



TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Francesc de Borja Ramis Ferrer

**An Approach to Automatically Distribute and Access
Knowledge within Networked Embedded Systems in
Factory Automation**



Julkaisu 1526 • Publication 1526

Tampere 2018

Tampereen teknillinen yliopisto. Julkaisu 1526
Tampere University of Technology. Publication 1526

Francesc de Borja Ramis Ferrer

An Approach to Automatically Distribute and Access Knowledge within Networked Embedded Systems in Factory Automation

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Festia Building, Auditorium Pieni Sali 1, at Tampere University of Technology, on the 9th of February 2018, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2018

Doctoral candidate: Francesc de Borja Ramis Ferrer
Laboratory of Automation and Hydraulic Engineering
Faculty of Engineering Sciences
Tampere University of Technology
Finland

Supervisor: Jose Luis Martinez Lastra, Prof.
Laboratory of Automation and Hydraulic Engineering
Faculty of Engineering Sciences
Tampere University of Technology
Finland

Pre-examiners: Ignacio Bravo, Prof.
Electronics Department
University of Alcala
Spain

Birgit Vogel-Heuser, Prof.
Department of Mechanical Engineering
Technical University of Munich
Germany

Opponent: Robert W. Brennan, Prof.
Department of Mechanical & Manufacturing Engineering
University of Calgary
Canada

Ramis Ferrer, Francesc de Borja: An Approach to Automatically Distribute and Access Knowledge within Networked Embedded Systems in Factory Automation

Tampere University of Technology, Faculty of Engineering Sciences, Finland 2017

Keywords: KNOWLEDGE ENGINEERING, KNOWLEDGE ACQUISITION, KNOWLEDGE REPRESENTATION, INTELLIGENT SYSTEMS, AUTONOMOUS SYSTEMS, CYBER-PHYSICAL SYSTEMS, INDUSTRIAL AUTOMATION

Abstract

This thesis presents a novel approach for automatically distribute and access knowledge within factory automation systems built by networked embedded systems. Developments on information, communication and computational technologies are making possible the distribution of tasks within different control resources, resources which are networked and working towards a common objective optimizing desired parameters. A fundamental task for introducing autonomy to these systems, is the option for represent knowledge, distributed within the automation network and to ensure its access by providing access mechanisms. This research work focuses on the processes for automatically distribute and access the knowledge.

Recently, the industrial world has embraced service-oriented as architectural (SOA) patterns for relaxing the software integration costs of factory automation systems. This pattern defines a services provider offering a particular functionality, and service requesters which are entities looking for getting their needs satisfied. Currently, there are a few technologies allowing to implement a SOA solution, among those, Web Technologies are gaining special attention for their solid presence in other application fields. Providers and services using Web technologies for expressing their needs and skills are called Web Services. One of the main advantage of services is the no need for the service requester to know how the service provider is accomplishing the functionality or where the execution of the service is taking place. This benefit is recently stressed by the irruption of Cloud Computing, allowing the execution of certain process by the cloud resources.

The caption of human knowledge and the representation of that knowledge in a machine interpretable manner has been an interesting research topic for the last decades.

A well established mechanism for the representation of knowledge is the utilization of Ontologies. This mechanism allows machines to access that knowledge and use reasoning engines in order to create reasoning machines. The presence of a knowledge base allows as clearly the better identification of the web services, which is achievable by adding semantic notations to the service descriptors. The resulting services are called semantic web services.

With the latest advances on computational resources, system can be built by a large number of constrained devices, yet easily connected, building a network of computational nodes, nodes that will be dedicated to execute control and communication tasks for the systems. These tasks are commanded by high level commanding systems, such as Manufacturing Execution Systems (MES) and Enterprise Resource Planning (ERP) modules. The aforementioned technologies allow a vertical approach for communicating commanding options from MES and ERP directly to the control nodes. This scenario allows to break down monolithic MES systems into small distributed functionalities, if these functionalities use Web standards for interacting and a knowledge base as main input for information, then we are arriving to the concept of Open Knowledge-Driven MES Systems (OKD-MES).

The automatic distribution of the knowledge base in an OKD-MES mechanism and the accomplishment of the reasoning process in a distributed manner are the main objectives for this research. Thus, this research work describes the decentralization and management of knowledge descriptions which are currently handled by the Representation Layer (RPL) of the OKD-MES framework. This is achieved within the encapsulation of ontology modules which may be integrated by a distributed reasoning process on incoming requests. Furthermore, this dissertation presents the concept, principles and architecture for implementing Private Local Automation Clouds (PLACs), built by CPS.

The thesis is an article thesis and is composed by 9 original and referred articles and supported by 7 other articles presented by the author.

Acknowledgements

Certainly, this has been a thrilling journey. Back in the end of 2013, Prof. Lastra asked me if I would be interested in applying for a doctoral student position at TUT after finishing my M.Sc. Although time will tell, I think that the continuation of my academic life was the right choice. Anyway, I would like to thank the people that, somehow, made my challenges easier, my waiting shorter and my life more beautiful.

First of all, I would like to thank Prof. Lastra for everything that he has given to me. I cannot describe with words how I feel about his help, support, guidance and encouragement, simply, as indescribable as unpayable. *¡Muchas gracias profesor!*

Then, I would like to thank my colleagues at FAST-Lab. and some mates that already left looking for new challenges out of our unit. Thanks to Anne and Andrei for all what we shared during these years. Further, thanks a lot to Sergii and Wael as I truly believe that a small portion of my success, it's yours. You taught me that true friendship can be found abroad, without caring about having different language, nationality or culture. *Luisito, ahí también entras tú, ¡baby!*

I would like to thank my close friends who supported me. Thanks to my namesake, Borja, for the countless hours that we spend together even being at different locations of the world. Also, thanks to Jaumet, the master of D&D masters and to Alex, an unexpected brother in law.

Here, I want to thank my relatives. Thanks to my parents, Jose Maria and Magdalena, and my sister Constança. *Ho hem fet i direu: "eres el mejor!" Però, sa veritat es que ho som un poc tots, vos estim molt!* Thanks also to my grandmothers, still in shape, Kika and Smith. Also, I want to thank Juani and Rafael for their love to me, *siempre me habeis tratado como vuestro hijo.*

Finally, I would like to thank my lovely wife, Amalia. If this journey has been at some (many) moments difficult, she has been the one to cheer me up, to show me the way, to do whatever is needed for helping to reach the goal. Definitely, without her, this would not be possible. *Muchísimas gracias princesa, TKMHEIYS! SJ! OUI!*

Francesc de Borja Ramis Ferrer (M.Sc., B.Sc.),

Tampere, Finland

18. 12. 2017

Foreword

The research outcome reported in this thesis was performed within the Factory Automation Systems and Technologies Laboratory (FAST-Lab.) currently belonging to the Laboratory of Automation and Hydraulic Engineering, in Tampere University of Technology, Finland, during the period 2014-2017. The research leading to these results has received funding from:

- I. the Graduate School of Tampere University of Technology
- II. the ARTEMIS Joint Undertaking¹ under grant agreement n°332946 and from the Finnish Funding Agency for Technology and Innovation (TEKES), correspondent to the project shortly entitled eScop², embedded systems for service-based control of open manufacturing and process automation.
- III. the European Union's Horizon 2020 research and innovation program under grant agreement n°636909, correspondent to the project shortly entitled C2NET³, Cloud Collaborative Manufacturing Networks.
- IV. the European Union's Horizon 2020 research and innovation programme under grant agreement n°644429 correspondent to the project shortly entitled MUSA⁴, Multi-cloud Secure Applications

¹ <http://www.artemis-ju.eu/>

² <http://www.tut.fi/escop/>

³ <http://c2net-project.eu/>

⁴ <http://musa-project.eu/>

“Els grassos sempre reben”

– Francisca Martorell Martorell (my grandmother)

Contents

| | | |
|---------|--|----|
| 1 | INTRODUCTION..... | 21 |
| 1.1 | Motivation and Justification | 21 |
| 1.2 | Problem Statement and Research Questions | 23 |
| 1.3 | Methodology | 24 |
| 1.4 | Objectives..... | 24 |
| 1.5 | Contributions | 25 |
| 1.6 | Thesis Outline..... | 25 |
| 2 | LITERATURE AND TECHNOLOGY REVIEW | 27 |
| 2.1 | Artificial intelligence | 28 |
| 2.1.1 | Knowledge representation and reasoning..... | 28 |
| 2.1.1.1 | Ontology | 29 |
| 2.1.2 | Automated planning and scheduling | 30 |
| 2.2 | Distributed systems | 31 |
| 2.2.1 | Problem solving..... | 31 |
| 2.2.2 | Resource allocation..... | 32 |
| 2.3 | Cloud computing..... | 32 |
| 2.4 | Architecture, methods and tools | 33 |
| 2.4.1 | Component..... | 34 |
| 2.4.1.1 | Service oriented..... | 34 |
| 2.4.1.2 | Function block oriented | 35 |
| 2.4.1.3 | Agent oriented | 37 |
| 2.4.2 | Interaction | 37 |

| | | |
|---------|--|----|
| 2.4.2.1 | Time triggered | 38 |
| 2.4.2.2 | Event driven..... | 38 |
| 2.5 | Summary of the literature and technology review | 39 |
| 3 | AN APPROACH TO SYSTEMATICALLY DISTRIBUTE, ACCESS AND REASON KNOWLEDGE WITHIN NETWORKED EMBEDDED SYSTEMS IN FACTORY AUTOMATION..... | 41 |
| 3.1 | Knowledge-based web service integration for industrial automation (Publication I)..... | 43 |
| 3.2 | Cyber–Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems (Publication II)..... | 44 |
| 3.3 | Towards the encapsulation and decentralization of OKD-MES services within embedded devices (Publication III) | 45 |
| 3.4 | Exemplifying the Potentials of Web Standards for Automation Control in Manufacturing Systems (Publication IV)..... | 46 |
| 3.5 | Product, process and resource model coupling for knowledge-driven assembly automation (Publication V) | 47 |
| 3.6 | Private local automation clouds built by CPS: Potential and challenges for distributed reasoning (Publication VI)..... | 48 |
| 3.7 | Management of distributed knowledge encapsulated in embedded devices (Publication VII) | 49 |
| 3.8 | An Architecture for Implementing Private Local Automation Clouds Built by CPS (Publication VIII)..... | 51 |
| 3.9 | Principles and risk assessment of managing distributed ontologies hosted by embedded devices for controlling industrial systems (Publication IX)..... | 52 |
| 3.10 | Summary | 53 |
| 4 | CONCLUSIONS AND RECOMMENDATION FOR FUTURE WORKS | 57 |
| 4.1 | Concluding Remarks | 57 |
| 4.2 | Further Work..... | 58 |

REFERENCES 61

PUBLICATIONS 73

List of Figures

| | |
|---|----|
| Figure 1: Structure of the literature and technology review | 27 |
| Figure 2: An example on ontology classes and their relationships [32] | 29 |
| Figure 3: Architecture, methods and tools..... | 33 |
| Figure 4: A Function Block model [95] | 36 |
| Figure 5: Sparse time base presented in [113] | 38 |

List of Tables

| | |
|---|----|
| TABLE. I: Main results and outcomes | 53 |
|---|----|

List of abbreviations

| | |
|-------|---|
| AI | Artificial Intelligence |
| BPEL | Business Process Execution Language |
| BPMN | Business Process Modeling Notation |
| C2NET | Cloud Collaborative Manufacturing Networks |
| CAN | Control Area Network |
| CC | Cloud Computing |
| CEP | Complex Event Processing |
| CN | Collaborative Network |
| DoS | Denial of Service |
| DPWS | Device Profile for Web Services |
| DS | Distributed Systems |
| ECC | Execution Control Chart |
| EDA | Event-Driven Architecture |
| ERP | Enterprise Resource Planning |
| eScop | Embedded systems for Service-based Control of Open manufacturing and Process automation |
| ET | Event-Triggered |
| FB | Function Block |
| FIPA | Foundation for Intelligent Physical Agents |
| GCE | Google Compute Engine |
| IaaS | Infrastructure-as-a-Service |
| ICT | Information and Communication Technologies |

| | |
|---------|---|
| IEC | International Electrotechnical Commission |
| INDIN | IEEE International conference on Industrial Informatics |
| I4.0 | Industry 4.0 |
| IIoT | Industrial Internet of Things |
| IoT | Internet of Things |
| ISA | International Society of Automation |
| ISO | International Organization for Standardization |
| JCR | Journal Citation Report |
| KB | Knowledge Base |
| KR | Knowledge Representation |
| KR&R | Knowledge Representation and Reasoning |
| MAS | Multi-Agent System |
| MES | Manufacturing Execution System |
| MESA | Manufacturing Execution Systems Association |
| MSO | Manufacturing System Ontology |
| OWL | Ontology Web Language |
| OWL-S | Ontology Web Language for Services |
| PaaS | Platform-as-a-Service |
| OASIS | Organization for the Advancement of Structured Information Standards |
| OKD-MES | Open Knowledge-Driven Manufacturing Execution System |
| OLE | Object Linking and Embedding |
| OPC | Object Linking and Embedding for Process Control |
| OPC-UA | Object Linking and Embedding for Process Control Unified Architecture |

| | |
|--------|---|
| ORL | Orchestration Layer |
| PHL | Physical Layer |
| PLAC | Private Local Automation Cloud |
| PPR | Product Process and Resource |
| PLC | Programmable Logic Controller |
| REST | Representational State Transfer |
| RDF | Resource Description Framework |
| RPL | Representation Layer |
| RPL-S | Representation Layer Service |
| SaaS | Software-as-a-Service |
| SCADA | Supervisory, Control and Data Acquisition |
| SOA | Service-Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SPAQRL | SPARQL Protocol and RDF Query Language |
| SPARUL | SPARQL Update Language |
| SWRL | Semantic Web Rule Language |
| TTA | Time-Triggered Architecture |
| UML | Unified Modeling Language |
| VDI | Verein Deutsche Ingenieure |
| VIS | Visualization Layer |
| W3C | World Wide Consortium |
| WS | Web Service |
| WS-* | Web Service standards |

| | |
|--------|---|
| WS-CDL | Web Service Choreography Description Language |
| WSDL | Web Services Description Language |
| XML | eXtensible Markup Language |

Refereed Publications

- I. B. Ramis Ferrer, L. Gonzalez, S. Iarovy, A. Lobov, J. L. Martinez Lastra, V. Vyatkin, and W. Dai, "Knowledge-based web service integration for industrial automation," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 733–739.
- II. S. Iarovy, W. M. Mohammed, A. Lobov, B. Ramis Ferrer, and J. L. Martinez Lastra, "Cyber–Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1142–1154, May 2016. (JCR Q1, JUFO 3)⁵
- III. B. Ramis Ferrer and J. L. Martinez Lastra, "Towards the encapsulation and decentralisation of OKD-MES services within embedded devices," *Int. J. Prod. Res.*, May 2017. doi: 10.1080/00207543.2017.1328141. (JCR Q1, JUFO 1)
- IV. B. Ramis Ferrer, S. Iarovy, W. M. Mohammed, A. Lobov, and J. L. Martinez Lastra. 2016. "Exemplifying the Potentials of Web Standards for Automation Control in Manufacturing Systems." *International Journal of Simulation Systems, Science & Technology* 17 (33): 3.1–3.12. doi:10.5013/IJSSST.a.17.33.03.
- V. B. Ramis Ferrer, B. Ahmad, D. Vera, A. Lobov, R. Harrison, and J. L. M. Lastra, "Product, process and resource model coupling for knowledge-driven assembly automation," *at-Automatisierungstechnik*, vol. 64, no. 3, pp. 231-243, 2016.
- VI. B. Ramis Ferrer and J. L. Martinez Lastra, "Private local automation clouds built by CPS: Potential and challenges for distributed reasoning," *Adv. Eng. Inform.*, vol. 32, pp. 113–125, Apr. 2017. (JCR Q1, JUFO 1)
- VII. B. Ramis Ferrer, S. Iarovy, L. Gonzalez, A. Lobov, and J. L. Martinez Lastra, "Management of distributed knowledge encapsulated in embedded devices," *Int. J. Prod. Res.*, vol. 54, no. 18, pp. 5434–5451, Sept. 2016. (JCR Q1, JUFO 1)
- VIII. B. Ramis Ferrer and J. L. Martinez Lastra, "An Architecture for Implementing Private Local Automation Clouds Built by CPS," in *IECON 2017 – 43rd Annual Conference on IEEE Industrial Electronics Society*, 2017.
- IX. B. Ramis Ferrer, S. O. Afolaranmi and J. L. Martinez Lastra, "Principles and risk assessment of managing distributed ontologies hosted by embedded devices for controlling industrial systems," in *IECON 2017 – 43rd Annual Conference on IEEE Industrial Electronics Society*, 2017.

⁵ JCR is the international Journal Citation Report quartile ranking and JUFO is a Finnish classification system for assessing the quality of the research output.

Author's Contribution on Refereed Publications

Publication I "Knowledge-based web service integration for industrial automation"

The scientific work was done by the doctoral student in collaboration of a set of research fellows within the research scope of the eScop project. The main contribution of the doctoral student was the development of the ontology and the web-based interface as well as leading the performance of the presented experiment. The work was supervised by the eScop technical project coordinator Dr. Lobov and by the close supervisor Prof Lastra. On the other hand, M.Sc. Gonzalez and M.Sc. Iarovyi provided programming support for the encapsulation as service of different component functionality. Prof. Vyatkin and Dr. Dai revised the manuscript and final results, and presented the work in the 12th IEEE International Conference on Industrial Informatics (INDIN), 2014. Furthermore, it is interesting to mention that Dr. Lobov exploited and disseminated the results reported in the article in the ARTEMIS-ITEA2 Co-Summit.

Publication II "Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems"

This is the only publication wherein the doctoral student does not have the corresponding author role. Nevertheless, this work is included in this doctoral thesis due to the importance of the provided results and collaboration which, indeed, served as inspiration for the thesis topic. The doctoral student contributed to the required research and performance of distributed systems and knowledge-driven approach paper sections. Moreover, the doctoral student belonged to the OKD-MES development team, more precisely in the part of designing and developing ontology-based experiments for the OKD-MES approach. M.Sc. Iarovyi was the main author of the publication and was involved in all parts of the presented research work. Then, M.Sc. Mohammed contributed in both development and description of the FASTory Simulator, which is described in [1]. Finally, the research was supervised by the eScop technical project coordinator Dr. Lobov and by the close supervisor Prof Lastra.

Publication III “Towards the encapsulation and decentralization of OKD-MES services within embedded devices”

The work reported in this manuscript was contributed by the doctoral student, supervised by Prof. Lastra.

Publication IV “Exemplifying the Potentials of Web Standards for Automation Control in Manufacturing Systems”

The scientific work was done by the doctoral student in collaboration of a set of research fellows within the research scope of the eScop project. The doctoral student belonged to the eScop development team, more precisely in the part of designing and developing ontology-based experiments for the eScop project proof of concepts. Also, as main author of this publication, the doctoral student wrote the majority of the article. Some effort on writing the article as well as the provision of examples on different web standards were provided by M.Sc. Iarovy. In addition, M.Sc. Mohammed contributed in both development and description of the FASTory Simulator. The work was supervised by the eScop technical project coordinator Dr. Lobov and by the close supervisor at TUT, Prof Lastra. Furthermore, it is interesting to mention that this journal article was performed after receiving and invitation for extending an article presented in the UKSim-AMSS 9th IEEE European Modelling Symposium on Mathematical Modelling and Computer Simulation [2].

Publication V “Product, process and resource model coupling for knowledge-driven assembly automation”

This international collaboration was proposed and led by the doctoral student. The scientific writing was also mainly done by the doctoral student but with the support of Dr. Ahmad. Moreover, the University of Warwick as an international collaborator, provided a testbed and supervision on the developments related to their industrial equipment. Therefore, Dr. Ahmad, Dr. Vera and Prof. Harrison were involved in the supervision of the approach implementation. Finally, from the side of Tampere University of Technology, the research and results were supervised and revised by Dr. Lobov and by the close supervisor Prof Lastra. Furthermore, it is interesting to mention that this publication is a research result from previous collaboration also led by the doctoral student in the same topic [3]–[5]. Currently, both organizations are working together in the same line of research and looking forward the publication of accepted manuscripts and the performance of more articles showing further results in the area.

Publication VI “Private local automation clouds built by CPS: Potential and challenges for distributed reasoning”

The work reported in this manuscript was contributed by the doctoral student, supervised by Prof. Lastra.

Publication VII “Management of distributed knowledge encapsulated in embedded devices”

The work reported in this manuscript was mainly contributed by the doctoral student. Nevertheless, M.Sc. Iarovyi supported with the research and discussion of ways to design the behavior of devices and M.Sc. Gonzalez supported with descriptions of the embedded devices for both the performance of the experiments and related information included in the article. The work was supervised by the eScop technical project coordinator Dr. Lobov and by the close supervisor Prof Lastra.

Publication VIII “An Architecture for Implementing Private Local Automation Clouds Built by CPS”

The work reported in this manuscript was contributed by the doctoral student, supervised by Prof. Lastra.

Publication IX “Principles and risk assessment of managing distributed ontologies hosted by embedded devices for controlling industrial systems”

The work reported in this manuscript was contributed by the doctoral student, supervised by Prof. Lastra. In addition, Mr. Afolaranmi had an important role in the research work performance since he supported in the research and documentation of related work on security as well as in the performance of the threat modelling and risk assessment.

1 Introduction

This section presents the research problem addressed by this dissertation. The chapter begins with the motivation and justification for this doctoral research. Then, the section introduces the methodology and the main research questions derived from it. The chapter is finished by outlining the main contributions achieved by this research.

1.1 Motivation and Justification

Technology is a main player in the development of the industrial world. The industrial revolution was driven by the mechanization of tasks and process at the factory floor, consecutively the introduction of electrification and informatization played a pivotal role for arriving to the current industrial solutions. The German Academy of Science and Technology numbers these as the first, second and third Industrial Revolutions. Technologies are mechanisms for facing problems, the current global volatile markets, shorter product life cycles and across-the -broad supply chains are calling for new solutions. These solutions are according to Germany the introduction of Internet of Things and Services into the manufacturing world, this new era is called Industrie 4.0 [6]–[9]. Germany is not alone trying to tackle these societal and economical issues affecting the manufacturing sector by increasing the presence of "smart" solutions, and similarly other European countries are working towards the Factories of the Future [10]. Western countries are not the exception working in these issues, for example China created a large technological program named Made in China 2025 basically addressing the same issues and embracing the same potential technological solutions [11]–[13].

Within the scope of these large technological programs, factory automation plays a major role contributing the entire digitalization of the product life cycle. Transparent factories, capable of reacting the customer needs and automatically trigger production capabilities are an area of major attention. In such scope, the factory automation domain seeks for novel types of Manufacturing Execution

Systems (MES) [14] which are interoperable with Web Service (WS) technologies that enable the orchestration of machine operations at factory shop floor. WS technologies are implementable into current embedded devices within few protocols, such as the Device Profile for Web Services (DPWS) which, in turn, enables the implementation of Service Oriented Architectures (SOA) [15].

Recently, the industrial world has adopted architectural SOA patterns for reducing the integration costs of factory automation systems. WS-enabled devices can act as gateways with factory shop floor machines and higher management and controlling systems as Supervisory, Control and Acquisition (SCADA), MES or Enterprise Resource Planning (ERP) systems, following the automation pyramid described by the ANSI/ISA95 [16]. In this scenario, and due to the fact that service requesters do not need to know where and how the required functionality is achieved by service providers, systems located at different levels of manufacturing enterprises can control and monitor factory shop floor operations.

For the last decades, the representation of human knowledge in a machine interpretable manner has been an interesting research topic. In the industrial domain, there is a trend of applying Knowledge Representation (KR) techniques [17] in order to model and describe system capabilities and functionalities in both human and machine-readable manner. This is achieved within the use of semantic technologies, such as ontologies [18] that may be processed and extended at system run time. This permits the implementation of intelligent-based solutions that automate processes and make decisions in order to enhance the productivity of modern factories.

Through the synergy of SOA and KR, the so-called Knowledge-Driven (KD) solutions permit the control and monitoring of semantic web service operations within both retrieval and update of Knowledge Base (KB) information [19]. Conceptually, KBs are repositories that contain semantic descriptions of system components. As an example of KD solutions, the embedded systems for service-based control of open manufacturing and process automation (eScop) project generated and validated the Open Knowledge-Driven Manufacturing Execution System (OKD-MES) framework [14]. The OKD-MES framework permits the implementation and control of MES functions within web-based standards [20]. One of the main characteristics for the utilization of the OKD-MES solution is its placement on top of Cyber-Physical Systems (CPSs) [21], such as specific industrial controllers [14]. These devices contain service operation descriptions which can be invoked by a service composition engine [22] and permit the vertical communication between different levels of the automation pyramid.

On the other hand, the advent of Cloud Computing (CC) permits the abstraction of computation resources that can be remotely stored and provisioned on user demand [23]. In fact, there is a trend on applying CC paradigm concepts to the industry [24]–[26]. The implementation of Collaborative Networks (CNs) and the combination of CC and the Internet of Things (IoT) concept, which is based on the connection of all the “things” (i.e., resources), can provide the optimization of manufacturing and logistics assets of supply chains [27], [28]. Therefore, in the context of I4.0, the in-

terconnectivity of cloud-based platforms with industrial equipment can be achieved within the use of IoT-based devices. Such kind of devices are the key to push information to the cloud. One of the major benefits for the industry of using this kind of devices is that meanwhile their computational capabilities increase on new device variants, the price does not grow accordingly. In fact, although these devices are cheap, they permit the performance of computationally expensive functions, e.g., KR and Reasoning (KR&R) at system runtime.

The integration cost at modern factories can be reduced within distributed intelligence [29] and the research community presented many approaches to solve this issue during last decade. Nevertheless, the emergence of the Industrial IoT (IIoT) and the increment of the computational power of embedded devices presents a novel scenario that must be explored.

More precisely, the embedded devices (i.e., control units) that are now used for industrial applications should not act only as gateways to enable connectivity and vertical communication but also to perform more functionalities which are currently managed at upper automation levels. In other words, there is a need of exploiting the unused resources of new embedded devices that may support to reduce the high cost of integration in the industrial field.

1.2 Problem Statement and Research Questions

Integration efforts are large when factory automation systems are built by interconnecting control units. These control units, very successful at the time of executing distributed tasks, are suffering when system changes overall objectives or the system is reconfigured. An approach to solve this problem is to increase machine interpretable knowledge, yet this usually generates new problems because of the centralized approach of building the knowledge base. Thus, the implementation of a knowledge base, easily to adapt over the time and with guaranties for being accessed by all the interested parties remained as an unanswered research problem.

In order to solve the previously stated research problem, this doctoral work shall answer the following main research questions:

1. How to distribute the knowledge among the different control units building the automation system?
2. How to access the knowledge by control units located in a different control unit?

1.3 Methodology

This doctoral work follows the inductive research method. The starting point has been the observation of the current needs faced by current, and near future, Discrete Event Dynamic Automation Systems in order to cope with the demands of Smart Factories. Special attention has been given to those needs calling for easy integration of devices at the factory floor. Witnessing the latest developments of the ratio price/computational power, systems are being built by a large amount of control nodes, interconnected via fast and reliable communication networks. These nodes need to collaborate in order to achieve common goals. A proposed approach for relaxing that collaboration process is to represent knowledge in a machine interpretable manner (i.e., ontologies) in order to automatically reason and decide. While the current approach to implement knowledge is a centralized one, this thesis tries to bring that concept to the next level by providing an approach for distributing that knowledge base within a large network of control units. Furthermore, the research presents a method for accessing that knowledge at runtime.

1.4 Objectives

This research work aims to create a solution for automatically distribute, access and reason knowledge descriptions within factory automation systems built by networked embedded systems (i.e., control units). Therefore, the three main research objectives are:

- I. To identify an approach for creating and distributing semantic knowledge representations among the embedded systems that build the automation system
- II. To design a mechanism that permits embedded devices to access knowledge descriptions hosted by different control units
- III. To develop a goal-oriented mechanism for integration of semantic resources
- IV. To propose an infrastructure that is capable of handling a set of networked control units which are capable of hosting, managing, accessing and integrating semantic descriptions at runtime

1.5 Contributions

The following is a list of main contributions of this research.

- I. A structure for ontologies in order to support reusability and expandability for describing factory manufacturing systems and devices' capabilities
- II. A methodology for integrating and reasoning system's knowledge within interconnected resource-constrained control nodes
- III. A reference architecture for implementing Private and Local Automation Clouds as a consequence of interconnecting networked control units targeting the distributed knowledge within the system as main base for reasoning at runtime

The doctoral research also provides another set of contributions, mainly derived as consequence of the core ones, or by the need of solving side problems associated to the main one. The main non-core contributions are:

- IV. A structure for ontologies in order to describe manufacturing equipment and services information for the OKD-MES framework
- V. A multiple domain ontology enriched with semantic rules for modelling and integrating Product, Process and Resource knowledge to be updated and accessed throughout the lifecycle of manufactured products
- VI. A solution for processing large amounts of events within ontology-based descriptions in highly dynamic environments, such as supply chain

These non-core contributions have been reported within several publications [2]–[5], [30]–[32].

1.6 Thesis Outline

The remaining of the document is structured as follows: Section 2 presents a literature and technology review in the scope of this research work. Then, Section 3 describes the design of private local automation clouds, built by CPS. Afterwards, Section 4 concludes the thesis work and suggests further work. Finally, this document presents a reproduction of each refereed publication that has been performed during the doctoral studies in order to achieve the aforementioned objectives in 1.4 Objectives.

2 Literature and Technology Review

This chapter presents a review and assessment of several computer science disciplines that are applied in the industrial automation domain. More precisely, this review focuses on the three main areas of study that are related to the presented research work: Artificial Intelligence (AI), Distributed Systems (DS) and Cloud Computing (CC). In addition, the overview of other subareas of study, such as Knowledge Representation or Problem Solving provide the necessary knowledge about relevant concepts that have been investigated, exploited and/or applied during this research work. Following Figure 1 depicts a hierarchy chart that presents the review structure of this review. Each block is separately described as subsections of this section.

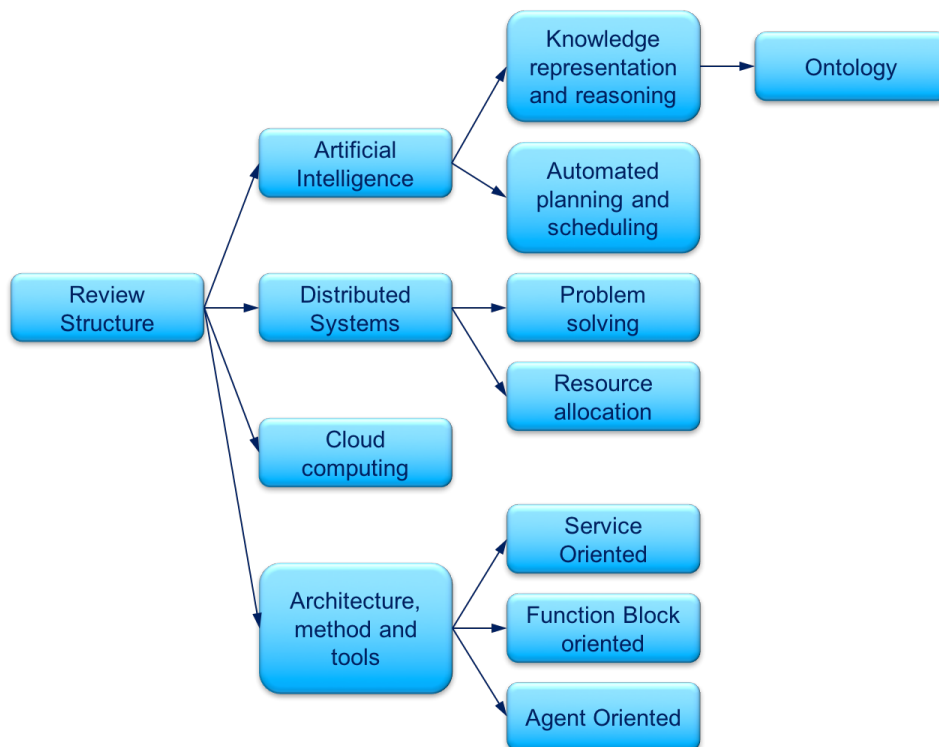


Figure 1: Structure of the literature and technology review

2.1 Artificial intelligence

Artificial Intelligence (AI) [33] is a field of study that focuses in the creation of intelligent machines and/or software. This is achieved throughout the design, implementation and deployment of intelligent behaviour that machines and/or computer programs will follow in order to perform actions. According to [33] there are many different features that are needed for achieving AI. These features are organized in several traits, which are concern in distinct aspects of AI e.g. problem solving, knowledge representation, planning, or natural language processing, among others.

2.1.1 Knowledge representation and reasoning

KR is a part of AI that is concerned with how computer systems use their knowledge about specific domains and decide what to do depending on certain situations. The domain knowledge is described as a set of statements forming a semantic repository, known as a Knowledge Base (KB). As the storage of information about the status of systems is frequently required in the industrial field to allow the adaptation and re-configurability of systems, the implementation of KR in current manufacturing systems becoming popular.

Examples of industrial automation research works that utilize a KB for storing data used for controlling processes is presented in [19], [34], [35]. Then, software and design engineers are now capable to describe physical and logical systems in a KB, which permit not only the collection of system data but also its accessibility from interested parties. It should be noted that, due to the abstraction of syntax that different tools offer to design KBs and because of the expressivity of employed languages in KR, the described knowledge is interpretable by both humans and machines.

The first decision for employing KR techniques to formally describe any domain knowledge to be used is the format of such representation. There are many different formalisms that can be employed e.g. ontologies, semantic nets, frames, production rules or even databases, among others. Probably, ontologies are the most current mean being used to describe knowledge in systems driven by computation of knowledge. On the other hand, semantic reasoning engines allow the conclusion of implicit knowledge that is inferred from explicit knowledge. In fact, KR is often referred as KR and reasoning (KR&R) because the field is not only concerned on describing explicit facts but inferring implicit information. This characteristic is crucial in knowledge-driven approaches because it is then possible to extend on runtime the KB with statements that are not included in design, configuration or operation phases. In addition, the inference of KB descriptions permits the validation of the model consistency, which can be altered when updating semantic descriptions with automated solutions.

2.1.1.1 Ontology

Ontologies permit the formal knowledge representation of any domain as e.g. manufacturing systems [36]. This kind of models are designed within the description of specific components that permit a rich description of the specific domain to represent semantically. The main components of ontologies are classes, relations, attributes, individuals, functions, axioms [18]. On the other hand, supporting the standardization of ontology design, there are some methodologies that guides ontology designers to implement ontological models [37]. In particular, aforementioned methodology consist on a set of steps that covers the main parts to be implemented when designing an ontology. As an example of an ontology employed in the factory automation field, Figure 2 shows an Unified Modeling Language (UML) class diagram that depicts the main objects and relationships of an ontology that belongs to the research work presented in [32].

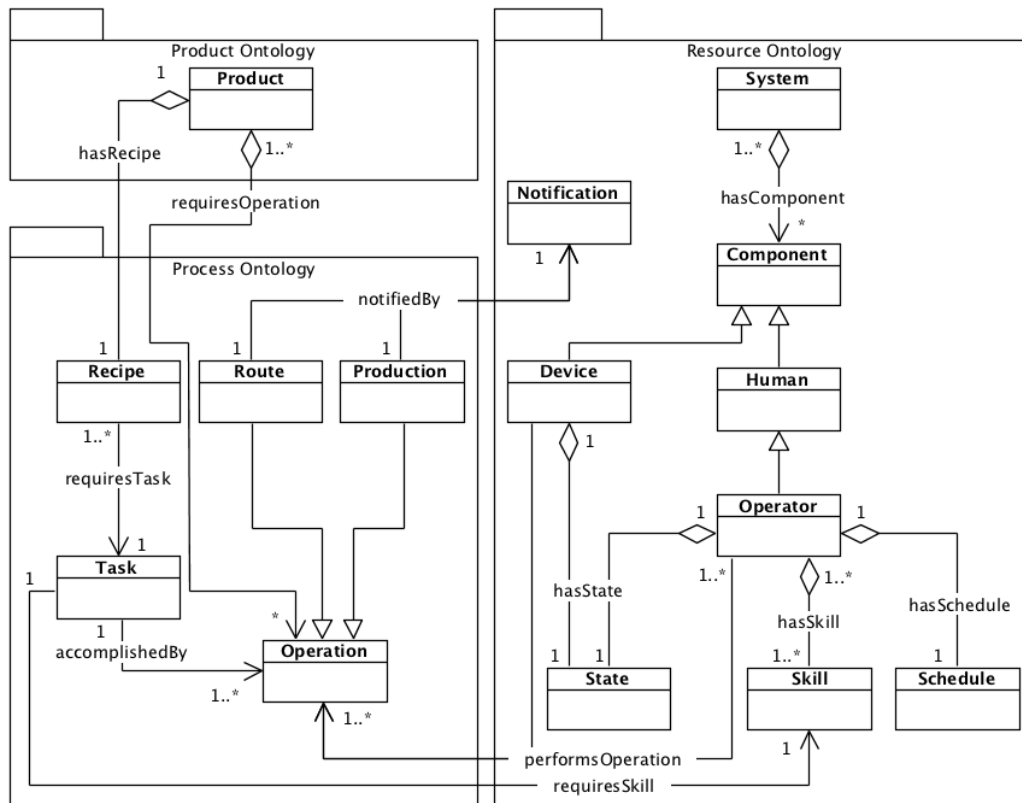


Figure 2: An example on ontology classes and their relationships [32]

There are many languages for implementing ontologies [38], [39]. Nevertheless, the common ontology language used in recent research works e.g., [34], [40] is the Web Ontology Language (OWL) [41].

OWL is presented as a mature language for implementing ontologies in the industrial automation field [42] because it offers higher degree of representation than other ontology languages as e.g. the Resource Description Framework (RDF) [43] language, which is based on the Extensible

Markup Language (XML) [44]. In fact, OWL is a RDF based language and, consequently, the use of RDF-based query languages for manipulating OWL KBs is possible.

The SPARQL Protocol and RDF Query Language (SPARQL) [45] can be employed for querying OWL models. SPARQL queries permit humans and systems to retrieve information of the knowledge. Moreover, the SPARQL Update Language (a.k.a. SPARUL) is a SPARQL extension that permit the update of the KB statements. Then, dynamic environment that uses a central repository of knowledge as e.g. manufacturing systems, can employ OWL models updated within SPARUL to actualize the status of systems [19], [46]. In addition, as ontologies can be accessed and modified on runtime, OWL model is a central component in knowledge-driven approaches, wherein behavior of systems is controlled according to the information of the system components [14].

In order to provide model inference, there are many semantic reasoning engines [47]. For example, the Pellet reasoner [48] is capable to understand and evaluate Semantic Web Rule Language (SWRL) rules which are added to the KB that, in turn, should be implemented within RDF-based languages [49]. Besides the inference of implicit knowledge, reasoning engines also provide the validation of ontologies by checking its consistency. This is particularly useful when knowledge of different domain is mapped in semantic models [49]–[51].

Aforementioned languages for implementing ontologies and rules are recommendations of the World Wide Web Consortium (W3C), which provides specifications for each language in [52]. The interrelation between different standards used for implementing ontologies can be understood within the Semantic Web, which is described in detail in [53]. Fundamentally, the concept of the Semantic Web is adopted by the industry in order to link resources in a web, which can be actually implemented within ontologies. The resulting map of semantic resources can then be used for diverse things e.g. checking dependencies of processes, discovering services or describing systems and/or services.

2.1.2 Automated planning and scheduling

Automated planning and scheduling is a part of AI that focuses on the organization and execution of activities in certain order which are needed to achieve certain goals. The sequence of activities, or plan, is normally managed by software engines that permits the monitoring and supervision of the process execution. Such plans may be created offline or even dynamically modified on runtime. The automated planning and scheduling is in relation with the decision-making of systems because the plans may vary depending on the requirements of the goal to be achieved. In addition, predefined plans can be optimized by same or external applications [27].

This area is relevant for the industrial automation because production systems are highly dynamic and require planning the execution of multiple operations to, e.g., manufacture customized products [54]. The planning and scheduling is not only concerned about the creation of products but

also in shipment and delivery, as part of the supply chain. In fact, the automated planning and scheduling is also applied in cases of failure or re-configuration of systems and unexpected events which should affect as less as possible to the value chain of the product [55].

Recent research work combines the implementation of semantic and web-based technologies for creating flexible and dynamic solutions for creating, controlling and supervising production plans [56]. Furthermore, the optimization of production plans and their distribution throughout the supply chain within a cloud-based platform is the core objective of contemporary European projects, such as the C2NET project.

2.2 Distributed systems

Distributed systems (DS) are formed by entities that are located in networked computers which coordinate their actions within the exchange of information. The handshake of messages permits DS to employ resources that are not owned by the same entity. Thus, one of the most important advantages of this systems is the possibility to have different entities that manage and share information. The main principles of distributed systems are described in [57].

In the industrial automation field, DS networks have been applied since many decades ago. In fact, besides the employment of Programmable Logic Controllers (PLCs) for distributed process control, modern production systems incorporate new types of embedded devices that allow building CPS. One of the valid approaches, which is in the scope of this research work, is the implementation of the SOA paradigm, which permits encapsulating the functionality of system components and exposing it as Web Services [58], [59]. The principles and a brief introduction on this topic is presented in 2.4.1.1 Service oriented.

2.2.1 Problem solving

Problem solving is concerned about techniques and methods that permit finding a solution for a problem. Diverse areas, such as computer science, AI, mathematics or medicine implement problem solving methods. For example, the industrial automation domain implements and applies different algorithms in order to solve specific problems [60]–[62].

Furthermore, problems can be solved within collaboration. In this scope, collective problem solving consists on utilizing the collection of efforts of multiple individuals in order to solve a specific problem [63]–[67]. As it can be seen in aforementioned research works, individuals (i.e., hardware and/or software) share and employs conjunctly their resources for solving a common problem.

2.2.2 Resource allocation

Resource allocation is a concern on different areas, e.g., computer science or project management. Conceptually, as resources are finite the resources must be employed efficiently in order to avoid delays or carrying out tasks without guarantees of a satisfactory performance. In computer science, any running application forces computers to allocate resources for such task. This may not be a concern for computers with high processing power or for performing small processes. However, resource-constrained embedded devices must be aware of the allocation of their resources due to their limitations. Then, hardware and software engineers make a great effort on researching about methods that allow the diminution of the computational power needed to perform tasks or, on the other hand, the reduction of size and cost of more powerful chips for such kind of embedded devices [68], [69]. In addition, other large systems, such as cloud-based systems, are also concerned about resource allocation due to the management of large amount of access requests to cloud resources [70].

2.3 Cloud computing

Cloud computing (CC) is a paradigm that is implemented in multiple domains, such as the industrial automation domain. Conceptually, CC permits not only the abstraction and storage of computational resources but also the remote access to such resources [71], [23]. Basically, a cloud-based model is composed by networked entities which manage and give access to their own resources. The computing resources are hosted locally by each entity, such as a server. From a higher perspective, the cloud is seen by outsiders as a unique system that contains many services that may be requested on-demand. This services can be understood as a type of WS [72].

There are three different service models of CC: IaaS, PaaS and SaaS. First, Infrastructure-as-a-Service (IaaS), e.g., the Google Compute Engine (GCE)⁶, employs virtual resources which are outsourced by an organization [73]. The users of IaaS are capable of remotely accessing and the cloud data. On the other hand, Platform-as-a-Service (PaaS), e.g., Microsoft Azure⁷, permits developers to implement their applications in the cloud. Finally, Software-as-a-Service (SaaS), e.g., Google Gmail⁸, permits users to access applications developed and distributed by different vendors within the web.

Moreover, the deployment of CC can be done in three different configurations: private, public or hybrid clouds. A comparison on such different options is presented in [74]. A private cloud is oper-

⁶ <https://cloud.google.com/compute/>

⁷ <http://azure.microsoft.com/en-us/>

⁸ <https://mail.google.com/>

ated by a specific organization although it can be hosted at internal or external location. Therefore, the services included in private clouds are offered only to users that belong or have an agreement to/with the private cloud organization. In the case of public cloud configuration, the services are available by anyone. Thus, the access to such services will be given by public networks. Finally, the hybrid configuration presents a type of cloud that is indeed a composition of several clouds. Hybrid clouds permit the combination of private and public clouds. The hybrid cloud configuration is useful when an organization needs extra computing resources which may be obtained from other clouds, such as public ones.

2.4 Architecture, methods and tools

As described along this section 2 Literature and Technology Review, there are multiple disciplines and concepts that are in relation to this research work. This subsection aims to present a collection of the most relevant methods and tools that inspired this investigation.

