Smart-M3-Based Robots Self-Organization in Pick-and-Place System

Alexander Smirnov[*†], Alexey Kashevnik[*†], Nikolay Teslya[*†], Sergey Mikhailov [*†], Anton Shabaev[‡]
[*]SPIIRAS, St.Petersburg, Russia
[†]ITMO University, St.Petersburg, Russia
[‡]Petrozavodsk State University (PetrSU), Petrozavodsk, Russia
{smir, alexey, teslya}@iias.spb.su, mikhaylovsergeyandreevich@gmail.com, ahabaev@petrsu.ru

***Abstract*—This paper presents an approach for robots self-organization for pick-and-place scenario based on Smart-M3 information sharing platform that provide possibilities of information sharing between different services in smart space. In scope of the approach the reference model for robots self-organization has been developed. To provide semantic interoperability, the ontologies for the robots participating in the scenario, have been built. The scenario implementation is based on Lego® Mindstorms EV3 set for robot construction, which is one of the most popular sets for education at the moment.**

I. INTRODUCTION

Self-organising systems are characterised by their capacity to spontaneously (without external control) produce a new organisation in case of environmental changes. These systems are particularly robust, since they adapt to changes, and are able to ensure their own survivability [1].

Self-organization of robots requires cyber-physical infrastructure allowing robots to operate in physical part while their interaction has to be organized in cyber part. Cyber-physical systems are spreading wide all over the world [2-5]. They are based on real time interaction between physical world and cyber world. Such systems rely on communication, computation and control infrastructures typically consisting of several levels for the two worlds with various resources as sensors, actuators, computational resources, services, etc.

One of the examples of cyber-physical system is a "smart home" system, which implements room cleaning scenario by automatically controlled robots interacting with each other [6]. In this scenario the self-organization of three types of robots is considered: (i) robot vacuum cleaner that can create room map using the light-sensitive sensors and clean the room using the map (e.g. Yujin Robot iClebo Arte YCR-M05), (ii) manipulating robot (e.g. FESTO Robotino XT), allowing to move light furniture (e.g. chairs, or coffee table) or decorative elements (pots with flowers, vases, etc.) for efficient cleaning, (iii) adaptive illumination control system for illumination level changing, depending on the current situation. All of them should interact with each other to solve different tasks. For example, vacuum cleaner needs light to create a room map. Therefore, it should "ask" illumination control system to raise illumination level in the room. Vacuum cleaner may find different obstacles while cleaning, and will have to "ask" manipulating robot to move them away until the cleaning has ended.

Scenario, developed in this paper, solves the task of pick-and-place an object from one point to another. Two robots participate in the scenario: the manipulating robot and the pipeline robot. The pipeline robot scans object characteristics and provides the object to the manipulating robot. The manipulating robot takes the object and based on it characteristics moves it to another place.

Presented scenario enhances the research work [7], which provides scenario for object searching by robots and organizes their behavior. Robots have to decide, which of them has to approach the object and bring it to the storage. The regular scenario can describe this example. Two or more robots receive a task to act, e.g. find an object and bring it to a storage. Only one robot should handle this task. For this purpose, robots should interact to decide, which one will bring the object to the storage. Interaction process includes the object finding, distances measurement and information sharing between robots.

For self-organization of the robots the smart space technology is used, which allows to provide information for sharing between different services of the system. This technology [8], [9] aims in the seamless integration of different devices by developing ubiquitous computing environments, where different services can share information with each other, make different computations and interact for joint tasks solving.

