm3-javakpi/
M3-Python KPI (m3_kp)	Language: Python. Network: TCP/IP. BSD license. A part of the Smart-M3 distribution, http://sourceforge.net/projects/smart-m3/
C# KPI library	Language: C#. Network: TCP/IP. University of Bologna (Italy), http://sourceforge.net/projects/m3-csharp-kpi/
High-level KP programming: OWL ontology	
Smart-M3 ontology to C-API generator	Language: Glib/C, Dbus/C. Network: TCP/IP, NoTA. BSD license. A part of the Smart-M3 distribution, http://sourceforge.net/projects/smart-m3/
Smart-M3 ontology to Python generator	Language: Python. Network: TCP/IP, NoTA. BSD license. A part of the Smart-M3 distribution, http://sourceforge.net/projects/smart-m3/
SmartSlog	Language: ANSI C, C#. Network: TCP/IP, NoTA. GPLv2. Petrozavodsk State University (Russia), http://sourceforge.net/projects/smartslog/

An ontology library simplifies building of the KP application logic by providing the developer with the programming language-like tools for dealing with the concepts defined for a given ontology. The number of domain elements is reduced since an ontology entity consists of many triples. The library API is generic: its syntax does not depend on a particular ontology, ontology-related names do not appear in names of API methods, and ontology entities are used only as arguments.

Let us take the SmartSlog development tool [Korzun et al., 2010] as an example. The ontology library structure consists of two parts: ontology-independent and ontology-dependent. The former is the same for any KP and implements generic API to access knowledge in the smart space. The latter is produced by SmartSlog CodeGen by a given OWL description (provided by the KP developer). It implements data structures for particular ontology entities. The library internally performs OWL-RDF transformations and calls a low-level KPI (e.g., KPI_Low) for data exchange with the SIB. If the low-level KPI is in a different language then wrappers are needed for corresponding calls.

Code generation uses a Jena-based back-end for analyzing ontology. Compared with the other generators, SmartSlog is more concerned with restrictions of low-end devices. It keeps dependencies to minimum and memory usage is predictable and bounded. Furthermore, SmartSlog is focused on efficiency optimization. For example, the search requests are written compactly by defining only what is needed for or known about the object that has to be found.

6. Conclusion

This paper addresses the key area of future services and applications studies, as provision of the well defined and easy to use tools for smart applications development is extremely important and shall boost the whole service industry. Implementation of the ubiquitous computing vision will affect a broad set of heterogeneous and transiently available devices around us. Allowing these devices to easily share information with other devices and architectures is a key requirement. The paper gives an overview of the Smart-M3 platform and the developed tools to support the new paradigm of programming smart devices, which allow effectively participating in various smart applications. The paper also introduces our recent tool called SmartSlog, which can be seen as a kind of ADK to simplify and easily productize future development of the Smart-M3 based solutions. The SmartSlog tool is already used by the Smart-M3 consortia and we expect that in the near future, thanks to

it more of independent developers will recognize new opportunities and horizons provided by the smart spaces concept and the Smart-M3 platform.

References

[Balandin et al., 2009] S. Balandin and H. Waris, “Key properties in the development of smart spaces,” in Proc. 5th Int’l Conf. Universal Access in Human-Computer Interaction. Part II: Intelligent and Ubiquitous Interaction Environments, ser. UAHCI’09. Springer-Verlag, 2009, pp. 3–12.

[Cook et al., 2007] D. J. Cook and S. K. Das, “How smart are our environments? an updated look at the state of the art,” Pervasive and Mobile Computing, vol. 3, no. 2, pp. 53–73, 2007.

[Eugster et al., 2003] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe,” ACM Comput. Surv., vol. 35, pp. 114–131, June 2003.

[Fensel, 2004] D. Fensel, “Triple-space computing: Semantic web services based on persistent publication of information,” in Proc. IFIP Int’l Conf. Intelligence in Communication Systems (INTELLCOMM 2004), LNCS 3283. Springer, November 2004, pp. 43–53.

[Gelernter 1985] D. Gelernter, “Generative communication in linda,” ACM Trans. Program. Lang. Syst., vol. 7, pp. 80–112, January 1985.

[Honkola et al., 2010] J. Honkola, H. Laine, R. Brown, and O. Tyrkk¨o, “Smart-M3 information sharing platform,” in Proc. IEEE Symp. Computers and Communications, ser. ISCC ’10. IEEE Computer Society, Jun. 2010, pp. 1041–1046.

[Korzun et al., 2010] D. Korzun, A. Lomov, P. Vanag, J. Honkola, and S. Balandin, “Generating modest high-level ontology libraries for Smart-M3,” in Proc. 4th Int’l Conf. Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2010), Oct. 2010, pp. 103–109.

[Liuha et al., 2009] P. Liuha, A. Lappetelainen, and J.-P. Soininen, "Smart objects for intelligent applications - first results made open," in ARTEMIS Magazine, no. 5, pp. 27–29, Oct. 2009.

[Luukkala et al., 2010] V. Luukkala and J. Honkola, “Integration of an answer set engine to smart-m3,” in Proc. 3rd Conf. Smart Spaces (ruSMART’10) and 10th Int’l Conf. Next Generation Wired/Wireless Networking (NEW2AN’10), ser. ruSMART/NEW2AN’10. Springer-Verlag, 2010, pp. 92–101.

[Martin-recuerda, 2005] F. Martın-Recuerda, “Towards Cspaces: A new perspective for the Semantic Web,” in Proc. 1st IFIP WG12.5 Working Conf. Industrial Applications of Semantic Web, M. Bramer and V. Terziyan, Eds., vol. 188. Springer, August 2005, pp. 113–139.

[Nixon et al., 2008] L. J. B. Nixon, E. Simperl, R. Krummenacher, and F. Martinrecuerda, “Tuplespace-based computing for the semantic web: A survey of the state-of-the-art,” Knowl. Eng. Rev., vol. 23, pp. 181–212, June 2008.

[Oliver et al., 2009] I. Oliver and S. Boldyrev, “Operations on spaces of information,” in Proc. IEEE Int’l Conf. Semantic Computing (ICSC’09). IEEE Computer Society, Sep 2009, pp. 267–274.