An architecture can be defined as a framework that allows the design, analysis and comparison of a range of systems over time [75]. The main elements of a system architecture are the components that form the system and their interactions. Therefore, the methods and tools to be considered in the scope of this research work are presented as i) components and ii) interactions of an architecture. This is depicted in Figure 3.

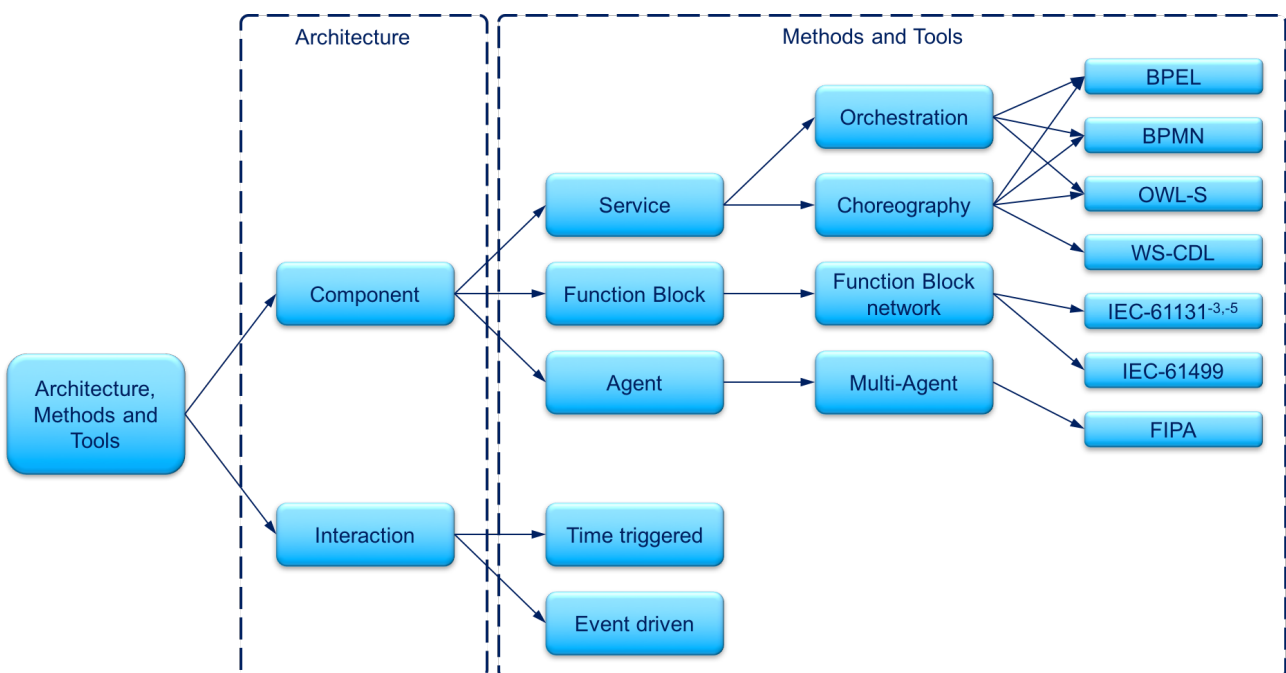


Figure 3: Architecture, methods and tools

2.4.1 Component

This research work considers three different components that can inspire system architectures: Service, Function block and Agent. Furthermore, as presented in [75], there are other types of components that might be included, such as API and Object.

2.4.1.1 Service oriented

Service-oriented architectures [59] permit the encapsulation of system functionality and its exposure in form of WS in the industrial automation domain [34], [76]. Some different applications of WS in the factory automation field are cross-layer communication and integration, which is needed in very dynamic environments wherein heterogeneous systems inhabit in collaboration for common goals [77]. In fact, the cross-layer integration permits the vertical communication between different International Society of Automation (ISA)-95 standard levels [16], which allow monitoring and controlling processes being performed in factory shop floors [78]. However, for achieving the complete re-configurability in large-scale systems, other concepts are still needed as the consumption and analysis of systems' knowledge to enable autonomous decision making. This can be achieved e.g. with repositories of data that become accessible to systems and even humans through WS.

WS may be deployed in industrial devices within the implementation of the WS-* from the Organization for the Advancement of Structured Information Standards (OASIS)⁹, the OPC Foundation¹⁰ OLE for Process Control Unified Architecture (OPC-UA) model or through the REST (Representational State Transfer) architectural style. Firstly, the set of OASIS standards (WS-*) include the DPWS. This specification stack is based on the Simple Object Access Protocol (SOAP) and can be implemented in combination with the Web Services Description Language (WSDL) in order to provide WS in resource-constrained embedded devices [22], [76]. Then, SOA can be handled by devices that are deployed at different layers of an enterprise for controlling and monitoring control processes which are executed at factory shop floors [79], [80]. On the other hand, the OPC-UA provides SOAP web services. In fact, DPWS and OPC-UA are both based on SOAP WS. Nevertheless, REST emerged as a natural competitor for similar implementations. In principle, REST requires a simpler infrastructure for its implementation and, thus, engineers tend now to use it more since last decade [14]. Interesting works based on OPC UA for enhancing the interoperability between automation systems can be found in [81]–[85].

Furthermore, the composition of WS can be implemented following different approaches, such as choreography and orchestration. Firstly, the orchestration suggests the composition of WS with predefined sequences that are centrally controlled by an orchestrator engine in order to execute

⁹ <https://www.oasis-open.org/>

¹⁰ <https://opcfoundation.org/>

WS operations. On the other hand, the choreography proposes a distributed and decentralized approach that allows the services to interact between themselves throughout a set of rules for exchanging messages. There are many languages for implementing WS compositions and it is important to select them according to the specific application needs and target scenarios [86]. For example, the Business Process Execution Language (BPEL) [87] is commonly used for implementing compositions of WS to be controlled by orchestrator engines [88].

Furthermore, the Business Process Modeling Notation (BPMN)¹¹ is a standard for describing business processes which maps directly to BPEL. In fact, current version of BPMN added an own XML format which makes now possible to execute BPMN based processes. On the other hand, the Web Service Choreography Description Language (WS-CDL) [89] is one standard language that can be used for modeling choreography WS compositions. The WS-CDL determines the workflow and behavior for service interaction [90]. Finally, the Ontology Web Language for Services (OWL-S) [91] may be used also for both orchestration and choreography compositions.

2.4.1.2 Function block oriented

The IEC-61499 standard is used for modeling distributed control systems [92], [93]. Principally, this standard presents Function Blocks (FBs) as the main components for the design of systems. This is why this is also referred also as the IEC-61499 function block standard. In fact, the standard's FB is an extended software unit based on the IEC-61131:3 [94], wherein the syntax and semantics of several languages for programming PLCs are described. From a high-level perspective, there are various models that form the architecture for a FB-based distributed control system: System, Device, Resource, Application, FB, distribution, Management and Operational State models. A detailed description of aforementioned models is found in [95].

Conceptually, IEC-61499 FBs are objects that create outgoing event and data flow throughout pre-defined and Execution Control Chart (ECC) and a set of algorithms. The ECC and algorithms require certain incoming event and data flow, respectively. In fact, there exist different types of FBs that depend on the defined kind of ECC and algorithms. Such algorithms can be implemented using several standard languages as e.g., Ladder Diagram or Structured Text, among others. Following Figure 4 depicts a Function Block model included in [95]. Such model shows the flow of both events and data as well as the different elements of Function Block instances.

¹¹ <http://www.bpmn.org/>

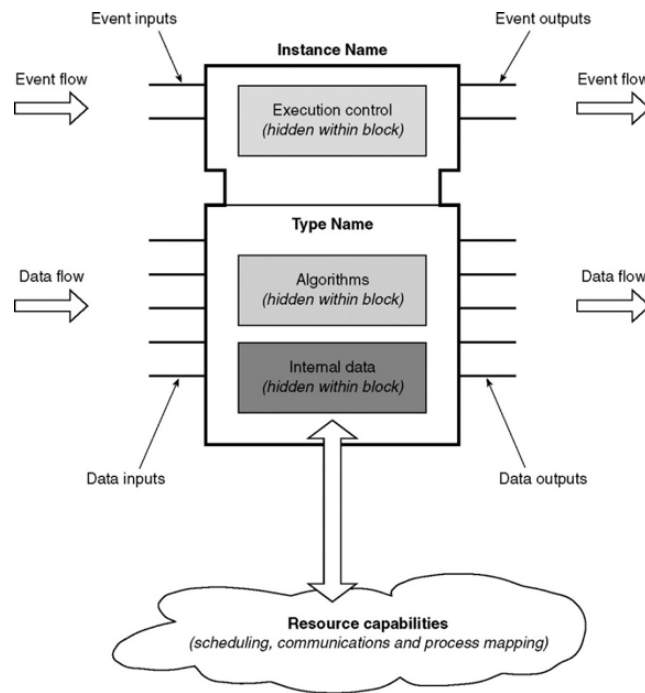


Figure 4: A Function Block model [95]

Meanwhile basic FBs defines fundamental blocks of the distributed control system being modelled; composite FBs are built by multiple FBs which are networked. Then, sub-application FB type consist on interconnected basic and composite FBs which performs part of the control of an application. This enhances the reusability of components and, then, the flexibility and re-configurability of the model with no need of editing monolithic large-sized implementations [90], [96]. In contrast to basic and composite FBs, sub-application type can be distributed in more than only one resource. This and other aspects of the interrelation between IEC-61499 models is represented in [97]. Moreover, there are other type of FBs as the Service Interface FB, for service sequences modelling, or the Adapter Interfaces, which is a special interface.

There are many research works and implementations for the industrial automation domain e.g., [93], [98]–[100]. In addition, Distributed Control Systems design and implementation within IEC-61499 is well-addressed in [101]. Previous cited works present different industrial environments wherein IEC-61499 permits the modelling of distributed control systems. But one of the most important feature to be implemented in industrial systems is the dynamic integration of heterogeneous data sources. Actually, industrial systems can be located at the same enterprise or in common supply chain of products being manufactured. This increases the complexity of solutions that must fill the gap in different environments.

2.4.1.3 Agent oriented

Inspired on different bibliographic sources that formally define the term *agent* it can be stated that an agent is an autonomous computer system that is capable of exchanging information with other peers throughout an agreed communication language [33], [102], [103]. Furthermore, the deployment of multiple agents that exchange messages and negotiate in order to work and collectively perform tasks for a common goal is known as a Multi-Agent System (MAS) [104]. According to [102], engineers that implement agents confront two types of design: agent design and society design. First, the agent design is concerned about the capabilities of agents that allow the performance of specific activities. Then, the society design establishes the behavior and procedures that the agents will develop for the exchange of information between other peers.

MAS are deployed in different domains. For example, such kind of autonomous, intelligent and dynamic systems may provide useful applications and domains e.g., game theory, travel agency systems, scheduling, logistics and industrial systems. Basically, MAS assist humans for carrying out tasks that are sometimes ineligible. Specifically, in the industrial field, MAS are usually employed for the control of distributed systems as it can be seen in following research works: [105]–[108]. Moreover, other applications in the same domain, such as integration, security or data processing may also be performed by MAS [109]. Among the benefits of MAS, aforementioned research works show enhancements for autonomous behavior, dynamism, data processing, negotiation, decision support, scalability and self-organization of agents.

Although, eventually, software engineers implement ad hoc MAS-based solutions, there are many frameworks that can be employed for implementing MAS. As one of the most known, the Foundation for Intelligent Physical Agents (FIPA)¹² is an organization that presents a set of specifications for developing agents following a standard manner. Furthermore, there are research works that provide comparisons of different MAS frameworks and tools [110], [111].

2.4.2 Interaction

As introduced at the beginning of this section, system architectures describe not only the components but also the interaction between them. This permits developers and users to understand how the different elements of the architecture share and exchange information in order to let processes to be executed. This research work presents two different types of interactions that are common in ICT-based solutions that are applied to multiple domains, such as the industrial automation field.

¹² <http://www.fipa.org/>

2.4.2.1 Time triggered

A Time Triggered Architecture (TTA) bases the execution of system tasks on a predefined schedule [112]. Similar to the description made about orchestration in section 2.4.1.1 Service oriented, a scheduler may be used for managing the execution of tasks at the required time according to the schedule.

In systems that follow the TTA, the notion of time is critical. As described in [113], one of the main characteristics of TTA is the use of real time as a primary quantity. To depict this, Figure 5 represents the sparse time base model. Basically, the real-time base is divided into time ticks which, in turn, are mapped to the duration of activities or silence, i.e. nothing occurs in such fraction of time.

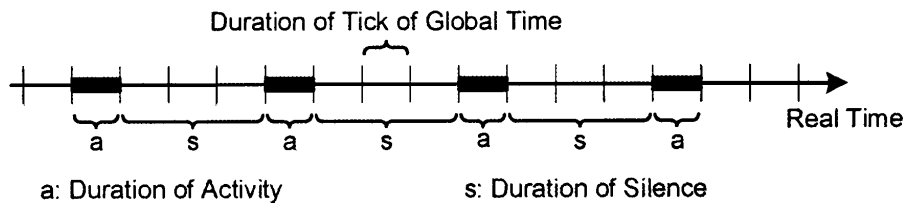


Figure 5: Sparse time base presented in [113]

One of the common applications of TTA is to develop safety-critical systems. In the industrial domain, this architecture is linked to the IEC 61508 standard [114]. Moreover, other domains, such as automotive [115] and medical systems [116] employ TTA based systems for safety applications.

Furthermore, TTA is also associated with Event Triggered (ET) or event-driven architectures. A theoretical comparison between ETA and TTA is found in [117]. In the scope of industrial automation, the aforementioned research work presents results on ET and TT based Control Area Network (CAN) measurements, as an industrial domain experiment. Following subsection introduces the event driven architectures.

2.4.2.2 Event driven

An Event Driven Architecture (EDA) is a software architecture for systems that produce, detects, consume and reacts to events in order to perform operations. Conceptually, an event is a change in the state of something that may be observed [118]. The EDA concept and its foundations are defined in [119]. Many domains have potential EDA applications, such as the industrial automation field. Nevertheless, as highlighted by [119], any domain system that produce or consumes data continuously and that requires responding to events on specific time is a potential employer of the EDA. In fact, the article [119] makes a special mention to the “*cyber infrastructures*”, clarified by the author as “*the integration of physical infrastructure with information technology*”. This, explained within current terminology, is mapped to the CPS concept, which is one of the main topics

of this research work and, indeed, refers to systems that are frequently based on event driven applications.

Furthermore, the combination of EDA and SOA provides the Event-driven SOA, which is also known as SOA 2.0. This form of SOA is extended by EDA in order to activate services by the occurrence of events. In this scope, [120] presents and describes separately SOA and EDA, and explains how the latter extends SOA and why this is important from the author perspective.

On the other hand, the order of the occurrence of events is an important knowledge for distributed systems. The research on this matter is not new [121]. Contemporary, this is covered by the area of Complex Event Processing (CEP), which is concerned about the application of techniques for processing streams of data about events that are triggered at some time. CEP emerged due to the need of exploiting the information that can be collected from the meaning, reason and time occurrence of events. This makes intelligent engines to conclude facts out of the processing of the incoming events, which may be in the range of thousands or even more depending on the environment that is monitored. Therefore, CEP requires systems that are not only capable to process a vast number of requests, but also to handle the asynchronous nature of event occurrence. Some works that present novel approaches on CEP for the industrial automation domain are [122], [123], [31], [81]. The aforementioned works are in the scope of this research because depict the combination of different areas and technologies included in this 2 Literature and Technology Review.

2.5 Summary of the literature and technology review

The previous subsections present the most relevant areas of study that are in relation with the implementation of CPS, in the scope of this research work. It can be concluded that the area of CPS is broad as it involves the knowledge of multiple disciplines. Although AI, DS and CC are not novel fields of research, there is a tendency on combining them in order to create CPS to be further deployed in the industrial domain. Conceptually, the synergy of such fields' applications enables the implementation of intelligent, autonomous and connectable systems thanks to the description and exchange of specific information about the working environment. Indeed, aforementioned characteristics are aligned with needed systems for achieving the I4.0 vision, which demands the implementation of significant ICT-based solutions.

While the technologies are prepared for performing required CPS functionalities, there is a lack of reference architectures that could be followed for implementing alike systems. Such architectures must present the different components and their interactions for performing factory automation operations. Commonly, *ad hoc* solutions are implemented for facing specific challenges. This reduces the reusability of such solutions. Therefore, the design of reference architectures could support the industrial domain with generic means for the implementation of CPS.

3 An Approach to Systematically Distribute, Access and Reason Knowledge within Networked Embedded Systems in Factory Automation

This chapter presents nine peer-reviewed manuscripts, related to this thesis work, that have been published in international journals and proceedings of international conferences which are linked to the domain of industrial automation.

Publication I contributes to the implementation of knowledge-driven solutions based on the employment of an ontology as the system KB. The provided model is used for i) describing the actual status of industrial resources and ii) provide any required information for orchestrating service operations executed at factory shop floors. In addition, the approach is presented as a web service integration that permits the encapsulation of each component as individual services. The research results served as a proof of principal concepts of the eScop project approach. Furthermore, the presented scenario was later used for teaching students of industrial informatics about features of knowledge-driven systems.

Publication II presents the so-called OKD-MES concept as a solution located on top of CPSs for controlling industrial equipment which is the main focus of the eScop project. This is shown throughout a concrete implementation in a production line previously retrofitted with web services technology [78]. One of the main and relevant facts claimed in the article is that there exist already a set of web standards and Internet-based technologies which are sufficiently mature in order to provide solutions as the OKD-MES that are fully functional, rapidly implementable and operable by end-users entirely from a web browser.

Publication III proposes to lower part of the OKD-MES framework functionality at the factory shop floor level. Conceptually, the OKD-MES concept defends a centric ap-

proach for managing and storing system semantic descriptions. Then, the article claims that such approach might be decentralized and deployed at the device level, closer to where manufacturing process data is generated within the employment of contemporary embedded devices. Then, the article i) sketches a set of diagrams that depict how some OKD-MES services can be encapsulated into such devices and ii) discusses the strengths and weaknesses of the new approach. This article that principal focus of this doctoral research.

Publication IV discusses and exemplifies the use of certain web standards that may be implemented along all the automation pyramid levels [16] for different and particular actions required for controlling and monitoring industrial process operations. The article provides a set of examples that demonstrate the implementation of web standards for enabling OKD-MES and other similar functionalities, such as service and ontology description, exchange of messages, query execution and web-based interaction and visualization.

Publication V proposes the employment of a modular ontology for integrating product, process and resource semantic descriptions. The presented approach is in the scope of model coupling and permits the matching of the requirements of products for executing assembly operations. Besides ontologies, the article proposes the employment of SWRL rules in order to link objects and instances of data models from diverse engineering domains and tools.

Publication VI presents the principles and main requirements of private local automation clouds which are built by CPS. In such context, the article defines Distributed Reasoning as a specific process that permits the integration of decentralized portions of knowledge that are located at the shop floor level throughout embedded devices. In addition, the article addresses how such devices interact in order to execute the process of distributed reasoning. On the other hand, the manuscript presents the ontology structure that permits the description of system knowledge. Finally, the potential and challenges of the approach are also provided.

Publication VII focuses on the execution of the distributed reasoning process which is required for integration of system knowledge descriptions in Private Local Automation Clouds (PLACs). First, a set of diagrams explain i) the mechanism for including devices in the PLACs, ii) the election of a device as leader of distributed reasoning processes and iii) the management and execution of a distributed reasoning process. In addition, the article presents simple scenarios for demonstrating the approach.

Publication VIII aims the presentation of an architecture for implementing PLACs which are built by CPS. In order to show a formal design of such architecture, the article shows a set of architectural views following the “4+1” view model. Then, this article presents the PLAC architecture within four views (i.e., logical, process, development and physical views) and a representative scenario. Furthermore, the article reviews relevant qualitative attributes of multiple CPS-based architectures, research works and solutions that have been published during the last years.

Publication IX suggests the implementation of techniques for protecting ICT-based solutions from different types of malicious access. More precisely, the article addresses the implementation of threat modeling and risk using the PLAC as a use case. The manuscript i) claims that presented CPS and ICT based research works do not frequently provide any kind of security validations and ii) discusses a set of recommendations for the implementation of PLACs.

3.1 Knowledge-based web service integration for industrial automation (Publication I)

The use of RDF-based models, such as OWL ontologies for describing the different kinds of information generated and consumed throughout all the automation pyramid layers may be used by other components in order to control and monitoring industrial processes. In this context, the synergy of knowledge models and web services permits the creation of knowledge driven systems.

This manuscript shows that the information included in an ontology can be queried and updated by independent services that need to exchange system information in order to execute operations. The approach distinguishes between three different component groups: *Shop floor*, *Cloud* and *Third-Party Server*. Then each service of the resulting knowledge-based web service integration approach will belong to one of such group. It should be mentioned that one of the main requirements for implementing similar solutions on modern product lines is the deployment of web service enabled controllers that permit the remote invocation of service operations.

Then, as shown in the article, the Shop floor includes an update manager, an orchestrator and the industrial controllers, which, in turn, are interconnected with the physical equipment. Without using the term, this article presents a cyber-physical integration. The orchestrator service is connected to the industrial controllers in order to be the component that triggers service operations in certain order. On the other hand, the up-

date manager, is an interface between the controllers and the ontology that is in charge of sending the updates on any resource of the system that changes its status. This update is done through SPARQL Update queries as they are compliant with RDF-based ontologies.

On the other hand, ontology service is deployed in the Cloud that acts as an interface to the ontology, which will be the central component of the solution since it will be accessed by different components. Besides the aforementioned access from the shop floor components, the user interface from the Third-Party Server may be used for sending SPARQL queries in order to retrieve information.

The presented knowledge-based service integration enables run-time reconfiguration of industrial systems because i) the ontology is updated during system operation and ii) the orchestration engine decides which operation to execute accordingly. Therefore, the system will adapt its behavior depending on KB changes. Furthermore, as a principal feature of ontologies as well as a recommendation in ontology design methodologies, such as [37], ontology models should be reused. In this context, the provided ontology may be reused for describing similar manufacturing systems that intend to implement a knowledge-based web service integration approach.

3.2 Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems (Publication II)

Manufacturing enterprises demand ICT-based solutions that enhances the efficiency of their systems in order to improve their productivity and scale up on the competitive market. This is achievable within the deployment of modern MES that facilitate the integration of heterogeneous systems located at different layers of the automation pyramid. There are several standards that describe the main functionalities that should be provided by any MES. For example, the Manufacturing Execution Systems Association (MESA)¹³ organization presents a set of functions in [124]. On the other hand, this article presents the mapping of similar set of functions defined by other organizations as the Verein Deutsche Ingenieure (VDI)¹⁴.

The work performed in this publication addresses the OKD-MES framework, which is the main contribution of the eScop project. Explicitly defined by its own term, this solu-

¹³ <http://www.mesa.org/en/index.asp>

¹⁴ <https://www.vdi.de>

tion is an open and knowledge-driven based MES that allow users to develop their functions in order to control and monitor industrial systems operation. As the principal duties of MES, the OKD-MES framework permits the integration of the Execution Resource Planning with shop floor levels. The followed methodology for implementing the modular OKD-MES is based on the SOA paradigm. More precisely, this is performed within the implementation of REST-WS [125]. In addition, the OKD-MES employs web-based standards in order to represent system information as well as for creating visualizations and interfaces for the end users.

The OKD-MES is presented as a stack of four layers: Physical layer (PHL), Representation layer (RPL), Orchestration layer (ORL) and Visualization layer (VIS). Firstly, PHL is composed by RTUs that expose i) the descriptions of controlled machines and ii) available services and data. Secondly, the RPL is concerned about the maintenance of the MES KR. Thus, the RPL includes an OWL ontology that describes the manufacturing system information. In addition, the ontology provides means for reasoning and querying any kind of system information. Thirdly, the ORL is in charge of executing sequences of operations which are described in the RPL. Finally, the VIS exposes the web user interfaces that are used for interacting with the multiple system components.

All important information on each layer is addressed in the article, which validates and discusses the approach with representative examples in a specific assembly line i.e., the FASTory line.

3.3 Towards the encapsulation and decentralization of OKD-MES services within embedded devices (Publication III)

At present, the industry seeks for novel, efficient and cheaper solutions that facilitate the access, management, control and monitoring of information of their systems. In addition, such solutions should provide remote accessibility in order to i) integrate isolated systems that must interact in the same value chain and ii) allow users to request system functionality from different locations and without critical delays.

The eScop framework presents a set of services that permits the control and monitoring of modern production systems ensuring the re-configurability of resources on operation time [14]. The eScop solution is based on the synergy between SOA, KR and the use of contemporary RTUs. The most particular characteristic of the OKD-MES framework in the context of knowledge management is that the RPL becomes a central service. The reason is that any component of the system that requires to access the KB

must interact with the RPL service (RPL-S), which might be presented as the unique access point of the stored data.

This research proposes the decentralization of the KB, which can be divided into portions that, in turn, may be hosted and managed by contemporary resource constrained embedded devices. This research work is motivated by the fact that such type of recent devices is enhanced with higher computational resources. This permits the management of more functionalities as well as the increment of storage capabilities. Then, the authors believe that the decentralization of some parts of the OKD-MES framework may lead i) to find a faster solution in terms of computation of information and ii) to employ available resources of embedded devices that are not yet exploited.

Then, this article presents a roadmap that may be followed to lower part of the OKD-MES at the device level. This is demonstrated with a set of diagrams that depict how the RPL-S may be handled by embedded devices. Conceptually, the RPL-S can be replicated and deployed on each device that is already located at the shop floor. Then, each device may include the portion of knowledge related to the controlled industrial equipment and, when needed, share it with peers. In this manner, updates to the knowledge model are directly triggered on the model, which is closer to where the data is generated.

In addition, the strengths and weaknesses of the proposed approach are discussed. In fact, this discussion is provided throughout a comparison of qualitative attributes of decentralized and centralized OKD-MES. Although the main focus of the proposal is to lower and implement the RPL-S at device level, it also states that, further, more OKD-MES might be descended. In the future, this will depend on the performance and capabilities of embedded devices.

3.4 Exemplifying the Potentials of Web Standards for Automation Control in Manufacturing Systems (Publication IV)

International consortiums as the W3C and the OASIS develop and release recommendations on standards for web based applications and systems. Such standards are prepared for dealing with common situations and fulfilling specific requirements of smart factories. For example, they allow the location of resources in networks, designing, defining and as well as dynamically routing system messages and graphical representation and navigation of models.

This article presents a selection of specific standard languages that are employed at different locations of the OKD-MES framework. In addition, important and representative examples are shown within simple and described scenarios. To demonstrate this, the article describes each of the OKD-MES framework layers. At PHL (i.e., device level), web-based standards are presented as an option for enabling vertical communication. The article highlights the utilization of the OPC UA, DPWS and REST. These standards permit the implementation of the SOA-based communication within industrial controllers which are deployed at the shop floor layer. As an example, the paper presents a sample of REST description for conveyor services. The following layer i.e., the RPL requires the implementation of a KB that must i) allow the system description and ii) be accessible through the web. that must be accessible by other service layers. Then, the article describes the implementation of models aligned with the Semantic Web stack and, more precisely, the utilization of RDF-based languages as the OWL. To show an example, a specific OWL model is presented i.e., Manufacturing System Ontology (MSO) [126]. Besides describing its main classes, a set of SPARQL and SPARQL Update queries that may be used for retrieving and updating model information are also depicted and explained. Afterwards, the article shows the potential of web-standards for the orchestration of services at the ORL. More precisely, the paper mentions WS-BPEL, BPMN and OWL-S. As an example, the orchestration of REST based services is discussed. Finally, in the scope of VIS, the article describes HTML, JS, CSS and SVG as a set of web-based standards that may be used for implementing web interfaces for user frontend visualization. This is exemplified throughout the description of the FASTory simulator interface, which is described in [1].

As a conclusion of the research work, the authors claim that there is a variety of web-based standards that may be used to create complete frameworks for controlling and monitoring systems. Such standards cover from the description of system capabilities and operations until the exposure of services and interfaces to users. In addition, the structure as well as the communication protocol of messages can also be handled with web-based standards.

3.5 Product, process and resource model coupling for knowledge-driven assembly automation (Publication V)

The amount of information about manufacturing systems that needs to be processed in order to control and monitor production processes has grown dramatically during the last decades. This is because the interconnection of physical resources with cyber systems in modern factories permits the supervision of different kinds of data that can lead

to a better understanding of the system performance. In this context, the industry faces the challenge of managing huge volumes of data that is heterogeneous due to the different formats in which it is presented. Moreover, the information is commonly allocated in multiple models that are isolated. Therefore, there is a need of novel solutions that permit the integration of data models which describes diverse domain information.

This research work presents a proof of concept for coupling domain ontologies designed for describing manufacturing systems information. More precisely, the article discusses the need of integrating product, process and resource (PPR) domains. To achieve such coupling, the approach suggests the implementation of SWRL rules that are included into a higher ontology which imports the PPR models. The implemented model not only permits the retrieval of different domain information, but also the inference of implicit knowledge throughout the use of reasoning engines. The latter model characteristic is a clear benefit because it permits to automatically discover information about the manufacturing system which is not known at design phase. Finally, the employment of reasoning engines permits the validation of the model. In other words, such engines allow to find semantic incoherencies in the ontology. Then, any conflict created i) in imported ontologies at model design phase or ii) in SWRL rule implementation process can be found, reported and, afterwards, fixed within semantic engines.

3.6 Private local automation clouds built by CPS: Potential and challenges for distributed reasoning (Publication VI)

This research work presents the main principles, requirements and challenges for the design and operation of PLACs built by CPS. Thus, this article is one of the most important documents of the doctoral research as it sets the bases of the upcoming investigation and experiments.

The combination of ontology-based knowledge representation and the SOA paradigm implemented within industrial controllers which, in turn, are deployed at factory shop layer can be a suitable approach for modelling and controlling industrial systems. This is actually realized in previous approaches e.g., the OKD-MES. However, the semantic models are located far away where the data is generated. This implies the need of cross-domain communication which adds delays and complexity to different processes of pushing information to the model. This might be changed within the implementation of Private Local Automation Clouds (PLACs).

This research proposes the employment of available resources of new embedded devices that are deployed at the factory device level for performing some tasks that are done at upper levels. For example, the encapsulation and management of ontological descriptions may be carried out by such devices. This permits a faster update of data model and the avoidance of cross-domain or vertical communications along the automation pyramid.

Then, the PLAC consists on a set of CPS devices that are networked and capable of not only sharing and exchanging information but also to execute operations. In this manner, upcoming user requests may be handled and responded by the PLAC. To achieve this, the article defines and describes the execution of the distributed reasoning process. Basically, the PLAC concept suggests that the systems' KB is divided into portions (i.e., smaller ontologies) that are encapsulated in embedded devices. As the devices exchange information within SPARQL over HTTP, any RDF-based model can be queried. Then, the distributed reasoning defines a set of steps that i) are managed by one of the embedded devices ii) permits the integration of semantic descriptions. Thus, any request can be responded because the PLAC is capable of i) executing incoming queries and ii) integrating all related information of such queries.

The article describes potential benefits as the enrichment of network descriptions, reduction of the high integration cost, the decentralization of knowledge-driven solutions using central KBs and the possibility of integrating the PLAC with other systems and even other cloud-based platforms throughout web-based technologies. On the other hand, some of the reported challenges are on: device behavior implementation, message development, horizontal scalability, fault tolerance, service composition, and cloud security. In addition, the research work benchmarks the implementation of the approach in the FASTory line in terms of production type, communication protocols, IT infrastructure, control architecture, number of computational nodes, data formats, or horizontal and vertical communication, among others.

3.7 Management of distributed knowledge encapsulated in embedded devices (Publication VII)

The implementation of PLACs requires the development of device behavior that permits the execution of the distributed reasoning process, which is needed for integrating decentralized semantic descriptions in order to respond to user queries. Therefore, this research work presents an approach for managing the distributed knowledge that is encapsulated in embedded devices.

To proof the main concept of this article requires the duplication of ontology model structure into different devices. The convention of similar ontologies to be deployed on each device is needed to avoid integration problems. Then, each device may populate the model with instances that represents the resources that are linked to the corresponding device. Once this is done, the set of behaviors presented in the research work may be executed.

The first behavior is the one that permits the devices to expose themselves to the rest of devices that inhabit in the PLAC. This is called device initialization and it mainly consists in a broadcast of the device address and ID for other peers to inscribe it in their KB. Then, the new device will receive a message from each device that is in the PLAC. In this moment, newer and older devices may interact further for solving incoming requests.

The second behavior consists in the election of the *requester device*, which is the given name to the device that will lead the distributed reasoning process per each incoming query. According to this article, the decision is to be made based on the first device that responds to a request of requester device. However, a more recent work explains that more complex processes might be implemented in order to select requester devices according to configurable parameters [127]. Once a requester device is elected, the distributed reasoning process for a specific query may be executed. This means that per each query a different election will be performed that might end up with a different requester device choice.

The third behavior is the distributed reasoning process itself. This is a more complex behavior compared with the previous ones because implies a larger number of steps and more computation of information. The requester device will receive the incoming query that will be then re-routed to all PLAC devices. Then, each device will respond to the query according to the encapsulated knowledge. In fact, such response is not a plain execution of the query but an aggregation of query execution and linked knowledge to the query result. This is because the addition of linked knowledge might end up with an answer that is not achievable within just a portion of knowledge. Once the information is sent to the requester device, it will execute the query on the combination of their own knowledge and all the information received. The result will be assumed as the final response to be sent to the user.

The article presents a proof of the concept in two scenarios. As the focus of the paper is the demonstration of management of distributed knowledge, the examples are general i.e., not focused on industrial domain processes. In any case, the results achieved at different stages of the distributed reasoning process are explained and depicted.

3.8 An Architecture for Implementing Private Local Automation Clouds Built by CPS (Publication VIII)

During the last years, several research works related to the development of PLCs built by CPS have been published and there is a need of presenting a reference architecture which may be followed by engineers willing to implement a similar solution. In this scope, this article presents an architecture for implementing PLCs.

The resulting architecture is designed within the known “4+1” view model, which permits a formal representation within different architectural views for identifying and describing the system components. The first architectural view of the presented architecture is the logical view. According to the 4+1 view model, this view must address the mechanisms and design elements of different system parts that will be implemented. Then, the article presents an UML class diagram that depicts different objects of PLCs. The main parts included in such diagram are the ones related to users and devices as e.g., the web user interface in case or user types of in the case of users and the KB or distributed reasoning process in case of devices. Afterwards, the process view describes the interaction of the main objects that are depicted in the logical view. To show this, the article presents a set of activity diagrams that describes the information flow and requirements when it is exchanged between different PLC components. This interactions are aligned with the behavior of devices presented in previous publications [128], [129]. Then, the article presents the development architectural view, which is shown as an UML component diagram. This diagram is composed by a set of components that must be developed and interconnected in order to achieve the main functionalities of the PLC-based approach. The last architectural view is the physical view, which presents a diagram of physical components that will be the ones to be interconnected from an engineer point of view. More precisely, this diagram includes, from bottom to up i) the industrial equipment, ii) the devices, iii) the web based server and iv) the user system. Finally, following the “4+1” a general scenario is presented in order to describe the role and involvement of each component in different activities i.e., Place Request, Check Response, Reasoning Process and Manage Devices.

Furthermore, the article presents a comparison of different CPS-based architectures and research works. This is done in order to present a discussion on critical qualitative attributes that should be addressed by any CPS-based solution.

3.9 Principles and risk assessment of managing distributed ontologies hosted by embedded devices for controlling industrial systems (Publication IX)

Industrial organizations employ different solutions that permits to handle different needs, such as interoperability of remote systems, system modelling and validation or data reasoning. These needs are addressed within IT-based solutions. As the employment of such kind of solutions is a trend in the industrial automation domain, the research community is continuously presenting novel research works that may increase the efficiency of solutions being used by companies. However, one of the bigger issues when such works are presented is the lack of security consideration.

Web-based technologies brings remote connectivity and accessibility but increases the exposure of system data over the Web. Obviously, this increases the vulnerability of systems which might be maliciously accessed. Thus, this article suggests the application of risk assessment and threat modelling techniques in order to identify weak points of solutions at design phase. In this manner, developers may bring security means to their implementations, ensuring the creation of less vulnerable solutions. Particularly, this research work focuses on solutions to be deployed in the industrial automation field.

This article presents the principles of risk assessment and threat modeling to enhance the security due to web exposure of the PLCs. This study is made at design phase in order to affect to the system design decisions. The benefit of this is that is more complex to modify decided interactions and/or technologies at advanced stages of a solution development than to do it at design phase i.e., when nothing is yet implemented.

In the particular case of PLCs, the implementation of aforementioned techniques identified the different points of system vulnerability. One of the advantages of PLCs is that there is no single point of failure because, due to the decentralization of knowledge, the system might operate with some devices down. Nevertheless, the risk assessment, suggests the backup of models in order to have a way of recovering damaged models. On the other hand, the study highlights the devices might be affected by different attacks, such as spoofing, tampering and Denial of Service (DoS). Meanwhile the authentication, authorization and identification of users is recommended to mitigate spoofing and DoS, the tampering may be avoided within the communication of devices through HTTPs.

3.10 Summary

This section outlines the main contribution of the research work throughout the TABLE. I, which summarizes the main results from the presented publications that supported the design and implementation tests of this doctoral research. In addition, TABLE. I links each publication with the objectives and contributions presented in sections 1.4 Objectives and 1.5 Contributions, respectively.

TABLE. I: Main results and outcomes

| # | Publication title | Main results | Doctoral research Objectives and contributions |
|-----|--|---|--|
| I | Knowledge-based web service integration for industrial automation | <ul style="list-style-type: none"> • Main components and demonstration of the knowledge-based web service integration solution for industrial systems • Structure of an ontology model that can be reused for describing production systems | Objective: 1.4 I Contribution: 1.5 I |
| II | Cyber-Physical Systems for Open Knowledge-Driven Manufacturing Execution Systems | <ul style="list-style-type: none"> • Presentation of the OKD-MES concept as the ultimate result of the eScop project • Discussion on application and application of the OKD-MES framework | Objective: 1.4 I Contribution: 1.5 I Non-core contribution: 1.5 IV |
| III | Towards the encapsulation and decentralization of OKD-MES services within embedded devices | <ul style="list-style-type: none"> • Proposal for decentralization of OKD-MES services and functionalities to be handled at device level • Strengths and weaknesses of the proposal which could be appreciated at early stage | Objective: 1.4 I Contribution: 1.5 II |

| | | | |
|------|--|---|---|
| IV | Exemplifying the Potentials of Web Standards for Automation Control in Manufacturing Systems | <ul style="list-style-type: none"> • Exemplification and discussion of using web standards at all the levels of the automation system, inspired in the needs of the OKD-MES development | <p>Objective: 1.4 I, 1.4 II</p> <p>Contribution: 1.5 I</p> <p>Non-core contribution: 1.5 IV</p> |
| V | Product, process and resource model coupling for knowledge-driven assembly automation | <ul style="list-style-type: none"> • PPR model coupling demonstration within RDF-based ontologies and SWRL rules. | <p>Objective: 1.4 I</p> <p>Contribution: 1.5 I</p> <p>Non-core contribution: 1.5 V</p> |
| VI | Private local automation clouds built by CPS: Potential and challenges for distributed reasoning | <ul style="list-style-type: none"> • Main principles and components description of PLCs build be CPS. • Distributed reasoning definition and exemplification • Benchmarking of the approach in a specific assembly line • Challenges and potential of the proposal | <p>Objective: 1.4 I, 1.4 II, 1.4 IV</p> <p>Contribution: 1.5 I, 1.5 II</p> |
| VII | Management of distributed knowledge encapsulated in embedded devices | <ul style="list-style-type: none"> • Description, implementation and demonstration of the distributed reasoning process • Structure of decentralized ontology models to be encapsulated in embedded devices • SPARQL query transformation in the distributed reasoning process | <p>Objective: 1.4 I, 1.4 II, 1.4 III, 1.4 IV</p> <p>Contribution: 1.5 II</p> |
| VIII | An Architecture for Implementing Private Local Automa- | <ul style="list-style-type: none"> • A reference architecture for developing PLCs, built by CPS | <p>Objective: 1.4 IV</p> <p>Contribution: 1.5 I, 1.5 II,</p> |

| | | | |
|----|---|--|--|
| | tion Clouds Built by CPS | <ul style="list-style-type: none"> • The architecture is presented throughout the “4+1” formal view model • Comparison of different research works, solutions and other architectures within a set of important qualitative attributes | 1.5 III |
| IX | Principles and risk assessment of managing distributed ontologies hosted by embedded devices for controlling industrial systems | <ul style="list-style-type: none"> • Recommendation of applying threat modelling and risk assessment techniques at design phase of IT-based solutions • Qualification of the security in PLCs | Objective: 1.4 II, 1.4 IV Contribution: 1.5 III |

4 Conclusions and Recommendation for Future Works

4.1 Concluding Remarks

The research work presented in this dissertation answers the original research questions outlined from the main research problem outlined in Chapter 1. The answer is grounded by:

Research Question #1:

The distribution of knowledge among the different control units that build automation systems has been achieved within the design and deployment of ontology models on each control unit. These models describe the knowledge of the system being controlled for the corresponding unit in RDF-based language. Then, the knowledge can be distributed within the exchange of messages between networked control units. Moreover, due to the technological features, the distribution, update and retrieval of knowledge descriptions can be done at run time. Practical examples of ontology models that can be deployed in control units are described and demonstrated in Publications I, II, III, IV, V, VI, VII and VIII.

Research Question #2:

The access of the knowledge by the controls units located in other control units has been achieved throughout the execution of the device behaviour presented and explained on several of the refereed publications. Meanwhile Publication VI and VIII describe such behaviour within a set of UML diagrams, Publication VII shows concrete scenarios that demonstrate the access and integration of knowledge in order to respond to user queries.

Therefore, this research work solves the original research problem by specifically answering the two research questions.

It has been proved that this research work advances the current state-of-the-art practices in the scope of knowledge representation and reasoning machines within the application of factory automation in general, and to the distribution and access of knowledge in particular.

4.2 Further Work

This section presents a set of suggestions for continuing with the development of this research theme. It sums up a few of the individual recommendations outlined in the refereed publications, and group them in a consistent manner.

As claimed in Publication III, solutions based on the employment of centralised models for the description, access and management of knowledge can be lightened on regarding the computation processing within the decentralization of such models and its distribution on control units currently deployed at factory shop floors.

Then, as pointed out in Publication VI, the distribution of knowledge descriptions among networked control units brings to the current structure of manufacturing systems a set of potential benefits, such as the reduction of high integration time and costs because of the diminution of cross-layer communication.

On the other hand, due to the irruption of cloud computing in the manufacturing domain, the execution of processes can be performed remotely by cloud resources. In this context, Publication VIII presents an architecture that can be followed for implementing the distribution and access of knowledge through networked control units which build the so-called private local automation cloud.

Furthermore, there are many technologies that can be used for building solutions based on the concepts described along this dissertation. Although some of them are addressed in Publication IV, the advances and appearance of new technologies must be considered in further implementations.

As a concluding remark, the author would like to point the attention of the research community towards the utilization of ontologies as an engineering artifact that can be used to describe the knowledge of system status at run time. The resulting knowledge model, in turn, can be hosted and managed with contemporary control units at the de-

vice level which provide sufficient computational power for model processing functionalities. Therefore, current functionalities being controlled at higher levels of manufacturing enterprises (e.g., MES and ERP), can be now performed by control units and, thus, saving time and costs on configurability and vertical communication.

References

- [1] W. M. Mohammed, A. Lobov, B. R. Ferrer, S. Iarovy, and J. L. M. Lastra, "A web-based simulator for a discrete manufacturing system," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 6583–6589.
- [2] B. R. Ferrer, S. Iarovy, A. Lobov, and J. L. M. Lastra, "Potentials of Web Standards for Automation Control in Manufacturing Systems," in *2015 IEEE European Modelling Symposium (EMS)*, 2015, pp. 359–366.
- [3] M. Ahmad *et al.*, "A knowledge-based approach for the selection of assembly equipment based on fuel cell component characteristics," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 001002–001007.
- [4] B. R. Ferrer, B. Ahmad, A. Lobov, D. Vera, J. L. Martinez Lastra, and R. Harrison, "A knowledge-based solution for automatic mapping in component based automation systems," in *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, 2015, pp. 262–268.
- [5] B. R. Ferrer, B. Ahmad, A. Lobov, D. A. Vera, J. L. Martinez Lastra, and R. Harrison, "An approach for knowledge-driven product, process and resource mappings for assembly automation," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 1104–1109.
- [6] K. Henning, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0," 2013.
- [7] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.
- [8] B. Sniderman, M. Mahto, and M. Cotteleer, "Industry 4.0 and manufacturing ecosystems," *DU Press*, Feb-2016. [Online]. Available: <https://dupress.deloitte.com/dup-us-en/focus/industry-4-0/manufacturing-ecosystems-exploring-world-connected-enterprises.html>. [Accessed: 07-Jun-2017].
- [9] S. Biffl and M. Sabou, "Introduction," in *Semantic Web Technologies for Intelligent Engineering Applications*, Springer, Cham, 2016, pp. 1–13.
- [10] International Electrotechnical Commission, "IEC - White Paper > Factory of the future," 2015. [Online]. Available: <http://www.iec.ch/whitepaper/futurefactory/>. [Accessed: 10-Oct-2017].