The open source Smart-M3 platform [10] has been used for organization of smart space infrastructure for robots self-organization. The use of this platform enables to significantly simplify further development of the system,

ISSN 2305-7254

include new information sources and services, and to make the system highly scalable. The key idea of this platform is that the formed smart space is device-, domain-, and vendor-independent. Smart-M3 assumes that devices and software entities can publish their embedded information for other devices and software entities through simple, shared information brokers. The Smart-M3 platform consists of two main parts: information agents and kernel 0. The kernel consists of two elements: Semantic Information Broker (SIB) and information storage. Information agents are software entities, installed on mobile devices of the smart space users and other devices, which host smart space services. These agents interact with SIB through the Smart Space Access Protocol (SSAP). The SIB is the access point for receiving the information to be stored, or retrieving the stored information. All this information is stored in the information storage as a graph that conforms with the rules of the Resource Description Framework (RDF) [12] and [13]. In accordance with these rules all information is described by triples "Subject - Predicate - Object".

For the robot constructing, authors are suggested to use the robot kits applicable in the education process, which allow to design and built robots from modules. The kit should meet several requirements to provide functionality for the scenario implementation: it should include at least distance and brightness sensors, gyroscope, a set of motors, and powerful control block with programming possibilities and wireless network connection to join the smart space. Lego® Mindstorms[1] is one of the most popular sets in robot constructing education. The latest system, called the Lego® Mindstorms EV3, was released on September 1, 2013 [14]. A standard Education EV3 Core set consists of the following components (Fig. 1 shows only control block with available sensors and motors):

- Control block called Brick (ARMv9 core CPU 300 MHz, 64 Mb RAM, 16 Mb flash memory and microSDHC port (supports microSD and microSDHC memory cards with capacity up to 32 Gb), USB host, WiFi through USB dongle, Bluetooth, speaker, LCD display and 6 hardware buttons).
- Motors (one medium motor and two large motors).
- Sensors: ultrasonic sensor (for distance measurement, from 3 to 150 cm), touch sensor (for handling touching), gyroscopic sensor (for measurement of turn angle and turn acceleration), light sensor (for brightness measurement and color detection).
- More than 550 Lego parts.

[1] http://mindstorms.lego.com

Fig. 1. Lego® Mindstorms EV3 Education kit

Infrared sensor with remote control and third-party sensors developed by other companies can be added. Up to four EV3 control blocks can be connected using a USB cable and thereby enabling robot to have sixteen output ports and sixteen input ports all controlled from the main EV3 Brick.

Lego® Mindstorms has own IDE for robot actions programming. It allows to program actions by graphical programming language using blocks with customized parameters. These blocks cover all robot functionality, but for the advanced features it is possible to use development environment with high-level program languages. Robot can also be controlled manually from iOS[2] and Android[3] applications via Bluetooth connection.

The rest of the paper is structured as follows. Analysis of operation systems for Lego® Mindstorms EV3 control block is presented in Section II. Section III describes developed reference model for robots self-organization. Section IV presents developed ontological models of two types of robots participating in the pick-and-place scenario. Scenario implementation is presented in Section V. The results are summarized in Conclusion.

II. Operating Systems for Lego® Mindstorms EV3 control block

There are many projects providing usage of the different program languages support for EV3[4]. One of them provides environment for compiling programs under existing control brick's OS (e.g. NXTGCC, Lego.NET, different libraries for GCC, etc.) The second class of projects provides controlling the EV3-based robot using different languages through the Bluetooth and/or USB interfaces (NXT_Python, OCaml-mindstorm, LabVIEW, etc). The

[2] https://itunes.apple.com/app/lego-mindstorms-robot-commander/id681786521

[3] https://play.google.com/store/apps/details?id=com.lego.mindstorms.robotcommander

[4] http://en.wikipedia.org/wiki/Lego_Mindstorms#Programming_languages_2

third class provides a replacement of the existing OS. Due to the fact, that Lego® Mindstorms EV3 control block is based on the ARMv9 processor and the main OS is Linux-based, it is possible to run another Linux-based OS, that is built for ARM architecture. Using Linux-based OS allows writing programs with any supported programming language. There are two options: replacement of the kernel embedded into the control block (ROBOTC), and installing additional OS on SD-card without replacing the existing OS (brickOS, LeJOS, ev3dev).