- [11] J. Li, "Analyzing 'Made in China 2025' Under the Background of 'Industry 4.0,'" in *Proceedings of the 23rd International Conference on Industrial Engineering and Engineering Management 2016*, Atlantis Press, Paris, 2017, pp. 169–171.
- [12] C. M. Shi, "„Made in China 2025“: Chinas Vision zu Industrie 4.0," *Wirtsch. Manag.*, vol. 9, no. 2, pp. 70–77, Apr. 2017.
- [13] H. Huang and G. Wang, "Comparison study on the industrial policies of additive manufacturing in U.S. and China," in *2016 International Conference on Industrial Economics System and Industrial Security Engineering (IEIS)*, 2016, pp. 1–6.
- [14] S. Iaroyvi, W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra, "Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems," *Proc. IEEE*, vol. PP, no. 99, pp. 1–13, 2016.
- [15] I. M. Delamer and J. L. M. Lastra, "Service-Oriented Architecture for Distributed Publish/Subscribe Middleware in Electronics Production," *IEEE Trans. Ind. Inform.*, vol. 2, no. 4, pp. 281–294, Nov. 2006.
- [16] ISA, "ANSI/ISA-95.00.03-2013 Enterprise-Control System Integration - Part 3: Activity Models of Manufacturing Operations Management," 2013. [Online]. Available: <https://www.isa.org/store/products/product-detail/?productId=116782>. [Accessed: 02-Aug-2017].
- [17] R. J. Brachman and H. J. Levesque, *Knowledge Representation and Reasoning*. Morgan Kaufmann, 2004.
- [18] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.
- [19] B. Ramis *et al.*, "Knowledge-based web service integration for industrial automation," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 733–739.
- [20] B. Ramis Ferrer, S. Iaroyvi, W. M. Mohammed, A. Lobov, and J. L. M. Lastra, "Exemplifying the Potentials of Web Standards for Automation Control in Manufacturing Systems," *Int. J. Simul. Syst. Sci. Technol.*, vol. 17, no. 33, p. 3.1-3.12, 2016.
- [21] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, and S. Yin, "Industrial Cyberphysical Systems: A Backbone of the Fourth Industrial Revolution," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 6–16, Mar. 2017.
- [22] J. Puttonen, A. Lobov, and J. L. Martinez Lastra, "Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2349–2359, Nov. 2013.
- [23] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, p. 50, Apr. 2010.

- [24] K. M. Alam and A. El Saddik, "C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems," *IEEE Access*, vol. 5, pp. 2050–2062, 2017.
- [25] B. Andres, R. Sanchis, and R. Poler, "A Cloud Platform to support Collaboration in Supply Networks," *Int. J. Prod. Manag. Eng.*, vol. 4, no. 1, pp. 5–13, Jan. 2016.
- [26] J. Puttonen, A. Lobov, M. A. C. Soto, and J. L. M. Lastra, "Cloud computing as a facilitator for web service composition in factory automation," *J. Intell. Manuf.*, pp. 1–14, Nov. 2016.
- [27] B. Ramis Ferrer, A. Nieto, and S. Iarovy, "C2Net | D6.6 - White Paper of C2NET platform / openness and portability - Deliverables," 2016. [Online]. Available: <http://c2net-project.eu/deliverables/-/blogs/d6-6-white-paper-of-c2net-platform-openness-and-portability>. [Accessed: 21-Apr-2017].
- [28] W. Mohammed M., D. Aleixo, B. Ramis Ferrer, C. Agostinho, and J. L. Martinez Lastra, "Configuring and Visualizing The Data Resources in a Cloud-based Data Collection Framework," in *2017 IEEE International Conference on Engineering, Technology and Innovation/ International Technology Management Conference (ICE/ITMC)*, 2017, pp. 1242–1249.
- [29] J. L. M. Lastra and I. M. Delamer, "Semantic web services in factory automation: fundamental insights and research roadmap," *IEEE Trans. Ind. Inform.*, vol. 2, no. 1, pp. 1–11, Feb. 2006.
- [30] S. Iarovy, B. Ramis, X. Xiangbin, A. Sampath, A. Lobov, and J. L. Martinez Lastra, "Representation of manufacturing equipment and services for OKD-MES: From service descriptions to ontology," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, 2015, pp. 1069–1074.
- [31] B. R. Ferrer, W. M. Mohammed, and J. L. M. Lastra, "A solution for processing supply chain events within ontology-based descriptions," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 4877–4883.
- [32] B. Ramis Ferrer, W. M. Mohammed, A. Lobov, A. Moreno Galera, and J. L. M. Lastra, "Including Human Tasks as Semantic Resources in Manufacturing Ontology Models," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017.
- [33] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.
- [34] A. Lobov, F. U. Lopez, V. V. Herrera, J. Puttonen, and J. L. M. Lastra, "Semantic Web Services framework for manufacturing industries," in *IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008*, 2009, pp. 2104–2108.

- [35] J. Puttonen, A. Lobov, and J. L. M. Lastra, "Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems," in *2013 IEEE 20th International Conference on Web Services (ICWS)*, 2013, pp. 419–426.
- [36] J. L. M. Lastra, I. M. Delamer, and F. Ubis, *Domain Ontologies for Reasoning Machines in Factory Automation*. ISA, 2010.
- [37] N. F. Noy, D. L. McGuinness, and others, *Ontology development 101: A guide to creating your first ontology*. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, 2001.
- [38] V. Maniraj and R. Sivakumar, "Ontology Languages—A Review," *Int. J. Comput. Theory Eng.*, vol. 2, no. 6, pp. 887–891, 2010.
- [39] E. Negri, L. Fumagalli, M. Garetti, and L. Tanca, "A review of semantic languages for the conceptual modelling of the manufacturing domain," in *Proceedings of the XIX Summer School Francesco Turco, 2014*, Senigallia, Ancona, 2014, pp. 1–8.
- [40] C. Hildebrandt *et al.*, *Semantic Modeling for Collaboration and Cooperation of Systems in the production domain*. 2017.
- [41] "OWL Web Ontology Language Reference," 2004. [Online]. Available: <http://www.w3.org/TR/owl-ref/>. [Accessed: 01-Apr-2015].
- [42] I. M. Delamer and J. L. M. Lastra, "Ontology Modeling of Assembly Processes and Systems using Semantic Web Services," in *2006 IEEE International Conference on Industrial Informatics*, 2006, pp. 611–617.
- [43] "RDF 1.1 Primer," 2014. [Online]. Available: <http://www.w3.org/TR/rdf11-primer/>. [Accessed: 01-Apr-2015].
- [44] "Extensible Markup Language (XML)," 2015. [Online]. Available: <https://www.w3.org/XML/>. [Accessed: 01-Jun-2016].
- [45] "SPARQL Query Language for RDF," 2008. [Online]. Available: <http://www.w3.org/TR/rdf-sparql-query/>. [Accessed: 01-Apr-2015].
- [46] J. Puttonen, A. Lobov, and J. L. M. Lastra, "On the Updating of Domain OWL Models at Runtime in Factory Automation Systems:," *Int. J. Web Serv. Res.*, vol. 11, no. 2, pp. 46–66, 32 2014.
- [47] S. Abburu, "A Survey on Ontology Reasoners and Comparison," *Int. J. Comput. Appl.*, vol. 57, no. 17, 2012.
- [48] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A Practical OWL-DL Reasoner," *Web Semant*, vol. 5, no. 2, pp. 51–53, Jun. 2007.

- [49] B. Ramis Ferrer, B. Ahmad, D. Vera, A. Lobov, R. Harrison, and J. L. Martínez Lastra, "Product, process and resource model coupling for knowledge-driven assembly automation," - *Autom.*, vol. 64, no. 3, Jan. 2016.
- [50] S. Feldmann, M. Wimmer, K. Kernschmidt, and B. Vogel-Heuser, "A comprehensive approach for managing inter-model inconsistencies in automated production systems engineering," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1120–1127.
- [51] E. Estevez and M. Marcos, "Model-Based Validation of Industrial Control Systems," *IEEE Trans. Ind. Inform.*, vol. 8, no. 2, pp. 302–310, May 2012.
- [52] W3C, "All Standards and Drafts - W3C," 2017. [Online]. Available: <https://www.w3.org/TR/>. [Accessed: 11-Jul-2017].
- [53] E. Friedman-Hill, *JESS in Action*, vol. 46. Manning Greenwich, CT, 2003.
- [54] C. Legat and B. Vogel-Heuser, "A configurable partial-order planning approach for field level operation strategies of PLC-based industry 4.0 automated manufacturing systems," *Eng. Appl. Artif. Intell.*, vol. 66, no. Supplement C, pp. 128–144, Nov. 2017.
- [55] D. Trentesaux *et al.*, "Benchmarking flexible job-shop scheduling and control systems," *Control Eng. Pract.*, vol. 21, no. 9, pp. 1204–1225, Sep. 2013.
- [56] J. Puttonen, *A Semantically Enhanced Approach for Orchestration of Web Services in Factory Automation Systems*. Tampere University of Technology, 2014.
- [57] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*. Pearson Education, 2005.
- [58] A. Lobov, J. Puttonen, V. V. Herrera, R. Andiappan, and J. L. M. Lastra, "Service oriented architecture in developing of loosely-coupled manufacturing systems," in *6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008*, 2008, pp. 791–796.
- [59] M. Qusay H., "SOA and Web Services," 2005. [Online]. Available: <http://www.oracle.com/technetwork/articles/javase/soa-142870.html>. [Accessed: 11-Jul-2017].
- [60] T. K. Liu, Y. P. Chen, and J. H. Chou, "Solving Distributed and Flexible Job-Shop Scheduling Problems for a Real-World Fastener Manufacturer," *IEEE Access*, vol. 2, pp. 1598–1606, 2014.
- [61] M. K. Tiwari, S. Kumar, S. Kumar, Prakash, and R. Shankar, "Solving Part-Type Selection and Operation Allocation Problems in an FMS: An Approach Using Constraints-Based Fast Simulated Annealing Algorithm," *IEEE Trans. Syst. Man Cybern. - Part Syst. Hum.*, vol. 36, no. 6, pp. 1170–1184, Nov. 2006.
- [62] N. Wu, C. Chu, F. Chu, and M. C. Zhou, "A Petri Net Method for Schedulability and Scheduling Problems in Single-Arm Cluster Tools With Wafer Residency

- Time Constraints," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 2, pp. 224–237, May 2008.
- [63] M. G. Hinchey, R. Sterritt, and C. Rouff, "Swarms and Swarm Intelligence," *Computer*, vol. 40, no. 4, pp. 111–113, Apr. 2007.
- [64] J. E. Inglett and E. J. Rodríguez-Seda, "Object transportation by cooperative robots," in *SoutheastCon 2017*, 2017, pp. 1–6.
- [65] H. Rezaee and F. Abdollahi, "A Decentralized Cooperative Control Scheme With Obstacle Avoidance for a Team of Mobile Robots," *IEEE Trans. Ind. Electron.*, vol. 61, no. 1, pp. 347–354, Jan. 2014.
- [66] Y. Su and J. Huang, "Cooperative Output Regulation of Linear Multi-Agent Systems," *IEEE Trans. Autom. Control*, vol. 57, no. 4, pp. 1062–1066, Apr. 2012.
- [67] W. Zhuang and M. Ismail, "Cooperation in wireless communication networks," *IEEE Wirel. Commun.*, vol. 19, no. 2, pp. 10–20, Apr. 2012.
- [68] A. Sehgal, V. Perelman, S. Kuryla, and J. Schonwalder, "Management of resource constrained devices in the internet of things," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 144–149, Dec. 2012.
- [69] M. Vögler, J. Schleicher, C. Inzinger, S. Nastic, S. Sehic, and S. Dustdar, "LE-ONORE – Large-Scale Provisioning of Resource-Constrained IoT Deployments," in *2015 IEEE Symposium on Service-Oriented System Engineering*, 2015, pp. 78–87.
- [70] Z. Xiao, W. Song, and Q. Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.
- [71] F. Xhafa and N. Bessis, Eds., *Inter-cooperative Collective Intelligence: Techniques and Applications*, vol. 495. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.
- [72] D. K. Barry, *Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's Guide*. Newnes, 2012.
- [73] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky Computing," *IEEE Internet Comput.*, vol. 13, no. 5, pp. 43–51, Sep. 2009.
- [74] S. Goyal, "Public vs Private vs Hybrid vs Community - Cloud Computing: A Critical Review," *Int. J. Comput. Netw. Inf. Secur.*, vol. 6, no. 3, pp. 20–29, Feb. 2014.
- [75] J. L. Martinez Lastra, *Reference mechatronic architecture for actor-based assembly systems*. Tampere University of Technology, 2004.
- [76] A. Lobov, F. U. Lopez, V. V. Herrera, J. Puttonen, and J. L. M. Lastra, "Semantic Web Services framework for manufacturing industries," in *IEEE International*

- Conference on Robotics and Biomimetics, 2008. ROBIO 2008, 2009*, pp. 2104–2108.
- [77] “IBM developerWorks : New to SOA and web services,” 05-Mar-2007. [Online]. Available: <http://www.ibm.com/developerworks/webservices/newto/service.html>. [Accessed: 01-Apr-2015].
- [78] L. E. G. Moctezuma, J. Jokinen, C. Postelnicu, and J. L. M. Lastra, “Retrofitting a factory automation system to address market needs and societal changes,” in *2012 10th IEEE International Conference on Industrial Informatics (INDIN)*, 2012, pp. 413–418.
- [79] I. M. Delamer and J. L. M. Lastra, “Self-orchestration and choreography: towards architecture-agnostic manufacturing systems,” in *20th International Conference on Advanced Information Networking and Applications, 2006. AINA 2006, 2006*, vol. 2, p. 5 pp.-.
- [80] A. N. Lee and J. L. M. Lastra, “Data aggregation at field device level for industrial ambient monitoring using Web Services,” in *2011 9th IEEE International Conference on Industrial Informatics (INDIN)*, 2011, pp. 491–496.
- [81] M. J. A. G. Izaguirre, A. Lobov, and J. L. M. Lastra, “OPC-UA and DPWS interoperability for factory floor monitoring using complex event processing,” in *2011 9th IEEE International Conference on Industrial Informatics (INDIN)*, 2011, pp. 205–211.
- [82] D. Schilberg, M. Hoffmann, S. Schmitz, and T. Meisen, “Interoperability in Smart Automation of Cyber Physical Systems,” in *Industrial Internet of Things*, Springer, Cham, 2017, pp. 261–286.
- [83] S. Iarovyi, J. Garcia, and J. L. M. Lastra, “An approach for OSGi and DPWS interoperability: Bridging enterprise application with shop-floor,” in *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, 2013, pp. 200–205.
- [84] F. Pérez, E. Irisarri, D. Orive, M. Marcos, and E. Estevez, “A CPPS Architecture approach for Industry 4.0,” in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, 2015, pp. 1–4.
- [85] M. Hoffmann, T. Meisen, and S. Jeschke, “OPC UA Based ERP Agents: Enabling Scalable Communication Solutions in Heterogeneous Automation Environments,” in *Advances in Practical Applications of Cyber-Physical Multi-Agent Systems: The PAAMS Collection*, 2017, pp. 120–131.
- [86] J. Minor, J. Garcia, J. Martinez, A. Lobov, and J. L. M. Lastra, “Evaluating Service-Oriented Orchestration Schemes for Controlling Pallet Flow,” presented at the ICONS 2012, The Seventh International Conference on Systems, 2012, pp. 88–92.
- [87] OASIS, “Web Services Business Process Execution Language,” 2007. [Online]. Available: <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>. [Accessed: 11-Jul-2017].

- [88] J. Puttonen, A. Lobov, and J. L. M. Lastra, "An application of BPEL for service orchestration in an industrial environment," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, 2008, pp. 530–537.
- [89] W3C, "Web Services Choreography Description Language Version 1.0," 2005. [Online]. Available: <https://www.w3.org/TR/ws-cdl-10/>. [Accessed: 11-Jul-2017].
- [90] B. Ramis, J. Garcia, and J. L. M. Lastra, "Assessment of IEC-61499 and CDL for Function Block composition in factory-wide system integration," in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 212–217.
- [91] W3C, "OWL-S: Semantic Markup for Web Services," 2004. [Online]. Available: <https://www.w3.org/Submission/OWL-S/>. [Accessed: 11-Jul-2017].
- [92] C. Sunder *et al.*, "Usability and Interoperability of IEC 61499 based distributed automation systems," in *Industrial Informatics, 2006 IEEE International Conference on*, 2006, pp. 31–37.
- [93] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," *IEEE Trans. Ind. Inform.*, vol. 7, no. 4, pp. 768–781, Nov. 2011.
- [94] IEC, "IEC 61131-3:2013 | IEC Webstore | water automation, water management, smart city," 2013. [Online]. Available: <https://webstore.iec.ch/publication/4552>. [Accessed: 11-Jul-2017].
- [95] R. W. Lewis, *Modelling Distributed Control Systems Using IEC 61499*. Stevenage, UK, UK: Institution of Electrical Engineers, 2001.
- [96] R. W. Brennan, M. Fletcher, and D. H. Norrie, "An agent-based approach to re-configuration of real-time distributed control systems," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 444–451, Aug. 2002.
- [97] O. J. L. Orozco and J. L. Lastra, "Adding Function Blocks of IEC 61499 Semantic Description to Automation Objects," in *2006 IEEE Conference on Emerging Technologies and Factory Automation*, 2006, pp. 537–544.
- [98] A. Dennert, A. Gössling, J. Krause, M. Wollschlaeger, and A. M. H. Montoya, "Vertical data integration in automation based on IEC 61499," in *2012 9th IEEE International Workshop on Factory Communication Systems*, 2012, pp. 99–102.
- [99] V. Vasyutynskyy, C. Hengstler, D. Nadoveza, J. McCarthy, K. G. Brennan, and A. Dennert, "Layered architecture for production and logistics cockpits," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, 2012, pp. 1–9.
- [100] S. Kožár and P. Kadera, "Integration of IEC 61499 with OPC UA," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–7.

- [101] V. Vyatkin, *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*. ISA, 2007.
- [102] W. Michael, *An Introduction to MultiAgent Systems*, Second edition. John Wiley & Sons, 2009.
- [103] F. Jaques, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman, 1999.
- [104] S. S. Walker, R. W. Brennan, and D. H. Norrie, "Holon job shop scheduling using a multiagent system," *IEEE Intell. Syst.*, vol. 20, no. 1, pp. 50–57, Jan. 2005.
- [105] V. V. Herrera, A. Bepperling, A. Lobov, H. Smit, A. W. Colombo, and J. Lastra, "Integration of Multi-Agent Systems and Service-Oriented Architecture for industrial automation," in *6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008*, 2008, pp. 768–773.
- [106] P. Leitao, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart Agents in Industrial Cyber-Physical Systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1086–1101, May 2016.
- [107] B. Vogel-Heuser, C. Diedrich, D. Pantförder, and P. Göhner, "Coupling heterogeneous production systems by a multi-agent based cyber-physical production system," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 713–719.
- [108] R. Priego, N. Iriondo, U. Gangoiti, and M. Marcos, "Agent-based middleware architecture for reconfigurable manufacturing systems," *Int. J. Adv. Manuf. Technol.*, vol. 92, no. 5–8, pp. 1579–1590, Sep. 2017.
- [109] W. Mohammed M., B. Ramis Ferrer, R. Sanchis, B. Andres, C. Agostinho, and J. L. Martinez Lastra, "A Multi-agent Approach for Processing Industrial Enterprise Data," in *2017 IEEE International Conference on Engineering, Technology and Innovation/ International Technology Management Conference (ICE/ITMC)*, 2017, pp. 1250–1256.
- [110] R. J. Allan, "Survey of Agent Based Modelling and Simulation Tools," Report DL-TR-2010-007, 2009.
- [111] C. N. and G. Madey, "Tools of the Trade: A Survey of Various Agent Based Modeling Platforms," 31-Mar-2009. [Online]. Available: <http://jasss.soc.surrey.ac.uk/12/2/2.html>. [Accessed: 21-Apr-2017].
- [112] M. J. Pont, *Patterns for Time-triggered Embedded Systems: Building Reliable Applications with the 8051 Family of Microcontrollers*. Addison-Wesley, 2001.
- [113] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proc. IEEE*, vol. 91, no. 1, pp. 112–126, Jan. 2003.

- [114] IEC, "IEC 61508: Functional Safety - IEC 61508 Explained," 2017. [Online]. Available: <http://www.iec.ch/functionalsafety/explained/>. [Accessed: 14-Jul-2017].
- [115] ISO, "ISO 26262-1:2011 - Road vehicles -- Functional safety -- Part 1: Vocabulary," 2011. [Online]. Available: <https://www.iso.org/standard/43464.html>. [Accessed: 14-Jul-2017].
- [116] IEC, "IEC 62304:2006 | IEC Webstore | Medical device software - Software life cycle processes," 2006. [Online]. Available: <https://webstore.iec.ch/publication/6792>. [Accessed: 14-Jul-2017].
- [117] A. Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," *Embed. World*, vol. 2004, pp. 235–252, 2004.
- [118] K. M. Chandy, M. Charpentier, and A. Capponi, "Towards a theory of events," in *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, 2007, pp. 180–187.
- [119] K. M. Chandy, "Event Driven Architecture," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Springer US, 2009, pp. 1040–1044.
- [120] J. Van Hoof, *How EDA extends SOA and why it is important*. unpublished, 2006.
- [121] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Commun ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978.
- [122] O. Etzion and P. Niblett, *Event Processing in Action*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2010.
- [123] Y. Evchina and J. L. M. Lastra, "Semantic-driven CEP for delivery of information streams in data-intensive monitoring systems," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, 2015, pp. 1251–1256.
- [124] MESA International, "MESA White Paper #39: MESA Model Evolution," 2011.
- [125] IBM, *REST: Advanced Research Topics and Practical Applications* | Cesare Pautasso | Springer. 2015.
- [126] M. Garetti, "P-PSO Ontology for Manufacturing Systems," 2012, pp. 449–456.
- [127] B. Ramis Ferrer and J. L. M. Lastra, "An Architecture for Implementing Private Local Automation Clouds Built by CPS," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017.
- [128] B. Ramis Ferrer, S. Iarovyj, L. Gonzalez, A. Lobov, and J. L. Martinez Lastra, "Management of distributed knowledge encapsulated in embedded devices," *Int. J. Prod. Res.*, pp. 1–18, Dec. 2015.

- [129] B. Ramis Ferrer and J. L. Martinez Lastra, "Private local automation clouds built by CPS: Potential and challenges for distributed reasoning," *Adv. Eng. Inform.*, vol. 32, pp. 113–125, Apr. 2017.

Publications

This part of the thesis present a reproduction of the original publications related to the doctoral research. As it can be seen, the reproduction of each publication has been requested to the corresponding organisation.

ORIGINAL PAPERS

I

KNOWLEDGE-BASED WEB SERVICE INTEGRATION FOR INDUSTRIAL AUTOMATION

by

Borja Ramis Ferrer, Luis Gonzalez, Sergii Iarovy, Andrei Lobov, José L Martínez Lastra, Valeriy Vyatkin, William Dai, July 2014

12th IEEE International Conference on Industrial Informatics (INDIN)

2014 IEEE. Reprinted, with permission, from Borja Ramis Ferrer, Luis Gonzalez, Sergii Iarovy, Andrei Lobov, José L Martínez Lastra, Valeriy Vyatkin, William Dai, Knowledge-Based Web Service Integration for Industrial Automation, 12th IEEE International Conference on Industrial Informatics (INDIN), July 2014.

Knowledge-based web service integration for industrial automation

Borja Ramis¹, Luis Gonzalez¹, Sergii Iarovy¹, Andrei Lobov¹,
José L. Martinez Lastra¹, Valeriy Vyatkin^{2,3}, William Dai³

¹Tampere University of Technology, Tampere, Finland

²Aalto University, Helsinki, Finland

³Luleå University of Technology, Luleå, Sweden

{borja.ramis, luis.gonzalezmoctezuma, sergii.iarovy, andrei.lobov, jose.lastra}@tut.fi, vyatkin@ieee.org, william.dai@ltu.se

Abstract—Web services are widely used for enterprise software development. Web service protocols simplify application integration thanks to interface description that can be processed at runtime and, in addition, due to mature and widely used standards for transportation and internetworking. Implementation of service-oriented architecture starts to get acceptance in the field of industrial automation ranging from international research project to the first implementations in industry including first commercial controller devices supporting web service communication protocols and executing deterministic control at the same time. The current research step is to allow knowledge-based integration of industrial automation systems and to exploit full potentials of run-time reconfiguration and adaptation of industrial systems. This paper demonstrates implementation of knowledge-based industrial system, the architecture and ontology model that can be generalized for implementation of other production systems.

Keywords—Industrial automation; web services; web service integration; knowledge-base; ontology; Web Ontology Language (OWL), SPARQL.

I. INTRODUCTION

Traditionally, industrial automation systems are designed and implemented in a hierarchical structure. This is also known as the industrial automation pyramid [31]. This design approach imposes several challenges from an integration perspective because horizontal and vertical communications are heterogeneous. Therefore gateways are needed between layers in order to communicate different protocols. Changes in the components or topology imply a big cost in reintegrating the system.

In the last years, software integration has been pushed towards Service Oriented Architecture (SOA) design, where the functionality of the components of a system, is encapsulated and exposed as services [1]. The most common implementation of SOA is by encapsulating software components in Web Services (WS). SOA offers many benefits, the most important of which are: seamlessly vertical and horizontal communications, deployment of components across different networks and flexible reconfiguration of the system.

The academia and industry have started to adopt SOA architecture for industrial systems. Several research projects have targeted the use of WS for controlling [2] and monitoring

[3] purposes. Lot of work has also been done for deploying WS at industrial devices. The result of this is the creation of DPWS; a protocol stack aligned with WS specifications and tuned to be implemented in resource-constrained embedded devices [4].

On the other hand ontologies have demonstrated to be a flexible, extensible and scalable mechanism to describe and store information that models the components of a system. The Web Ontology Language (OWL) [5] offers rich properties to describe knowledge about entities and their relations. An important feature of OWL models is that they can be distributed and accessed through networks.

This work proposes the use of OWL to describe completely all the levels and parts that compose an industrial automation system. Different components of the industrial system are modeled, such as: equipment, actuators and sensors. Control parameters that are used for the orchestration of the processes like physical layout of the equipment or current location of the products across the production system. Enterprise level information like requested orders or definition of operations per product are also described in OWL.

The model is exposed to the other components of the industrial system as a Knowledge-based Web Service. The information contained in models can be queried and updated by other entities as, for instance, industrial orchestration engines, user interfaces and devices. All this components converge in the use of a Knowledge-based service, which acts as an integrator node.

This paper proposes architecture to achieve the previous functionality. The different components of the system have been deployed as WSs across different networks to demonstrate the versatility and scalability of the approach.

The rest of the paper is structured as follows: Section II presents a start of the art of the core developments, focused in the fields that this work encompasses. Section III describes the proposed architecture that allows the integration of the different components of an industrial automation system, through a Knowledge-based service. The description of knowledge modeling and reasoning is detailed in Section IV. Finally, Section V summarizes and identifies points for further work.

II. STATE OF THE ART

In this section, the overview of research in implementation of SOA and Knowledge-based systems concepts and related technologies for factory automation domain is described.

A. Web Services

WS is widely assumed to be a preferred set of standards to implement SOA [6], [7]. W3C defines two main classes of WSs in the WS architecture: Representational State Transfer (REST)-compliant WS and Arbitrary WS. Both service architectures provide operations to be invoked, but REST restricts these operations to be “stateless”, i.e. context or state of service provider does not have impact on the operation result [8].

Arbitrary WS, also referred as SOAP WS, are more mature and employ more generic architecture and include more specified standards to address discovery, addressing, security and functionality of WS. Having more standards SOAP WS are more complicated in development, although provide more functionality, such as dynamic discovery, asynchronous messaging, notification mechanisms and others. SOAP WS standard was defined earlier and together with the benefits it provides it is often employed for enterprise integration applications [9], [10]. In addition, several endeavors took place last years to adapt SOAP WS technology for devices. The most prominent realizations are Device Profile for Web Services (DPWS) and OPC-UA. Both of these technologies were applied by real devices [11], [12], [13], [14].

REST architecture is also defined by employment of uniform interface, which allows decoupling functionality [15]. This WS architecture was developed considering mainly HTTP protocol and as result RESTful WS can be interacted with via web browser with minimal additional software, which simplifies development of user interfaces for the system. REST services are widely considered to be a lightweight and simple solution for SOA implementation. These and other benefits in application to enterprise applications were studied in several research works, but still, to the knowledge of authors, the concept has not been implemented for industrial controller devices [11], [12].

Web services are also compatible with Event-driven Architecture. Business processes require more than one service to be employed in order to execute business logic and these processes should be invoked by the events in the system. This leads to event driven nature of the processes in dissimilar systems including industrial ones. To address this issue Event-driven SOA (EDSOA) concept was introduced. The EDSOA concept combines benefits provided by SOA with the ones provided by event-driven architecture such as asynchrony, modularity and concurrency [18], [19].

B. Knowledge-based systems

While SOA provides facilitation to interactions for knowledge-based system, the service description should be machine-readable. The means for such description lies in semantics, which can provide metadata about devices. In [20], it is defined that for explicit knowledge representation of

services employed in factory automation domain Web Ontology language as a mature mean to implement a Knowledge Base (KB).

OWL is RDF (Resource Definition Framework) based XML language, which provides level of abstraction required for Knowledge Representation (KR). Employing object-oriented perception, it introduces concepts of classes, properties and individuals in order to describe the structured information. There are several OWL dialects that enable creation of the complex models, which are both machine and human readable [21].

SPARQL Protocol and RDF Query Language (SPARQL) allows retrieving the data from ontology using simple querying. SPARQL standard was developed to satisfy needs of data access [22]. In order to modify RDF graphs, SPARQL update may be employed. This language allows updating RDF graphs by adding, removing or modifying resources and their relations [23]. OWL being RDF based also can be queried within SPARQL and SPARQL Update queries. OWL and SPARQL languages are providing sufficient means to keep the KR of system up to date [24] and even sufficient to control the workflow execution based on the knowledge base [25].

III. ARCHITECTURE

Usually, information and communication technology (ICT) infrastructure of industrial automation systems follows a layered architecture, commonly known as automation ICT pyramid. The proposed architecture resembles more a star topology having a Knowledge-based service as an integrator element that is used by other components. Four main elements compose this architecture:

- Knowledge-based service,
- Physical equipment,
- Orchestration service and
- User Interface.

The star topology central element is the Knowledge-based service component, which holds the information of the whole system. The physical equipment refers to the actual industrial sensors, actuators and controllers, which are located in the factory floor. These devices map their status into the knowledge-based component. The orchestration service is in charge of controlling and driving the physical equipment to achieve the desired task and the user interface, where operators or managers can visualize the state of the production, or request an order to be produced. The proposed architecture is depicted in Fig. 1.

The proposed architecture follows the SOA paradigm, where all the components are encapsulated and exposed as WSs. Most of them are deployed as RESTful WSs, while few others are deployed as DPWS. This provides four main features:

- Components can communicate between themselves across networks. Requests can pass easily through firewalls.

- Reconfigurability is improved, since components can be moved between platforms without major configuration changes.
- Seamless horizontal and vertical communications.
- Reuse of tools and libraries used in the Information and Communication Technologies (ICT) realm.

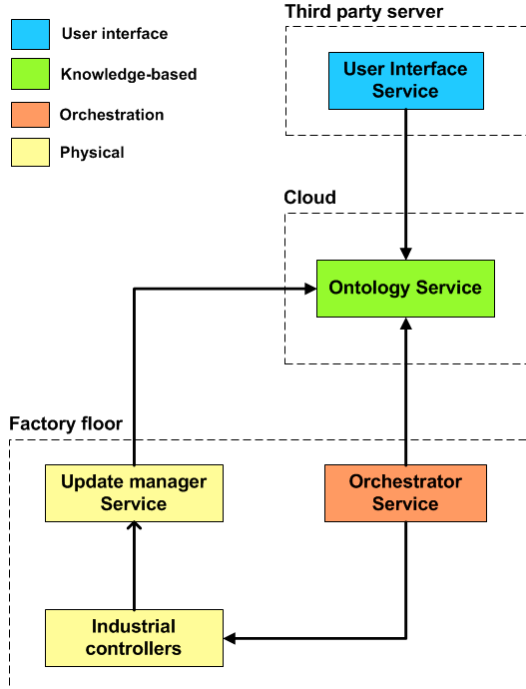


Fig. 1. Proposed Architecture

The diagram in Fig. 1 shows in solid line boxes the implementations that compose each of these groups. The architecture groups are highlighted by the background color of the boxes and the dotted boxes specify the actual location, where these components were deployed.

A. Knowledge-based service

In order to describe and model all the components and information of the system, OWL models are used. OWL models inherit the great flexibility of ontologies making them readable to humans and machines.

In this work, an ontology API is encapsulated as a RESTful WS. This opens the possibility to execute SPARQL and SPARQL Update queries on the OWL models through RESTful WSs. The implementation of this component is referred as Ontology Service.

This service was deployed on a cloud platform; therefore it is accessible through the Internet from everywhere. This is necessary since the other components of the architecture can be deployed on different geographical places.

B. Physical equipment

The industrial controllers used in this work are connected to real sensors and actuators. The used controllers are S1000 from Inico Technologies [26]. These controllers perform scan cycle logic as a basic industrial controller, but use DPWS for communication purposes. The controller encapsulates and exposes its functionality as WS operations. These operations are invoked by the orchestration service to indicate the controller actions to be performed in the physical equipment.

The industrial controllers can notify to other services when relevant events occur in the process. This is done by using the WS-Eventing protocol [27], which is implemented by the DPWS stack. WS-Eventing allows publishing notifications just to the endpoints, which are subscribed to specific topics. Event-driven systems reduce considerably the network overhead when compared with polling systems and improves considerably its reconfigurability by using the subscription mechanism.

As it has been previously mentioned, the industrial equipment maps its current status to the knowledge-based OWL model. This is achieved by translating the WS events generated by the devices into SPARQL Update queries. This translation is done by a component called Update Manager service.

The Update Manager service works with a Rules-Actions list. This list contains a mapping of which SPARQL Update queries should be executed based on the received WS events. For instance, if a motor starts to move, the headers of the event are detected and the corresponding query that updates the OWL model is selected. This SPARQL Update query is then executed on the ontology service and the model is updated. Thus, the system ontology gets the latest status of a system.

C. Orchestration service

The orchestration service contains the business logic that controls the physical equipment. It gets the status of the equipment and the requested order from the Knowledge-based service. Therefore, the orchestration service is capable of submitting SPARQL queries and reading the SPARQL response document.

Once the information is acquired, the ontology service plans a set of actions that is executed in the physical equipment. The orchestration service invokes in the right sequence these operations by invoking WS operations on the physical equipment, which as mentioned earlier, implements the DPWS stack.

D. User interface

The User Interface (UI) is the access point for human interaction with the system. The UI offers the possibility to entry new orders to the system and visualize graphically the state of the equipment. Since the orders and equipment status is contained in OWL models, the user interface service is capable of executing SPARQL Update (INSERT type) queries for adding new orders and SPARQL (SELECT type) queries to read the status of the equipment. Hence, the UI service

consumes the functionality exposed by the Ontology Service through RESTful WSs.

This UI service is implemented with HTML and JavaScript; therefore, the Ontology Service is consumed through AJAX requests.

It is important to notice, from architecture proposed in Fig. 1, that all the services could be deployed within the same local network, for example, at the factory floor facilities. Nevertheless, by placing the ontology service on the cloud, third parties can interact with the system and offer services, for example an user interface, which can be located in another server across the Internet. The selected deployment is to demonstrate the potential of the architecture and different deployment configurations can be achieved with minor settings.

IV. KNOWLEDGE MODEL AND REASONING

A. Production Line system description

An ontology model is developed for representation of the production line system, which is visualized in the web browser by the UI service. The layout of the modelled system is shown in the following Fig. 2.

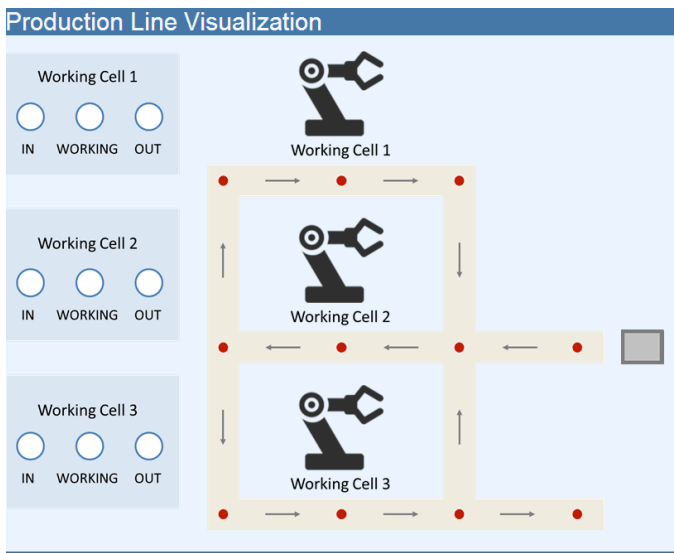


Fig. 2. Production Line system visualization

Previous visualization sketches a production line system that consists of three manufacturing cells. Each cell contains one robot that can perform distinct assembling operations, in which their corresponding components are needed to produce a desired product. In addition, each cell includes one conveyor that is the transport mechanism and link between different manufacturing cells. Observe that meanwhile the red circles depict conveyor zones; the grey arrows show the flow of pallets within the system. In addition, the pallet is represented by a rectangle that is animated according to the system model updates. Finally, some light indicators are represented in the left side of the visualization to show users to verify that the lights, pallet flow and system model coincide about the pallet

location. Note that the light indicators are not described in the model since is a feature of the visualization for facilitating the user interpretation of the pallet flow.

B. Modelling the system using OWL

During the last years a large amount of researches have been used OWL for modelling systems at different domains. Although it is true that a system can be modelled using an ontology from scratch, it is likewise true that following a methodology the ontology designer can perform models in an easier manner. A simple knowledge-engineering methodology for designing ontologies is described in [28] presenting seven steps to design an ontology. The second step of the methodology, describes several advantages about reusing existing ontologies. For the system modelling presented in this article, an existing model in has been reused.

In fact, [29] presents an OWL domain model that describes a generic production system ontology, adaptable to different use cases. Then, the model of the production line presented in previous Fig. 2 is based on the one presented in [29] because, for instance, the domain is the same, there is a common terminology between models and class hierarchy can be reused.

The following Fig. 3 presents a class diagram that is useful for representing the hierarchy of ontology concepts using the UML standard notation.

The presented class diagram is not only useful for representing given 14 main ontology concepts (classes), but also describes object properties that are included in the OWL model showing relationships that class instances inherit.

The OWL model of the system describes a set of default instances (individuals), which are listed in below Table I. The individuals that are real world devices are easily identifiable elements in the presented system visualization (Fig. 2). On the other hand, some status, assembly operations and components are also defined. These reflect operational status of the system.

TABLE I. DEFAULT MODEL INSTANCES

| Class | Instances |
|-------------------------|---|
| Conveyor | conveyor_1, ..., conveyor_3 |
| ConveyorZone | input_Cell_1, ..., input_Cell_3 |
| | output_Cell_1, ..., output_Cell_3 |
| | systemInput, systemOutput |
| | workingPosition_Cell_1, ..., workingPosition_Cell_3 |
| ManufacturingCell | manufacturingCell_1, ..., manufacturingCell_3 |
| Robot | robot_1, ..., robot_3 |
| ManufacturingCellStatus | CelldownStatus, CellworkingStatus |
| RobotStatus | downStatus, executingStatus, idleStatus |
| AssemblyOperation | assemblyOperation_1, ..., assemblyOperation_6 |
| Component | component_A, ..., component_F |

Table I is labelled as 'default model instances' because as it is explained in Section III, the model changes based on execution of SPARQL Update queries. Thus, by default, there are no pallet or product individuals defined. These are instantiated at runtime as actual physical pallets enter the system and products are made. Similarly, the instances can be deleted from the knowledge base, as if it is no longer needed (e.g. the pallet has successfully gone through the

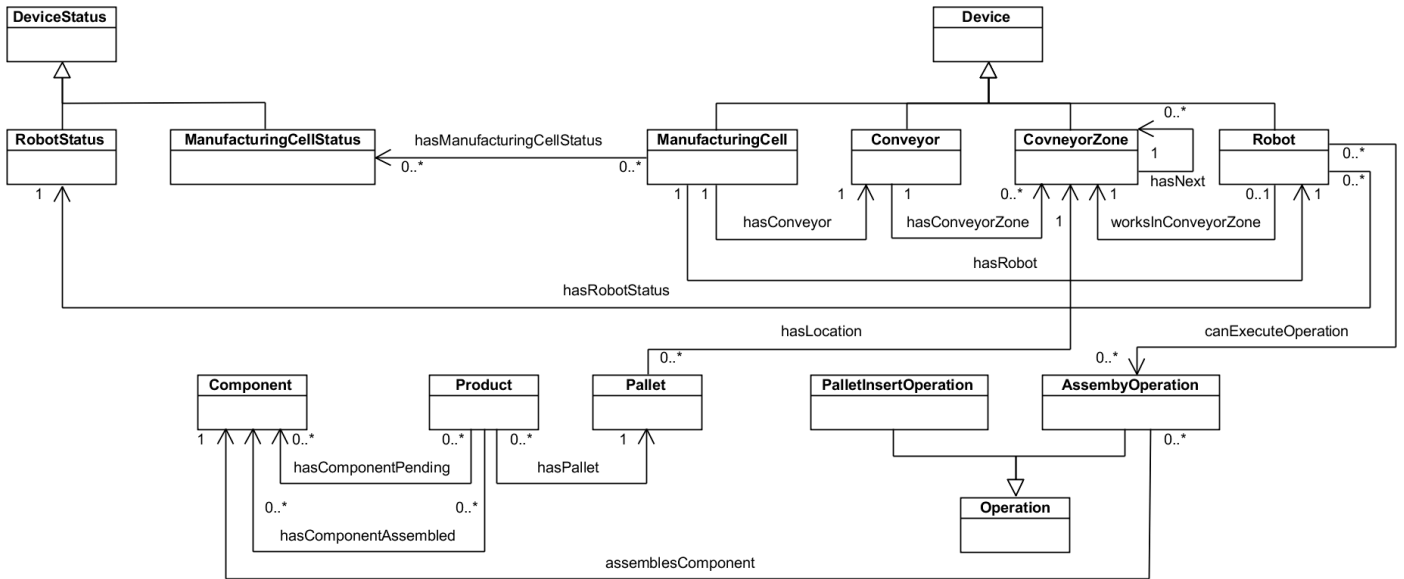


Fig. 3. Production Line system ontology class diagram

system and the product has been made; thus, no need anymore to represent information on the pallet and its location). In the same way, the starting relationships between device and status of robots and manufacturing cells are *idleStatus* and *CellworkingStatus* respectively, which can be updated as production system is running.

C. Query execution for updating and checking the model state

Section III described how the implemented architecture includes an Ontology Service that permits the interaction of other modules and the OWL model. Meanwhile SPARQL queries are used for checking desired information about the actual state of the model; SPARQL Update queries are used for manipulating it to achieve the representation of, for instance, the location of pallets in the system.