The use of additional OS without replacing the existing is more preferred due to the fact that it cannot corrupt the data on the brick. Existing OS still can be booted and used. Interaction with control block in the case of replacement is faster than through the Bluetooth interface.

One of the Linux-based systems is a project called brickOS[5]. This is an open-source embedded operating system providing a C and C++ programming environment for the Lego® Mindstorms Robotics Kits.

The alternative to brickOS is a project called LeJOS[6] (acronym from Lego Java Operating System). It provides the full featured embedded OS with GUI. LeJOS provides Linux environment with Java Runtime Environment. LeJOS Java bindings implement access to the robot's hardware. It allows Lego Mindstorms robots to be easily programmed using Java language. Some of LeJOS main features are [14]:

- object oriented language (Java);
- pre-emptive threads (determined context switching);
- (multi-dimensional) arrays;
- synchronization;
- exceptions;
- types of variable including float, long, and String;
- most of the standard Java classes are available;
- well-documented robotics APIs.

The ev3dev[7] is another interesting project, which provides a Debian-based OS for EV3. It provides a console system available through SSH, which can be updated and extended by using extensive Debian repository. For hardware access, ev3dev provides drivers and language bindings for C++, Google Go, Node.js and Python programming languages. Other languages also can be used, but interface libraries then need to be implemented by yourself.

Due to the Linux kernel, all systems include firmware for the wireless network USB-adapters. It makes possible to connect robots to the local area network and unite them to the smart space. For the robot realization, the LeJOS has been used, because it provides the most powerful software development environment (threads with synchronization, standard classes and types of variable) as well as wide device type support (all native devices and 3rd party sensors and motors).

III. Reference Model

Fig. 2 presents the reference model of the pick-and-place system, proposed in the paper. The system is divided into two main paths: physical space and smart space. Two kinds of robots interact with each other in the physical space: the manipulating robot and the pipeline robot. Each robot has physical part, which implements manipulations in the physical space, and control service, which interacts with other services in the smart space and controls the physical part. People, observing and interacting with robots in the physical space, can control them through a smart space service installed on their personal mobile device. Information services interact with other services in smart space. They provide calculations and different types of information for the robots and people.

To provide semantic interoperability between robots their interaction in smart space is based on ontologies. Each robot uploads its ontology to the smart space ontology library when it connects to the system. The ontology represents the robot. It contains information about robot requirements and possibilities. Robot requirements represent the information, which the robot needs for starting its scenario. Robot possibilities is the information that robots can provide in scope of the considered system.

Proposed robots interaction scheme in the smart space is presented in Fig. 3. $Robot_1$ and $robot_2$ connect to the system and upload their ontologies to smart space ontology library. For this purpose special services are used that implement robots logic and interaction in the smart space.

When a service has information that can be helpful for other services in the smart space, it uploads this information according to previously uploaded own ontology. If a service requires information for performing an action according to the system scenario, it uses the ontology matching service to determine, if needed information is accessible in the smart space or not. Ontology matching service [15] and [16] implements matching of the service ontology with the smart space ontology library and determines, which information in the smart space corresponds to the required one. If the ontology matching service finds this information, the service downloads it from the smart space.

[5] http://brickos.sourceforge.net/
[6] http://www.lejos.org/
[7] https://github.com/ev3dev

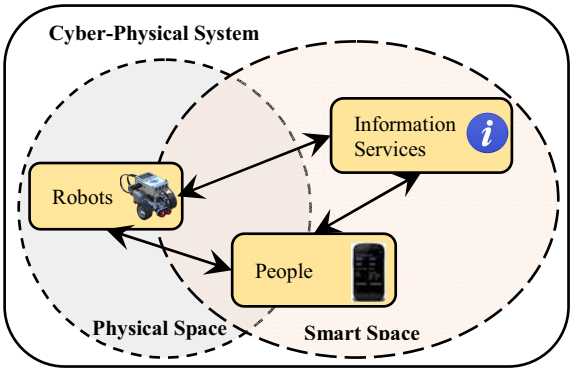

Fig. 2. Robots Interaction Reference Model in Cyber-Physical System

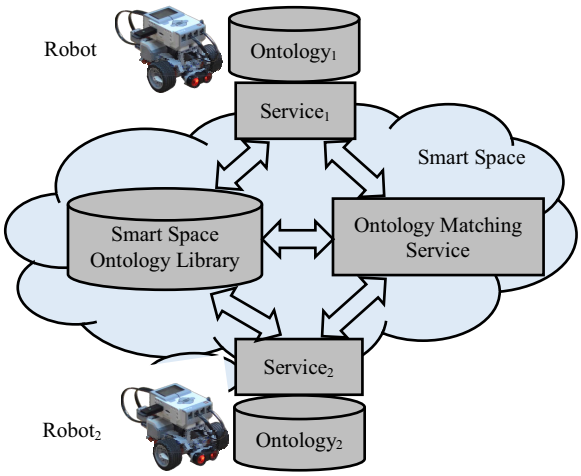

Fig. 3. Robots Interaction in Smart Space Based on Ontology Matching

IV. Robot Ontologies

Authors present their expertise in ontology development for cyber-physical systems in [17]. The manipulating robot ontology consists of the following main classes (see Fig. 4). There are three top level classes: *"Actions"* (actions the manipulation robot implements), *"Sensors"* (sensors installed in the manipulating robot), and *"Object"* (object the manipulating robot moves). Class *"Actions"* is classified as movements (class *"Movement"*) and gripping (Class *"Grip"*). There are two sensors is installed to the manipulating robot (gyroscope and ultrasonic sensor). In the proposed ontology the corresponding classes (*"Gyroscope"* and *"Ultrasonic"*) is associated with class *"Sensors"* with relationship *"is_a"*. At the same time manipulating robot movement requires gyroscope and ultrasonic sensor (classes *"Gyroscope"* and *"Ultrasonic"* is associated with class *"Movement"* with relationship *"requires"*). Type of object determines position where the object has to be placed (class *"Object"* is associated with class *"Movement"* with relationship determines). Gripper of manipulating robot has to take the object and move it (class *"Grip"* is associated with class *"Object"*).

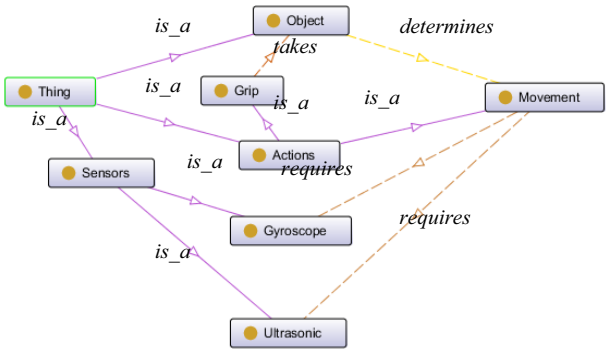

Fig. 4. Manipulating Robot Ontology

The following properties have been defined in the manipulating robot ontology:

- Property *"has_color"* for the class *"Object"*. List of possible values are (*"red"*, *"green"*, *"blue"*). Based on value of this property the manipulating robot determines the places where the object has to be placed.
- Property *"has_gripping_velocity"* for the class *"Grip"*. Value is a positive number that determines at the moment gripper speed. The property is used by the pipeline robot to understand that at the moment manipulating robot is taking the object and pipeline has to be remain stopped.
- Property *"has_movement_velocity"* for the class *"Movement"*. Value is a positive number that determines at the moment manipulating robot movement speed. The property can be used by other system services to determine the time when the object can be accessible in destination position.