Fig. 4 shows a SPARQL Update query used for inserting a pallet into the system. This action involves the definition of three statements that can be understood through the SPARQL syntax. Firstly, a *pallet_product_1* instance is introduced in the *Pallet* class of the model. Secondly, the pallet instance location is defined as *systemInput*, which means that the pallet is placed in the input of the production line and, hence, transported in that position of the visualization. Finally, a product is linked to the pallet that is introduced into the system to map the product and pallet instances. Observe that the product must be previously included to the system within another SPARQL Update query, which also defines its pending components that have to be assembled.

```
PREFIX bo:<http://www.tut.fi/FAST/productionLineSystem#>
INSERT DATA {
bo:pallet_product_1 a bo:Pallet.
bo:pallet_product_1 bo:hasLocation bo:systemInput.
bo:product_1 bo:hasPallet bo:pallet_product_1.
}
```

Fig. 4. SPARQL Update query for inserting a pallet in the system

As it has been stated previously, the state of the model can be checked using SPARQL queries. An example of using a SELECT type query for checking the location of a specific pallet in the system is shown in following Fig. 5.

```
PREFIX bo:<http://www.tut.fi/FAST/productionLineSystem#>
SELECT ?pallet ?product ?location
WHERE {
?product bo:hasPallet ?pallet. FILTER (?product = bo:product_1)
?pallet bo:hasLocation ?location.
}
```

Fig. 5. SPARQL SELECT query for checking the pallet location in the system

Then, the above code presents how a pallet that flows in the system can be located through a SPARQL. In this case, the pallet is selected harnessing the mapping that it has to a specific product. Then, the location of the found pallet is directly retrieved within last statement.

D. Model usage

Revising the architecture explanations in Section III, the WS interface developed in the Ontology Service allows the interaction with the model, which is kept in the cloud. The Update Manger service is an entity that its objective is to transform event notifications of the physical equipment into SPARQL Update queries. Moreover, observe that the model is not only available for different module functionalities, but also permits human users to interact through the designed UI.

In fact, the web browser UI allows the insertion of queries in commented text areas to directly check and/or manipulate the model. In this way, actions as pallet insertion demonstrated in Fig. 4 are possible. More detailed descriptions of the designed OWL model, as well as a guided test for interacting with the production line model, are available in [30].

V. CONCLUSION

This paper outlined the architecture and production system ontology. It has shown the main components, their use and querying (reasoning) principles for ontology model. The knowledge-based system and ontology is accessible online to test queries and get additional details on implementation. It should be noted that online system is not connected at the moment to actual physical layer, as it is not an intension to allow everyone to control the system, but to demonstrate the role and use of ontology. In the given online setup, the queries executed on cloud-hosted ontology service result in update of user interface, which in its turn also retrieves information on system status through the ontology service.

Currently, there are many solutions of production and manufacturing systems dealing with knowledge that do not use the SOA paradigm. Then, observe that adapters or gateway devices would need to be integrated with information systems as the one that is presented in this paper.

This approach permits a knowledge-based integration of industrial automation systems. Thus, the presented knowledge-based service integration exploits full potentials of run-time reconfiguration of industrial systems. The key to the reconfigurability of industrial automation system resides in capacity of the system to adapt its behavior according to changes in system KR.

In addition, another benefit of the proposed approach is the reusability of the OWL domain model. The presented model describes a generic production system ontology, adaptable to different use cases in the manufacturing domain.

The equipment is located in FAST lab at Tampere University of Technology and it has been demonstrated during ARTEMIS-ITEA2 Co-Summit in December 2013.

In the future work we plan to elaborate basic architecture blocks performance and handling of exceptional cases at the production floor. In addition, runtime knowledge aggregation principles have to be elaborated, as the use of distributed Knowledge Bases and reasoning capabilities at embedded device level may be a key enabler for wide acceptance of knowledge-based systems for different domains.

On the other hand, updates on the model are performed by using an algorithm of queries. Further on, we plan to add a set of rules to support the knowledge and we expect to infer the model with the use of reasoner. Observe that in the actual implementation, the reasoner allows to check the consistency of the model once new instances are added due to the production process progress.

ACKNOWLEDGMENT

This work has been funded by the ARTEMIS First Call 2012 Program under Grant Agreement number 332946, correspondent to the eScop project: *Embedded systems for Service-based control of Open Manufacturing and Process Automation*.

REFERENCES

- [1] A. Lobov, J. Puttonen, V. V. Herrera, R. Andiappan, and J. L. M. Lastra, "Service oriented architecture in developing of loosely-coupled manufacturing systems," in 6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008, 2008, pp. 791–796.
- [2] I. M. Delamer and J. L. M. Lastra, "Self-orchestration and choreography: towards architecture-agnostic manufacturing systems," in 20th International Conference on Advanced Information Networking and Applications, 2006. AINA 2006, 2006, vol. 2, p. 5 pp.–.
- [3] A. N. Lee and J. L. M. Lastra, "Data aggregation at field device level for industrial ambient monitoring using Web Services," in 2011 9th IEEE International Conference on Industrial Informatics (INDIN), 2011, pp. 491–496.
- [4] A. V. Ramos, I. M. Delamer, and J. L. M. Lastra, "Embedded service oriented monitoring, diagnostics and control: Towards the asset-aware and self-recovery factory," in 2011 9th IEEE International Conference on Industrial Informatics (INDIN), 2011, pp. 497–502.
- [5] "OWL Web Ontology Language Reference." [Online]. Available: <http://www.w3.org/TR/owl-ref/>. [Accessed: 05-Mar-2014].
- [6] "SOA and Web Services." [Online]. Available: <http://www.oracle.com/technetwork/articles/javase/soa-142870.html>. [Accessed: 05-Mar-2014].
- [7] "IBM developerWorks: New to SOA and web services." [Online]. Available: <http://www.ibm.com/developerworks/webservices/newto/service.html>. [Accessed: 05-Mar-2014].
- [8] "Web Services Architecture." [Online]. Available: <http://www.w3.org/TR/ws-arch/>. [Accessed: 04-Mar-2014].
- [9] M. Benaissa, A. Benabdelhafid, M. Baccouche, and A. Alim, "Integration of manufacturing production and planning based in SOAP system," in 2004 IEEE International Conference on Industrial Technology, 2004. IEEE ICIT '04, 2004, vol. 2, pp. 938–943 Vol. 2.
- [10] V. V. Herrera, A. Bepperling, A. Lobov, H. Smit, A. W. Colombo, and J. Lastra, "Integration of Multi-Agent Systems and Service-Oriented Architecture for industrial automation," in 6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008, 2008, pp. 768–773.
- [11] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains," in International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006, 2006, pp. 43–43.
- [12] A. Cannata, M. Gerosa, and M. Taisch, "SOCRADES: A framework for developing intelligent systems in manufacturing," in IEEE International Conference on Industrial Engineering and Engineering Management, 2008. IEEM 2008, 2008, pp. 1904–1908.
- [13] J. M. Mendes, A. Rodrigues, P. Leitao, A. W. Colombo, and F. Restivo, "Distributed Control Patterns using Device Profile for Web Services," in Enterprise Distributed Object Computing Conference Workshops, 2008 12th, 2008, pp. 353–360.
- [14] E. Zeeb, G. Moritz, D. Timmermann, and F. Golatowski, "WS4D: Toolkits for Networked Embedded Systems Based on the Devices Profile for Web Services," in 2010 39th International Conference on Parallel Processing Workshops (ICPPW), 2010, pp. 1–8.
- [15] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, 2000.
- [16] W. Junye, M. Lirui, and C. Hongming, "A REST-Based Approach to Integrate Enterprise Resources," in International Forum on Computer Science-Technology and Applications, 2009. IFCSTA '09, 2009, vol. 3, pp. 219–223.
- [17] G. Mulligan and D. Gracanic, "A comparison of SOAP and REST implementations of a service based interaction independence middleware framework," in Simulation Conference (WSC), Proceedings of the 2009 Winter, 2009, pp. 1423–1432.
- [18] S. Overbeek, B. Klievink, and M. Janssen, "A Flexible, Event-Driven, Service-Oriented Architecture for Orchestrating Service Delivery," IEEE Intell. Syst., vol. 24, no. 5, pp. 31–41, Sep. 2009.

- [19] Z. Laliwala and S. Chaudhary, "Event-driven Service-Oriented Architecture," in 2008 International Conference on Service Systems and Service Management, 2008, pp. 1–6.
- [20] J. L. M. Lastra and I. M. Delamer, "Semantic web services in factory automation: fundamental insights and research roadmap," *IEEE Trans. Ind. Inform.*, vol. 2, no. 1, pp. 1–11, Feb. 2006.
- [21] "OWL Web Ontology Language Reference." [Online]. Available: <http://www.w3.org/TR/owl-ref/>. [Accessed: 05-Mar-2014].
- [22] "SPARQL Query Language for RDF." [Online]. Available: <http://www.w3.org/TR/rdf-sparql-query/>. [Accessed: 05-Mar-2014].
- [23] "SPARQL 1.1 Update." [Online]. Available: <http://www.w3.org/TR/2013/REC-sparql11-update-20130321/#sec-intro>. [Accessed: 05-Mar-2014].
- [24] A. Lobov, F. U. Lopez, V. V. Herrera, J. Puttonen, and J. L. M. Lastra, "Semantic Web Services framework for manufacturing industries," in *IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008*, 2009, pp. 2104–2108.
- [25] J. Puttonen, A. Lobov, and J. L. Martinez Lastra, "Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2349–2359, Nov. 2013.
- [26] "Inico Technologies." [Online]. Available: <http://www.inicotech.com/>. [Accessed: 05-Mar-2014].
- [27] "Web Services Eventing (WS-Eventing)." [Online]. Available: <http://www.w3.org/Submission/WS-Eventing/>. [Accessed: 05-Mar-2014].
- [28] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," 2001.
- [29] J. Puttonen, A. Lobov, and J. L. M. Lastra, "Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems," in 2013 IEEE 20th International Conference on Web Services (ICWS), 2013, pp. 419–426.
- [30] "eScop Demonstration Wizard." [Online]. Available: <http://www.escop-project.eu/teaser/>. [Accessed: 05-Mar-2014].
- [31] ISA, "ISA-95 Manufacturing Enterprise Systems Standards", March 2011

II

**CYBER–PHYSICAL SYSTEMS FOR OPEN-KNOWLEDGE-
DRIVEN MANUFACTURING EXECUTION SYSTEMS**

by

Sergii Iarovyj, Wael M Mohammed, Andrei Lobov, Borja Ramis Ferrer, Jose L
Martinez Lastra, May 2016

Proceedings of the IEEE, Volume: 104, Issue: 5

2016 IEEE. Reprinted, with permission, from Sergii Iarovyj, Wael M Mohammed, Andrei Lobov, Borja Ramis Ferrer, Jose L Martinez Lastra, Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems, Proceedings of the IEEE, May 2016.

Cyber-Physical Systems for Open Knowledge-driven Manufacturing execution systems

Sergii Iarovy, Wael Mohammed, Andrei Lobov, Borja Ramis Ferrer and Jose L. Martinez Lastra

Tampere University of Technology, FAST laboratory, PO Box 600, 33101 Tampere, Finland

{sergii.iarovy, wael.mohammed, andrei.lobov, borja.ramisferrer, jose.lastra}@tut.fi

Abstract—Manufacturing execution systems play an important role of bridging high-level enterprise functions and production or manufacturing operations. The embedded systems are usually in charge of controlling execution of the operations. Modern embedded systems have become capable of simultaneous and deterministic execution of control algorithms and IP-based communication, making it possible to create complex cyber-physical systems (CPSs), where the computational and communication resources of a device can be used directly for various control, supervisory, or monitoring functions. The complexity for defining open-knowledge-driven manufacturing execution system (OKD-MES) is in maintaining awareness of overall system state to avoid disruptive actions as various functions may be requested from a system. The problem is that obtaining such information on system state may necessitate collecting data from a number of devices, as there may not be a single data point for state information. This paper describes and illustrates an approach for designing OKD-MES on top of CPSs that controls robot workstations and conveyor-based transportation system of a pallet-based production system.

Index Terms—Cyber-physical systems (CPSs); knowledge-based systems; manufacturing automation; manufacturing systems

I. INTRODUCTION

CONTEMPORARY manufacturing enterprises both big and small require new generation information systems to enable efficient operation of factories by reducing the time and costs of building and extending manufacturing system functionality, being aware of the current state and therefore being able to make better decisions. The desired efficiency can be achieved through the deployment of affordable manufacturing execution systems (MESs) that are easy to integrate with heterogeneous cyber-physical systems (CPSs) [1] operating at different levels of an enterprise. In order to improve the integration capabilities, various standards and methodologies have to be used for building such distributed networked systems. In addition, it could be beneficial to have system information represented in a common format at all the levels, so that the same language is used to describe heterogeneous components, which in its turn may reduce the time for system components integration.

The next section provides background in the areas of MESs, CPSs, distributed systems, and knowledge-driven approach, which are required to illustrate the approach and case study

presented, respectively, in Sections III and IV. These sections demonstrate how the conjunction of the mentioned areas could improve the development and exploitation of CPS in manufacturing. The main improvements and challenges related to the proposed approach are given in Section V. Section VI presents the conclusions and proposals for future work.

II. BACKGROUND

A. Manufacturing Execution Systems

The information systems applied in manufacturing enterprises are complex systems which should provide various functionalities and be adjusted to the needs of particular enterprise. The more complex the manufacturing process becomes the more challenging is the problem of efficient management of the enterprise. Addressing a problem of complexity, it is usually beneficial to apply a divide and conquer approach defining the independent problem parts and resolving them separately. The efforts to determine which functions the information system components have to provide have been duly applied and resulted in several standards, the most prominent of which are Purdue Enterprise Reference Architecture (PERA)¹ and ANSI/ISA-95².

ANSI/ISA-95 defines a hierarchy of control between the enterprise resource planning (ERP) systems and control systems. In this hierarchy, the MESs assume the role of connecting the office ERPs with the shop-floor equipment implementing manufacturing operational management (MOM) functions in the enterprise. In [2], the following definition of MES is proposed: “A manufacturing execution system... is an online integrated computerized system that is the accumulation of methods and tools to accomplish production,” where “online” is used as connected. Another important organization Manufacturing Execution Systems Association (MESA) [3] generally assigns MESs the same role. Moreover, as Table 1 shows, ISA-95, MESA, and Verein Deutsche Ingenieure (VDI)³ propose that almost identical sets of functions be provided on MOM level.

¹ <http://www.pera.net/>

² <https://www.isa.org/isa95/>

³ <https://www.vdi.de/>

Table 1. Functions of MES systems

| MESA | ISA-95 | VDI |
|---------------------------------|---------------------------------|---|
| Operations / detail scheduling | Detailed Production | Detailed planning and detailed scheduling control |
| Resource allocation and status | Scheduling | |
| Document control | Production data collection | Operating resources management |
| Dispatching production units | Production resource management | |
| Performance analysis | | |
| Labor management | Product definition Management | Personnel management |
| Maintenance management | Product tracking | Data acquisition and processing |
| Process management | Production dispatching | Interface management |
| Quality management | Production execution | Performance analysis |
| Data collection and acquisition | Production performance analysis | Quality management Information management |
| Product tracking and genealogy | | |

For this study, MES function's classification by MESA model [3] was selected. In the model, "resources allocation and status" functions provide access to all resources in MESS and manage the needs for material from suppliers. "Operations/detail scheduling" functions translate orders to shop-floor schedules and may provide the estimated time for product delivery. Decomposition of an order to a task for the shop floor and updates of the order during execution are implemented within "dispatching production unit" functions. "Document control" functions facilitate document generation and flow in the manufacturing system. "Data collection/acquisition" functions accumulate all data in the manufacturing system, especially from shop-floor hardware and provide these data to other functions. Management of shift schedules for workers belongs to the scope of "labor management" functions. "Quality management" functions facilitate the quality assurance in the manufacturing system in general. "Process management" function takes control to ensure the correct process on the shop floor and issues instantaneous alarms to anticipate and mitigate faults early. "Maintenance management" function provides maintenance schedules for shop floor by alarm management or by periodic maintenance. "Product tracking and genealogy" functions record production information for each product in order to apply performance analysis or for faults recall. "Performance analysis" function analyzes the performance of the facility (production rate, energy consumption, and order manipulation) with the support of graphs.

Commercially available MES solutions are mostly centralized systems, tightly coupled horizontally (between MES functions) and vertically (with ERP and shop floor). Such systems are usually developed by business intelligence or industrial automation companies (e.g., SAP and Siemens, respectively). This leads to a strong connection between such MES solutions and the ecosystem of other software provided by the company and arguably reduces the openness for integration with third party components. A tightly coupled approach increases the challenge of customization for enterprise needs, and consequently the introduction of an MES solution

usually entails a tradeoff between the modifications of software solution and business process.

Currently, MESSs are mostly used by large enterprises following a mass production paradigm. For such enterprises the production volumes are significant, while the variation in the products is limited [4]. Nevertheless, the trends in industry are heading toward increasing customization [5] and an increasing role of SME in economy [6].

The adaptation of conventional MES solutions may be possible, but the authors discuss a more efficient concept for MES system. The requirement of modularization for MESSs is suggested by Kletti in 2007 [7]. In more detail, the concept of modularized MESSs is proposed in [8]. Bratukhin and Sauter in 2011 suggest the concepts for distributed MESSs. The eScop⁴ solution—open-knowledge-driven MES (OKD-MES)—proposes combining the modularity of the system with the knowledge-driven approach.

The core concepts on which the architecture of OKDMES is based are CPSs, service orientation, and knowledge-driven approach. Contemporary ERP solutions commonly provide web service interfaces for integration. Given the service-oriented nature of OKD-MES, the challenge of integration with ERP could and should be addressed through the shared knowledge about the services available in MESSs and ERP. Integration with the factory shop floor in turn is more challenging, as in addition to the shared knowledge it is mandatory to enable web services on the constrained equipment of the factory shop floor. More details on OKDMES concepts in relation to CPSs will be provided in Section III, while later in this section the background on the named core concepts will be provided.

B. Cyber-Physical Systems

As the computational power of embedded systems has increased and information and communication technologies (ICT) standards have been refined to support the development and integration of large-scale networked systems, it has become possible to design, implement, and distribute networked

⁴ <http://escop-project.eu/>

embedded systems currently known as CPSs.

Rajkumar et al. [9], [10] outlined the profound effects that development of CPSs may have on society. These include improvements in the energy sector, which could start using smart power grids [11]. CPS also tends to affect the automotive industry and transportation systems in general, the first successful experiments having already been carried out [12] with autonomous vehicles. In agriculture, deployment of CPS may increase yields due to better monitoring of the condition of crop fields as well as the utilization of autonomous machines. CPS requires new approaches to create real-time embedded software [13].

The development of CPS requires multidisciplinary knowledge. Besides knowledge of the domain in which the system and its components physically operate, knowledge of software, communication, control theories, methods, and tools is required. Therefore, the management of interdisciplinary knowledge in order to build a CPS demands good engineering methodologies. In order to address issues arising in the implementation of generally loosely coupled CPS [14], a cyber-application framework is proposed. The framework advocates the use of a number of patterns to handle partially ordered knowledge for building pervasive applications.

New standards and new methodologies make it possible to seamlessly integrate information available on different levels into a single CPS. The decisions in such a system can be made autonomously based on the most recent situation observed at the “physical end”—the process, and throughout the “cybernetics” part of the system—the control functions which can be integrated at all the levels. Also, the manufacturing domain starts to benefit from recent approaches such as cloud computing [15] that provides tools, methods, and techniques to enlist additional computational resources on demand to support the enterprise.

OKD-MES can be positioned at level 3 according to ISA-95, between the factory shop floor and ERP components. MES, as an intermediate element in this perspective, should contain all the necessary functionality not only for covering traditional MES functions, but also to seamlessly link factory floor and enterprise business functionality.

C. Distributed Systems

Distributed systems are formed by components residing in networked computers, which coordinate their actions through the exchange of messages. One of the most valuable benefits and the main motivation for using distributed systems is the option to utilize resources that do not belong to the entity. The definitions and principles of distributed systems are described in [16].

In recent decades, aside from the utilization of PLCs for distributed process control, the factory automation domain has also introduced new types of embedded devices that permit the implementation of CPS in modern production lines. In the last few years, one approach that has been intensively researched and tested is the implementation of the service-oriented architecture (SOA) paradigm, which permits encapsulating the functionality of system components and exposing it as web

services (WS) [17], [18]. WS may be deployed in industrial devices employing different approaches such as WS-* from OASIS, OPC-UA model, or following RESTful architectural style. For WS-* one of the most important implementations is a device profile for web services (DPWS), which is a modified WS specification stack based on SOAP protocol that can be implemented in resource-constrained embedded devices. In this manner, SOA can be handled by networked devices located at different levels of an enterprise for controlling and monitoring control processes [19], [20]. These processes are physically executed on shop floors. Being based on the previous OPC standards, OPC-UA is widely accepted in industry and also provides SOAP web services. OPC-UA defines a complex data model which introduces the limits of the application, as it may bring unnecessary complexity to the business applications.

DPWS and OPC-UA are both based on SOAP-based web services rivalled significantly in recent years by the newer representation states transfer (REST) web services. The architectural constraints of REST have been developed to utilize the nature of the web, allowing more scalability and requiring simpler infrastructure to support the applications. The community using REST for general applications exceeded that of SOAP⁵ at the beginning of the 2010s and as a result the share of open RESTful APIs is currently over 70%.⁶ Constrained application protocol (CoAP) in turn is commonly employed in the Internet of Things domain, implementing simple representational state transfer (RESTful) architecture which enables simple mapping to generic HTTP RESTful services. Basic HTTP RESTful service may also find its niche in the industrial applications domain, mainly for monitoring and non-time-critical tasks.

The embedded devices enabling web services can serve as gateways permitting cross-layer integration between physical equipment and cyber-systems, managed and coordinated within the distribution of information and referred to as remote terminal units (RTUs). As a commercial example, S1000 (by Inico Technologies)⁷ is a WS-enabled controller that has been used in several research works for controlling the operations of modern production lines. Each device exposes the equipment functionality as service operations which can be invoked. Moreover, information messages are distributed among all the networked devices that participate on the process control, which allows the coordination of each device operation. Some studies using these devices for distributing data and controlling processes are described in [21] and [22].

D. Knowledge-Driven approach

The preceding sections described the application of CPSs in the industrial domain and presented tools and technologies enabling it in modern manufacturing enterprises. Today, one of the biggest problems that engineers confront is managing the large volumes of information, which flow between the

⁵ <http://www.google.com/trends/explore#q=%2Fm%2F077dn%2C%20%2Fm%2F03nsxd&cmpt=q&tz=Etc%2FGMT-2>

⁶ <http://www.programmableweb.com/search>

⁷ <http://www.inicotech.com/>

different layers of enterprises. In fact, the issue is not only to handle it but also to extract valuable data that permit the analysis, tracking, evaluation, and understanding of ongoing processes in production lines. This need is crucial in a time when each decision on manufacturing processes directly affects the economies of organizations.

Knowledge representation (KR) and reasoning are a part of the artificial intelligence field that is concerned with describing world information in certain formalisms that can be interpreted and used by computer systems for accomplishing complex tasks. Main concepts and several formalisms (e.g., frames and ontologies) are described in [23]. In the industrial automation domain, the main benefit of KR is the creation of a system model which incorporates all the required information that is generated and consumed by manufacturing systems in both human and machine-readable form. Recent works in the field describe how the utilization of KR and its combination with SOA implementation facilitate the management of manufacturing system information [24], [25]. In fact, the scenarios presented in these works envision requirements for implementation of the CPS integration.

Recently, the knowledge-driven approach and main concepts for manufacturing systems have been presented in [26] as the solution for achieving the eScop project objectives. Garetti et al. present a multilayer architecture for organizing the entire manufacturing system that manages the orchestration of service operations execution by consulting a central system model, which is updated on runtime with the actual status of the system.

This knowledge-driven approach proposes the use of ontologies as a technology that in recent years has been utilized in the industrial automation domain. Among other benefits, ontologies offer a hierarchical and well-organized framework for the description of the system model. Additionally, ontologies enable reasoning—the process of the generation of new facts in the model based on available facts and predefined sets or axioms and rules. The reasoning automates runtime update and consistency check for the model. Applying semantic rules in ontologies it is also possible to implement data mapping and classification.

There are many languages for implementing ontologies [27] but the most used ones are the resource definition framework (RDF)-based ones [28]. Following the semantic web stack presented in [29], RDF is built on the top of the XML syntax, which does not include any semantic constraint, into the structured documents. Then, RDF Schema is used for defining the taxonomy of document resources, which means semantic constraints (attributes and types) that allow the interrelation of RDF resources in XML-based documents. Ontology models implemented within RDF are RDF graphs that are a set of RDF statements, or RDF triples. Such RDF triples consist of a concrete relation: subject–predicate–object. Ontology web language (OWL) [30] is an extension of RDF language that permits richer knowledge descriptions because it adds more vocabulary for describing types (e.g., disjoint and cardinality constraints) and attributes (e.g., symmetry). It should be noted that OWL is classified into distinct sublanguages that offer

different description capabilities: OWL-Lite, OWL-DL, and OWL-Full.

Since the decision by manufacturing system components depends on the status of the system, model information can be consulted within queries. Queries can be formulated within different languages but for querying RDF-based models, SPARQL protocol and RDF query language (SPARQL) [31] are the most used. On the other hand, RDF-based graphs can be updated within SPARQL update (SPARUL) [32], which is an extension of SPARQL. Briefly put, SPARUL is a language used for adding, removing, or modifying RDF triples. In fact, RDF graphs can be totally populated with instances through SPARUL queries. Recent studies showing some of the functionality of SPARQL and SPARUL languages in the industrial automation domain are described in [33]–[36].

III. APPROACH

This section presents a description of the approach for the creation of an open-knowledge-driven manufacturing execution system. As mentioned in the background, the MES operates on the level between ERP and shop floor and should connect them. From the bottom-up perspective ERPs are centralized cybernetic solutions (although technically ERP may be implemented in several components), while the control equipment on the shop floor is distributed and resource constrained. This means that the OKD-MES needs to interconnect two systems of different natures.

The basic set of requirements for an MES solution has been summarized in [7] by Kletti and includes ERP production system requirements mapping, modular and expandable nature, adaptability to process and functional needs, and standardized interfaces on all levels.

RTUs are commonly used as a gateway between physical shop-floor devices and systems on higher levels of the automation hierarchy. Considering the increasing computation and communication capabilities of such RTUs, the devices may provide their functions in the form of services, data, and descriptions from their physical components. Such capabilities reduce the required depth of coordination during the development of the system components. Although the recent developments in the field of embedded systems enable more functionality and data on the factory shop-floor level, such functions and data have limited scope and are machine centric (e.g., manufacturing operations, or device-specific data).

The MES solution should handle the shop-floor CPS and provide the functionalities of higher levels of abstraction and complexity. Moreover, a contemporary MES solution should be able to use data and descriptions hosted by the shop-floor device. This information provided by this device can be processed in order to obtain higher level knowledge of the manufacturing system. Such knowledge may be used to make the MES solution more flexible and loosely connected to the underlying shop floor.

Besides providing flexibility on the factory shop-floor level, a contemporary MES must itself be flexible. Such consideration transforms to a design requirement for modularity of MES architecture. The modular architecture should allow the selec-

tion of the optimal solution for MES functionality. Such an approach provides great benefit to the domain and is constantly requested by consumers. The concept for OKD-MES is being developed taking the aforementioned points into consideration. In the following sections the architecture of OKD-MES will be described.

The methodology to implement modular OKD-MES is based on SOA. Service orientation should allow the required level of loosely coupled integration to keep the parts interchangeable while maintaining the capability for the interaction of the system. A particular approach for the system architecture REST is selected in this approach for several reasons. First, REST RESTful-WS exploits the web-based nature of the system and uses ubiquitous Internet standards, thus providing an accessible toolkit for interactions with other systems and users. Finally, the application of RESTful services provides easy integration with the tools to place the orders in the system, whether through a third-party ERP system or directly from the system user.

The components of the OKD-MES may be grouped into two sets—those that provide core functionalities to the system and those introducing the additional MES functions exploiting the underlying core. The core functionalities should:

- define and handle configuration of the MES with relation to internal and external components;
- facilitate interactions with shop-floor equipment;
- facilitate interactions with MES users;
- facilitate interactions between the internal components of MES and external systems.

Following the needs for named core functionalities four main layers of OKD-MES core were defined: physical layer (PHL), representation layer (RPL), orchestration layer (ORL), and visualization layer (VIS). PHL in OKD-MES is embodied in service-enabled RTUs which expose the descriptions of controlled devices as well as available services and data.

PHL implements discovery protocols based on multicasting of Hello/Bye messages and on listening for discovery probing/heartbeat requests. Discovery functionality allows synchronization representation of the factory shop floor in RPL with real-world in-system runtime.

RPL serves to maintain the KR of the MES and related components. The knowledge about MES is represented in the manufacturing system ontology (MSO) [24]. Exploiting OWL as a language to describe the manufacturing system enables reasoning and querying for the required information when required in the system. RPL contains the system configuration, such as knowledge about component functionality and relations. Employing discovery protocols the system representation is being synchronized with the real world.

ORL enables the execution of sequences of operations provided by the system components according to defined processes. Considering that for the execution of most simple RESTful operations only URL is required, ORL is capable of requesting the next operations from other system components and of dynamically executing them in the system while maintaining the closed loop in the system.

Finally, the visualization layer exposes the user interfaces to

interact with system components. This layer is developed to dynamically adjust the user interfaces based on the system configuration available in the ontology. Such an approach reduces the burden on the system reconfiguration related to the development of the user interfaces. Fig. 1 presents the structure for OKD-MES including core layers and complimentary MES functions defined by MESA.

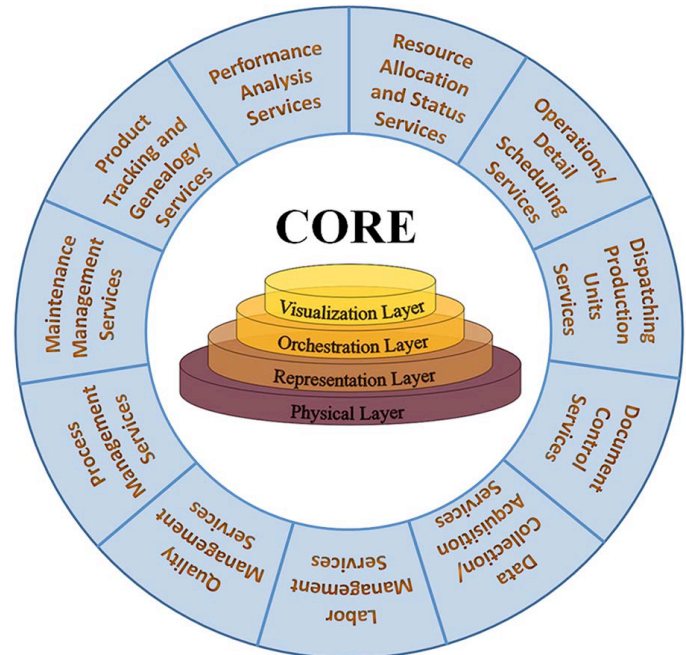


Fig. 1. Structure for OKD-MES.

The system status is primarily defined by the status of factory shop-floor equipment. Efficiency of dispatching is based on how precise the representation of the system status is at the moment of dispatching decisions and the possibility to predict the status of the system when the dispatched operation is executed. In case of an integrated OKD-MES solution the decision about dispatch may be made close to the moment of execution of the required operation, thereby reducing the possibility of an erroneous prediction of system status. Moreover, as the status of the equipment is provided by the equipment itself, possible distortions in the precision of system status representation can be improved. The improvement is possible because each piece of equipment starts to assess its status locally instead of reporting often-limited set of data to some central diagnostic application, which may become difficult to change or extend to support the extension of a production line. In mission-critical applications, where also the situation of wrong status reporting by a machine should be avoided or mitigated, the role of “external” observers for particular equipment can be taken by its neighbor machines. It becomes more computationally expensive in comparison to the use of a single supervisor, but can be paid back by simplified dynamic reconfigurability available at system runtime.

IV. CASE STUDY

To demonstrate the capabilities of the OKD-MES the sample MES function implementation will be described. This

section presents selected MES function, its design and implementation, and the application case.

The case study for a proposed solution is based on dispatching production unit (DPU) MES function. The DPU function should analyze the production order and dispatch it on the manufacturing equipment in the production line. This demonstrates vertical and horizontal integration for visual and understandable process. As well in OKD-MES DPU has to interact with all layers and may interact with other MES function implementations. Concluding, the DPU function implementation provides a comprehensive example of MES functionality, and consequently was chosen to be used as for current case study.

A. FASTory Line

As a testbed for the implementation of the DPU the FASTory production line (see Fig. 2) was selected. The line is used to simulate the process of mobile phone assembly. The real operations of mobile phone assembly are imitated by drawing the components on the pallet. Considering the nature of the robotic operations required for the real assembly process, scribed simulation is a relatively close approximation. The imitation process includes drawing the three main parts of mobile phone (frame, screen, and keyboard) in different colors and shapes, these variations provide 729 different products.

The FASTory line contains ten identical workstations which can draw the phone parts on paper, one buffer station for loading/storing empty pallets and one station for loading new paper onto the pallet and unloading the ready papers. In this sense, the paper represents the product while the colored shapes represent the components of the mobile phone.

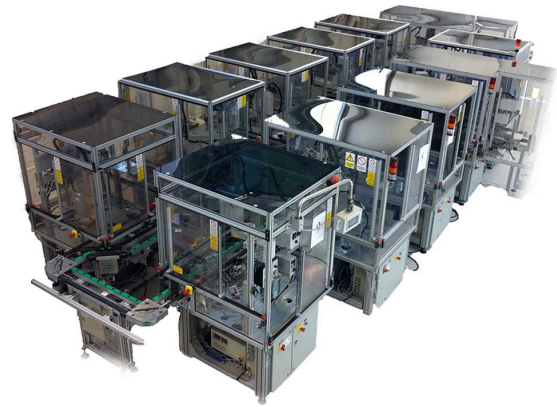


Fig. 2. FASTory line.

Fig. 3 presents the physical structure of FASTory workstations (to be referred to as W#). The pallet buffer station is labeled W7, paper loading station is labeled W1, and finally, the ten processing workstations are labeled W2–W6 and W8–W12. Each processing workstation contains two conveyor paths: a main conveyor to deliver a pallet to the robot and a bypass conveyor moving the pallet to the next station once the workstation is busy. The production line arranged in the closed loop. Such typology provides a continuous path for pallets, thereby increasing the productivity/space ratio [21].

All conveyors are divided into different zones (to be referred to as Z#) as marked in Fig. 3. The inlets and outlets of the workstations are located at Z1 and Z5, respectively, in all stations but W7, where the inlet is in Z2. The possible positions of the pallets in main and bypath routes are marked as Z2, Z3, and Z4. The processing point of each station is in Z3.

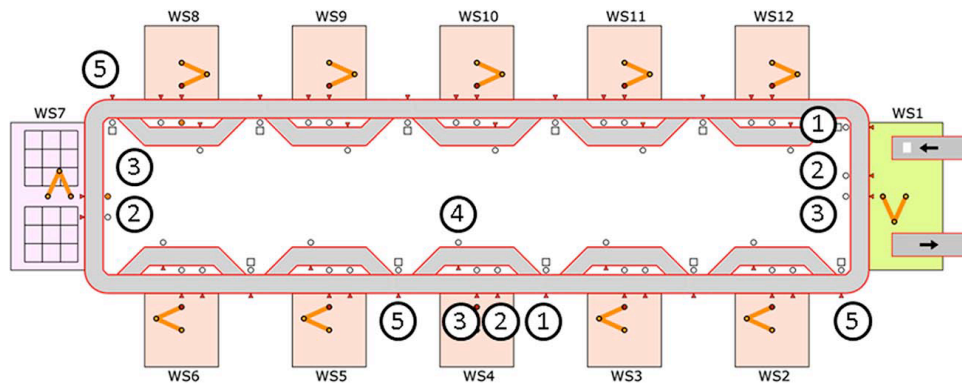


Fig. 3. FASTory line typology.

With this structure, FASTory is considered to be a flexible assembly line as a pallet can reach any workstation from any position. For each zone, there is a sensor to detect the pallet and a stopper to precisely stop the pallet. Z1 of each station also contains an RFID reader to read the pallet ID.

The FASTory line is equipped with S1000, WS-enabled controllers, managing the shop floor hardware. In addition to the generic controller functionality S1000 is capable of exposing the procedures and data from the line equipment in the form of RESTful services. Among such service the event subscription mechanism is developed. Such mechanism enables event-driven behavior in the system. The component based on

its internal logic may send the predefined events to the dynamic list of subscribers. For example, if orchestrator should trigger some process in response to the appearance of the pallet in Z1 of certain station, it may subscribe to corresponding event—*Z1_Changed*—provided by the controller in the line. For subscription, the client should provide the event sink to which the notification should be sent when status of Z1 changes.

Following the service-oriented constrains on development of control logic it is possible to encapsulate the underlying complexity and expose relevant level of abstraction to shop-floor service consumers. Additionally such approach enables

exploitation of the web simulators. The simulator for the FASTory line is described below.

B. FASTory Simulator

FASTory simulator was developed as a platform-independent solution. The web application is considered the strongest candidate among other solutions [35]. The FASTory simulator is a web server which hosts RESTful services and web pages that can be accessed via internet browser. The services are virtualizing the shop-floor functionalities, while the set of web pages provides basic information about simulator as well user interface including the visual representation of the status of simulated system.

Simulator significantly accelerates the development process and reduces the potential risks and cost of running a real system. Being online, web-based solution, the simulator provides realistic development environment open for general public. It simulates the real line in terms of interface and functionality.

In the scope of development of OKD-MES the simulator is used for development of ORL, RPL, and MES functions. The solutions developed and tested in the simulated environment are then deployed to real system. If the constraints of simulator were properly addressed in the development, migration from the virtual to real line is a seamless process. In the OKD-MES exploiting the capabilities of RPL and PHL the migration process is reduced to basic connection to the proper networks.

FASTory simulator is accessible online⁸.

C. DPU implementation

The implementation of the DPU function in OKDMES requires interaction with PHL, RPL, ORL, and other MES functions. DPU function should provide a service to deploy assign the operation to shop-floor equipment based on the order and status of the system. To facilitate the information about the status DPU should interact with RPL, requesting the information about the available and required functionalities. In order to enforce the decision execution the ORL can be used to manage the complexity of direct interactions with the PHL. Finally some other services such as implementation of operation/detail scheduling (ODS) or resource allocation and status (RAS) functions may be used in dispatcher for decision.

An example of the DPU configuration for FASTory line is provided below. The assumption is made that a certain number of pallets are constantly available to be introduced to the system. In such a case the appearance of the pallet on the inlet of any workstation is the event (*ZI_Changed*) which should trigger the ORL to request the displacement function for particular order in particular station. ORL analyzes the notification to retrieve pallet ID and workstation ID. These are the parameters for which ORL requests DPU to provide the list of production tasks to be dispatched.

Fig. 4 shows the sequence diagram for the scenario. In this representation, ORL focuses only on the request and execution of the task list. RPL provides information once it is required and dispatcher is the part which makes decisions for the production sequence.

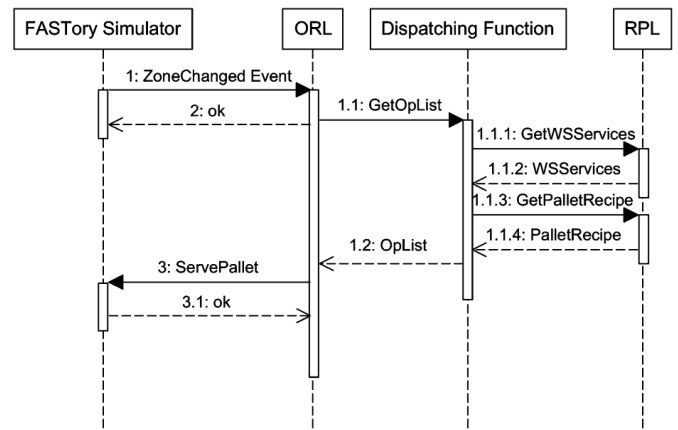


Fig. 4. Sequence diagram for the dispatching scenario

The particular decision in the scenario depends on the circumstances in which the DPU was called. If a pallet enters a zone of a workstation, DPU should analyze if the current or any of the following zones of the workstation may provide the services required for the current product. If there are no services to be provided for a product, the pallet is moved from the workstation in a predefined optimal path. If the operations provided by the workstation are required for a product and achievable by the pallet, the pallet has to be moved to the required zone and possible operations are to be executed. In the case of complex and interdependent operations in the product recipe, an additional request for dispatching may be issued after the execution of the manufacturing operations are complete, as new operations may become enabled for the product or in the equipment.

Such a decision-making process requires certain models for the representation of the required knowledge in RPL. The part of eScop MSO used in the FASTory DPU use case is depicted in Fig. 5. Conveyor may have event emitted when the pallet appears in one of its zones. Description of event is defined as a triggering event for the DPU, so ORL can subscribe to it. Such an event includes information about the location and ID of the pallet. The pallet belongs to the container concept in MSO and is related to a product from the production order. Product in turn has a production routing or routing, which is a sequence of operations which have to be performed in order to manufacture the product. The operations may match the description of services from the processor in a workstation. In addition to the semantic description, the service may have a technical description required for the invocation of the operation. In the FASTory scenario, the complete technical details are embedded in the URL of the service. Employing such a representation DPU by the set of interactions with the PHL and RPL may retrieve a set of executable URLs which invokes real operations in the manufacturing system based only on information about the location and ID of the pallet.

⁸ <http://escop.rd.tut.fi:3000/>

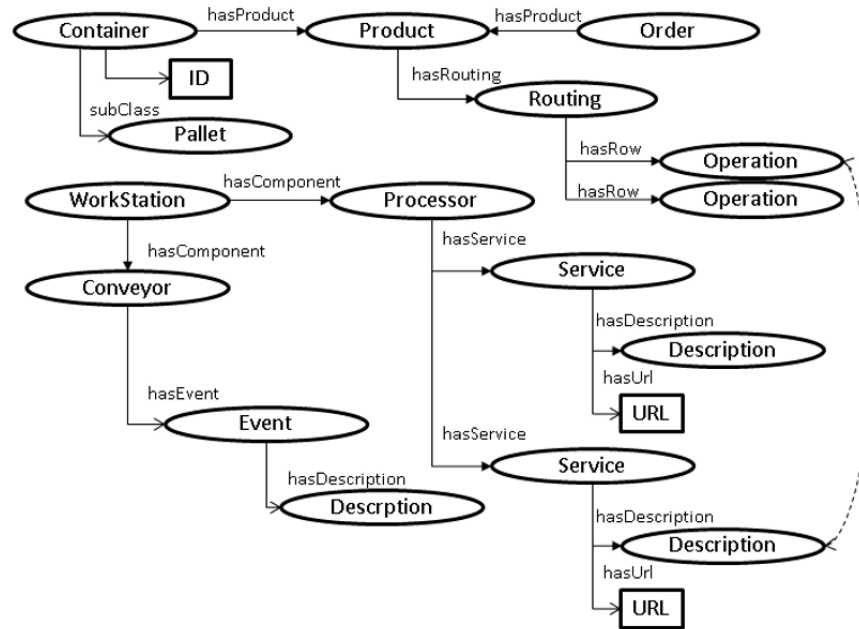


Fig. 5. Excerpt from eScop MSO required for DPU service.

V. DISCUSSION

Manufacturing systems would generally appear to be doing the same or similar things as before. Just as the appearance of an autonomous vehicle on the road [12] will not generally change its main function, which is to transport people and/or goods from some arbitrary point A to another point B, manufacturing systems equipped with OKD-MES will continue to deliver their main functionality—manufacturing of corresponding products. The adoption of new technologies, methodologies, and standards changes rather the nonfunctional requirements or qualities of a manufacturing system. It is possible to list the four most appealing improvements.

The first one is “time and cost reduction” for the development of MES. The use of common web standards and technologies developed and matured in the field of general ICT reduces development time and costs due to widely available and advanced tools in addition to available expertise and APIs for application development. For instance, it takes just several days for an engineer to develop a fully functioning online simulator of the production system presented in the previous section. It enables fast prototyping evaluating different ideas. Furthermore, the only tool needed to operate the simulator and the actual production line is a basic web browser.

The web browser can be used to visualize information on the line and to interact with it by invoking services on the controllers. For instance, among other tools Advanced REST Client by Google can be mentioned,⁹ which is accessible via a web browser. The tool can be directly used as an engineering tool, for example, to test the operations of line controllers. On the other hand, sales and marketing personnel get new opportunities to present solutions to their potential customers as the

majority of people are familiar with handling a web browser. In comparison to commercially available products, such as, for example, totally integrated automation¹⁰ by Siemens, presented approach is open for different service-enabled controllers. In fact, the approach is meant for service-oriented systems. Although major PLC vendors did not pick up the approach yet, the general IT sector moves and evolves around the web standards. The authors predict that similarly to the Ethernet, which was adapted to various industrial Ethernet protocols in industry, the adaptation of software engineering paradigms will follow. The authors also understand that major vendors may still want to bind their customers to their solutions by providing virtuous kinds of “integrated” solutions. As good as such solutions can be they will cause a vendor lockup problem. OKD-MES approach can be followed by those, who may also want to avoid being locked up to the particular vendor.

Second, system “extendibility” is increased. New functions can be relatively easily added using web service standards, thus extending the overall system functionality. In general, since web standards and IP-based networks are used on the factory floor, the CPS can be integrated into a global network if necessary. In the example of dispatch function outlined in the previous sections, the set of services required to address particular product needs may be extended or changed over time, but without requiring reprogramming of MES functions. This is possible due to late binding in the loosely coupled OKD-MES based on the domain knowledge. Consequently to introduce new manufacturing system entities in most cases it should suffice to use the same concept vocabulary or to map the domain vocabularies of the application. In the case of more complex modifications, a change of OWL (knowledge) models may be required to facilitate the representation of addition-

⁹ Advanced REST client, <https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbjeloo>

¹⁰ <http://www.industry.siemens.com/topics/global/en/tia/pages/default.aspx>

al concepts. Yet even in this case, an engineer can already use online tools¹¹ (requiring just a web browser). The complete eScop MSO ontology can be accessed using the aforementioned online tools.

Third, systems can be characterized by “adaptability” with respect to new conditions, which may be due to changes in production processes or in production equipment. The change is noted in the knowledge model and the model can be directly updated via a SPARUL query sent to RPL by a device integrated into the physical environment. The word “query” should not be confused here. The “update query” is sent to make a change in the knowledge model rather than to retrieve some information. Thus, an engineer should develop and test valid queries when deploying devices. The same online tools mentioned for “extendibility” can be used for developing and testing the queries.

Finally, system “availability” is improved due to better awareness of system resources and their status. Introduction or removal of the hardware which follows certain constraints can be reflected in the system model within seconds. Given the event-driven and service-oriented nature of the proposed system, the status of system resources can also be updated. For more elaborated resource representation, additional MES services (such as resource status and allocation) may be developed. The awareness of configuration and status of the components contributes to improved decision making and failure handling.

In addition to qualitative improvements, there are certain risks or challenges associated with the use of CPS for OKD-MES. The three most relevant of these can be described as follows.

“Security” is a general demand for any networked system. Since it is now easy to access CPS on the factory floor, as even a basic tool such as a web browser can be used not only to obtain information on the status of a device or the process it controls, but also to invoke an operation on the device, therefore security measures should be carefully implemented. Again, general IT policies and standards developed for global networks can be applied here. As soon as public networks are used for data transmission, the data can be encrypted decreasing the chances of security breaches. The factory floor should be isolated behind network firewalls, with analytical tools and procedures in place for detecting possible attacks.

“Observability” can be seen as a more important challenge specifically for manufacturing. Observability is an ability to know the system state, as and when it is needed. As decision making is pushed to the lower levels to increase system responsiveness and adaptability, it becomes more challenging to observe the overall system state. In manufacturing, many products may be handled in parallel, competing for the same resources of the manufacturing system, and the system must provide mechanisms for handling conflict resolution. Orchestrators that do service composition for making a product should be aware of other production workflows and their statuses.

Another issue for implementing CPS is fast changing standards. For example, there are different versions of simple object access protocol (SOAP)¹² that can be used for invoking web services. There are different versions of device profile for web services (DPWS)¹³ that can be used to build service-enabled CPS. There are a number of versions for business process execution language (BPEL)¹⁴ that can be used for service composition. There are different versions of HTML, and so on. A CPS following web standards needs to use several standards and protocols at the same time. Aiming a functional integration of such standards and protocols, the implementation of OKD-MES is developed within open standards, which are mature enough to perform required features. Although web standards evolve, they are often compatible extensions of their predecessors, providing more complex, richer or, simply, new features (e.g., RDF and OWL or SPARQL and SPARUL). On the other hand, APIs developed for one version of the standard may not have forward compatibility, thus solutions developed earlier may not be directly integrated with newer applications. In order to mitigate this risk, a web browser can be used as a benchmarking tool. That is, if the technology and/or protocol is supported by a web browser, then it is more likely to be around for a longer time. Due to the application scale of Internet technologies, those which are supported by the majority of the web browsers would tend to have the most mature and highly developed APIs. Then, applications developed within web standards, which are directly supported in different web browsers, will be compatible with future technologies.

The presence of MES solutions in SMEs is limited due to the complexity, inflexibility, and high implementation and customization costs of an existing solution. OKD-MES is being developed to address this niche. In order to address the MES system migration for enterprises already having an MES solution, the migration approach is required. Such an approach might be based on the advantage of the open and knowledge-driven nature of the system. Openness makes it possible to develop the solution to bridge the gap once and share it with the community, while knowledge-driven nature supports the reusability of a developed block providing a shared but flexible model of required knowledge.

The challenge of “performance” is often an obstacle to the widespread application of knowledge-driven solutions. In light of the dependence of the performance of the interactions with the knowledge base on the complexity of queries and the size of the knowledge base, the surest approach to verify the required performance may be achieved only through extensive testing and benchmarking. The prototype implementation of OKDMES solution was capable to process tens of mixed queries per second in persistent mode and hundreds in nonpersistent mode. Regarding the application of the prototype in several medium-sized pilot cases (e.g., FASTory line) it was estimated that it is realistic to maintain the representation of system configuration in the knowledge base and use it for

¹¹ eScop online tools, <http://www.escop-project.eu/tools>

¹² <http://www.w3.org/TR/soap/>

¹³ <http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01>

¹⁴ https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

decision making in MES functions. Additional research and optimization may enable better performance and as a result should be capable of managing larger systems.

Having discussed four qualities and challenges of CPS for manufacturing, it is also important to stress that the manufacturing of devices that can be used for controlling industrial equipment is currently in the hands of only a few rather small companies. As APIs, standards, and tools reach maturity, the use of CPSs in conjunction with knowledge-driven approaches in domain of manufacturing will become common. Meanwhile, the adaptation of the approach to the legacy systems can be carried out in two basic ways.

The first approach is servitization of high-level supervisory and monitoring applications providing service interfaces for other high-level enterprise applications. The supervisors or monitors can continue to use traditional fieldbuses to communicate with the equipment wrapping and translating information between the equipment and enterprise functions.

The second approach is servitization of controller devices by installation of gateway devices at the factory floor. The role of the device is the same as in the first case—to translate and wrap the functionality of a controller as a set of web services, but it can be a dedicated solution for particular industrial controller. The cost of the gateway hardware can be relatively low, in terms of tens of euros¹⁵ and it is already capable to run applications using latest service APIs

VI. CONCLUSION AND FUTURE WORK

The use of CPSs for OKD-MES was illustrated using FAS-Tory production line. The main challenges as well as the improvements of the approach proposed were discussed. Web standards and mature Internet-based technologies made it possible to develop and integrate an application using less time due to affordable and widely used basic tools such as a web browser. Future work on developing integrated methodology that would combine methods, tools, and techniques used in heterogeneous disciplines may be required to improve the adoption of the approach in the manufacturing domain. The development of the consumer market for handheld devices making, for instance, a smartphone a common and widely used tool, the functions of which can be extended with the installation of new applications using the same web standards to interact with the manufacturing systems may contribute to a paradigm shift toward the use of open-knowledge-driven manufacturing execution systems.

Because of the advantages mentioned in Section V, the OKD-MES concept may become more desirable in the manufacturing domain. CPS is one of key systems with significant potential in implementing the OKD-MES concept. Since traditional manufacturing systems can in principle deliver the basic functionality expected from such systems, the OKD-MES concept faces challenges in the adoption of the approach by industrial community. A solution to this challenge would require new system architecture which homogenizes the manufacturing ecosystem applying the proposed approach.

ACKNOWLEDGMENT

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n° 332946 and from the Finnish Funding Agency for Technology and Innovation (TEKES), correspondent to the project short title eScop¹⁶, Embedded systems for service-based control of open manufacturing and process automation.

REFERENCES

- [1] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang, "Toward a Science of Cyber-Physical System Integration," *Proc. IEEE*, vol. 100, no. 1, pp. 29–44, Jan. 2012.
- [2] M. McClellan, *Applying Manufacturing Execution Systems*. CRC Press, 1997.
- [3] "MESA International - Home." [Online]. Available: <http://www.mesa.org/en/index.asp>. [Accessed: 03-Oct-2014].
- [4] A. Bratukhin and T. Sauter, "Functional Analysis of Manufacturing Execution System Distribution," *IEEE Trans. Ind. Inform.*, vol. 7, no. 4, pp. 740–749, Nov. 2011.
- [5] V. Modrak, S. Bednar, and D. Marton, "Generating product variations in terms of mass customization," in *2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMII)*, 2015, pp. 187–192.
- [6] "Manufacturing Execution Systems - Accenture," 2010.
- [7] J. Kletti, *Manufacturing execution systems-MES*. Springer, 2007.
- [8] F. K. Johnson, "Future of manufacturing execution systems: the brave new modular world of manufacturing intelligence," *Rev. Manag.*, vol. 1, no. 1, pp. 4–14, 2011.
- [9] R. Rajkumar, "A Cyber-Physical Future," *Proc. IEEE*, vol. 100, no. Special Centennial Issue, pp. 1309–1312, May 2012.
- [10] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 731–736.
- [11] S. Galli, A. Scaglione, and Z. Wang, "For the Grid and Through the Grid: The Role of Power Line Communications in the Smart Grid," *Proc. IEEE*, vol. 99, no. 6, pp. 998–1027, Jun. 2011.
- [12] E. Guizzo, "Autonomous Vehicle Driving from Italy to China," 21-Sep-2010. [Online]. Available: <http://spectrum.ieee.org/automaton/robotics/robotics-software/autonomous-vehicle-driving-from-italy-to-china>. [Accessed: 04-Jun-2015].
- [13] J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou, "Distributed Real-Time Software for Cyber-Physical Systems," *Proc. IEEE*, vol. 100, no. 1, pp. 45–59, Jan. 2012.
- [14] J.-S. Choi, T. McCarthy, M. Yadav, M. Kim, C. Talcott, and E. Gressier-Soudan, "Application patterns for cyber-physical systems," in *2013 IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, 2013, pp. 52–59.
- [15] A. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, and J. L. Lastra, *Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach*. Springer Publishing Company, Incorporated, 2014.
- [16] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*. Pearson Education, 2005.
- [17] "SOA and Web Services." [Online]. Available: <http://www.oracle.com/technetwork/articles/javase/soa-142870.html>. [Accessed: 04-Jun-2015].
- [18] A. Lobov, J. Puttonen, V. V. Herrera, R. Andiappan, and J. L. M. Lastra, "Service oriented architecture in developing of loosely-coupled manufacturing systems," in *6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008*, 2008, pp. 791–796.
- [19] I. M. Delamer and J. L. M. Lastra, "Self-orchestration and choreography: towards architecture-agnostic manufacturing systems," in *20th International Conference on Advanced Information Networking and Applications, 2006. AINA 2006*, 2006, vol. 2, p. 5 pp.–.
- [20] A. N. Lee and J. L. M. Lastra, "Data aggregation at field device level for industrial ambient monitoring using Web Services," in *2011 9th*

¹⁵ <https://www.raspberrypi.org/>

¹⁶ <http://www.escop-project.eu/>

- IEEE International Conference on Industrial Informatics (INDIN)*, 2011, pp. 491–496.
- [21] L. E. G. Moctezuma, J. Jokinen, C. Postelnicu, and J. L. M. Lastra, “Retrofitting a factory automation system to address market needs and societal changes,” in *2012 10th IEEE International Conference on Industrial Informatics (INDIN)*, 2012, pp. 413–418.
- [22] S. Iarovy, J. Garcia, and J. L. M. Lastra, “An approach for OSGi and DPWS interoperability: Bridging enterprise application with shop-floor,” in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 200–205.
- [23] R. J. Brachman, *Knowledge representation and reasoning*. Amsterdam ; Boston: Morgan Kaufmann, 2004.
- [24] L. Fumagalli, S. Pala, M. Garetti, and E. Negri, “Ontology-Based Modeling of Manufacturing and Logistics Systems for a New MES Architecture,” in *Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World*, B. Grabot, B. Vallespir, S. Gomes, A. Bouras, and D. Kiritsis, Eds. Springer Berlin Heidelberg, 2014, pp. 192–200.
- [25] B. Ramis, L. Gonzalez, S. Iarovy, A. Lobov, J. L. Martinez Lastra, V. Vyatkin, and W. Dai, “Knowledge-based web service integration for industrial automation,” in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 733–739.
- [26] M. Garetti, L. Fumagalli, A. Lobov, and J. L. Martinez Lastra, “Open Automation of Manufacturing Systems through Integration of Ontology and Web Services,” presented at the 7th IFAC Conference on Manufacturing Modelling, Management, and Control, 2013, 2013, vol. 7, pp. 198–203.
- [27] D. Kalibatiene and O. Vasilecas, “Survey on Ontology Languages,” in *Perspectives in Business Informatics Research*, J. Grabis and M. Kirikova, Eds. Springer Berlin Heidelberg, 2011, pp. 124–141.
- [28] “RDF 1.1 Concepts and Abstract Syntax.” [Online]. Available: <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>. [Accessed: 03-Oct-2014].
- [29] “Semantic Web Architecture - Introduction to ontologies and semantic web - tutorial.” [Online]. Available: <http://obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html>. [Accessed: 27-Oct-2015].
- [30] “OWL 2 Web Ontology Language Document Overview (Second Edition).” [Online]. Available: <http://www.w3.org/TR/owl2-overview/>. [Accessed: 05-Mar-2014].
- [31] I. Kollia, B. Glimm, and I. Horrocks, “SPARQL query answering over OWL ontologies,” in *The Semantic Web: Research and Applications*, Springer, 2011, pp. 382–396.
- [32] “SPARQL 1.1 Update.” [Online]. Available: <http://www.w3.org/TR/2013/REC-sparql11-update-20130321/#sec-intro>. [Accessed: 05-Mar-2014].
- [33] J. Puttonen, A. Lobov, and J. L. Martinez Lastra, “Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services,” *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2349–2359, Nov. 2013.
- [34] J. Puttonen, A. Lobov, and J. L. M. Lastra, “Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems,” in *2013 IEEE 20th International Conference on Web Services (ICWS)*, 2013, pp. 419–426.
- [35] B. R. Ferrer, S. Iarovy, L. Gonzalez, A. Lobov, and J. L. M. Lastra, “Management of distributed knowledge encapsulated in embedded devices,” *Int. J. Prod. Res.*, vol. 0, no. 0, pp. 1–18, Dec. 2015.
- [36] T. Preobrazhenskaya, M. Bakaev, and T. Avdeenko, “The role of data structures design in modern web applications development,” in *2013 8th International Forum on Strategic Technology (IFOST)*, 2013, vol. 2, pp. 276–279.

ABOUT THE AUTHORS



Sergii Iarovy received the M.Sc. degree (with distinction) in electromechanics from the National Technical University “KhPI,” Kharkiv, Ukraine, in 2011 and the M.Sc. degree in factory automation from Tampere University of Technology, Tampere, Finland, in 2014.

Since 2012, he has worked at the FAST Laboratory, Tampere University of Technology, as a Project Researcher. His research interests are in the application of semantic web services, cyber– physical systems, and enterprise integration for factory automation



Wael M. Mohammed received the B.Sc. degree in mechatronics engineering from Jordan University, Amman, Jordan, in 2010.

From 2010 to 2011, he was a Research Assistant at Tampere University of Technology, Tampere, Finland, and from 2011 to 2013, he worked as Head of the Technical Department at Etihad Alafandi LLC in Eastern Province, Kingdom of Saudi Arabia. Since 2015, he has been a Research Assistant with the FAST Laboratory at Tampere University of Technology.



Andrei Lobov received the B.S. degree in computer and system engineering from Tallinn University of Technology, Tallinn, Estonia, in 2001, the M.S. degree in automation engineering from Tampere University of Technology, Tampere, Finland, in 2004, and the Dr. Tech. degree on the subject of formal methods in factory automation from Tampere University of Technology in December 2008.

He is a University Lecturer at Tampere University of Technology. His research interests include development of architectures, methodologies, and technologies for industrial manufacturing systems. He is a technical coordinator in the eScop project.



Borja Ramis Ferrer received the B.Sc. degree in electrical engineering from the Universidad de las Islas Baleares, Islas Baleares, Spain, in 2011 and the M.Sc. degree (with distinction) in factory automation from Tampere University of Technology, Tampere, Finland, in 2013, where he is currently working toward the Ph.D. degree.

He is a Fellow of the President’s Doctoral School at Tampere University of Technology. His research interests include the deployment of knowledge-based and cyber–physical systems in factory automation.



Jose L. Martinez Lastra received the Ingeniero Tecnico Industrial degree in electrical engineering from the Universidad de Cantabria, Santander, Spain, and the M.Sc. degree (with Distinction) and the Dr.Sc. in Technology degree (with Commendation) in automation engineering from the Tampere University of Technology, Tampere, Finland.

He joined Tampere University of Technology, Tampere, Finland, in 1997, and became University Full Professor in 2006. His research interest is in applying information and communication technologies to the fields of factory automation and industrial systems. He leads the FAST Laboratory, Tampere University of Technology, with the ultimate goal of seamlessly integrating the knowledge of humans and ma-

chines. He has co/authored over 250 scientific papers and holds a number of patents in the field of industrial informatics and automation.

Dr. Martinez Lastra serves as an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and he is a Technical Editor of the IEEE/ASME TRANSACTIONS ON MECHATRONICS.



TOWARDS THE ENCAPSULATION AND DECENTRALIZATION OF OKD-MES SERVICES WITHIN EMBEDDED DEVICES

by

Borja Ramis Ferrer, Jose Luis Martinez Lastra, May 2017

International Journal of Production Research.

Reproduced with permission: 'Towards the encapsulation and decentralisation of OKD-MES services within embedded devices' by Borja Ramis Ferrer & Jose Luis Martinez Lastra International Journal of Production Research pp. 1-13 (2017). This is the authors accepted manuscript of an article published as the version of record in International Journal of Production Research on 13th May 2017. www.tandfonline.com/http://dx.doi.org/10.1080/00207543.2017.1328141



Towards the encapsulation and decentralisation of OKD-MES services within embedded devices

Borja Ramis Ferrer & Jose Luis Martinez Lastra

To cite this article: Borja Ramis Ferrer & Jose Luis Martinez Lastra (2017): Towards the encapsulation and decentralisation of OKD-MES services within embedded devices, International Journal of Production Research, DOI: [10.1080/00207543.2017.1328141](https://doi.org/10.1080/00207543.2017.1328141)

To link to this article: <http://dx.doi.org/10.1080/00207543.2017.1328141>



Published online: 13 May 2017.



Submit your article to this journal [↗](#)



Article views: 43





View related articles [↗](#)



View Crossmark data [↗](#)

Towards the encapsulation and decentralisation of OKD-MES services within embedded devices

Borja Ramis Ferrer*  and Jose Luis Martinez Lastra 

Factory Automation Systems and Technologies Laboratory (FAST-Lab.), Tampere University of Technology, Tampere, Finland

(Received 9 June 2016; accepted 28 April 2017)

Traditionally, the resources of embedded devices which are employed for process control at shop floors were resource constrained. However, advances in embedded system technologies permit the enhancement of the processing and storage capabilities of embedded devices. Therefore, semantic descriptions of manufacturing systems can now be hosted and computed at the device level. This fact permits the creation of a decentralised solution for controlling processes at the lowest level of the manufacturing enterprises and the reduction in the time and effort requirements for the configuration and information exchange. The eScop project presented the Open Knowledge-Driven Manufacturing Execution System (OKD-MES) solution, which enables monitoring and controlling production systems openly and allows runtime re-configurability of interconnected industrial equipment and services. This research work presents how part of the OKD-MES functionality can be handled at lower level. More precisely, the OKD-MES representation and management of knowledge can be decentralised and handled at the shop floor level, where the industrial machines are connected to devices that are capable of controlling the execution of processes. The main objective of this paper is to describe a decentralised vision for the OKD-MES framework, which is a centric solution in terms of knowledge management. Moreover, the article also discusses some of the advantages to be gained from decentralising the management of knowledge model semantic descriptions.

Keywords: cyber-physical systems; service oriented architecture; OKD-MES; decentralised knowledge bases; industrial automation

1. Introduction

Nowadays, the industry is facing an incessant demand of new, efficient and cheaper platforms, frameworks and tools that permit different stakeholders to access, monitor and manage information about production along the supply chain. Furthermore, interested parties request those systems to be remotely accessible and configurable. Therefore, the industry is now employing a large amount of new information and communication technologies-based solutions that permit the manipulation of information through networks and, more precisely, the Internet. The advent of the Internet of Things (IoT) makes possible the connection and information exchange for the ‘things’, such as sensors, wearables, industrial equipment and computers of different domains across the Internet.

Recently, the EU eScop project¹ has presented and validated the eScop solution, which is based on the Open Knowledge-Driven Manufacturing Execution System (OKD-MES) approach (Iarovyi et al. 2016). Conceptually, the eScop framework provides a reference architecture formed by several services that permit the supervision and control of production systems in an open manner, allowing runtime re-configurability and initialization of new equipment and services. The OKD-MES has been implemented within the combination of a Service-Oriented Architecture (SOA), Knowledge Representation (KR) and the use of contemporary industrial controllers. Therefore, the OKD-MES can be considered as a cyber-physical system implementation because it enables the synergy between cyber and physical domains.

One of the particular features of the eScop solution is that the management of knowledge model descriptions is performed within a central service: The Representation Layer Service (RPL-S). Basically, the RPL-S provides the unique access point to the data stored in the central Knowledge Base (KB), which describes the system being controlled. Therefore, in terms of knowledge management, trade and consumption, the OKD-MES is a centralised solution (Fumagalli et al. 2014).

This research work is motivated by the enhancement of computational resources of resource-constrained embedded devices that are only used for service description and reduced control algorithms. Due to aforementioned highlight, these

*Corresponding author. Email: borja.ramisferrer@tut.fi

devices are now capable of (i) encapsulating more data and (ii) managing new functionalities that demand computation at device level. In such scope, the main objective of this article is to propose placing at lower level certain parts of the OKD-MES functionality to be implemented and managed by cooperative embedded devices. Hence, this research work describes an alternative for the eScop centric solution. The presented approach depicts how to achieve the decentralisation of the entire system KB and discusses some of the advantages that can offer if compared to the OKD-MES implementation.

The rest of the paper is structured as follows: Section 2 provides a short review of the literature and industrial practices related to the scope of this research work. Afterwards, Section 3 describes briefly the OKD-MES architecture. Then, Section 4 presents the main principles of the decentralised proposal. Section 5 discusses the proposed alternative and compares it with the actual OKD-MES implementation. Finally, Section 6 concludes the article.

2. Review of literature and industrial practices

2.1 Manufacturing Execution Systems

A Manufacturing Execution System (MES) is a control system implemented in the industrial automation domain for monitoring and managing runtime processes on a factory shop floor. A formal definition and foundations of MES and main features were early addressed in McClellan (1997). Since then, different research works describing strategies and technologies for implementing and modelling new MES have emerged as e.g. the ones described in Gao, Li, and Chen (2015), Iarovyi et al. (2016) and Witsch and Vogel-Heuser (2012). Nevertheless, the main features and requirements for these systems remain. Basically, MES must be an online system connected to both business and factory shop floor levels' systems. In addition, the system has to be integrated with the levels so that cross-level communication is feasible.

The need of employing MES solutions in manufacturing systems can be understood within a set of functions that the Manufacturing Enterprise Solutions Association (MESA) defines in MESA International (2011). In fact, aforementioned white paper describes how the MESA model has evolved until current version from its first definition in 1992.

Initially, the first MESA model identified 11 main functions that should be provided by a MES solution to facilitate the manufacturing production. Other organisations, such as the International Society of Automation (ISA) and the Verein Deutsche Ingenieure provide similar set of functions, which can be mapped as shown in Iarovyi et al. (2016). The functionality of MES is critical to enable the production of a manufacturing system within the integration of different levels depending on the model. For example, the current MESA model levels show in 'MESA International – MESA Model' (2016) are: *Strategic initiatives*, *business operations*, *manufacturing/production operations* and the *manufacturing/production* bottom level. On the other hand, according to the ISA-95 model levels show in ISO (2007) are: *business planning and logistics*, *manufacturing operations management*, *batch control*, *continuous control*, *discrete control* and *production process* bottom level. In fact, the ISA-95 automation pyramid presents the MES an intermediate level between manufacturing operations management level systems, such as the Enterprise Resource Planning (ERP) and control level systems as, for example, the Supervisory Control and Data Acquisition (SCADA). Hence, the MES fills the gap between the management and the shop floor control. In this case, MES is integrated with the ERP for scheduling and supporting the dispatch of orders to the control system, which increases the productivity by reducing its cycle time.

2.2 Related work: a knowledge-driven solution for the industrial automation domain

The eScop project has provided the OKD-MES solution. Through this platform, researchers plan the deployment of a new MES that improves the competitiveness of the European industry (eScop 2016; Tampere University of Technology 2016). The developers claim that the eScop framework enhances the industrial efficiency within e.g. the reduction of time and effort for configuration, supervision and control of assembly line equipment and processes. The OKD-MES solution has been implemented through the synergy of three important areas: SOA, KR and embedded devices. Each of these areas is briefly introduced in following subsections to describe its role and functionality in this particular knowledge-driven solution to be deployed in the industrial automation domain.

2.2.1 SOA related work for the OKD-MES

SOA (Qusay 2005) started to be used in the industrial automation domain for encapsulating functionality of system components and exposing it as services (Lobov et al. 2008). The software encapsulation is performed using the Web Service (WS) technology, which includes a set of standards to implement SOA.

Some of the important applications of WS for the industrial automation domain are: cross-layer communication, integration of components that manages heterogeneous data and adaptation of existing applications ('IBM developerWorks' 2007). Moreover, SOA is used for monitoring and control of manufacturing system processes (Moctezuma et al. 2012). Nevertheless, the degree of re-configurability and interoperability required for large-scale dynamic production systems is not covered entirely by the SOA paradigm. Therefore, the OKD-MES proposes the combination of SOA with KR so that knowledge of the system can be manipulated by any service of the platform.

2.2.2 KR related work for the OKD-MES

The abstraction and storage of knowledge about the system status is required to allow the adaptation and re-configurability of systems. This is traduced in the implementation of KR in current manufacturing systems. Through KR, engineers are able to describe physical and logical systems, allowing the collection of system data, which is later transformed into information that can be used by other subsystems. Examples of industrial automation research works that utilise a KB for storing data used for controlling processes are presented in Lobov et al. (2009), Puttonen, Lobov, and Lastra (2013) and Ramis et al. (2014).

Ontologies are used as a formal representation of manufacturing systems. Although there are many options for designing ontologies (Lastra, Delamer, and Ubis 2010) and a wide variety of languages (Maniraj and Sivakumar 2010; Negri et al. 2016), the most prominent ontology language is the Web Ontology Language (OWL) ('OWL Web Ontology Language Reference' 2004).

OWL has already been presented as a mature language for KR in factory automation (Delamer and Lastra 2006) and it has a higher degree of representation than other ontology languages. OWL is a Resource Description Framework (RDF) ('RDF – Semantic Web Standards' 2014), which is an Extensible Markup Language (XML)-based language ('Extensible Markup Language (XML)' 2015). Consequently, the use of RDF-based languages for retrieving information of OWL KBs is appropriated. It should be noted that cited research works use SPARQL Protocol and RDF Query Language (SPARQL) ('SPARQL Query Language for RDF' 2008) for querying ontological OWL models. Moreover, due to the adaptation and re-configurability of production lines, implementations must use another language, which permits the update of KBs. The SPARQL Update ('SPARQL 1.1 Update' 2013) permits the modification of OWL models. An example on how SPAQRL and SPARQL Update can be utilised on industrial systems can be found in Puttonen, Lobov, and Martinez Lastra (2013). Fundamentally, the implementation of KD approaches permits the runtime process control of systems in the industrial automation field. Then, the enormous advance that the use of ontologies offers in this domain is not only the knowledge representation, but also letting them play an important role in the system control.

Furthermore, reasoning engines permit the inference of implicit facts that are concluded from explicit knowledge. This is a powerful feature provided by ontologies because machines can extend on runtime the system's KB, which contains the data from the system being controlled. Semantic reasoning is commonly achieved within semantic reasoners as e.g. Pellet (Sirin et al. 2007) and, if desired, Semantic Web Rule Language for defining rules in RDF-based ontologies (Ramis Ferrer, Ahmad, et al. 2016).

2.2.3. Embedded devices related work for the OKD-MES

Finally, as presented in Garetti et al. (2013, Iarovyi, Garcia, and Lastra (2013) and Moctezuma et al. (2012), Remote Terminal Units (RTUs) are a type of industrial controllers that can be used in production lines for process control implementing SOA, as a gateway between the lowest and higher levels of manufacturing systems. More precisely, these RTUs may employ the Device Profile for Web Services (DPWS) technology as an alternative for having the service functionality. Then, such DPWS devices are capable of real-time control, Web-based monitoring and integration to SCADA systems, among other applications.²

Current RTUs not only have increased their communication and networking capabilities, but also the processing power of CPUs. Therefore, due to the evolution of embedded devices the expanded resource power can be used for other functionalities, rather than just work as gateways for vertical communication through different levels of automation systems. Among other functionalities, the possibility of hosting KR models is proposed in Ramis Ferrer et al. (2015). It should be noted that there are nowadays so many IoT commercial-embedded devices that can be employed to manage functionality of factories as e.g. the addition of certain shields for increasing the computational power and connectivity to devices as Raspberry Pi,³ Arduino⁴ or Onion Omega.⁵

3. The OKD-MES architecture

The objective of the OKD-MES is to exploit the synergy of new generation of industrial controllers with a KD SOA for monitoring and controlling an entire automated manufacturing environment. The research work (Garetti et al. 2013) presented an innovative solution for the control processes in manufacturing systems, based on the integration of web-services technologies and an ontological model, allowing an easy configuration, update and scalability of the control system. In fact, this is the first presentation of the OKD-MES.

Furthermore, Garetti et al. (2013) presents a comparison between the architecture of conventional MES and the OKD-MES. The main difference between the two architectures is the appearance of an ontology in the control layer. In addition, the control layer is divided in two layers for differentiate between the functionalities of representation and orchestration of WSs.

On the other hand, Fumagalli et al. (2014) presents the main concept of the eScop project, which is the combination of embedded systems and an ontology-driven service-based architecture for realising the OKD-MES, defined also as a fully open-automated manufacturing environment. The architecture of the OKD-MES is presented in Figure 1, which is an abstract representation of the initial eScop project kernel model view described in Fumagalli et al. (2014). The main differences between initial and final implementation are described throughout this section.

Initially, the architecture of the OKD-MES consisted in five layers: Physical (PHL), Representation (RPL), Orchestration (ORL), Interface (INT) and Visualisation (VIS) layers. It should be noted that VIS is not shown in Figure 1 because it was not included in the kernel module representation. Moreover, INT functionality was finally not implemented as a separated layer but as a module in each layer. In other words, INT is implemented within each layer satisfying the corresponding requirements of authentication and authorization where appropriate.

The lowest layer is the PHL, which can be mapped to the hardware components located in the shop floor: machines, sensors, actuators, control units and the RTUs, which are the devices appearing in Figure 1. The objective of PHL is the real-time control of devices. These are the devices that expose the functionalities of the production line as services and provide access to service description. Therefore, RTUs enable cross-layer communication because they serve as an interface between the PHL and other layers of the architecture.

Then, the RPL contains the ontological model of the production system i.e. the KB. The Production Systems Model, which is based on the Politecnico di Milano Production Systems Ontology (P-PSO) (Garetti 2012), is an example of a model that can be included and exploited in the RPL. According to the P-PSO approach, the system KB will contain information about the different domains of the system e.g. physical, technological and control. Moreover, the formal representation of the production line status and service functionality supports the operation of SOA based orchestration tools, located at the ORL.

The ORL hosts the Supervisory Control System of the shop floor equipment. This system is composed by a set of orchestration tools and a production scheduler. Through these entities, the ORL is capable of composing and control the process that must be executed in the shop floor. The composition starts with incoming orders from the factory Order Entry System, which is indeed exposed to the user by the corresponding MES function through the VIS layer. Therefore, the ORL consists both in the service composition and orchestration for controlling not only the shop floor equipment, but also for coordinating actions of the entire system.

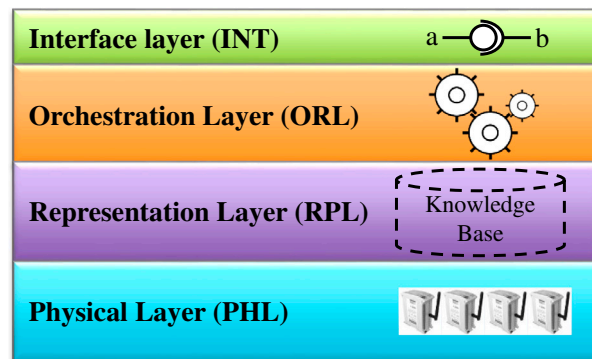


Figure 1. Abstract representation of the initial eScop project kernel model view.

In addition, the OKD-MES includes the VIS, which is as layer that is in charge of offering graphical visualisation of the actual status of components and user interfaces. This layer is used for runtime monitoring of the production system that is possible within an interface with the RPL, which permits the monitoring of the line status at any time.

Moreover, the research work presented in Ramis et al. (2014) is a first prototype implementation that fulfils the main requirements of the presented architecture in Figure 1 within two main differences: (i) the VIS and the INT layers are developed jointly in the User Interface service and (ii) the ORL is situated in the shop floor. Such research work tested the main concepts of the OKD-MES by exposing layer functionalities as services and presents the solution as a knowledge-based WS integration.

The characteristic of any of the system versions described in this section is that RPL becomes a central component because any of the services or layers require the interaction with the model of the system for retrieving and updating the KB. To highlight that the management and consumption of system knowledge included in the KB is done centrally, Figure 2 shows another perspective of the eScop kernel. The diagram extends the names of each module with '-S' for indicating that the entire layer and, hence, its functionality is implemented as a service.

4. The alternative

The processing power of CPUs at resource-constrained embedded devices, which serve as industrial controllers, has grown dramatically. This fact permits the growth of the communication and networking capabilities of these devices. Due to the use of larger memories, RTUs have also increased the storage for different types of resources as i.e. libraries, data and user programmes. Then, recent embedded devices have sufficient resources to perform deterministic control and networking functions.

On the other hand, Figure 2 demonstrates that the actual OKD-MES architecture is centralised by means of interaction with the system KR. More precisely, the KB hosted in the RPL and managed by the RPL-S becomes a central component for the execution of system processes. This happens because any layer of the system must interact with the RPL-S.

Then, this research work proposes the implementation of the RPL-S functionality inside the embedded devices, which are used in shop floors mostly as gateways between PHL and other layers of the OKD-MES. The resulting embedded system is a device that not only keeps the objective of interfacing shop floor equipment with the rest of the MES, but also includes a similar instance of the RPL-S that is used for interaction with a KB. A conceptual view of the proposed service encapsulation and device structure is depicted in Figure 3. In addition, the following considerations are applied to the device architecture presented in Figure 3:

- (1) '*RP-S*' is a service, which is encapsulated by the embedded device, which implements the functionality of the OKD-MES RPL-S.
- (2) The '*x*' notation is used for indicating a specific device that hosts its own KB. For example, a *Device₁* would host the *KB₁*.

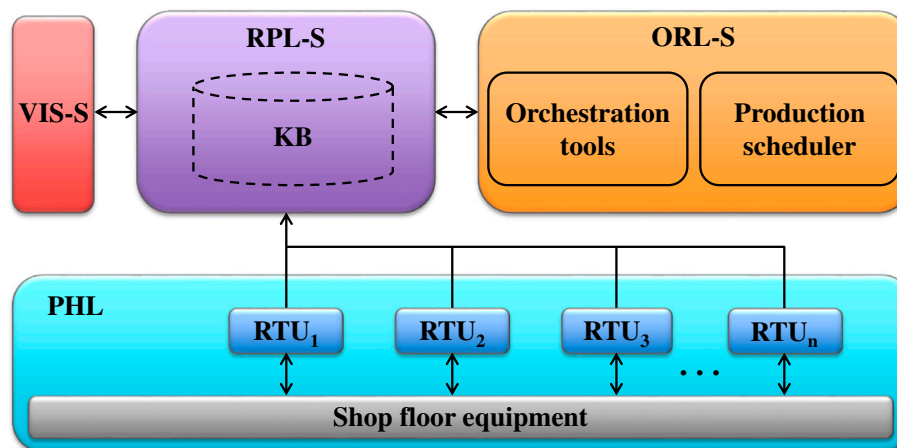


Figure 2. Interaction of OKD-MES layers, implemented as services, with the RPL-S.

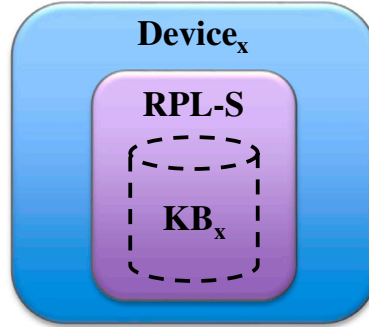


Figure 3. Encapsulation of RPL-S functionalities in embedded devices.

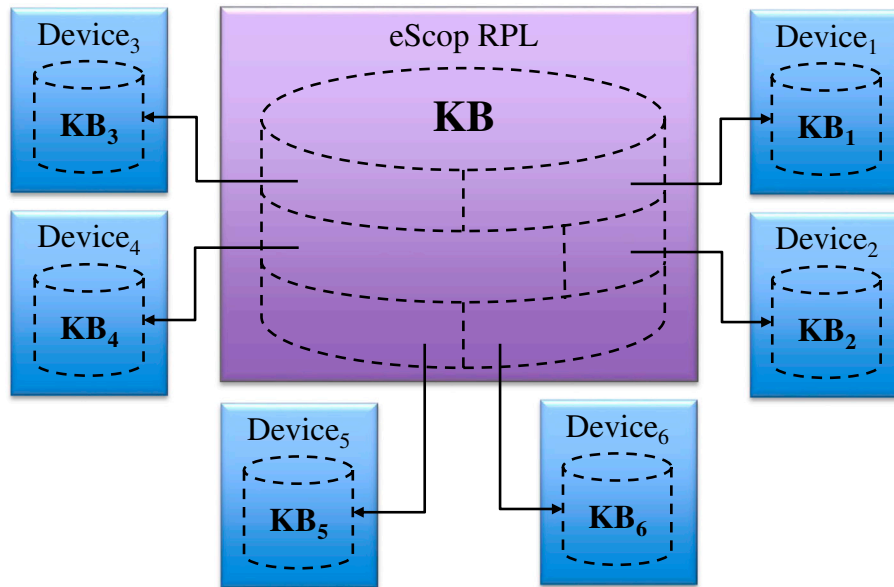


Figure 4. Representing the decentralisation of the system KB into separate KBs.

- (3) The KB of each device will be a portion of the entire system's KB, which must be integrated when an incoming request needs to be answered. This is represented in Figure 4.

The main objective of this research work is to propose a decentralised architecture, as an alternative of the KR utilisation in OKD-MES. At least, each device hosts the description of (i) itself, (ii) controlled equipment capabilities and (iii) distributed network, in which all the devices are interconnected. This is translated to an ontological model i.e. the systems' KB. A first version of such kind of model has been presented in Ramis Ferrer and Martinez Lastra (2017). Such ontological model is accessible by other devices and *Layer-Services*⁶ within SPARQL and SPARQL Update endpoints. This is to be implemented as other embedded device interfaces.

This approach is looking towards the cooperation of intelligent embedded devices. Conceptually, the presented approach designs a distributed network, in which devices will enhance their autonomy for performing different tasks, such as knowledge management and decision-making. Then, the implementation on new industrial controllers, which intends to implement this approach, must include the capability of:

- *WS implementation*: embedded devices must be capable of implementing services for (i) controlling processes in the industrial automation domain and (ii) handling the functionality of the RP-S.
- *Ontological model (RDF-based) hosting*: the device must have sufficient resources for storing its own KB. Such model describes the required knowledge that the device needs for operating and performing specific tasks.

- *User programme implementation*: designers must be capable to implement algorithms. As an example, Ramis Ferrer et al. (2015) presents the knowledge management and device behaviour algorithms to be executed by devices.
- *Interface for connection with shop floor equipment*: an interface is required (i) for activating machine inputs, which are triggered from invoked WS operations and (ii) for reading system outputs in order to update the actual system status, which is represented in the ontological model.

Once the device architecture is described, the disposition of them and interaction with remaining Layer-Services are depicted in Figure 5. This diagram represents the interconnection between devices and the shop floor equipment. In comparison with the connections shown in Figure 2, the represented knowledge is now decentralised due to the encapsulation of a RP-S instance into each device. The union of all device KBs result in the entire manufacturing system model, as previously represented in Figure 4. Each device hosts the description of its own related functionality for avoiding redundant data in different embedded devices. Fundamentally, the duplication of data must be avoided because it occupies memory that can be used for other purposes, such as for the OWL expansion due to update queries.

The implementation of this approach imposes the need of managing the knowledge at device level. This management allows the response to incoming queries of other Layer-Service. For instance, if the ORL-S is invoking operations of certain process, it will send a request to the device network in order to know which device is in charge of the desired service operation to be invoked. Therefore, such request is delivered to the devices, which in cooperation, will find a response and send it back to the requestor. The management of KBs, which are encapsulated in embedded devices is described in Ramis Ferrer et al. (2015). In this work, the algorithm and flow of queries within the distributed network of embedded devices is presented and demonstrated.

Moreover, as the encapsulation of KR into embedded devices will reduce the number of requests sent from the field level to higher levels, efforts and time in both configuration and information exchange will be also reduced. Basically, this particular benefit of the proposed approach, if compared with the OKD-MES solution, is due to the encapsulation

Figure 5. Ontology design within Protégé and an OWL format fragment of the implemented model.

of the RP-S into the devices, which allows the direct interconnection between shop floor equipment and its model, with no need of intermediate layers.

The rest of the interactions and functionalities remain in a similar distribution to the recent OKD-MES architecture. The VIS-S is used for monitoring of shop floor processes, for example, within a Human Machine Interface (HMI). Besides, the ORL-S interacts with devices for executing operations that are described in the local resources of the industrial controllers. The interconnection of each Layer-Service with devices is implemented as a communication channel that implements its corresponding interface in the RP-S instance. In this manner, devices handle requests from any Layer-Service independently of its type. More precisely, cause the ontological model is described using OWL, queries from any Layer-Service are sent to the device SPARQL over HTTP endpoint, which is described in ‘SPARQL 1.1 Graph Store HTTP Protocol’ (2013). This interface is a bidirectional interconnection that devices use also for sending responses to Layer-Service requests.

4.1 KR implementation through ontologies

There are many types of formal representation, such as frames, production rules and ontologies for describing knowledge of specific domains (Brachman and Levesque 2004). This research work proposes the implementation of KBs within ontologies. One of the difficulties on implementing models within such formalism is the verbosity of languages. This can lead to make some mistakes if the models are created manually. However, the design of ontologies is nowadays easier within the use of ontology editors which allow users to abstract from the model syntax. For instance, Protégé⁷ is an open source ontology editor that provides an intuitive graphic user interface. Figure 5 shows the interface of Protégé and a fragment of the generated ontology, which is written in OWL format.