The pipeline robot ontology consists of the following main classes (see Fig. 5). There are three top level classes: *"Movement"* (different types of movement that pipeline robot can implement), *"Sensors"* (sensors installed in the manipulating robot), and *"Object"* (object the pipeline robot moves and determine color). Class *"Pipeline"* has taxonomical relationship *("is_a")* with the class *"Movement"* as this type of movement is supported by the pipeline robot. Class *"ColorSensor"* has taxonomical relationship *("is_a")* with the class *"Sensor"* as this sensor type is installed for the pipeline robot. Sensors are installed in the pipeline (class *"Sensors"* has *"installed_in"* relationship with the class *"Pipeline"*). Object is placed in pipeline that moves it (class *"Object"* has *"placed_in"* relationship with the class *"Pipeline"*). For determining color of the object when it moves in pipeline the color sensor is used (class *"ColorSensor"* has relationship *"determine_color"* with class *"Object"*).

The following properties have been defined in the pipeline robot ontology.

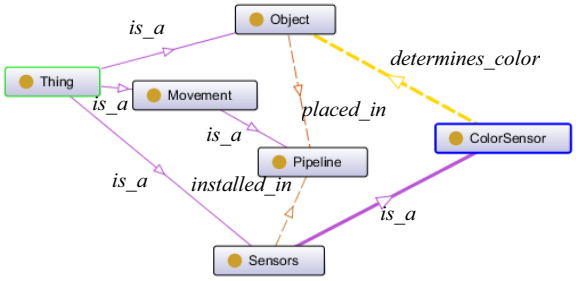

Fig. 5. Pipeline Robot Ontology

- Property *"has_color"* for the class *"Object"*. List of possible values are (*"red"*, *"green"*, *"blue"*). Based on the color sensor output the property is take on the appropriate value.
- Property *"has_velocity"* for the class *"Pipeline"*. Value is a positive number that determines at the moment the pipeline speed. The property can be used for smart space services for estimation of time when the object will be ready for manipulation.
- Property *"is_ready_for_manipulation"* for the class object. Value of this property is a Boolean variable that is *"true"* if pipeline robot is completing move the object and *"false"* if the object is moving.

V. Pick-and-Place System Scenario

Two types of robots participat in the scenario: pipeline robot (Fig. 6) and manipulating robot (Fig. 7). The first one is stationary and has a pipeline that moves objects from the location to the destination. It has a color sensor that determines the color of the moved object. When the robot is moving the object, the pipeline velocity is shared with smart space by the following triple in according with the pipeline robot ontology (see Section IV).

("Pipeline", "has_velocity", [pipeline velocity])

When the color is determined, it is shared with smart space by the following way.

("Object", "has_color", [object color])

When the object has been moved to the destination point and is ready for manipulation by the manipulating robot, the related triple is shared with smart space by pipeline robot.

("Object", "is_ready_for_manipulation", 1)

("Pipeline", "has_velocity", 0)

The second robot has possibility to load an object from pipeline at the location, move to destination, unload the object, and return to the initial location. The manipulating robot subscribes to the information in smart space if an object is ready for manipulation.

("Object", "is_ready_for_manipulation", None)