Then, the main objectives of Figure 5 are (i) to depict the editor interface being used at the design phase of the ontology and (ii) to show a specific ontology format. At left hand side of such picture, the interface of Protégé shows a set of views of an ontology for describing a production line, inspired on the one shown in Ramis et al. (2014). More precisely, Figure 5 shows two hierarchy views i.e. class and object property hierarchies. In addition, meanwhile the *instances* view presents all robot instances; the *property assertions* view displays the relationships of the *robot_1* individual. On the other hand, the right side of the Figure 5 presents the interface of a text editor that shows the ontology format, which in this case is written in OWL.⁸ In fact, as the ontology file is automatically generated by the editor, it is possible to generate ontologies in several formats, such as RDF/XML,⁹ Turtle¹⁰ or JSON-LD.¹¹

The screenshot shows the Protégé SPARQL query editor. The query is a SELECT statement with various variables and filters. The results are displayed in a table with 10 columns: pallet, product, statusProperty, component, assemblyOperation, robot, robotStatus, location, ManufacturingCell, and ManufacturingCellStatus. Two rows of data are shown.

```

SPARQL query:
PREFIX bo:<http://www.tut.fi/FAST/LineControllerProject#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>

SELECT ?pallet ?product ?statusProperty ?component ?assemblyOperation ?robot ?robotStatus ?location ?ManufacturingCell ?ManufacturingCellStatus
WHERE {
  ?product bo:hasPallet ?pallet. FILTER (?product = bo:Product_1)
  ?product bo:hasComponent ?component.
  ?assemblyOperation bo:assemblesComponent ?component.
  ?robot bo:canExecuteOperation ?assemblyOperation.
  ?robot bo:hasRobotStatus ?robotStatus.
  ?ManufacturingCell bo:hasRobot ?robot.
  ?robot bo:worksInConveyorZone ?location.
  ?statusProperty rdfs:subPropertyOf bo:hasComponentStatus. FILTER (?statusProperty = bo:hasComponentPending)
  ?product ?statusProperty ?component.
  ?ManufacturingCell bo:hasManufacturingCellStatus ?ManufacturingCellStatus.
}

```

| pallet | product | statusProperty | component | assemblyOperation | robot | robotStatus | location | ManufacturingCell | ManufacturingCellStatus |
|----------|-----------|---------------------|-------------|---------------------|---------|-------------|------------------------|---------------------|-------------------------|
| Pallet_1 | Product_1 | hasComponentPending | component_A | assemblyOperation_1 | robot_1 | idleStatus | workingPosition_Cell_1 | manufacturingCell_1 | CellworkingStatus |
| Pallet_1 | Product_1 | hasComponentPending | component_E | assemblyOperation_5 | robot_3 | idleStatus | workingPosition_Cell_3 | manufacturingCell_3 | CellworkingStatus |

Figure 6. SPARQL SELECT execution through Protégé.

Furthermore, as described in this Section 4, ontologies will be queried in order to retrieve and update information within SPARQL¹² and SPARQL Update,¹³ respectively. The result of SPARQL queries can be provided in different formats, such as XML or JSON. In addition, there are different forms of SPARQL queries: *SELECT*, *ASK*, *CONSTRUCT* and *DESCRIBE*. Each form will provide a different type of result. For instance, *SELECT* queries provide results in tabular form. This is demonstrated through Figure 6 which shows the result of executing a query in the Protégé interface. More precisely, the displayed query is used for checking relevant information about the production of specific products being manufactured in a production line. Moreover, SPARQL Update queries only return an acknowledgment notification as a result in order to indicate if the update operation has been successfully performed. Such kind of updates can perform different actions e.g. *ADD*, *MODIFY* or *DELETE* data graphs of the ontology. The research work (Ramis Ferrer, Iaroyvi, et al. 2016) demonstrates the combination of SPARQL and SPARQL Update queries in the industrial domain.

Due to the increment of the utilisation of semantic descriptions in the industrial domain, there are many research works that implement solutions with domain ontologies for describing industrial systems. Common practices and methodologies are useful for different organisations that develop ontologies in the same domain for supporting the knowledge sharing and distributed collaboration (Lin et al. 2011). In the industrial domain, the research work (Negri et al. 2017) provides a representation of industrial logistics aspects within existing ontologies that are compared and reused. In fact, one of the most important attributes of ontologies is the reusability which implies the need of standardising the terminology used for different models in the same domain (Usman et al. 2013). A review on ontologies performed in the context of product lifecycle management is provided in Kadiri and Kiritsis (2015). On the other hand, semantic rules permit the inference of implicit knowledge. For example, the research work (Zhang et al. 2016) shows how semantic rules can be used for predicting process and resource modifications when product requirements change.

5. Discussion

This research work presents an approach to manage the knowledge of the system in a decentralised manner, based on employing unused computational resources of current embedded devices. This section discusses the advantages and drawbacks of the presented approach, which propose the decentralisation of some OKD-MES functionality. As it can be seen in Figure 7, the main focus of the presented alternative is the distribution and decentralisation of (i) system KB and (ii) functionality of the RPL-S. Then, the management of portions of KB will be performed by a network of interconnected devices which are the ones used currently for the process control at factory floor shops. From a conceptual perspective, this network could be defined as a cloud of devices that is capable of receiving requests that are solved within the cooperation of devices.

Although it is arguable when a centralised and decentralised approach are more suitable in the industrial domain, the proposed alternative in this research work can provide a more robust, flexible, dynamic and scalable than the

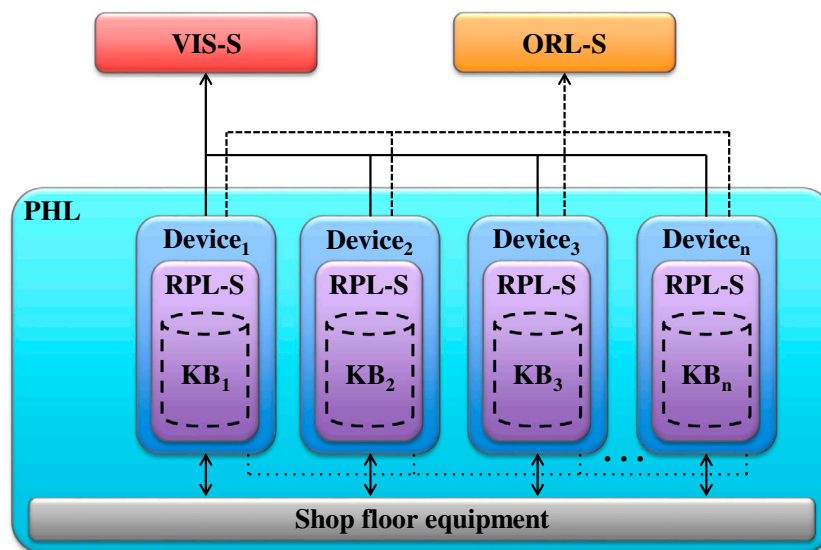


Figure 7. Decentralised RPL-S alternative for the actual OKD-MES.

Table 1. Strengths and weaknesses of the decentralised OKD-MES approach.

| Feature | Strength | Weakness | Description |
|-----------------------------------|----------|----------|---|
| Robustness | Yes | No | One of the main disadvantages of current OKD-MES is that all the knowledge resides in a central point. Thus, if it fails, the system will not work until the central RPL components are fixed. In the proposed approach, the knowledge is decentralised and distributed along a network of embedded devices. Devices are easily replaceable and knowledge can be automatically migrated or reproduced in new devices so that the information is not lost. As happens in other domains, in which topologies with no central node are more robust, the proposed OKD-MES alternative would be more prepared, having to resist to small nodes' failures, rather than resisting a failure on the central component of the system |
| Flexibility | Yes | No | The proposed alternative is flexible in terms of adding new devices to the system. Devices can be re-configured and added to the network because they will include on its RPL-S functionality the required behaviour to perform when appearing in a network or discovering new devices |
| Plug and play, even for knowledge | Yes | No | One important feature of the proposed alternative is that devices can be added on runtime. This feature is inherited from the current OKD-MES, wherein devices of the PHL that control factory shop floor equipment can be added without the need of stopping the system. In addition, the proposed alternative permits the OKD-MES to add description of knowledge also in runtime. This is because the knowledge is encapsulated in device. Thus, the alternative enables plug and play, even for knowledge |
| Scalability | Yes | No | Throughout the alternative, the OKD-MES would become fully scalable. Currently, the only module or functionality that impedes the system to be scalable is the RPL, which must be unique for any system. Within the employment of devices that hosts the functionality of the RPL-S in a decentralised approach, the OKD-MES becomes scalable |
| Resource costly on devices | No | Yes | The first limitation of the proposed alternative is obviously the availability of resources in devices. In the decentralised approach, the system will have actually more resources according to the number of devices employed. Then, it can be stated that the implementation of presented approach can be expensive when more powerful devices are employed |
| Possible device conflicts | No | Yes | A new weakness that appears in this alternative is the management of device conflicts. The conflict may come when knowledge of the system represented in different devices is redundant or contradictory. For example, this problem can be avoided within the employment of semantic reasoners that validate the entire system KB frequently, or introducing a conflict resolution approach within a distributed querying protocol |
| Time in small systems | Yes/No | Yes/No | In relation to the scalability, it may be argued that the performance, in terms of time, of the presented alternative in small systems using the proposed approach might be slower than within current OKD-MES. In principle, it is expected that at higher system's complexity the performance of the alternative will be faster, but using many devices for controlling a simple (small) system would be slower due to the number of connections and communication between devices |

OKD-MES solution. However, the efficiency of the decentralised approach might be limited by the computational resources of embedded devices and affected by possible conflicts in device communication. To compare the proposed approach with the current OKD-MES framework, Table 1 presents strengths and weaknesses of both system solutions.

As this research work is presented as a source for establishing the grounds of an alternative to the current OKD-MES, there are not actual results to enforce some of the stated advantages. In any case, it can be claimed that the current OKD-MES is not scalable. VIS-S, ORL-S and even PHL are layers that can be easily decentralised because they can reside out of the framework and can be reproduced several times with no harm to the system. However, each system must work with a unique RPL. Such requirement disallows the OKD-MES to be fully scalable. This would be allowed throughout the proposed decentralised approach. The encapsulation of RPL-S functionality and the partition of the entire system KB enable the solution to be scalable within the addition of needed embedded devices. Simply, whenever more resources are needed, extra devices may be deployed on runtime, due to both *flexibility* and *plug and play* stated features in Table 1.

On the other hand, due to the described *time in small systems*, wherein the complexity is not high, the performance of the presented approach would be weaker than the current OKD-MES. Nevertheless, the performance of the proposed

solution would be better when the size and complexity of the system increases. In such case, the decentralisation of RPL-S would provide a faster approach for process control and decision-making to the manufacturing system. Fundamentally, the deployment of embedded devices close to where data are generated permits the transformation of such data into information with no need of intermediate layers and vertical communication. This means that interoperability problems, delays in transporting data and network overload may be dramatically reduced. Therefore, the decentralised approach may perform decision-making tasks faster than centralised approaches that require the transformation and exchange of information with systems that are located at higher levels of the manufacturing enterprise. Moreover, this matter is enhanced with the scalability principles of the decentralised proposal, which would permit the OKD-MES accommodating to larger systems.

6. Conclusion

This article presents the organisation and different functionality of the OKD-MES layered solution. Afterwards, the presented research work proposes a decentralised architecture as an alternative for moving down the management of knowledge to the device level. In fact, one of the direct impacts of such approach is the reduction of vertical communication since part of the interactions between the PHL and RPL are replaced by horizontal communication. This means that some of the required OKD-MES layer-to-layer communication is replaced by a device-to-device interaction, due to the encapsulation of functionality at device level (i.e. PHL).

Moreover, Section 5 aims to emphasise in explaining the strengths and weaknesses of the presented alternative. In principle, the implementation of the presented approach would create a solution that is more robust, flexible and scalable in terms of knowledge management than the OKD-MES solution. However, the cost of resource on devices and possible conflicts when communicating among each other must be considered and further investigated.

Another important fact to conclude with the proposed alternative is that the encapsulation of the RPL-S OKD-MES functionalities inside embedded devices could be just a first step in the modification of the eScop project solution. More precisely, the encapsulation of other Layer-Service functionalities into embedded devices could be tried as long as there are available resources which allow the management of such functionalities. Then, this progressive approach would offer same kind of benefits, such as avoiding vertical communication for the orchestration of service execution. This could be actually achieved within the encapsulation of the ORL-S into devices. Nevertheless, this requires (i) a deep study about the limits of encapsulation in the embedded devices, which are employed at the factory shop floor and (ii) a definition of required algorithms for device cooperation in order to substitute other layer functionalities.

Hence, the further work of this research includes the evaluation of aforementioned limits and the definition of a methodology for descending functionalities to be managed at the device level, wherein the processes are executed. Then, the next step of this research work is to define how devices, aside from RP-S functionalities, are able also to host the visualisation of themselves, which e.g. can feed the information required of dedicated HMIs. Afterwards, the implementation and testing of a complete solution will be possible.

Acknowledgement

The authors gratefully acknowledge the support of the graduate school funding of Tampere University of Technology in carrying out this work.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes

1. <http://www.escop-project.eu/>.
2. <http://www.inicotech.com/>
3. <https://www.raspberrypi.org/>
4. <https://www.arduino.cc/>
5. <https://www.onion.io/>
6. *Layer-Service* refers to the service naming in actual OKD-MES architecture as e.g. RPL-S or ORL-S.
7. <http://protege.stanford.edu/>
8. <https://www.w3.org/TR/owl-guide/>
9. <https://www.w3.org/TR/rdf-syntax-grammar/>
10. <https://www.w3.org/TR/turtle/>

11. <https://www.w3.org/TR/json-ld/>
12. <https://www.w3.org/TR/rdf-sparql-query/>
13. <https://www.w3.org/TR/sparql11-update/>

ORCID

Borja Ramis Ferrer  <http://orcid.org/0000-0002-0525-163X>

Jose Luis Martinez Lastra  <http://orcid.org/0000-0001-6227-3408>

References

- Brachman, R. J., and H. J. Levesque. 2004. *Knowledge Representation and Reasoning*. Morgan Kaufmann.
- Delamer, I. M., and J. L. M. Lastra. 2006. "Ontology Modeling of Assembly Processes and Systems Using Semantic Web Services." In *2006 IEEE International Conference on Industrial Informatics*, 611–617. doi:10.1109/INDIN.2006.275631.
- Extensible Markup Language (XML). 2015. Accessed 1 June 2016. <https://www.w3.org/XML/>
- Fumagalli, L., S. Pala, M. Garetti, and E. Negri. 2014. "Ontology-based Modeling of Manufacturing and Logistics Systems for a New MES Architecture." In *Advances in Production Management Systems. Innovative and Knowledge-based Production Management in a Global-local World*, 192–200. Springer. http://link.springer.com/chapter/10.1007/978-3-662-44739-0_24.
- Gao, Q., F. Li, and C. Chen. 2015. "Research of Internet of Things Applied to Manufacturing Execution System." In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 661–665. doi:10.1109/CYBER.2015.7288019.
- Garetti, M.. 2012. "P-PSO Ontology for Manufacturing Systems." In edited by B. Theodor, 449–456. Elsevier. doi:10.3182/20120523-3-RO-2023.00222.
- Garetti, M., L. Fumagalli, A. Lobov, and J. L. M. Martinez Lastra. 2013. "Open Automation of Manufacturing Systems through Integration of Ontology and Web Services." In edited by B. Natalia, 198–203. Elsevier. doi:10.3182/20130619-3-RU-3018.00169.
- Iarovyi, S., J. Garcia, and J. L. M. Lastra. 2013. "An Approach for OSGi and DPWS Interoperability: Bridging Enterprise Application with Shop-floor." In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, Bochum, Germany, 200–205. IEEE: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6622882.
- Iarovyi, S., W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra. 2016. "Cyber-physical Systems for Open-knowledge-driven Manufacturing Execution Systems." *Proceedings of the IEEE* 104 (5): 1142–1154. doi:10.1109/JPROC.2015.2509498.
- IBM developerworks: New to SOA and Web Services. 2007. [CT801], March 5. Accessed 1 April 2015. <http://www.ibm.com/developerworks/webservices/newto/service.html>
- ISO. 2007. "IEC 62264-3:2007 – Enterprise-control System Integration – Part 3: Activity Models of Manufacturing Operations Management." Accessed 2 June 2016. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40949
- Kadiri, S. E., and D. Kiritsis. 2015. "Ontologies in the Context of Product Lifecycle Management: State of the Art Literature Review." *International Journal of Production Research* 53 (18): 5657–5668. doi:10.1080/00207543.2015.1052155.
- Lastra, J. L. M., I. M. Delamer, and F. Ubis. 2010. *Domain Ontologies for Reasoning Machines in Factory Automation*. ISA.
- Lin, L. F., W. Y. Zhang, Y. C. Lou, C. Y. Chu, and M. Cai. 2011. "Developing Manufacturing Ontologies for Knowledge Reuse in Distributed Manufacturing Environment." *International Journal of Production Research* 49 (2): 343–359. doi:10.1080/00207540903349021.
- Lobov, A., F. U. Lopez, V. V. Herrera, J. Puttonen, and J. L. M. Lastra. 2009. "Semantic Web Services Framework for Manufacturing Industries." In *IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008*, Bangkok, Thailand, 2104–2108. doi:10.1109/ROBIO.2009.4913327.
- Lobov, A., J. Puttonen, V. V. Herrera, R. Andiappan, and J. L. M. Lastra. 2008. "Service Oriented Architecture in Developing of Loosely-coupled Manufacturing Systems." In *6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008*, Daejeon, Korea, 791–796. doi:10.1109/INDIN.2008.4618209.
- Maniraj, V., and R. Sivakumar. 2010. "Ontology Languages – A Review." *International Journal of Computer Theory and Engineering* 2 (6): 887–891.
- McClellan, M. 1997. *Applying Manufacturing Execution Systems*. CRC Press.
- MESA International. 2011. "MESA White Paper #39: MESA Model Evolution." <https://services.mesa.org/ResourceLibrary/ShowResource/73b23d9e-133e-456b-b844-d7ba5ff8278a>.
- MESA International – MESA Model. 2016. Accessed 2 June 2016. <http://www.mesa.org/en/modelstrategicinitiatives/MESAModel.asp>
- Moctezuma, L. E. G., J. Jokinen, C. Postelnicu, and J. L. M. Lastra. 2012. "Retrofitting a Factory Automation System to Address Market Needs and Societal Changes." In *2012 10th IEEE International Conference on Industrial Informatics (INDIN)*, Beijing, China, 413–418. doi:10.1109/INDIN.2012.6301202.
- Negri, E., L. Fumagalli, M. Garetti, and L. Tanca. 2016. "Requirements and Languages for the Semantic Representation of Manufacturing Systems." *Computers in Industry* 81: 55–66. doi:10.1016/j.compind.2015.10.009.
- Negri, E., S. Perotti, L. Fumagalli, G. Marchet, and M. Garetti. 2017. "Modelling Internal Logistics Systems through Ontologies." *Computers in Industry* 88: 19–34. doi:10.1016/j.compind.2017.03.004.

- OWL Web Ontology Language Reference. 2004. Accessed 1 April 2015. <http://www.w3.org/TR/owl-ref/>
- Puttonen, J., A. Lobov, and J. L. M. Lastra. 2013. "Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems." In *2013 IEEE 20th International Conference on Web Services*, 419–426. Santa Clara, CA.
- Puttonen, J., A. Lobov, and J. L. Martinez Lastra. 2013. "Semantics-based Composition of Factory Automation Processes Encapsulated by Web Services." *IEEE Transactions on Industrial Informatics* 9 (4): 2349–2359. doi:10.1109/TII.2012.2220554.
- Qusay, H. M. 2005. "SOA and Web Services." Accessed 2 March 2016. <http://www.oracle.com/technetwork/articles/javase/soa-142870.html>
- Ramis Ferrer, B., and J. L. Martinez Lastra. 2017. "Private Local Automation Clouds Built by CPS: Potential and Challenges for Distributed Reasoning." *Advanced Engineering Informatics* 32: 113–125. doi:10.1016/j.aei.2017.01.007.
- Ramis Ferrer, B., S. Iarovy, L. Gonzalez, A. Lobov, and J. L. Martinez Lastra. 2015. "Management of Distributed Knowledge Encapsulated in Embedded Devices." *International Journal of Production Research* 1–18. doi:10.1080/00207543.2015.1120902.
- Ramis Ferrer, B., B. Ahmad, D. Vera, A. Lobov, R. Harrison, and J. L. Martinez Lastra. 2016. "Product, Process and Resource Model Coupling for Knowledge-driven Assembly Automation." *At – Automatisierungstechnik* 64 (3): 231–243. doi:10.1515/ato-2015-0073.
- Ramis Ferrer, B., S. Iarovy, W. M. Mohammed, A. Lobov, and J. L. M. Lastra. 2016. "Exemplifying the Potentials of Web Standards for Automation Control in Manufacturing Systems." *International Journal of Simulation Systems, Science & Technology* 17 (33): 3.1–3.12. doi:10.5013/IJSSST.a.17.33.03.
- Ramis, B., L. Gonzalez, S. Iarovy, A. Lobov, J. L. Martinez Lastra, V. Vyatkin, and W. Dai. 2014. "Knowledge-based Web Service Integration for Industrial Automation." In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, Porto Alegre RS, Brazil, 733–739. doi:10.1109/INDIN.2014.6945604.
- RDF – Semantic Web Standards. 2014. Accessed 1 June 2016. <https://www.w3.org/RDF/>
- eScop. 2016. "Media Releases | ESCop." <http://www.escop-project.eu/media-releases/>.
- Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. 2007. "Pellet: A Practical OWL-DL Reasoner." *Web Semantics: Science, Services and Agents on the World Wide Web* 5 (2): 51–53. doi:10.1016/j.websem.2007.03.004.
- SPARQL 1.1 Graph Store HTTP Protocol. 2013. Accessed 1 April 2015. <http://www.w3.org/TR/sparql11-http-rdf-update/>
- SPARQL 1.1 Update. 2013. Accessed 1 April 2015. <http://www.w3.org/TR/sparql11-update/>
- SPARQL Query Language for RDF. 2008. Accessed 1 April 2015. <http://www.w3.org/TR/rdf-sparql-query/>
- Tampere University of Technology. 2016. "New Manufacturing Execution System Improves the Competitiveness of European Industry." Tampere University of Technology. Accessed 2 June 2016. <http://www.tut.fi/en/about-tut/news-and-events/new-manufacturing-execution-system-improves-the-competitiveness-of-european-industry-x154451c2>
- Usman, Z., R. I. M. Young, N. Chungoora, C. Palmer, K. Case, and J. A. Harding. 2013. "Towards a Formal Manufacturing Reference Ontology." *International Journal of Production Research* 51 (22): 6553–6572. doi:10.1080/00207543.2013.801570.
- Witsch, M., and B. Vogel-Heuser. 2012. "Towards a Formal Specification Framework for Manufacturing Execution Systems." *IEEE Transactions on Industrial Informatics* 8 (2): 311–320. doi:10.1109/TII.2012.2186585.
- Zhang, J., B. Ahmad, D. Vera, and R. Harrison. 2016. "Ontology Based Semantic-predictive Model for Reconfigurable Automation Systems." In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, Poitiers, France, 1094–1099. doi:10.1109/INDIN.2016.7819328.

IV

EXEMPLIFYING THE POTENTIALS OF WEB STANDARDS FOR AUTOMATION CONTROL IN MANUFACTURING SYSTEMS

by

Borja Ramis Ferrer, Sergii Iarovy, Wael M. Mohammed, Andrei Lobov, José L.
Martinez Lastra, 2016

International Journal of Simulation Systems, Science & Technology. Volume: 17,
Number: 33, pp 3.1–3.12

*Reproduced with kind permission: The paper was published in the International Journal
of Simulation: Systems, Science & Technology, Volume 17, Issue number 33, paper
number 3 with DOI: 10.5013/IJSSST.a.17.33.03.*

Exemplifying the Potentials of Web Standards for Automation Control in Manufacturing Systems

Borja Ramis Ferrer, Sergii Iarovy, Wael M. Mohammed, Andrei Lobov, José L. Martinez Lastra
Factory Automation Systems and Technologies Laboratory (FAST-Lab),
Tampere University of Technology,
Tampere, Finland
{borja.ramisferrer, sergii.iarovy, wael.mohammed, andrei.lobov, jose.lastra}@tut.fi

Abstract — Web standards developed mainly by W3C and OASIS shape the general IT domain and its applications. Due to the scale of web applications, the web standards have matured to deal with typical situations of finding the right node on the network, reconfiguring the routing for messaging, using common standards for representing graphical information and many others. Industrial manufacturing can benefit from the web standards due to interoperability and simplified application integration. This article reviews the current use of web standards in the industrial automation domain. In addition, it describes and discusses the potential of using web standards at all the levels of the automation system: from high level web-based user interfaces to the industrial controllers located in the lowest layer of the well-known automation pyramid. Aligned with such a description, the article presents a framework for Open, Knowledge Driven Manufacturing Execution Systems (OKD-MES), which enables the systematic use of web standards and technologies in factories. Finally, the manuscript exemplifies the use of web standards for automation control in real implementations in a mobile phone assembly line.

Keywords - web standards; industrial automation; OKD-MES; Semantic Web; Service-oriented Architecture; Cyber-physical systems

I. INTRODUCTION

In the domain of industrial manufacturing throughout last decade the level of competition for customers and resources has been constantly increasing. One of the most important approaches to reducing operating costs related to manufacturing and the customization of goods is linked to the possibilities of more interoperable automation systems. The concept of Industrial Internet is one of the possible approaches to achieve new quality in factories.

The concept of Industrial Internet requires more capable devices which can interact in the web system. In the last few years, certain efforts have been made to create such smart devices. Some of these devices are able to interact with significant levels of autonomy and to provide more interactive and adaptable functionalities for industrial systems [1], [2], [40].

Modern factory shop floor equipment and software provide more heterogeneous and interoperable infrastructure for automation. One of the key problems in such large-scale systems is lack of interoperability and a need for customized integration of components. In fact, as the nature of the Industrial Internet system is very similar to that of the consumer Internet, it affords an opportunity to employ solutions of the latter in factories.

Web standards are critical enabling factors of Internet success. The World Wide Web Consortium (W3C) and OASIS are the leading standardization organizations in the domain and provide a comprehensive and versatile set of standards to enable web based systems. The application of

such web standards in industrial systems may make it possible not only to resolve technical difficulties, but also to remove the actual barrier that prevents the complete integration of two domains: general IT and factory automation.

According to their use, specifications, and demonstrated implementations, web standards are open, mature, and usually efficient. Some of the standards, such as the concept of Web Services (WS), and some specific implementations of this, have already been successfully adapted to factory shop floor control devices [3]–[5]. In fact, based on service-enabled devices, the application of Semantic WS for manufacturing has already been researched and documented in several articles [6]–[8]. Also, an evaluation of applying web standards for system and knowledge representation systems, decision support and visualization is presented in [9].

The ongoing EU Project eScop¹ (Embedded systems Service-based Control for Open manufacturing and Process automation) currently employs web standards for multiple purposes in a factory wide Manufacturing Execution System (MES). Within the use of web standards, the eScop project is developing a framework for realizing a new concept: Open, Knowledge Driven Manufacturing Execution Systems (OKD-MES). Descriptions of the novel OKD-MES concept can be found in [15], [32], [46]. The purpose of this article is to demonstrate the possibilities and potentials of employing web standards at different levels of manufacturing systems

¹ <http://www.escop-project.eu/>

with the example of the implementation of the OKD-MES framework.

The research on applying web standards at different levels of manufacturing systems is presented in this paper as follows: Section I describes the possibilities of different applications of WS standards for factory shop floor devices. Section III describes a set of standards that are applicable for modeling industrial systems. Section IV then presents the possibility of applying web standards for coordinating systems, the use of which is described in Section II and Section III. Section V presents discussion on some of the possibilities for exploiting some of the web standards for industrial system visualization. In Section VI, the synergy of the standards applied at all levels of the system is described and practical examples are also given. In fact, the description of such integration of standards permits the presentation of the OKD-MES concept. Aiming at a demonstration of how web standards are actually implemented, Section VII provides examples of developments employed in a mobile phone assembly line for automation control. Finally, Section VIII concludes the paper and offers suggestions for further work. It should be noted that this article is an extension of the research work presented in [48].

II. DEVICES

The main goal of industrial automation is to control and monitor processes in an automated manner. To achieve this, industrial controllers are used for controlling sensors and actuators that, conjunctly, are the devices enabling the physical execution of process. Usually, industrial controllers are deployed near to where the sensors and actuators reside. The deployment of industrial controllers in a facility implies that these units have a small form factor that, in return, constrains its computational resources [10]. In addition, industrial controllers need to execute the control logic and exchange information with other controllers on same level (horizontal communication) or with higher-level components (vertical communication) of the entire industrial automation system.

Several communication protocols enable horizontal integration between controllers, such as CAN, etherCAT and Modbus, among many other fieldbuses. Nevertheless, web-based technologies are the most convenient and suitable for allowing the vertical communication of controllers and other industrial components like MESs or even Enterprise Resource Planning (ERP) modules [11].

A. Web standards for devices

Some of the service-oriented communication standards and approaches attracting special scholarly and industrial attention include: OPC Unified Architecture (OPC UA), Device Profile for Web Services (DPWS) and Representational States Transfer (REST) architecture [12], which are described below.

OPC UA is the successor to Object Linking and Embedding (OLE) for process control (OPC). This communication stack offers several features such as scalability, multi-threading, and security. An outstanding characteristic is that OPC UA is a Service Oriented

Architecture (SOA). It exposes methods and device functionality as services which are protocol independent. Two protocols are defined for this purpose: binary Transmission Control Protocol (TCP) protocol and Web-service oriented. The binary protocol is highly efficient and reduced significantly the transaction overheads [13]. Web Service Simple Object Access Protocol (SOAP) protocol facilitates integration with traditional IT components, tools, and technologies. It is easily understood by firewalls and uses Hypertext Transfer Protocol (HTTP), which is the foundation protocol of the World Wide Web. The main drawback of OPC UA is that it is required to be part of the OPC foundation in order to gain access to stack specification.

The DPWS standard [14] defines and tunes a set of SOAP based WS protocols for devices. This stack profile enables WS capabilities on resource-constrained devices. Such capabilities include secure invocation of WS operations, description, and dynamic discovery of WSs and mechanisms to subscribe and receive events from WSs. An important characteristic is that devices that implement DPWS are fully aligned with the WS technology. This facilitates the vertical integration of devices with high-level applications. This synergy between physical devices and cyber systems realizes one of the topics currently widely discussed in industrial automation: cyber-physical systems (CPS) [1], [2], [38], [39]. Nevertheless, one drawback of this protocol is its verbosity due to the fact that SOAP messages are XML formatted. It is important to note that DPWS is an open standard, hence any manufacturer can adopt it. In the industrial domain there are already commercial industrial controllers that implement the DPWS stack such as, for example, the S1000 by Inico Technologies².

Both OPC-UA and DPWS use Remote Procedure Call (RPC) as an architectural style for the services. In this case most of the HTTP capabilities are not used. It may be useful if the system employs services utilizing different protocols, but this is rather rare. RPC services likewise tend to be more tightly coupled with clients than some other implementations of WS.

Representational States Transfer (REST) is another software architectural style applicable for WS. REST defines a set of constraints which makes WS more compatible with Web infrastructure and technologies. The constraints of REST enable more scalable, loosely-coupled and efficient services compared to RPC. The RESTful WSs often employ HTTP/HTTPS as the transport and application protocol [16]. These web mechanisms are very well understood and can be easily ported over the web. Currently the payload in RESTful WSs is often formatted as JavaScript Object Notation (JSON), which is less verbose than XML and still human readable. A vast number of technologies, frameworks, and tools from traditional IT can be exploited in the industrial automation field if industrial controllers are implemented with RESTful capabilities [17].

For the formal description of WS, several description languages may be applied. The most important for the

² <http://www.inicotech.com/>

description of SOAP services is Web Service Description Language (WSDL). WSDL provides a complete technical description of services. In order to enrich this description with metadata about services, Semantic Annotations for WSDL (SAWSDL) is generally used. It makes it possible to connect the service with related concepts in the model. It should be noted that WSDL2.0 can be equally well applied to RESTful services. However, a different approach is usually applied in this case for reasons some of which are described below.

B. Application of REST

Application of RESTful WS on the factory shop floor level may require an approach different than the implementation of RPC services. This relates mainly to the different perception of the system in these approaches. If RPC usually focuses on the actions in the system, REST suggests concentrating on resources. HTTP verbs such as *GET*, *PUT*, *POST*, and *DELETE* are used with the resources to describe the action to be performed on the resource. Furthermore, HTTP response codes and other headers may provide additional information on service execution.

The resources in REST are defined by their representation. The concept of Hypermedia (linked media) is usually applied to resource representation in order to define resource relations with other resources. This approach allows further exploitation of web architecture, as links to other resources may be dereferenced without any extra tools or technologies. Furthermore, Hypermedia is recommended for use as the Engine of Application States (HATEOAS). This approach allows services to instruct clients on further possible operations and, hence, to remove the need for a client to guess the next operation. It means that in case of changes in service implementation it will inform the client immediately on execution of the service. An approach for semantically rich HATEOAS descriptions is being developed within Hypermedia Driven Web APIs (Hydra) [18], [19].

III. MODELS

Models are another important element for contemporary industrial automation systems. According to stated conditions, knowledge of system properties and their interrelations allows adequate acting and reacting in the modeled system. While the previous section concentrated on how to provide interaction of the automated system with the physical world, this section focuses on the representation of available knowledge about the system.

A. Models for design and runtime support

Enabling the management of manufacturing processes by information systems necessitates transforming real world assumptions into a formal model, which must be understandable by cyber systems. Ontologies may be used for representing the real world because they provide a pragmatic means of modeling domain specific knowledge. An ontology is an explicit and formal specification of a shared conceptualization [20]. In other words, ontologies are employed to represent knowledge of a certain domain as a

set of concepts, their definitions and interrelationships. Web Ontology Language (OWL) is one of the languages used for constructing ontologies. The language is characterized by formal semantics and RDF/XML-based serializations for the Semantic Web. As illustrated in Figure 1, a hierarchical stack represents the architecture of the Semantic Web [35].

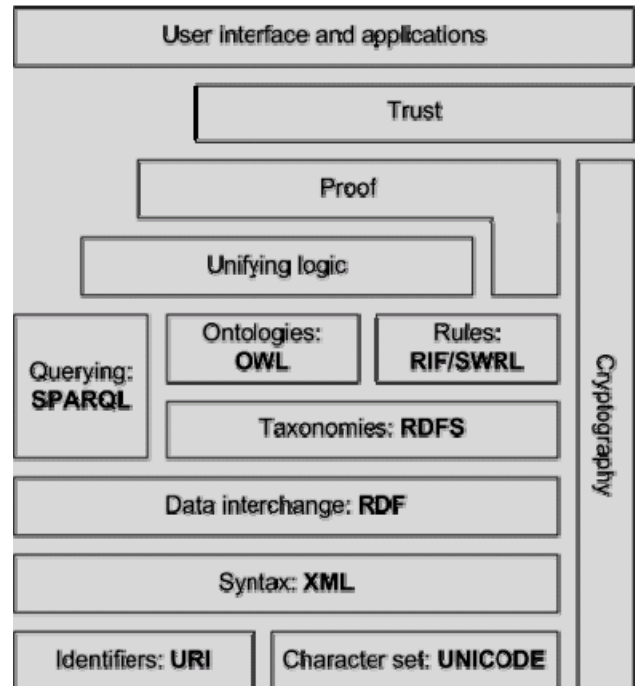


Figure 1. The Semantic Web Stack [35]

In the stack presented, XML is a surface syntax of structured documents and imposes no semantic constraints on the document. Then, XML Schema defines the structure constraints of XML documents. RDF [22] is a data model of resources and their relationships expressed by XML syntax that provides simple semantics for the data model. RDF Schema [23] is a vocabulary describing the attributes and types of the RDF resources. Hence it provides generic semantics for the attributes and types. Finally, OWL adds more vocabulary to describe attributes and types, such as disjoint and cardinality constraints in types and symmetry in attributes.

OWL is often presented as one of the richer ontology languages because it has more mechanisms for representing semantics in comparison with XML, RDF, and RDFS. Nevertheless, Table I shows a comparison of various syntaxes for OWL 2, which is an extension and revision of OWL developed by the W3C Web Ontology Working Group and published in 2004 [21]. As shown in Table I, several syntaxes can be used to store OWL 2 ontologies and to exchange them among tools and applications. It should be noted that the primitive exchange syntax for OWL 2 is RDF/XML. Thus RDF/XML is the syntax that must be supported by all OWL 2 tools.

TABLE I. COMPARISON OF SYNTAX FOR OWL 2

| Name of Syntax | Status | Purpose |
|-------------------|-----------|--|
| RDF/XML | Mandatory | Interchange ^a |
| OWL/XML | Optional | Easier to process with XML tools |
| Functional Syntax | Optional | Easier to see the formal structure of ontologies |
| Manchester Syntax | Optional | Easier to read/write DL Ontology |
| Turtle | Optional | Easier to read/write RDF triples |

^aCan be written and ready by OWL 2 software

Users and applications can interact with ontologies and data by querying the ontology model using SPARQL query language [24], which was standardized in 2008 by W3C. The standard query evaluation mechanism is based on sub-graph matching and is called simple entailment because it can equally be defined in terms of the simple entailment relation between RDF graphs [25]. Like other database query languages, SPARQL uses several keywords to form a pattern to query data. A SPARQL query contains three parts. Firstly, the pattern matching part includes fundamental features of pattern matching of graphs such as optional parts, union of patterns, nesting, filtering (or restricting) values of possible matches. Secondly, the solution modifiers part allows modifying the obtained query values applying classical operators like *DISTINCT*, *ORDER* and *LIMIT*. Thirdly, the output of a SPARQL query can be of different types: Boolean queries (true/false) for *ASK* query type, selections of values of the variables which match the patterns for *SELECT* query type, construction of new triples from these values for *CONSTRUCT* query type. Moreover, SPARQL Update [34] is an extension of SPARQL, which permits the update of the model within different operations such as *DELETE* or *INSERT* for deleting or inserting triples. It should be noted that the output of such queries is an execution message for indicating the success of the update operation.

Ontology also supports reasoning which derives facts that are not explicitly asserted in the ontology. For example, if a model describes that A is an ancestor of B and B is an ancestor of C, then, although the conclusion that A is also an ancestor of C is trivial, the model needs an additional engine called a reasoner, which is capable of inferring this new fact. Then, a reasoner (or reasoning engine) is a piece of software capable of performing reasoning tasks. In other words, a reasoner is capable of inferring logical consequences from a set of assertions in the ontology. Several reasoners are currently available, such as FaCT++, Pellet and HermiT. Nevertheless, Pellet seems to be one of the most common reasoning engines used for reasoning OWL models in current implementations. It should be noted that rule languages, such as Semantic Web Rule Language (SWRL) [41], are used for defining rules in ontological models that are also interpretable by reasoning engines. Within the employment of semantic rules, automatic mapping of ontological individuals can be achieved, as demonstrated in [42], [45].

According to these explanations, it is obvious that ontologies enable the construction of models for design and

runtime support of systems. This is feasible because ontologies allow the description of models that are understandable in machine-to-machine, human-to-machine and even human-to-human communications. In addition, the computational inference achieved thanks to reasoning engines permits an automatic solution for deriving new facts, which are later included in the model. Finally, it should be noted that ontologies are reusable, extendible, and flexible. This means that population of data in ontological models is possible after model definition at runtime. This feature makes them even more powerful in the industrial domain because the system model changes continuously due to the large number of events that occurred in manufacturing systems. The benefits of ontologies for manufacturing and logistics operations management are presented in [47].

B. Use of ontologies at runtime

Modeling within ontology is one aspect of knowledge driven information systems involving the use of ontologies in the development phase. The other aspect is its usage during runtime. In the latter case the ontological models act as semantic data as opposed to the mere syntactic data managed in databases. An Ontology Management System (OMS) is needed to manage the semantic data such as Sesame [36] and Jena [37] frameworks.

The OMS serves as a repository where ontology models can be imported or exported. On the other hand, an OMS may need to manage the ontology models, for instance, updating the concepts and relationships. Implementing a SPARQL query engine in OMS is one of the easy ways to manage ontology models upon the demand of external use. An inference engine of the OMS can also be used so that for retrieving information this engine can return the entailed information derived by a reasoner from the asserted facts and axioms. In addition, OMS can offer universal access to data sources. Heterogeneous data sources can be accessed by mapping schema based on certain rules so that various specific data sources can be wrapped into one source. Users or applications can simply use OMS to obtain information from different sources independent of their locations and schemas.

IV. SYSTEM COMPOSITION

Automated service-oriented manufacturing system composition requires coordinating service executions. One of the problems for composing systems is the description of processes in a certain format that can later be executed by machines. This problem is generally addressed by using workflow or orchestration engines. Such engines are capable of executing services according to their technical descriptions. This approach works well with static service inventories. But once there are changes in the service inventory, due, for example, to equipment replacement, the process must be reconfigured. This raises a second problem: that of process composition. In systems with rich knowledge models, it should be possible to address the second problem using more abstract process descriptions, which are later mapped to particular service implementations. This section will first review some of applicable web standards for the

system composition and then describe the composition process available within the proposed set of technologies.

A. System composition standards

One of the standards which may be applied to describe the executable process is Web Services Business Process Execution Language (WS-BPEL) [26]. WS-BPEL defines the sequences and patterns of service executions. There are several engines capable of processing WS-BPEL description and executing the related process. The applicability of WS-BPEL for industrial automation was presented by J. Puttonen in 2008 [27]. Some other efforts were directed in another language for process description: Business Process Model and Notation (BPMN) [28]. Originally developed for modeling processes, BPMN 2.0 addresses the problem of executing such compositions.

Another approach to composition is the application of OWL to WS is OWL-S [29]. OWL-S aims to provide means for service discovery, invocation, and composition. It describes services from technological aspects (grounding) through the description of communication patterns to the logical pre-conditions of service execution and the impact of execution on the system [30], [31].

B. Composition process in automation systems

Service composition is a knowledge-intensive process. Information is required about service inventory, equipment status, processes and requirements, schedules, and others as well as relations between domains. In most cases it is also a time consuming process.

Querying ontology is slower than direct data access, noting that the complexity of queries will directly affect the speed of the process, being slower with more complex queries. Therefore, the load on the ontology should be minimized by simplifying queries and reducing their number when possible.

Firstly, it is possible to use the ontologies to configure the process executions. In such cases the knowledge base provides not particular data but rather the source of this data as direct data access is faster than querying. Once the system composition tool is connected to the required data source it may directly interact with it as long as the configuration of the system does not change.

In addition to access to data sources information from ontologies, the engine executing an automated process may require some data not separately available in any component of the system. But as regards the reasoning capabilities of ontology the knowledge base may be used as a source for such data. Following this approach, it is possible to provide the orchestration tools with the required information, which can be efficiently accessed in the system.

V. VISUALIZATION

Interaction with a human is another important function of industrial systems in which web standards may be applied. Currently multiple hand-held or even wearable devices are becoming one of the most practical interfaces for human-machine interactions. The application of web browser based

user interfaces provides the benefits of being cross-platform, powerful, and a user-friendly framework.

The most basic standard for web-based visualization is Hyper Text Markup Language (HTML). Besides its ability to represent the industrial system, it provides some out-of-the-box compatibility with RESTful architecture. In fact, through its integration with JavaScript (JS), HTML may be employed to provide the user with access to all WS functionality. Additionally, to enrich the data driven capabilities for representation Cascading Style Sheets (CSS) and Scalable Vector Graphics (SVG) may be applied.

A. Visualization supported by ontologies

Visualization applications have been developed for monitoring the manufacturing system. These applications form the front end of the OKD-MES concept by providing a graphical visualization of the system in web-enabled devices such as computers, smart phones, or tablets. Manufacturing System Ontology (MSO) for OKD-MES aims at a general description of the components of the manufacturing system [32]. These descriptions should also include information to generate visualization of the system based on certain rules. Such rules allow mapping the MSO data required, for example, for configuring visualization symbols.

Visualization symbols may be simple or complex. Simple symbols adopting visualization standards like SVG can be directly represented in the knowledge base with different levels of detail in the graphics. A basic SVG symbol defines the shape, position of the shape (x, y), its size (height, width) and style (fill, stroke, and color) [33]. By modeling these features, the data required to create SVG symbols can be stored in the ontology.

To support visualization, the ontology must provide mapping between the real component of the system and its visualization symbol. For example, if a conveyor is located in the system and is represented by an instance of *Conveyor* class in the MSO, then it can be mapped to its corresponding symbol instance in *Symbol* class using the *hasSymbol* object property. This kind of mapping offers flexibility to store the visualization related information about the components in MSO and provides the information to create visualization displays. Also, it enables a dynamic graphical visualization that reflects the changes in the system during runtime. Whenever a new device is added to the system, the ontology will be updated (e.g. via SPARQL Update for an RDF-based model) with the description of the device, which also includes visualization data. Hence, the added device appears in the visualization display making the visualization dynamic.

Another important concept to be represented in the ontology is the display composition. The display usually includes visual elements like screens, graphs, tables, maps, buttons, and objects (single symbol). An element can be composed of other elements, e.g. a screen may include graphs, objects etc., and these are present at a certain position on the element. The same element can be in a different position in different displays. All these concepts related to display composition should be represented in MSO to facilitate the creation of the visualization display.

Some visualization information may not be available during modeling and may depend to some extent on the user to provide the missing data to configure the display screen. The MSO will not contain metadata information like the position of an element if it is not initially possible to pre-define such data. User Interface (UI) could be a solution in this case to obtain the missing information from the user. MSO is flexible enough for storing the metadata obtained from the user and, together with other visualization information, it is feasible the configuration of the display screen.