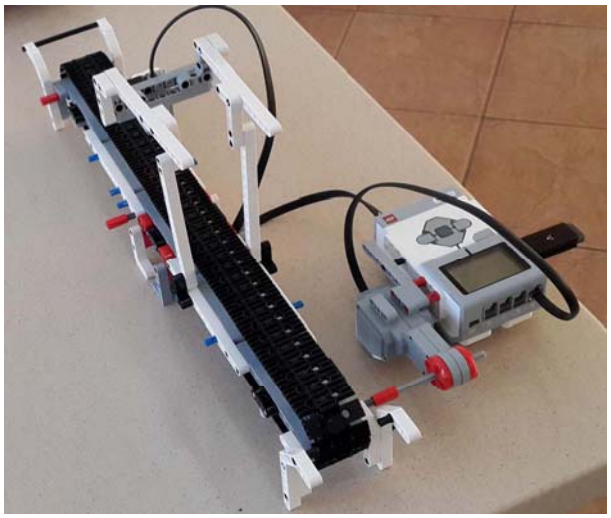

Fig. 6. Pipeline robot

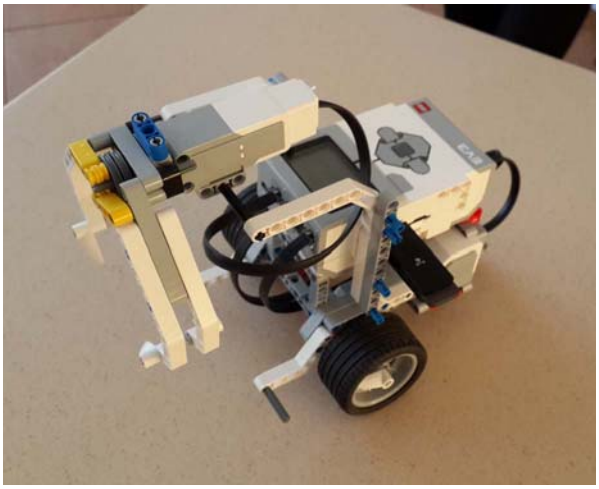

Fig. 7. Manipulating robot

When the pipeline robot moves the object to the destination point and shares with the smart space appropriate information the manipulating robot gets notification and moves to the object location. It shares with the smart space the movement velocity.

("Movement", "has_movement_velocity", velocity)

Then it takes the object using the onboard gripper. When the gripper is taking the object the robot shares the gripping velocity with the smart space.

("Grip", "has_gripping_velocity", velocity)

The manipulating robot queries information of object color from the smart space to determine the place, where the object has to be unloaded.

("Object", "has_color", None)

Then the robot moves the object to the place according to the object color obtained from smart space (Fig. 8).

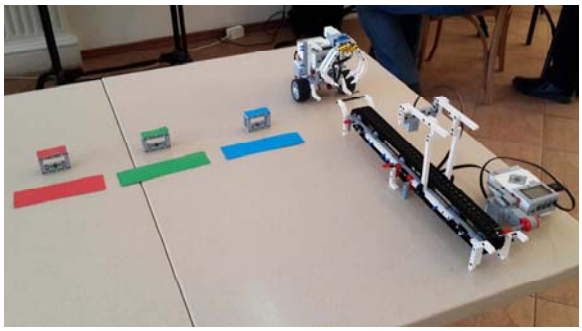

Fig. 8. Pick-and-Place System Scenario

VI. Conclusion

The paper presents reference model and scenario implementation of Smart-M3-based robots self-organization in pick-and-place system. Developed ontologies for pipeline and manipulating robots support semantic interoperability between them. Interaction between robots is organized based on smart space technology. For the implementation, the Smart-M3 information sharing platform is used, which enables to significantly simplify further development of the system, include new information sources and services, and to make the system highly scalable. Two robots are used for the pick-and-place system scenario implementation. In the future work authors plan to extend the scenario for more than two robots.

During the implemented scenario evaluation, the following problems have been identified and will be studied in the future work.

- Lego Mindstorm EV3 sensors have low accuracy, which sometimes does not allow to complete the scenario. E.g., gyroscope sensor accumulates error that causes inaccurate robot turn. In the future authors plan to use different sensors for robot positioning to increase the accuracy (e.g., for turn measurement the gyroscope, and ultrasonic sensor that measures distances to walls can be used).
- The lack or excessive room light cause inaccurate color determination by color sensor. Scenario room has to be isolated from the natural light and artificial light has to be enough for the ultrasonic sensor.
- Half-discharged battery of the robots causes unpredictable behavior of the sensors and engines.

Acknowledgment

The presented results are part of the research carried out within the project funded by grants # 13-07-00336, 13-07-12095, 13-01-00286 of the Russian Foundation for Basic Research, program P40 “Actual Problems of Robotics” of the Presidium of the Russian Academy of Sciences. This work was partially financially supported by Government of Russian Federation, Grant 074-U01, and by the Ministry of Education and Science of Russia within research project # 14.574.21.0060 (RFMEFI57414X0060) of Federal Target Program "Research and development on priority directions of scientific-technological complex of Russia for 2014-2020".

References

[1] G. Serugendo, M. Gleizes, A. Karageorgos, “Self-Organisation and Emergence in MAS: An Overview”, *Informatica*, 30, 2006, pp. 45–54.

[2] L. Monostori, “Cyber-physical Production Systems: Roots, Expectations and R&D Challenges”, *Procedia CIRP*, vol. 17, 2014, pp. 9–13.

[3] M. Mikusz, “Towards an Understanding of Cyber-physical Systems as Industrial Software-Product-Service Systems”, *Procedia CIRP*, vol. 16, 2014, pp. 385–389.

[4] Y. Chen, J. Luo, W. Li, E. Zhang, J. Shi, “Self-Organization Framework and Simulation Realization of Transportation Cyber-Physical System”, CICTP 2014, 2014, pp. 584–595.

[5] T. Li, J. Cao, J. Liang, J. Zheng, Towards context-aware medical cyber-physical systems: design methodology and a case study, Cyber-Physical Systems, 2014, Web: http://www.tandfonline.com/doi/full/10.1080/23335777.2014.972686#abstract.

[6] A.M. Kashevnik, A.V. Ponomarev, S.V. Savosin, “Hybrid Systems Control Based on Smart Space Technology”, *SPIIRAS Proceedings*, vol. 4(35), 2014, 212-226 (in Russian).

[7] N. Teslya, S. Savosin, “Smart-M3-Based Robot Interaction in Cyber-Physical Systems”, *Proceedings of the 16th Conference of Open Innovations Association FRUCT*, 2014, pp. 108-114.

[8] 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, 2007, pp. 53–73.

[9] 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*, Springer-Verlag, 2009, pp. 3–12.

[10] Smart-M3 at Sourceforge, Web: http://sourceforge.net/projects/smart-m3.

[11] J. Honkola, H. Laine, R. Brown, O. Tyrkko, “Smart-M3 Information Sharing Platform”, in Proc. *ISCC 2010, IEEE Comp. Soc.*; Jun. 2010, pp. 1041-1046.

[12] T. Berners-Lee, R. Fielding, L. Masinter, RFC 3986 – Uniform Resource Identifier (URI): Generic Syntax, URL: http://tools.ietf.org/html/rfc3986.

[13] Resource Description Framework (RDF). W3C standard, Web: http://www.w3.org/RDF/.

[14] Coming Fall 2013: LEGO MINDSTORMS EV3, URL: http://www.lego.com/en-us/mindstorms/news/2013/january/announcing-lego-mindstorms-ev3/.

[15] A. Smirnov, A. Kashevnik, N. Shilov, S. Balandin, I. Oliver, S. Boldyrev, “On-the-Fly Ontology Matching in Smart Spaces: A Multi-Model Approach”, *Proceedings of the Third Conference on Smart Spaces*, 2010, pp. 72–83.

[16] S. Balandin, S. Boldyrev, I. J. Oliver, T. Turenko, A.V. Smirnov, N. G. Shilov, A. M. Kashevnik, “Method and apparatus for ontology matching”, US 2012/0078595 A1, 2012.

[17] A. Smirnov, T. Levashova, N. Shilov, K. Sandkuhl, “Ontology for Cyber-Physical-Social Systems Self-Organisation”, *Proceedings of the 16th Conference of Open Innovations Association FRUCT*, Oulu, Finland, 27-31 October 2014, pp. 101-107.