Therefore, the visualization of the system is supported by the ontology because it makes possible a dynamic graphical visualization, supporting the extendibility and evolution of UIs. The visualization information is also represented in the knowledge base like any other system information providing universal information representation format and simplifies system management.

VI. INTEGRATING DIFFERENT LAYERS OF AUTOMATION SYSTEMS WITH WEB STANDARDS: THE OKD-MES POTENTIALS

This section will discuss the benefit exploitation of the approaches and standards proposed before in integrated industrial automation system. The general discussion of the possible benefits and risks of web standards exploitation in automation will be followed by practical outcomes of the application of such standards in manufacturing systems. Firstly, the most important features of the OKD-MES architecture developed in the EU project eScop will be outlined. In fact, some of the features related to the representation of manufacturing equipment and services for OKD-MES have recently been presented in [43]. Some smaller cases of use will be described below.

A. An overview of the OKD-MES architecture

EU Project eScop endeavors to create a framework for OKD-MES. In OKD-MES, web standards are employed at all system levels. The main goal of the project is to develop a system for providing a set of core components that facilitate the basic manufacturing functions. The core components must be able to accommodate new services, equipment, or processes that are introduced into the system. Due to its distributed nature, interoperability and community driven development are among the core features that OKD-MES provides by employing web standards. Figure 2 presents the OKD-MES architecture.

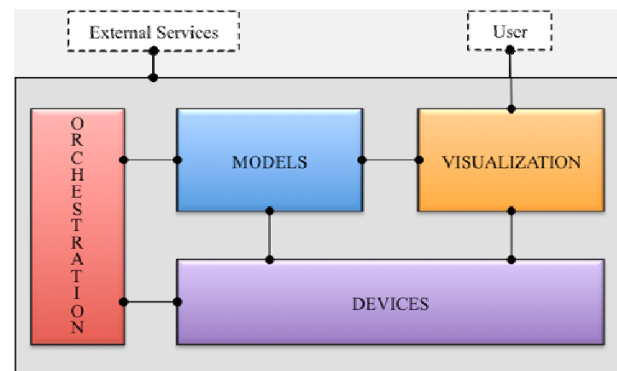


Figure 2. The OKD-MES architecture

The integration of different layers of industrial automation systems can be implemented as shown in the OKD-MES architecture. Each of the OKD-MES modules has been described separately in the preceding sections, which also present which web standards are employed. It should be noted that the orchestration module is referred to in Section IV as the system composition, which is required for orchestrating service executions. Then, following the order in which each layer appears in this article, the OKD-MES is formed by four core modules: *Devices*, *Models*, *Orchestration* and *Visualization*.

Firstly, the *Devices* module represents the equipment that is mostly located on factory shop floors. Devices are interconnected with any other module of the architecture. The connection with the *Models* and *Visualization* modules is needed for representing and accessing the status of the equipment during runtime. On the other hand, the connection with the *Orchestration* module is required for the execution of services, which are located in the industrial controllers.

Secondly, the *Models* module is formed by (1) the system descriptions and (2) the interfaces that permit access, update and retrieval of any model information. Similarly to the connections of the *Device* module, the *Models* module is interconnected with any other module in the architecture. The connection to the *Devices* module is needed for representation. In addition, the *Models* module is connected to the *Visualization* module because it must be capable of answering requests in the form of data that it will be needed for web UI management. On the other hand, as the hosted models also contain service descriptions of devices, the *Models* module must also be interconnected with the *Orchestration* module.

Thirdly, the *Orchestration* module is connected within the *Models* and *Devices* modules. As explained before, the reason for this connection is the need for the system composition to have access to all service descriptions. These descriptions include information about operations that will be invoked for executing processes in the system.

Finally, the *Visualization* module is connected to the *Models* and *Devices* modules. As explained previously, the reason for this connection is the need for access to system status information. This information permits the *Visualization* module to create and display web UIs.

As illustrated in Figure 2 above, the system also contemplates outside interaction with (a) external services and (b) users. External services are supposed to be interconnected all the architecture components so they can be consumed by any module, which requests any service which is not provided inside the system. On the other hand, users are only interconnected with the *Visualization* module because they interact within the system through the displayed web UI.

B. Exploitation of web standards: FASTory as the use case

This section will discuss the exploitation of the approaches and standards proposed before in an integrated industrial automation system. The general discussion of the possible benefits and risks of web standards exploitation in automation will be followed by practical outcomes of the application of such standards in a specific manufacturing system called FASTory (shown in Figure 3).

The FASTory is an automated mobile phone assembly line which has been used as a testbed in many EU research projects. This line is composed of a set of independent workstations. Each workstation is provided with safety equipment, a segment of the central transport system and a Selective Compliance Assembly Robot Arm (SCARA) robot. The FASTory line was retrofitted [44] within the addition of S1000 for controlling each component, which enabled the implementation of WSs.



Figure 3. The FASTory line in Tampere University of Technology

The transportation of the work pieces in the line is performed by a closed loop modular conveyor system. Among the robotic cells, there is a pallet buffer station, which loads and unloads the pallets to the conveyor line. Another robotic cell provides material deployment on the pallet. The remaining robotic cells provide the production operations. To reduce the line operating costs, the real assembly process is simulated by drawing the mobile components. Nevertheless, the complexity of the drawing process is similar to that of the assembly operations.

During the development of OKD-MES, FASTory is employed as a solution demonstrator. The FASTory line is controlled by WSs enabled devices, combining real-time PLC like control with the exposure of the RESTful WSs. Among the services presented, the control devices provide

subscription functionality for enabling event driven behavior of the solution. The configuration and status of the system are represented in the ontology. Then the access to the ontology is provided in the form of the RESTful services by the ontology management component. The information about available services from the devices is automatically discovered by the ontology management component and presented to the consumers. The system status and configuration are used by the orchestration component. The orchestration component executes the manufacturing process in the line according on the process description based on BPMN. Finally, the user interactions for the system in general are provided by the visualization component. Besides the four basic modules presented in Figure 2, the additional MES functions are provided by external services. Specifically, the important functions provided by the external services are dispatching and scheduling. The scheduling service provides the high level decision to which order to assign an incoming pallet. The dispatching function handles lower level decisions on which if any operations are to be performed in the station when the pallet reaches a workstation.

By applying the web standards on all levels, the system developed enables seamless integration of all of its components and other web based solutions, including ERP. Dynamically updating the representation of the system status, based on the availability and configuration of the shop-floor devices, enables easy re-configurability of the system hardware as well as the introduction of new products in manufacturing systems. The orchestration with decision support from MES functions is able to handle repetitive preconfigured modules in various configurations. Additionally, the managers, operators, customers, and other manufacturing stakeholders have direct controlled browser access to the data generated in the manufacturing process. Furthermore, the dynamic generation of screens makes it possible to visualize such basic system data. In conclusion, the FASTory demonstrator provides a significant level of adaptability to configuration changes on all levels of the system (representation, control, monitoring) in case of exploitation of predefined modules or their variations, within implementing web standards.

VII. EXEMPLIFYING DIFFERENT IMPLEMENTATIONS OF WEB STANDARDS IN THE FASTORY LINE

Many web standards can be used for a cross-layer implementation that integrates all levels from high web-based UIs to the industrial controllers located on the factory shop floor. The last section discussed how all levels are integrated within the implementation of the OKD-MES architecture. The FASTory line is one of the EU eScop project demonstrators used for testing the OKD-MES features. The objective of this section is to present how to implement some of the web standards described for automation control within real developments in the FASTory line. Such developments demonstrate the use of web standards at different layers of the architecture presented. It should be noted that due to the extension of the entire architecture, some examples are simplified versions of the

real development and that this section does not cover everything needed for implementing the OKD-MES architecture in the mobile phone assembly line described.

A. Service descriptions at device level

As described in Section II, Web Service enabled devices are connected to factory shop floor equipment for controlling the operations that are physically performed for assembly products, for example. Then operations can be invoked within web services, which are represented by web standards (e.g. REST) descriptions.

```
{
  "id": "services",
  "links": {
    "self": "http://192.168.1.1:3000/RTU/CNV1/services",
    "info": "http://192.168.1.1:3000/RTU/CNV1/services/info"
  },
  "class": "services",
  "children": {
    "TransZone12": {
      "id": "TransZone12",
      "links": {
        "self": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone12",
        "info": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone12/info",
        "notifs": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone12/notifs"
      },
      "class": "process",
      "children": {}
    },
    "TransZone23": {
      "id": "TransZone23",
      "links": {
        "self": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone23",
        "info": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone23/info",
        "notifs": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone23/notifs"
      },
      "class": "process",
      "children": {}
    },
    "TransZone35": {
      "id": "TransZone35",
      "links": {
        "self": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone35",
        "info": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone35/info",
        "notifs": "http://192.168.1.1:3000/RTU/CNV1/services/TransZone35/notifs"
      },
      "class": "process",
      "children": {}
    },
    "Z1": {
      "id": "Z1",
      "links": {
        "self": "http://192.168.1.1:3000/RTU/CNV1/services/Z1",
        "info": "http://192.168.1.1:3000/RTU/CNV1/services/Z1/info",
        "notifs": "http://192.168.1.1:3000/RTU/CNV1/services/Z1/notifs"
      },
      "class": "query",
      "children": {}
    },
    "Z2": {
      "id": "Z2",
      "links": {
        "self": "http://192.168.1.1:3000/RTU/CNV1/services/Z2",
        "info": "http://192.168.1.1:3000/RTU/CNV1/services/Z2/info",
        "notifs": "http://192.168.1.1:3000/RTU/CNV1/services/Z2/notifs"
      },
      "class": "query",
      "children": {}
    },
    "Z3": {
      "id": "Z3",
      "links": {
        "self": "http://192.168.1.1:3000/RTU/CNV1/services/Z3",
        "info": "http://192.168.1.1:3000/RTU/CNV1/services/Z3/info",
        "notifs": "http://192.168.1.1:3000/RTU/CNV1/services/Z3/notifs"
      },
      "class": "query",
      "children": {}
    },
    "Z5": {
      "id": "Z5",
      "links": {
        "self": "http://192.168.1.1:3000/RTU/CNV1/services/Z5",
        "info": "http://192.168.1.1:3000/RTU/CNV1/services/Z5/info",
        "notifs": "http://192.168.1.1:3000/RTU/CNV1/services/Z5/notifs"
      },
      "class": "query",
      "children": {}
    }
  }
}
```

Figure 4. Sample hypermedia description of conveyor services

As an example, Figure 4 shows hypermedia describing the services available for a conveyor in the FASTory line. The hypermedia presented is a sample of a response

returned by the services on invocation. The hypermedia includes the links to other services which can be used in the system. This example presents the services related to the manufacturing operations in one of the conveyors of the previously presented FASTory line.

The service description provided includes the information that the client needs to start operating the conveyor. Firstly, it lists all services available in Conveyor 1 of the production line. Each service has an ID (*id* fields) which uniquely identifies the service in the device. The services are categorized into *process* and *query* groups (value of attribute *class* in the description). The *process* group represents the RESTful resources for a technological process, while the *query* group resources are related to the programs in the controller making available information on system status. Finally, the *links* define the particular resources related to a service. The *notifs* link defines the notification endpoint, to which the clients may subscribe to in order to receive updates on service executions. The semantic description of the service can be obtained on *info* link. Finally, the *self* is an identifier of the service resource that can be invoked. Invocation of any among *TransZone* processes will trigger the logic in the controller required to transmit the container between predefined zones in the conveyor. Request in the form of a *Z* query will return the container ID if one is available in the particular zone of the conveyor. Using the description defined and sharing minimal taxonomy the client should be able to operate the web service enabled component of the production line.

B. Ontology model interaction at model level

As described in Section III, the knowledge of the system is described in a centralized ontology called the MSO, which includes different domain descriptions i.e., physical, technological, control, and visualization domains. The principal domain classes of the MSO are presented in Figure 5. It should be noted that a detailed description of the structure of the MSO and its role in the eScop project is presented in [49].



Figure 5. The four domain classes of the MSO

The MSO is implemented within standardized OWL. As OWL is an RDF-based language, it can be queried within SPARQL and SPARUL. Therefore, any level of the system wishing to extract or update information to the MSO located at the model level must send SPARQL or SPARUL queries.

As the MSO describes by default around a 1.5 thousand triples (ignoring the instances populated during system runtime) the RDF-based model cannot be presented in this article. Nevertheless, Figure 6 depicts a few classes and instances of the MSO with the objective of presenting product related information that can be monitored and manipulated within queries.

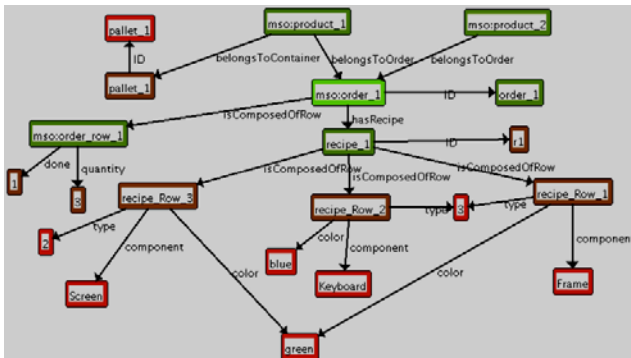


Figure 6. Container to recipe model snapshot

The diagram above is an ontology graph extracted from Olingvo, which is an ontology editor developed in Tampere University of Technology. Olingvo is a Graphical User Interface (GUI) application that permits, among other features, the design and navigation of ontologies, which are implemented in RDF-based languages.

Figure 6 illustrates how the MSO describes the relation between the container, product, the order it belongs to and the related recipe. Following this model, it is possible to obtain the list of descriptions for the required operations from the pallet ID. Pallet ID makes it possible to find a relation to the product hosted on the pallet. The product in turn belongs to the order. The order has a recipe, which consists of several recipe rows. These rows include a description of the operations to be executed. One or more SPARQL queries need to be executed to follow the defined linked data chain. The query collecting the information about a particular order and its relations to products and the recipe is defined in the first part (*Query 1*) of Figure 7.

Another example of ontology model information manipulation is the case of new order introduction. When the order is inserted through the visualization layer, related information should be added to the ontology. The SPARUL query defining the order properties and its relation to the recipe of products is required. An example of such a query is also found in the second part (*Query 2*) of Figure 7.

It should be noted that the queries shown in Figure 7 are proper executable queries written in compliance with the SPARQL query language specification.

```
#Query 1
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX mso: <http://www.escop-project.eu/MSO.owl#>
SELECT ?id ?quantity ?done
WHERE {
  ?o mso:ID "order_1"^^xsd:string.
  ?o mso:ID ?id.
  ?o mso:isComposedOfRow ?r.
  ?r mso:quantity ?quantity.
  ?r mso:done ?done.
}

#Query 2
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mso: <http://www.escop-project.eu/MSO.owl#>
INSERT {
  ?o rdf:type mso:Order.
  ?o mso:ID "order_1"^^xsd:string.
  ?o mso:hasRecipe ?res.
}
```

```
?o mso:isComposedOfRow ?r.
?r rdf:type mso:Order_Row.
?r mso:quantity 3.
?r mso:done 1.
?prd_0 rdf:type mso:Product.
?prd_0 mso:belongsToOrder ?o.
?prd_1 rdf:type mso:Product.
?prd_1 mso:belongsToOrder ?o.
}
WHERE {
  ?res mso:ID "r1"^^xsd:string.
  BIND(IRI("mso:order_1") AS ?o)
  BIND(IRI("mso:order_row_1") AS ?r)
  BIND(IRI("mso:product_1") AS ?prd_0)
  BIND(IRI("mso:product_2") AS ?prd_1)
}
```

Figure 7. Executable SPARQL queries for order management

C. Service composition at orchestration level

As described in Section IV, a system composition tool is used for interacting with the model, which describes the sequence of operations to be performed on the factory shop floor. Then the orchestration level is capable of invoking the services encapsulated in the industrial controllers with a specific order. The sequence of service invocations will be determined by both the process to be performed and the status of all resources linked to any process operation.

The FASTory production line is dynamic and reconfigurable. The introduction and modifications of the orders may take place at any stage of production. As a result, the effective orchestration solution should be capable of adapting to the changes in the system in runtime.

The one of the most complex challenges in such a dynamic orchestration is to make a decision if a particular workstation can and should execute some production operations on the product. Firstly, from the description of the device description, it is known that it is possible to obtain the ID of a pallet present in a certain zone of the conveyor. Secondly, as described in the previous subsection, it is possible to get a list of descriptions of production operations which have to be executed on the product based on the knowledge of pallet ID. Thirdly, using /info service from the PHL, the client can get a description of the service provided by a shop-floor device.

All the functionalities described above are available in the form of RESTful web services. The orchestrator, as a tool, should be able to execute the RESTful services and obtain data on the services available and required for a certain pallet in a certain workstation. Assuming the capability of the orchestrator to compare the sets of services it should be able to find an intersection between them. As in turn the intersection will be a set of RESTful services, which the orchestration module is able to execute, the whole cycle of detection-decision-action can be performed by the orchestrator tool.

If the comparison of the requirements and capabilities is not a trivial task, a dedicated service may be developed and used by the orchestrator to make a decision. This makes it possible to isolate and reuse certain blocks of logic, and as result to significantly reduce the complexity of the service composition. A possible composition for such approach is illustrated in Figure 8.

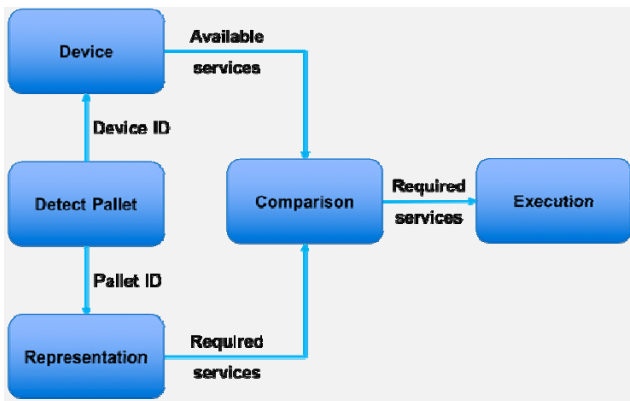


Figure 8. Orchestration diagram

D. Web UIs at visualization level

As described in Section V, a web UI can be displayed in the interaction of the visualization and model layers. Briefly, the interface retrieves the visualization domain descriptions that are included in the MSO via SPARQL queries. Such an interface may be used not only as a monitoring tool but also as a UI for low-level interaction with the production line. Thanks to the use of standards, the interface is accessible through any web browser, which means that the assembly line can be controlled from remote locations. Figure 9 shows an implemented web UI in the eScop project.

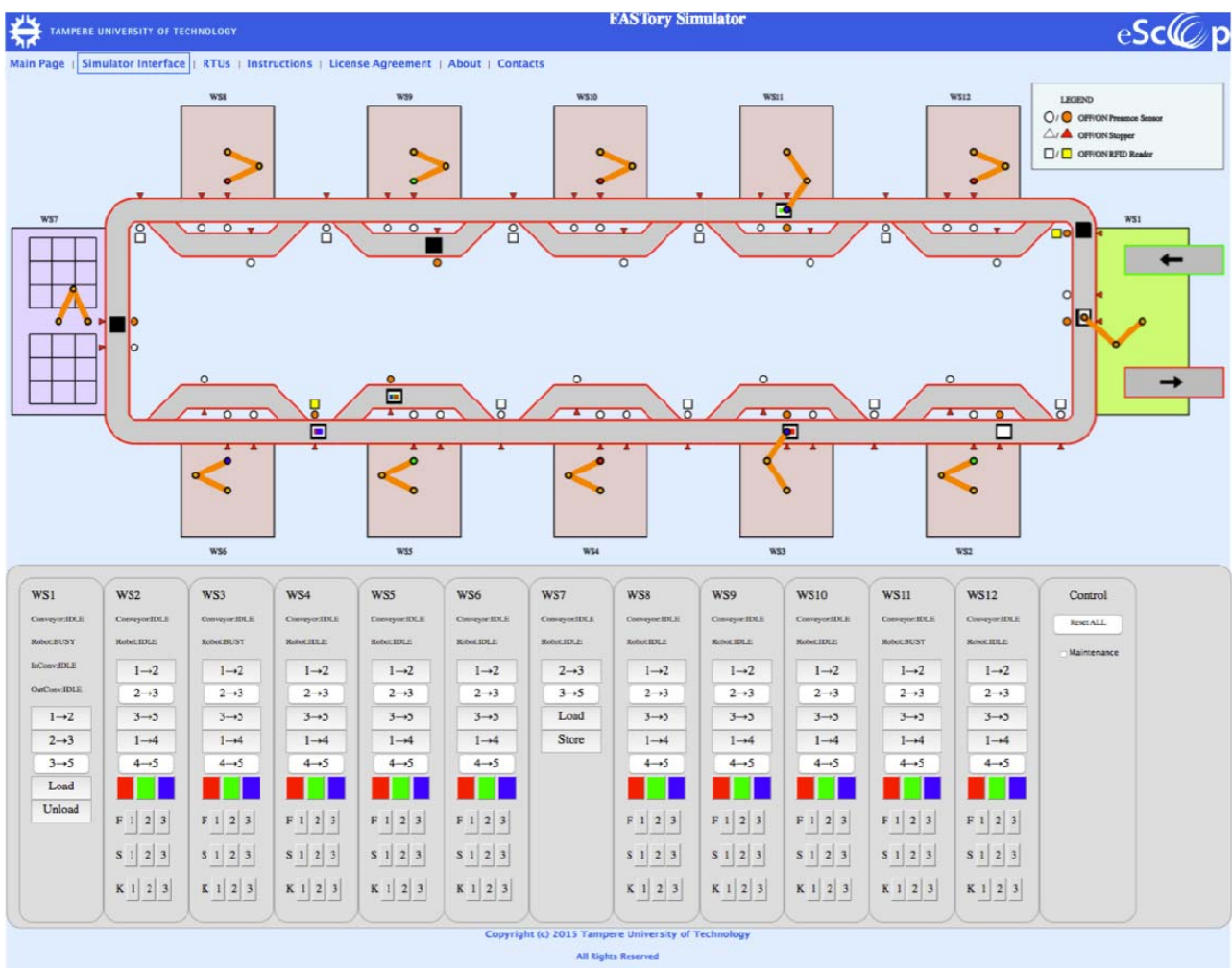


Figure 9. FASTory simulator visualization

The layout of the FASTory line is visualized in the web UI presented. Basic physical operations can be monitored in the interface because it displays e.g. the pallet movement when it is transferred between different pallet locations or the movement of the SCARA robots when operations on parts being transported in pallets are performed.

This web UI was implemented utilizing basic SVG and HTML for the representation of the components and JavaScript for their animation. The connection to the data sources and data sinks was developed using RESTful services. This set of technologies enables usability of the visualization provided across platforms and devices.

VIII. CONCLUSION

This paper first presented an overview of different web standards currently used in ongoing research work in the industrial automation domain. It then described the potential of web standards for automation control in manufacturing systems with results achieved on the ongoing EU project eScop.

One of the general benefits of exploiting open web standards is the availability of an existing ecosystem around it. The ecosystem was built in recent decades during expansion of the consumer internet. The communities of developers and users of the solutions based on the open web standards make it possible to reduce the cost of introducing the technology used, as gaining public acceptance is one of the most complex challenges in technology dissemination and exploitation. Beyond public acceptance the ecosystem drives the creation of support tools, frameworks, approaches, and other concepts accelerating the utilization of the technology. The extensive application of the open web standards in recent years has facilitated the development of networking hardware, user interfaces, programming languages, development, testing and deployment environments, as well as the approaches and techniques for their efficient exploitation.

The promising benefits of the solution proposed are discussed within a presented framework, namely OKD-MES. This particular application for the industrial domain demonstrates how the integration of different automation levels is possible using web standards and the synergy between cyber and physical systems, realizing the implementation of CPS in manufacturing systems.

The authors argue that the use of web standards, especially those presented here, which are directly supported in the web browsers, gives greater chances for applications developed following these standards to be compatible with future technologies. This may in turn reduce the effort needed to update or integrate the application into its evolving environment in the future. The potentials of web standards come from the scale and spread of web-based applications resulting in the development of mature proven standards that are used daily by billions of people. There are web standards for different aspects including the organization of basic communication (HTTP), service orientation (REST), the presentation of information (OWL) or even visualization (SVG). The examples presented show that all modules of the OKD-MES described can be interrelated within the implementation of web standards. In fact, it was discussed and demonstrated that having a centralized ontology model, to which different modules of the system can retrieve or manipulate information, the OKD-MES architecture is capable of effectively processing, controlling, and executing demands for assembling products in assembly lines, for example. The authors claim that such control and execution are complex because the production lines in dynamic systems constantly change their status. Nevertheless, the implementation of web standards at all levels of the OKD-MES architecture permits web-based control of systems,

which is possible by means of a cross-domain exchange of information.

ACKNOWLEDGMENT

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n° 332946 and from the Finnish Funding Agency for Technology and Innovation (TEKES), corresponding to the project shortly entitled eScop³, Embedded systems for service-based control of open manufacturing and process automation.

REFERENCES

- [1] E. COMPUTING, 'Cyber-physical systems', 2009.
- [2] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, 'Cyber-physical systems: the next computing revolution', in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 731–736.
- [3] J. M. Mendes, A. Rodrigues, P. Leitao, A. W. Colombo, and F. Restivo, 'Distributed Control Patterns using Device Profile for Web Services', in *Enterprise Distributed Object Computing Conference Workshops, 2008 12th*, 2008, pp. 353–360.
- [4] A. Cannata, M. Gerosa, and M. Taisch, 'SOCRADES: A framework for developing intelligent systems in manufacturing', in *IEEE International Conference on Industrial Engineering and Engineering Management, 2008. IEEM 2008*, 2008, pp. 1904–1908.
- [5] H. Bohn, A. Bobek, and F. Golatowski, 'SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains', in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006*, 2006, pp. 43–43.
- [6] A. L. Juha Puttonen, 'A Semantic Web Services-based approach for production systems control', *Adv. Eng. Inform.*, no. 3, pp. 285–299, 2010.
- [7] A. Lobov, F. U. Lopez, V. V. Herrera, J. Puttonen, and J. L. M. Lastra, 'Semantic Web Services framework for manufacturing industries', in *IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008*, 2009, pp. 2104–2108.
- [8] J. Puttonen, A. Lobov, and J. L. Martinez Lastra, 'Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services', *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2349–2359, Nov. 2013.
- [9] B. Ramis, L. Gonzalez, S. Iarovyi, A. Lobov, J. L. Martinez Lastra, V. Vyatkin, and W. Dai, 'Knowledge-based web service integration for industrial automation', in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 733–739.
- [10] N. A. N. Lee, L. E. G. Moctezuma, and J. L. M. Lastra, 'Visualization of Information in a Service-Oriented Production Control System', in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 2013, pp. 4422–4428.
- [11] S. Iarovyi, J. Garcia, and J. L. M. Lastra, 'An approach for OSGi and DPWS interoperability: Bridging enterprise application with shop-floor', in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 200–205.
- [12] S. Sucic, B. Bony, and L. Guise, 'Standards-compliant event-driven SOA for semantic-enabled smart grid automation: Evaluating IEC 61850 and DPWS integration', in *2012 IEEE International Conference on Industrial Technology (ICIT)*, 2012, pp. 403–408.
- [13] J. Intiaz and J. Jasperneite, 'Scalability of OPC-UA down to the chip level enables "Internet of Things"', in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 500–505.

³ <http://www.escop-project.eu/>

- [14] ‘OASIS Devices Profile for Web Services (DPWS)’. [Online]. Available: <http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01> [Accessed: 16-Jul-2015]
- [15] G. Marco, L. Fumagalli, A. Lobov, and J. L. Martinez Lastra, “Open Automation of Manufacturing Systems through Integration of Ontology and Web Services,” 2013, pp. 198–203.
- [16] S. Karnouskos and V. Somlev, ‘Performance assessment of integration in the cloud of things via web services’, in *2013 IEEE International Conference on Industrial Technology (ICIT)*, 2013, pp. 1988–1993.
- [17] V. Gilart-Iglesias, F. Maciá-Pérez, D. Marcos-Jorquera, and F. J. Mora-Gimeno, ‘Industrial Machines as a Service: Modelling industrial machinery processes’, in *2007 5th IEEE International Conference on Industrial Informatics*, 2007, vol. 2, pp. 737–742.
- [18] M. Lanthaler and C. Gütl, ‘Hydra: A Vocabulary for Hypermedia-Driven Web APIs.’, in *LDOW*, 2013.
- [19] ‘Hydra Core Vocabulary’. [Online]. Available: <http://www.w3.org/ns/hydra/spec/latest/core/>. [Accessed: 16-Jul-2015].
- [20] T. R. Gruber, ‘A translation approach to portable ontology specifications’, *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.
- [21] ‘OWL Web Ontology Language Semantics and Abstract Syntax’. [Online]. Available: <http://www.w3.org/TR/owl-semantics/>. [Accessed: 16-Jul-2015]
- [22] ‘RDF 1.1 Concepts and Abstract Syntax’. [Online]. Available: <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>. [Accessed: 16-Jul-2015].
- [23] ‘RDF Schema 1.1’. [Online]. Available: <http://www.w3.org/TR/rdf-schema/>. [Accessed: 03-Jul-2015].
- [24] ‘SPARQL Query Language for RDF’. [Online]. Available: <http://www.w3.org/TR/rdf-sparql-query/>. [Accessed: 05-Jul-2015].
- [25] I. Kollia, B. Glimm, and I. Horrocks, ‘SPARQL query answering over OWL ontologies’, in *The Semantic Web: Research and Applications*, Springer, 2011, pp. 382–396.
- [26] ‘OASIS Web Services Business Process Execution Language (WSBPEL) TC | OASIS’. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. [Accessed: 16-Mar-2015].
- [27] J. Puttonen, A. Lobov, and J. L. M. Lastra, ‘An application of BPEL for service orchestration in an industrial environment’, in *IEEE International Conference on Emerging Technologies and Factory Automation, 2008. ETFA 2008*, 2008, pp. 530–537.
- [28] ‘BPMN 2.0’. [Online]. Available: <http://www.omg.org/spec/BPMN/2.0/>. [Accessed: 10-May-2015].
- [29] ‘OWL-S: Semantic Markup for Web Services’. [Online]. Available: <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/#6>. [Accessed: 25-May-2015].
- [30] D. Redavid, L. Iannone, T. Payne, and G. Semeraro, ‘OWL-S Atomic Services Composition with SWRL Rules’, in *Foundations of Intelligent Systems*, A. An, S. Matwin, Z. W. Raś, and D. Ślęzak, Eds. Springer Berlin Heidelberg, 2008, pp. 605–611.
- [31] H. Li, L. Zhang, and R. Jiang, ‘Study of manufacturing cloud service matching algorithm based on OWL-S’, in *Control and Decision Conference (2014 CCDC), The 26th Chinese*, 2014, pp. 4155–4160.
- [32] L. Fumagalli, S. Pala, M. Garetti, and E. Negri, ‘Ontology-Based Modeling of Manufacturing and Logistics Systems for a New MES Architecture’, in *Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World*, B. Grabot, B. Vallespir, S. Gomes, A. Bouras, and D. Kiritsis, Eds. Springer Berlin Heidelberg, 2014, pp. 192–200.
- [33] ‘Scalable Vector Graphics (SVG) 1.1 (Second Edition)’. [Online]. Available: <http://www.w3.org/TR/SVG11/>. [Accessed: 16-Mar-2015].
- [34] ‘SPARQL 1.1 Update.’ [Online]. Available: <http://www.w3.org/TR/sparql11-update/>. [Accessed: 11-July-2015].
- [35] ‘Semantic Web Architecture - Introduction to ontologies and semantic web - tutorial.’ [Online]. Available: <http://obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html>. [Accessed: 26-Jul-2015].
- [36] ‘Sesame.’ [Online]. Available: <http://rdf4j.org/>. [Accessed: 26-Jul-2015].
- [37] ‘Apache Jena - Home.’ [Online]. Available: <https://jena.apache.org/>. [Accessed: 26-Jul-2015].
- [38] A. W. Colombo, S. Karnouskos, and T. Bangemann, ‘Towards the Next Generation of Industrial Cyber-Physical Systems,’ in *Industrial Cloud-Based Cyber-Physical Systems*, A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, and J. L. Lastra, Eds. Cham: Springer International Publishing, 2014, pp. 1–22.
- [39] B. B. Jay Lee, ‘A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,’ *SME Manuf. Lett.*, 2014.
- [40] S. Iarovy, José L. Martinez Lastra, Rodolfo Haber, and Raúl del Toro, ‘From artificial cognitive systems and open architectures to cognitive manufacturing systems’, in *2015 13th IEEE International Conference on Industrial Informatics (INDIN)*, 2015, pp. 1225–1232.
- [41] ‘SWRL: A Semantic Web Rule Language Combining OWL and RuleML’ [Online]. Available: <http://www.w3.org/Submission/SWRL/>. [Accessed: 28-July-2015].
- [42] B. Ramis Ferrer, B. Ahmad, A. Lobov, D. Vera, José L. Martinez Lastra, R. Harrison, ‘A knowledge-based solution for automatic mapping in component based automation systems’, in *2015 13th IEEE International Conference on Industrial Informatics (INDIN)*, 2015, pp. 262–268.
- [43] S. Iarovy, B. Ramis Ferrer, X. Xiangbin, A. Sampath, A. Lobov, José L. Martinez Lastra, ‘Representation of manufacturing equipment and services for OKD-MES: from service descriptions to ontology’, in *2015 13th IEEE International Conference on Industrial Informatics (INDIN)*, 2015, pp. 1069–1074.
- [44] L. Gonzalez, J. Jokinen, C. Postelnicu, J. M. Lastra, ‘Retrofitting a factory automation system to address market needs and societal changes,’ *Industrial Informatics (INDIN)*, 2012 10th IEEE International Conference, pp.413- 418, 2012.
- [45] B. Ramis Ferrer, B. Ahmad, A. Lobov, D. Vera, José L. Martinez Lastra, R. Harrison, ‘An approach for knowledge-driven product, process and resource mappings for assembly automation’, in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 1104-1109.
- [46] S. Iarovy and X. Xiangbin, ‘Developing Open Knowledge-Driven Manufacturing Execution System (2015)’. In: Strzelczak S., Balda P., Garetti M., Lobov A. (Eds.), *Open Knowledge Driven Manufacturing and Logistics - the eScop Approach*, Warsaw University of Technology Publishing House, pp. 295-310, 2015. ISBN 978-83-7814-440-3.
- [47] Strzelczak S., *Ontology-Aided Manufacturing and Logistics (2015)*. In: Strzelczak S., Balda P., Garetti M., Lobov A. (Eds.), *Open Knowledge Driven Manufacturing and Logistics - the eScop Approach*, Warsaw University of Technology Publishing House, pp. 23-50, 2015. ISBN 978-83-7814-440-3.
- [48] B. Ramis Ferrer, S. Iarovy, A. Lobov, José L. Martinez Lastra, ‘Potentials of web standards for automation control in manufacturing systems’, in *2015 UKSim-AMSS 9th IEEE European Modelling Symposium on Mathematical Modelling and Computer Simulation*. EMS 2015, 2015, pp. 359–366.
- [49] M. Garetti, E. Negri, L. Fumagalli, *Ontology-Based Modeling of Production and Logistics Systems (2015)*. In: Strzelczak S., Balda P., Garetti M., Lobov A. (Eds.), *Open Knowledge Driven Manufacturing and Logistics - the eScop Approach*, Warsaw University of Technology Publishing House, pp. 215-234, 2015. ISBN 978-83-7814-440-3.

V

**PRODUCT, PROCESS AND RESOURCE MODEL COUPLING
FOR KNOWLEDGE-DRIVEN ASSEMBLY AUTOMATION**

by

Borja Ramis Ferrer, Bilal Ahmad, Daniel Vera, Andrei Lobov, Robert Harrison,
José Luis Martínez Lastra, May 2016

at-Automatisierungstechnik, vol. 64, no. 3, pp. 231-243

Reproduced with permission.

Anwendungen

Borja Ramis Ferrer*, Bilal Ahmad, Daniel Vera, Andrei Lobov, Robert Harrison, and José Luis Martínez Lastra

Product, process and resource model coupling for knowledge-driven assembly automation

Kopplung von Produkt-, Prozess- und Ressourcenmodell für die wissensgetriebene Montageautomation

DOI 10.1515/auto-2015-0073

Received October 15, 2015; accepted December 30, 2015

Abstract: Accommodating frequent product changes in a short period of time is a challenging task due to limitations of the contemporary engineering approach to design, build and reconfigure automation systems. In particular, the growing quantity and diversity of manufacturing information, and the increasing need to exchange and reuse this information in an efficient way has become a bottleneck. To improve the engineering process, digital manufacturing and Product, Process and Resource (PPR) modelling are considered very promising to compress development time and engineering cost by enabling efficient design and reconfiguration of manufacturing resources. However, due to ineffective coupling of PPR data, design and reconfiguration of assembly systems are still challenging tasks due to the dependency on the knowledge and experience of engineers. This paper presents an approach for data models integration that can be employed for coupling the PPR domain models for matching the requirements of products for assembly automation. The approach presented in this paper can be used effectively to link data models from various engineering domains and engineering tools. For proof of concept, an example implementation of the approach for modelling and integration of PPR for a Festo test rig is presented as a case study.

Keywords: Knowledge driven systems, model coupling, ontology matching, assembly automation.

*Corresponding author: **Borja Ramis Ferrer**, Tampere University of Technology – FAST-Lab. P.O. 600, FI-33101, Tampere, Finland, e-mail: borja.ramisferrer@tut.fi

Andrei Lobov, José Luis Martínez Lastra: Tampere University of Technology – FAST-Lab. P.O. 600, FI-33101, Tampere, Finland

Bilal Ahmad, Daniel Vera, Robert Harrison: WMG, University of Warwick, Coventry, CV4 7AL, United Kingdom

Zusammenfassung: Die Berücksichtigung häufiger Produktänderungen innerhalb kurzer Zeiträume ist eine Herausforderung für Ingenieure aufgrund der Limitierungen aktueller Ansätze zum Entwurf, Aufbau und Rekonfiguration von Automatisierungssystemen. Insbesondere die steigende Menge und Vielfalt an Informationen aus der Produktion, die auf effiziente Art und Weise ausgetauscht und wiederverwendet werden sollen, führt zu einem Engpass. Zwei vielversprechende Ansätze um Entwicklungskosten und Entwicklungszeit zu reduzieren und gleichzeitig eine effiziente Entwicklung sowie die Rekonfiguration des Montagesystems zu ermöglichen, sind die digitale Fabrik und kombinierte Produkt-, Prozess- und Ressourcenmodelle (PPR). Aufgrund der bisher ineffizienten Verknüpfung der PPR-Daten ist die Entwicklung und die Rekonfiguration von Montageanlagen noch immer eine sehr anspruchsvolle Aufgabe aufgrund der starken Abhängigkeit vom Fachwissen und der Erfahrung der Ingenieure. Dieser Artikel präsentiert einen Ansatz zur Integration von Datenmodellen, der zur Kopplung der PPR-Domänenmodelle eingesetzt werden kann, um die Anforderungen der Produkte an die Montageautomatisierung zu erfüllen. Die hier vorgestellte Methode dient dazu, Datenmodelle und ihre Werkzeuge aus verschiedenen Ingenieursdisziplinen zu koppeln. Zur Evaluation wird eine beispielhafte Implementierung des Modell und Integrationsansatzes die Fallstudie eines FESTO-Prüfstandes präsentiert.

Schlüsselwörter: wissensbasierte Systeme, Koppelmodell, ontologische Übereinstimmung, Montageautomatisierung, Produktionsautomatisierung.

1 Introduction

The increasing market turbulence and the rise of new product technologies represent challenges and force industry to manufacture new product variants and adjust

production volume constantly. For example, the automotive industry is expected to accommodate changes frequently due to increased environmental concerns, technological advancements, changes in the market requirements etc. As a result, product assembly is becoming a challenging task. Industries have to frequently change manufacturing process plans and subsequently reconfigure/build assembly lines to manufacture new products [1].

Assembly systems within the automotive manufacturing sector are typically based on commonality approaches. Around 80% of existing off-the-shelf manufacturing resources remain unchanged or slightly modified to accommodate a new version of a product. However, due to the use of ad hoc engineering methods the knowledge utilised from previous similar projects is only a small percentage of the available knowledge [2]. It is reported in [3] that companies are often unaware of the extent of in-house knowledge and therefore spend significant time searching for relevant information from previous similar projects.

To accommodate product changes, the required manufacturing process and associated resource constraints are typically checked manually due to the lack of tools that can support analysis of how the product changes will affect manufacturing operations. As a result, feasibility studies associated with the manufacturing of new products on existing assembly lines, design of new process plans and required reconfiguration of manufacturing resources remain subjective. Such ad hoc approaches rely heavily on the knowledge and experience of engineers that result in prolonged lead-times and increased engineering costs [4].

To assist in the design, planning and reconfiguration of assembly systems, a number of digital modelling and simulation tools have been developed. These tools provide a number of benefits such as visualisation, verification and optimisation of the manufacturing systems before the physical build [5, 6]. However, manufacturing process and resource modelling is still a challenging and complex task. This is mainly due to the gaps that exist in the engineering process due to the reliance on ad hoc mechanisms for knowledge sharing, integration and reusability. Typically, product, process, and resource information and data sets exist within a given organisation, but they are not effectively coupled [7], which represents a big challenge in this field.

One promising approach to address this issue is the effective linking of the digital descriptions of the product attributes and requirements, to the characteristics of the manufacturing resources. The resulting knowledge-base derived from such integrated digital description can potentially increase the efficiency of the engineering pro-

cess. The focus of this paper is to improve PPR modelling of assembly automation systems by integrating virtual engineering tools with an ontology-based knowledge modelling system. To realise such modelling approach, a generic concept is presented that couples data models of PPR. The matching of different ontology domain models in a knowledge-driven system can potentially provide reusable knowledge-based PPR description to interconnect product attributes with related manufacturing resources.

In this article, the authors present a method for coupling PPR data models that can be achieved within the implementation of the rule-based approaches described in [4, 8] for mapping data of different concepts. In other words, this research work presents an application of previous work for multiple domain ontologies matching. It should be noted that previous work focused directly on mapping data in a unique and artificial model. In fact in a real scenario, these models can be generated from the domain specific engineering tools by corresponding domain experts. For this, the use of standard terminology and taxonomies is a promising approach, where standardisation can be driven by relevant communities holding the expert knowledge on a particular domain. Based on query rules, the presented methodology allows mapping between product, process and resource ontology models that will become modules of the PPR model. The ontology-based representation of engineering knowledge permits the dynamic modelling and mapping of products to required assembly processes and resources. Moreover, the use of standard languages for designing ontological models means that data can be accessed by third party applications, allowing retrieval and updating of the data model and thus makes the proposed approach extensible and scalable.

The remainder of this paper is organised as follows. Section 2 presents the background of the research work. Section 3 describes the approach for PPR model coupling within semantic descriptions. Section 4 presents an implementation of the approach with a case study based on a Festo test rig. Finally, Section 5 concludes the research work and discusses further tasks.

2 Background

The recent developments in manufacturing system engineering methods and software tools have focused on extending the data set based on which the design of production system is conducted; PLM (Product Lifecycle Management) and PDM (Product Data Management) have been

integrated with MPM (Manufacturing Processes Management) tools in order to achieve better integration between product design and manufacturing system design processes [9]. Paradigms such as design for assembly, fabrication, manufacture, etc. focus on integrating critical design parameters and constraints, early in the product design phase, in order to achieve higher level of concurrency and avoid deviations between product and production system design processes [10].

Within PLM systems, the concept of PPR extends the above mentioned concepts by integrating the notion of standardised manufacturing processes and resources, and the relation that exists between processes and resources [11]. In the domain of manufacturing, resources typically represent standardised modular machine units defined at various level of granularity (e.g. component, station, cell, zone, etc.). Each resource supports a specific process and may consist of sub-processes and sub-resources (e.g. a station-level transport system consisting of sub process and resources such as pre-stop, machine-stop and clamping operations).

A typical PPR-based engineering workflow consists of initiating the product design while the manufacturing process and resources required to manufacture this product are concurrently derived from the PPR information mapping. Existing PPR-based systems (e.g. Dassault Enovia, Siemens TeamCenter) typically implement PPR capabilities through the storage and integration (i.e. cross-referencing) of PPR data models. The increasing availability and usage of 3D based virtual modelling and simulation tools within PPR tool chain have significantly contributed to i) extending the data set integrated into PPR systems (mechanical/3D data, station/line layout, PLC control data in the form of re-usable Function Blocks, etc.) and ii) improving design support and validation functions through the use of dynamic 3D-based simulation environment that provide an intuitive view of the complete data set and therefore of the final systems design.

PPR systems rely heavily on the use of relational database systems to store information and to define relations between PPR data sets; relationships between data sets are based on physical or logical constraints and relations (e.g. processing time, product size/resource capacity, etc.), or on specific relations derived from experience collected during past projects (e.g. specific product variant/resource mapping). However, today's deployment of advanced ICT systems and the exponentially increasing quantity of digital data generated, represents a real challenge in not only managing the data (i.e. storage, categorisation, storage), but also in defining and maintaining the

relation between various engineering data sets and models.

The emergence of cyber-physical systems (CPS) integration and the expected industry evolution (Industry 4.0), Knowledge Representation (KR) is a promising solution to manage the referencing and integration of large and expanding data sets. This permits the reduction of data processing overload and, at the same time, the support of advanced reasoning and design capabilities provided by knowledge-based systems.

KR is the field of study concerned with describing facts in a human and machine-readable format, which makes machines capable of automating processes with the usage of semantic descriptions. Aside from formally defining KR, [12] presents several formalisms that can be used for implementing KR. During the last decade, the industrial automation research community has tended to implement ontologies for formal description of manufacturing systems [13]. However, it should be noted that Linked Data [14] is considered as a simple method for managing interrelated data in knowledge-based systems but with several challenges for reasoning [15]. On the other hand, other KR formalisms such as semantic rules or frames can be used for representing knowledge.

Ontologies contain formal descriptions, which can be queried and even reasoned. An ontological model is employed as a data storage that contains the actual status of systems, which are semantically described through axioms and relationships between instances of real world objects. In addition, the design of models based on standards permits different designers to convey and describe knowledge with same terminologies and taxonomies. Although there are numerous semantic languages that can be used for modelling manufacturing systems, Resource Definition Framework (RDF)¹ based languages are the dominant ones for model implementation in factory automation [16]. RDF is an XML-based language that belongs to the W3C standard recommendations.

RDF-based models, or RDF graphs, are sets of RDF triples that permit the semantic description of any domain. Syntactically, triples are structured within a relation of three terms: subject, predicate and object. Furthermore, the Ontology Web Language (OWL) [17] is an RDF-based language that extends RDF within the declaration of class descriptions (e.g. enumeration, property restriction, cardinality constraints) and axioms allowing the enrichment of model descriptions. Hence, ontologies not only link data but also add semantics (or meaning) to model resources.

¹ <http://www.w3.org/RDF/>

The information can be retrieved from models and modified with the use of standard query languages. This feature allows the data of graph accessible by other applications. In fact, any RDF-based graph datasets can be queried within SPARQL [18] and updated via its extension SPARQL Update (SPARUL). SPARQL permits the extraction of model data in the form of result sets or RDF graphs. SPARUL is an extension of SPARQL that includes a set of operations for updating, creating and removing data from RDF-based models.

The utilisation of semantic descriptions in factory automation domain permits the development of knowledge-driven solutions that are capable of managing and even orchestrating the execution of operations in contemporary production lines. For instance, [19] present useful applications of KR in the factory automation domain that allows scalability and reconfigurability of intelligent industrial automation systems. Implicit data of ontologies can be inferred by semantic reasoning engines to evaluate model descriptions and define new facts, which are not beforehand explicitly described [12]. This is a powerful characteristic of ontologies since the evaluation of models, inclusion at runtime, can result in new classifications or assertions of data. Furthermore, reasoning allows restructuring of models with statements which are not visible or deducible in the design phase. In addition, description of semantic rules using Semantic Web Rules Language (SWRL) [20] can be employed for mapping data from different domains [4, 8] to achieve model coupling.

Due to the number of domains involved in the engineering of manufacturing systems, the data generated during the design phase is heterogeneous and is defined in different domain models. Therefore, effective model coupling and ontology matching becomes an important research focus. [21] describes a conceptual methodology for merging heterogeneous data from upper levels. On the other hand, an example for lower level data coupling is shown in [22], which shows an implementation of algorithms for generating unique models as aggregation of sub-models. The evaluation of mentioned studies and previous work done in [4, 8] motivated and inspired the presented approach in this article.

3 Approach

The extensive use of virtual engineering tools in assembly automation for manufacturing process planning, optimisation and validation is promising. Virtual engineering tools have shifted work activity from serial to parallel and advanced information and communication infrastructure

has replaced paper-based processes. Despite this, launching a new product variant in the automotive industry remains a challenge. The selection of required manufacturing resources heavily rely on knowledge and experience of engineers. Although the PPR information exist separately, the lack of mappings between those data sets makes it difficult to identify whether the manufacturing process could be executed using available resources. To address this, an approach for automating the mappings of products, processes and resources using semantic descriptions for assembly automation systems is presented in this section.

The use of modular ontologies (each module belongs to a different domain) and standard terminology and taxonomies is a promising approach to achieving such PPR ontology coupling for assembly systems. Integration of modules in a unique representation creates a common source from which system information can be accessed. However, the main problem with using modular ontology is to define a manner of mapping data to be carried out by designers or with automatic approaches. In fact, the use of common terminology and taxonomies already presented in standards that are being implemented in the industry might avoid data naming convention issues. This requires employment of several standards for covering data managed in all levels of automation systems. For instance, the ISA 88 (used for defining the control batch processes) and the ISA 95 (used for describing the interface between enterprise) can be integrated [23] and used for describing related taxonomy in the PPR domain ontologies.

On the other hand, one of the problems that exists nowadays is the lack of a large infrastructure for sharing ontologies that could be reused for describing different industry domains. In fact, the common practice is to develop in-house models that are frequently described in other formats than ontologies e.g. XML, UML models, MS Office documents etc. Nevertheless, models can be transformed to ontologies automatically using existing approaches [24, 25] besides creating them manually and from the scratch. Moreover, APIs as e.g. Apache POI² permits the manipulation of MS documents with Java so they can be transformed e.g. to XML objects.

The focus of this paper is working with different domain ontologies and demonstrate how they can be coupled by mapping ontology elements, achieved by importing ontologies and combining them through a rule-based approach. The implementation of this approach allows merging modular ontologies and therefore create a link be-

² <https://poi.apache.org/>

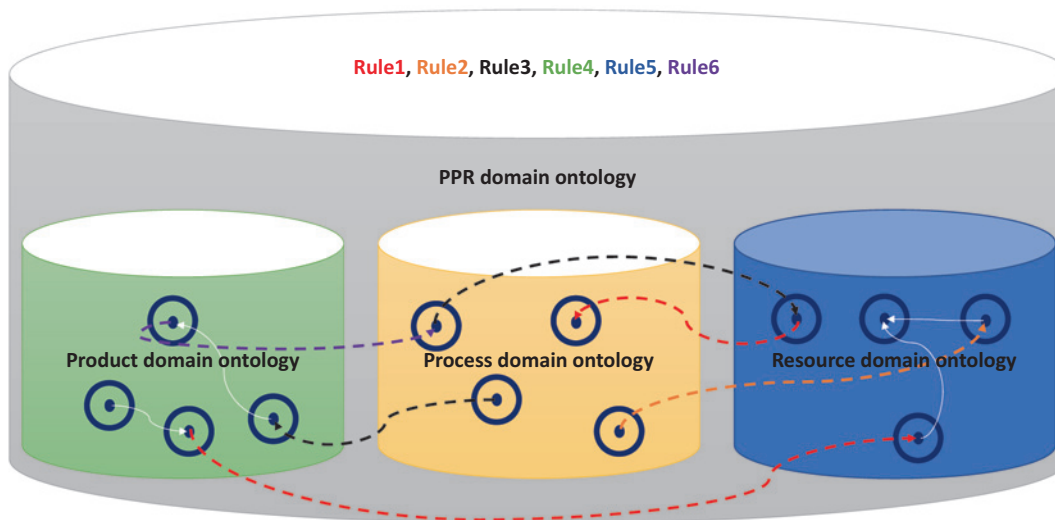


Figure 1: Representation of the resulting PPR domain ontology through different domain model coupling.

tween concepts and data covered by different domains that are frequently described with non-connected standards.

As this approach is intended for the case of having data in different ontologies, the mapping of data is in fact an interconnection between different domain models to address the model coupling of Product, Process and Resource domain ontologies. Figure 1 depicts that how instances belonging to different domain ontologies are matched with rules.

Taking into account the design of modular ontologies [26], different models can be merged to form a unique higher level ontology. This is achieved by using existing ontology editors (e.g. Protégé³ editor) that form a graph dataset containing all graphs of imported ontologies.

Nevertheless, merging ontologies does not result in automatic creation of link between data belonging to different domains because tools (or an import action) only create a model containing all instances of imported models. Thus, data coupling requires an extra process step. As shown in Figure 1, the model coupling is achieved through the definition of rules that link different data domains. A semantic reasoner that evaluates the model and stated rules infers the interrelation between different data models. It should be noted that creating the PPR ontology and data mapping semantic rules does not remove existing links of the individual domain ontologies, provided they do not create conflict with new rules. The important characteristic of achieving model coupling with the use of rules

and a reasoning engine is that the reasoner automatically validates the consistency of the model.

The approach for domain model coupling of PPR data used in this research consist of following four main steps:

1. Importing domain models in a unique ontology
2. Define a set of object properties that relate different domain concepts
3. Define a set of rules that can be understood by reasoning engines to infer model coupling
4. Automatically infer links between instances and validate model consistency

In the first step, a first graph data set, known as the PPR ontology, is created as an aggregation of all imported models. This new model contains all the elements of each ontology, and have its own Internationalised Resource Identifier (IRI). In the second step, additional object properties that interrelate different domain concepts are defined. These new properties are added in the PPR ontology IRI. Once object properties are added, a set of rules are defined in the next step. Finally, the semantic reasoning engines infer implicit data based on the rules defined in the previous step. The implementation of each step is further described in the case study.

The proposed approach presents a method for model coupling that can be used for e.g. mapping the required data needed for manufacturing products in assembly lines. The data mapping is achieved with an ontological model that allows not only the description of assembly lines, but also the representation of the processes that components can perform. The ontology that is formed by the three PPR domains and main classes used for the im-

³ <http://protege.stanford.edu/>

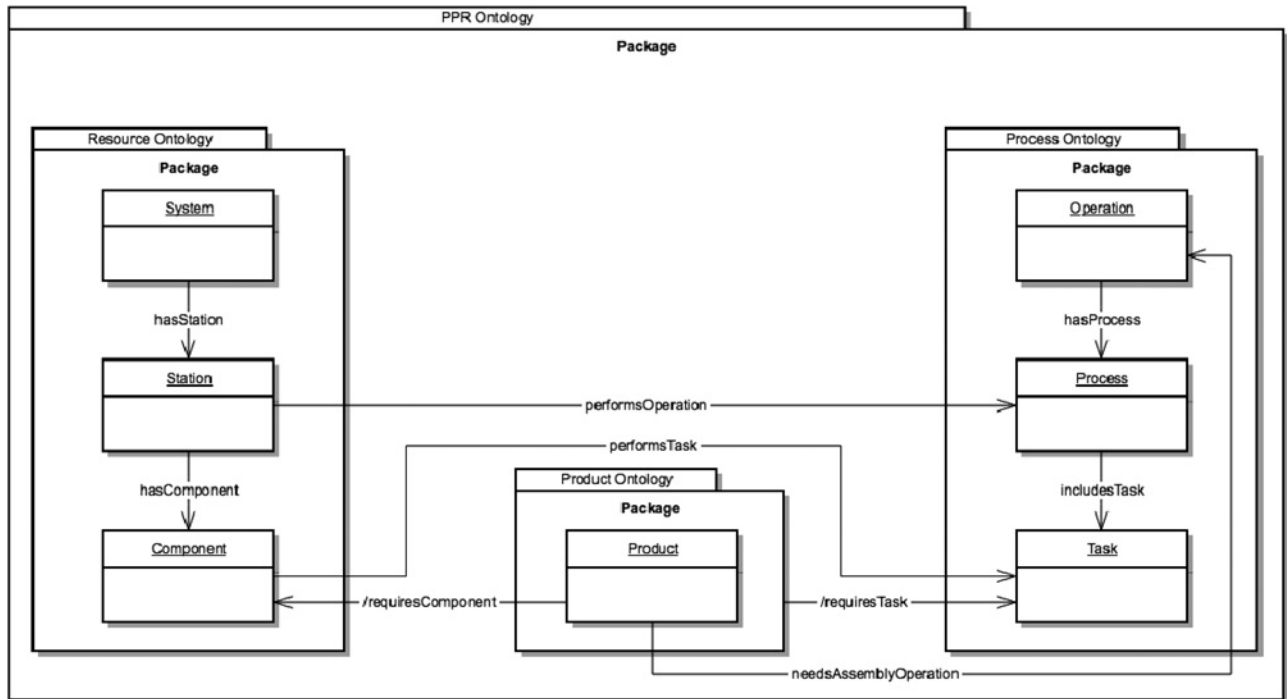


Figure 2: Processing Station ontology model.

plementation of the four step approach are shown in Figure 2 with a UML class diagram. The diagram depicts the hierarchical distribution of packages (ontologies) and included classes, which permit the description of processes and resources of assembly lines for manufacturing products.

Figure 2 depicts the resultant model. This diagram is used as a reference for required descriptions of the research work implementation. Although domain models are usually larger than the ones presented, the described classes are sufficient for demonstrating how the approach can be implemented. It should be noted that the model is reused from the research work presented in [4]. However, different concepts are now separated in different ontologies which are treated as modules (or higher classes) of the PPR ontology model.

4 Case study

This section describes the implementation details of the approach presented in previous section with the help of a case study. The case study is based on a station of a Festo Modular Production System, which presents a small-scale realistic industrial test bench widely used for teaching and research purpose. Figure 3 shows the Festo test rig, composed of four stations: *Distributing*, *Buffering*, *Processing*

and *Handling* stations. The *Processing Station* is selected for this use case.

4.1 Creating the PPR ontology model

The first step of the approach is accomplished by importing all the ontology models in a unique model. This is carried out using an ontology editor that permits importing models and saving the resulting ontology in a file. Protégé is an ontology editor that is used to implement the presented approach. Figure 4 shows the *Active Ontology* tab from the Protégé interface, in which the imported ontologies information is shown.

As it can be seen in Figure 4, imported ontologies retain the IRI which allow the differentiation between elements that belong to different domain models (even if the name of the instances is same). Therefore, the difference in IRI forms a basis for querying and defining rules in the model.

After the importation of the process ontology, the ontology contains three classes that allow the process related descriptions: *Operation*, *Process* and *Task*. *Operation* class is composed of operations that are performed in system stations. *Process* class is composed of processes linked to operations and *Task* is composed of tasks of processes.

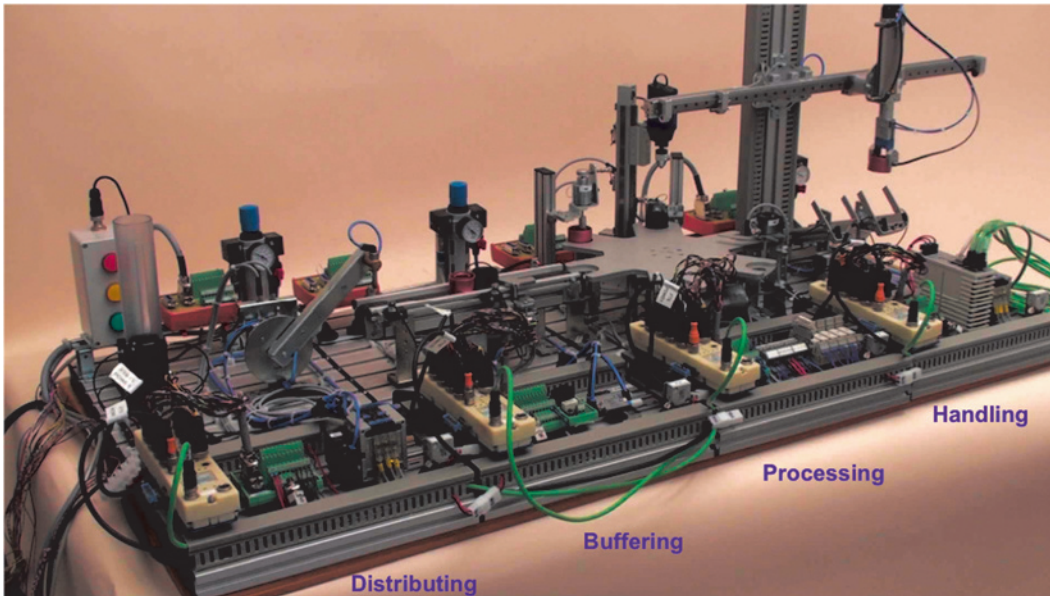


Figure 3: Festo test rig.

Ontology header:

Ontology IRI: <http://www.tut.fi/en/fast/PPR.owl>

Ontology Version IRI: e.g. <http://www.tut.fi/en/fast/PPR.owl/1.0.0>

Annotations:

- comment
- PPR ontology

Ontology metrics:

| Metrics | |
|-----------------------|----|
| Axiom | 93 |
| Logical axiom count | 56 |
| Class count | 7 |
| Object property count | 9 |
| Data property count | 0 |
| Individual count | 21 |
| DL expressivity | AL |

Imported ontologies:

Direct Imports:

- <<http://www.tut.fi/en/fast/processOntology.owl>>
 - pro
 - Ontology IRI: <<http://www.tut.fi/en/fast/processOntology.owl>>
 - Location: [/Users/ramisfer/Desktop/ProcessOnt.owl](#)
- <<http://www.tut.fi/en/fast/productOntology.owl>>
 - pro
 - Ontology IRI: <<http://www.tut.fi/en/fast/productOntology.owl>>
 - Location: [/Users/ramisfer/Desktop/ProductOnt.owl](#)
- <<http://www.tut.fi/en/fast/resourceOntology.owl>>
 - res
 - Ontology IRI: <<http://www.tut.fi/en/fast/resourceOntology.owl>>
 - Location: [/Users/ramisfer/Desktop/ResourceOnt.owl](#)

Indirect Imports:

To use the reasoner click Reasoner->Start reasoner Show Inferences

Figure 4: Imported ontologies in the PPR ontology.

A process is defined as a set of tasks performed in a certain sequence.

With the import of product ontology, *Product* class is imported and included in the PPR ontology which is used for collecting types of products that are manufactured in the assembly line. All required relations for PPR mappings

are shown in the class diagram of Figure 2, which is described in section 4.2. These relations are implemented as ontology object properties because they are used as a relationship between class instances.

Finally, after importing the resource ontology, three classes are included in the PPR ontology that allow the

Table 1: Instances Property Assertions included in the PPR ontology model.

| Instance (<i>type</i>) | Object property | Instance (<i>type</i>) |
|--|------------------------|---|
| product_1 (<i>Product</i>) | needsAssemblyOperation | op_01 (<i>Operation</i>) |
| product_2 (<i>Product</i>) | | op_02 (<i>Operation</i>) |
| festoSystem (<i>System</i>) | hasStation | processingStation (<i>Station</i>) |
| processingStation (<i>Station</i>) | hasComponent | rotaryInTableModule (<i>Component</i>) drillingModule (<i>Component</i>) testingModule (<i>Component</i>) clampingModule (<i>Component</i>) sortingGateModule (<i>Component</i>) |
| rotaryInTableModule (<i>Component</i>) | performsTask | advance_position (<i>Task</i>) |
| drillingModule (<i>Component</i>) | | drill (<i>Task</i>) |
| testingModule (<i>Component</i>) | | move_up (<i>Task</i>) move_down (<i>Task</i>) |
| clampingModule (<i>Component</i>) | | push_clamp (<i>Task</i>) release_clamp (<i>Task</i>) |
| sortingGateModule (<i>Component</i>) | | push_sort (<i>Task</i>) release_sort (<i>Task</i>) |
| op_01 (<i>Operation</i>) | hasProcess | process_1 (<i>Process</i>) |
| op_02 (<i>Operation</i>) | | process_2 (<i>Process</i>) |
| process_1 (<i>Process</i>) | includesTask | advance_position (<i>Task</i>) drill (<i>Task</i>) push_clamp (<i>Task</i>) move_down (<i>Task</i>) move_up (<i>Task</i>) release_clamp (<i>Task</i>) push_sort (<i>Task</i>) release_sort (<i>Task</i>) |
| process_2 (<i>Process</i>) | | advance_position (<i>Task</i>) push_sort (<i>Task</i>) release_sort (<i>Task</i>) |

physical concept description of the assembly line: *System*, *Station* and *Component*. *System* class is used for assembly line instances, representing an entire production line. *Station* contains different stations of a system which represent an assembly operation (such as piston stuffing). *Component* includes elements that are used for performing station operations (such as clamping).

4.2 Adding required object properties in the ontology model

As described previously, some object properties are already asserted when models are imported. This is because relationships described in models are not modified during the import process, which is just an aggregation of graphs

into the generated graph dataset. Hence, graphs that are linking instances with properties are not affected.

However, the new links that relate elements of different domain models are inserted once the unique ontology is being created. In this case study, the properties *performsOperation* and *performsTask* are added and linked between corresponding PPR ontology instances. The IRI of these properties is same as of the PPR ontology. These added properties in the second step of the approach are represented as UML direct association. Table 1 presents the relationships between instances of the model after the definition of the PPR ontology object properties. The names of instances are based on the information available in [27].

The two products which are added to the model (i.e. *product_1* and *product_2*) are shown in Table 1. These products are manufactured through slightly different man-

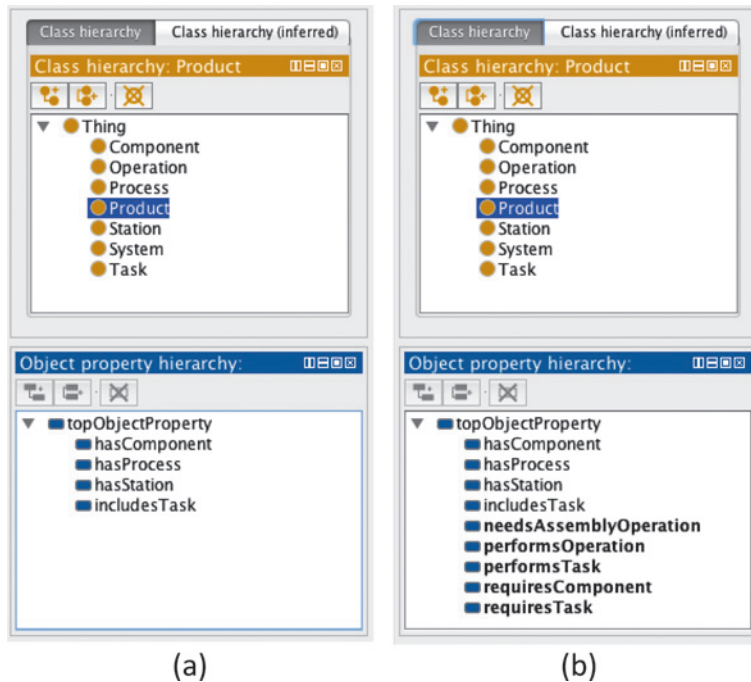


Figure 5: Class hierarchy and object property hierarchy (a) before and (b) after enriching the PPR ontology.

ufacturing processes. Thus, they are related to *op_01* and *op_02* that, in turn, are related to *process_1* and *process_2*. In this use case scenario, *product_1* is executed using *process_1*. The main steps constitute: advancing through the station, position testing, drilling and sorting. On the other hand, *product_2* does not require the manufacturing operations of processing station and is transported to the next station through the positions of the station, which are the steps of the *process_2*.

It should be noted that in addition to the shown object properties in Table 1, there are two more object properties represented in the UML ontology model: *requiresComponent* and *requiresTask*. These properties are used for determining which components and tasks are required to manufacture a product. However, they are represented within an UML derived association notation because such properties are derived from information included in the model. Basically, *requiresComponent* and *requiresTask* are not explicitly described because this approach implements such relation within semantic rules, as described in following sub-sections. Figure 5 shows the hierarchy of classes and object properties (a) before and (b) after the enrichment of object properties in the PPR ontology.

As it can be seen in Protégé views, the class hierarchy remains same in (a) and (b); the object property hierarchy changes because (b) contains the object properties shown in Figure 2 that are used for interrelating different domain models.

Figure 6 depicts that *processingStation* resource domain instance is linked to two domain instances. First, it

is still linked with *hasComponent* object property to other resource domain instances (from the imported resource ontology model) and, afterwards, it is linked with different process domain instances (*op_01* and *op_02*) with *performsOperation* object property.

4.3 Adding required SWRL rules and coupling data in the ontology model

In addition to presented classes and object properties that are imported from different domain models and object properties that are defined in the PPR ontology, the automatic coupling of data is achieved with SWRL rules. This section presents two rules that allow the mapping between product, component and tasks of assembly lines. As the rules are expressed through SWRL they are suitable for any RDF-based ontology model. Table 2 shows the two SWRL rules that are defined for this use case.

The first SWRL rule maps products with components by taking into account *performsOperation*, *includesTask*, *hasProcess*, *needsAssemblyOperation*, *hasComponent* and *performsTask* object properties and the classes that are linked to several model properties (i.e. *Component*, *Task*, *Operation*, *Station* and *Product*). The second SWRL rule maps products with tasks by taking into account *includesTask*, *hasProcess* and *needsAssemblyOperation* object properties and the classes that are linked to such properties (i.e. *Task*, *Process*, *Operation* and *Product*).

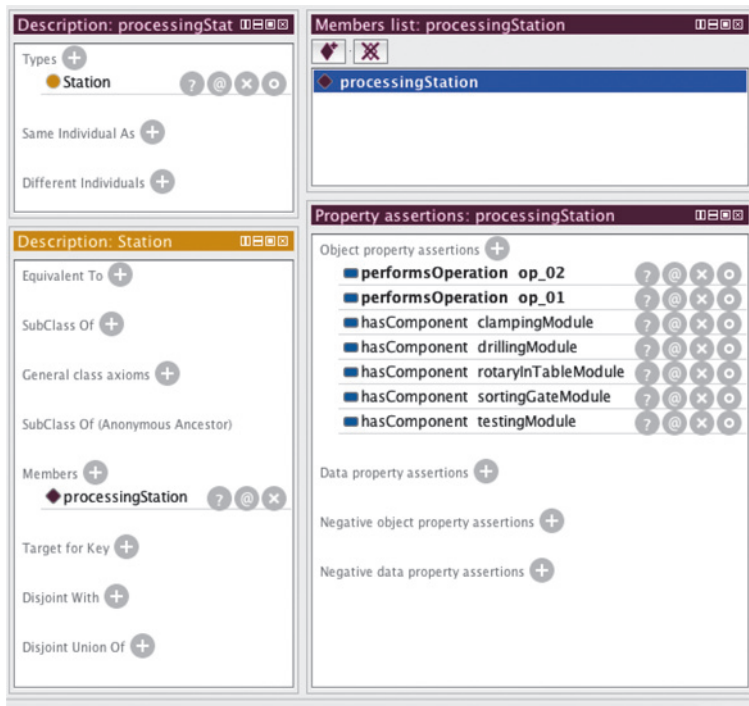


Figure 6: Object property assertions of *processingStation* in the PPR ontology.

Table 2: SWRL rules for automatic data coupling.

| Rule 1: Linking products with required components | Rule 2: Linking products with required tasks |
|--|--|
| $\text{Component}(?c) \wedge \text{Task}(?t) \wedge \text{Operation}(?o) \wedge \text{Station}(?s) \wedge$ $\text{Product}(?pr) \wedge \text{performsOperation}(?s, ?o) \wedge$ $\text{includesTask}(?p, ?t) \wedge \text{hasProcess}(?o, ?p) \wedge$ $\text{needsAssemblyOperation}(?pr, ?o) \wedge \text{hasComponent}(?s, ?c) \wedge$ $\text{performsTask}(?c, ?t) \rightarrow \text{requiresComponent}(?pr, ?c)$ | $\text{Task}(?t) \wedge \text{Process}(?p) \wedge \text{Operation}(?o) \wedge$ $\text{Product}(?pr) \wedge \text{includesTask}(?p, ?t) \wedge$ $\text{hasProcess}(?o, ?p) \wedge \text{needsAssemblyOperation}(?pr, ?o)$ $\rightarrow \text{requiresTask}(?pr, ?t)$ |

4.4 Automatic data mapping within semantic reasoning engines

Using the presented SWRL rules, products can be inter-related with tasks and components. Thus, through SWRL rules and reasoning engines, the relation between products, tasks and components are inferred. It should be noted that these rules could be merged in one rule, which would give the same mappings. However, the implementation of two independent rules allows identifying which classes and properties are required for each type of mapping. It should be noted that for this approach implementation, Pellet reasoner [28] has been used as the reasoning engine for obtaining the mappings. Thus, before SWRL rules are executed, Pellet must be started.

Once the reasoning engine runs the final graph, dataset is generated because it includes the inferred data. Figure 7 shows the inferred data mappings of *product_1* (visible in property assertions tab) created with the help of SWRL rules and the evaluation done by the semantic

reasoner. The mappings demonstrate that the data of all models is successfully coupled because *product_1* (product domain) is linked to *op_01* and to a set of tasks (process domain) and, at the same time, to a set of components (resource domain).

The presented implementation demonstrates successful coupling of models by (1) merging different domain models, (2) assertion of object properties between class instances and (3) automatic mappings through semantic reasoning, achieved through SWRL rules and model evaluation. If desired, the mappings of products and their requirements can be extracted from the model. As the model has been implemented in OWL (RDF-based) syntax SPARQL queries can be used for monitoring any information related to the model.

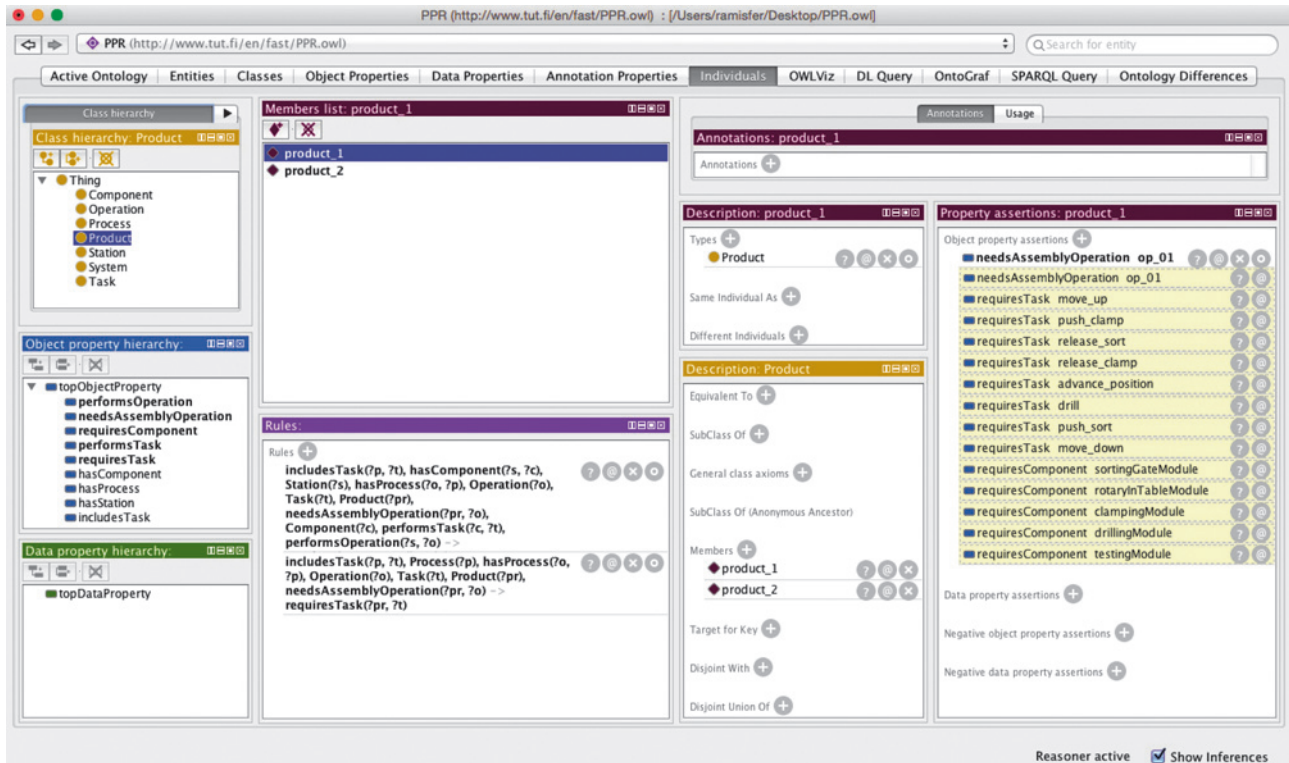


Figure 7: Inferred mappings in the PPR ontology.

5 Conclusion

This paper presents the application of a knowledge-based PPR mapping approach that can support dynamic configuration of assembly systems because the model interrelations may change when populating the ontology with new instances. Such approach relies on capturing and formalising in house engineering knowledge that link Product, Process and Resource in order to reduce system development and reconfiguration time. To illustrate the concept, generic models of products, processes and manufacturing resource components are created for a Festo test rig with two product variants scenario. The authors believe that such approach can be extended to allow reconfiguration of existing manufacturing systems in order to include more complex products (e.g. product structure and complexity, product variants) querying the available facilities to manufacture a new product variant. From the results of the case study, it can be concluded that ontological mapping of product, process and resource data and their representation within virtual modelling and simulation environment can, to a certain extent, allow the automation of the process that consists in instantiating a specific system from a library of PPR components and therefore to accelerate the configuration of manufacturing system for a specific production requirement.

The approach is implemented for a table-size test rig with a small number of components. Preliminary results are the automatic generation of a list of resource components based on Process and Product requirement input. Such results suggest that further development of ontology based system can potentially be used to achieve automatic configuration of manufacturing resources and contribute to the self-configuration paradigm introduced by Industry 4.0. An important aspect of the presented approach is the mapping between models based on query rules. The use of best practices to create these rules and serve as a reference to develop and advance the expertise is an important research area which needs to be addressed to enable wide application of such approach. A wider application of the proposed approach is required in order to assess the robustness, practicality and efficiency in terms of timesaving of this approach. Further research will focus on using semantics to characterise data collect from IoT devices deployed on the real system (e.g. PLC, energy monitors, sensors, production data) in order to further extend the data set based on which ontology and case based reasoning can be conducted.

Acknowledgement: The authors gratefully acknowledge the support of the UK EPSRC through the Knowledge-Driven Configurable Manufacturing (KDCM) research

project under the Flexible and Reconfigurable Manufacturing Initiative and the graduate school funding of Tampere University of Technology in carrying out this work.

References

- Kong, X., B. Ahmad, R. Harrison, Y. Park, and L.J. Lee. *Direct deployment of component-based automation systems*. In *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*. 2012.
- Raza, M.B., *A knowledge based approach to integration of products, processes and reconfigurable automation resources*. 2012, Loughborough University.
- Murray, S., *Knowledge management in manufacturing*, in *The Economist*. 2007.
- Ferrer, B.R., B. Ahmad, A. Lobov, D.A. Vera, J.L.M. Lastra, and R. Harrison. *An approach for knowledge-driven product, process and resource mappings for assembly automation*. In *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*. 2015. IEEE.
- Kong, X., *An approach to open virtual commissioning for component-based automation*, *Doctoral Thesis*, in *Wolfson School of Mechanical and Manufacturing Engineering*. 2013, Loughborough University: Loughborough, Leicestershire.
- Falkman, P., F. Westman, and C. Modig. *Verification of operation sequences in process simulate by connecting a formal verification tool*. In *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*. 2009.
- Raza, M., B. and R. Harrison, *Design, development & implementation of ontological knowledge based system for automotive assembly lines*. *International Journal of Data Mining & Knowledge Management Process*, 2011. 1(5): p. 21–40.
- Ferrer, B.R., B. Ahmad, A. Lobov, D. Vera, J.L.M. Lastra, and R. Harrison. *A knowledge-based solution for automatic mapping in component based automation systems*. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*. 2015. IEEE.
- Haq, I., T. Masood, B. Ahmad, R. Harrison, B. Raza, and R. Monfared. *Product to process lifecycle management in assembly automation systems*. In *7th CIRP international conference on digital enterprise technology*. 2011. Athens, Greece.
- Urrutia, U.A., P. Webb, and M. Summers. *Analysis of Design for X Methodologies for Complex Assembly Processes: A Literature Review*. In *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. 2014. American Society of Mechanical Engineers.
- Choi, S.S., T.H. Yoon, and S.D. Noh, *XML-based neutral file and PLM integrator for PPR information exchange between heterogeneous PLM systems*. *International Journal of Computer Integrated Manufacturing*, 2010. 23(3): p. 216–228.
- Brachman, R. and H. Levesque, *Knowledge representation and reasoning*. 2004: Elsevier.
- Lastra, J.L.M., I.M. Delamer, and F. Ubis, *Domain ontologies for reasoning machines in factory automation*. 2010: ISA.
- Bizer, C., T. Heath, and T. Berners-Lee, *Linked data-the story so far*. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, 2009: p. 205–227.
- Polleres, A., A. Hogan, R. Delbru, and J. Umbrich, *RDFS and OWL reasoning for linked data*, in *Reasoning Web. Semantic Technologies for Intelligent Data Access*. 2013, Springer. p. 91–149.
- Kalibatiene, D. and O. Vasilecas, *Survey on ontology languages*, in *Perspectives in Business Informatics Research*. 2011, Springer. p. 124–141.
- OWL Web Ontology Language Semantics and Abstract Syntax*. 2015 [cited 2015 01 March]; Available from: <http://www.w3.org/TR/owl-semantics>.
- SPARQL Query Language for RDF*. 2015 [cited 2015 01 March]; Available from: <http://www.w3.org/TR/rdf-sparql-query>.
- Ramis, B., L. Gonzalez, S. Iarovy, A. Lobov, J.L. Martinez Lastra, V. Vyatkin, and W. Dai. *Knowledge-based web service integration for industrial automation*. In *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*. 2014. IEEE.
- SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. [cited 2015 12 January]; Available from: <http://www.w3.org/Submission/SWRL/>.
- Yap, C.E. and M.H. Kim. *Instance-based ontology matching with rough set features selection*. In *IT Convergence and Security (ICITCS), 2013 International Conference on*. 2013. IEEE.
- Nath, R.P.D., H. Seddiqui, and M. Aono. *Resolving scalability issue to ontology instance matching in semantic web*. In *Computer and Information Technology (ICCIT), 2012 15th International Conference on*. 2012. IEEE.
- Vegetti, M. and G. Henning, *ISA-88 Formalization. A Step Towards its Integration with the ISA-95 Standard*. *FOMI'2014 Formal Ontologies meet Industry*: p. 17.
- Bosch, T. and B. Mathiak. *XSLT transformation generating OWL ontologies automatically based on XML Schemas*. In *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*. 2011. IEEE.
- Belghiat, A. and M. Bourahla. *Transformation of UML models towards OWL ontologies*. In *Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on*. 2012. IEEE.
- Ensan, F. and W. Du, *A knowledge encapsulation approach to ontology modularization*. *Knowledge and information systems*, 2011. 26(2): p. 249–283.
- Festo Didactic. *Processing station: Purely electrical*. 2015 [cited 2015 12 May]; Available from: <http://www.festo-didactic.com/int-en/learning-systems/mps-the-modular-production-system/stations/processing-station-purely-electrical.htm>.
- Sirin, E., B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz, *Pellet: A practical owl-dl reasoner*. *Web Semantics: science, services and agents on the World Wide Web*, 2007. 5(2): p. 51–53.

Bionotes

M. Sc. Borja Ramis Ferrer

Tampere University of Technology – FAST-Lab. P.O. 600, FI-33101, Tampere, Finland
borja.ramisferrer@tut.fi

Borja Ramis Ferrer received the Ingeniero Técnico Industrial degree in electrical engineering from the Universidad de las Islas Baleares, Islas Baleares, Spain, in 2011 and the M.Sc. degree (with Distinction) in Factory Automation from Tampere University of Technology, Tampere, Finland, in 2013. He is currently working towards his Dr. Tech degree at Tampere University of Technology and is President's Doctoral School fellow. His research interests include the deployment of knowledge-based and cyber-physical systems in factory automation.

Dr. Bilal Ahmad

WMG, University of Warwick, Coventry, CV4 7AL, United Kingdom
b.ahmad@warwick.ac.uk

Bilal Ahmad is a Research Fellow at WMG, University of Warwick. He received his MSc in Mechatronics and PhD in Automation Systems from Loughborough University. He specialises in the area of industrial automation. He has worked on a number of UK and EU engineering research projects in collaboration with automotive manufacturers, machine builders and control vendors to develop tools and methods to support lifecycle of automation systems.

Dr. Daniel Vera

WMG, University of Warwick, Coventry, CV4 7AL, United Kingdom
d.a.vera@warwick.ac.uk

Daniel Vera has been working in the domain of manufacturing engineering for over ten years. His research interests are focused on various aspects of manufacturing from the modelling, analysis and optimisation of engineering processes to the design and development of 3D-based virtual engineering tools for supporting the manufacturing system lifecycle, which formed the focus of his PhD thesis. Dr Vera has been involved in numerous UK and European projects as a Research Associate at Loughborough University and now a Research Fellow at the University of Warwick. He is currently taking a leading role in the commercialisation of new automation systems and methods.

Dr. Andrei Lobov

Tampere University of Technology – FAST-Lab. P.O. 600, FI-33101, Tampere, Finland
andrei.lobov@tut.fi

Andrei Lobov is lecturing at the Tampere University of Technology. He received his PhD in Formal Methods of Factory Automation, in 2008. He holds BSc in Computer and System Engineering from the Tallinn University of Technology (2001). Then, he continued his education at the Tampere University of Technology and received MSc in Automation Engineering (2004). His research interests include development of architectures, methodologies and technologies for manufacturing systems. He is a technical coordinator of the eScop project.

Prof. Dr. Robert Harrison

WMG, University of Warwick, Coventry, CV4 7AL, United Kingdom
robert.harrison@warwick.ac.uk

Robert Harrison is Professor of Automation Systems at WMG, University of Warwick and has been principal investigator on more than 35 EU, UK government, and commercial R&D projects related to manufacturing automation with current projects focusing on lifecycle engineering and virtual commissioning, control deployment and augmented reality. He led the UK research related to Ford's Technology Cycle Plan for powertrain manufacturing automation and was recipient of a RAEng Global Research Award to study "Lifecycle Engineering of Modular Reconfigurable Manufacturing Automation".

Prof. Dr. José L. Martínez Lastra

Tampere University of Technology – FAST-Lab. P.O. 600, FI-33101, Tampere, Finland
jose.lastra@tut.fi

José L. Martínez Lastra joined Tampere University of Technology in 1997, and became University Full Professor in 2006. His research interest is on applying Information and Communication Technologies to the fields of Factory Automation and Industrial Systems. Prof. Lastra leads the Factory Automation Systems and Technologies Laboratory with the ultimate goal of seamlessly integrating the knowledge of humans and machines. Prof. Lastra has co/authored over 250 scientific papers and holds a number of patents in the field of Industrial Informatics and Automation. He serves as Associate Editor of the IEEE Transactions on Industrial Informatics, and he is a Technical Editor of the IEEE/ASME Transactions on Mechatronics.

VI

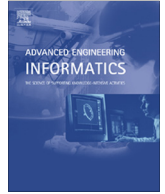
PRIVATE LOCAL AUTOMATION CLOUDS BUILT BY CPS: POTENTIAL AND CHALLENGES FOR DISTRIBUTED REASONING

by

Borja Ramis Ferrer, Jose Luis Martinez Lastra, April 2017

Advanced Engineering Informatics vol. 32, pp. 113-125

Reproduced with permission.



Private local automation clouds built by CPS: Potential and challenges for distributed reasoning



Borja Ramis Ferrer*, Jose Luis Martinez Lastra

Tampere University of Technology, Fast-Lab., P.O. Box 600, FIN-33101 Tampere, Finland

ARTICLE INFO

Article history:

Received 19 March 2016

Received in revised form 23 January 2017

Accepted 25 January 2017

Keywords:

Industrial automation
Cyber-physical systems
Local cloud
Ontology
Distributed reasoning

ABSTRACT

The employment of cyber-physical systems allows the control of processes in modern production lines. On the other hand, several research works have recently presented how ontology-based knowledge representation can be a suitable method for modelling industrial systems. However, system models are located far away from where the data is generated which adds complexity for cross-domain communications and resource management. Current embedded devices can encapsulate ontological models that can be accessed as local resources. This article presents the integration of interconnected devices as the computational nodes of a cloud which is private and local. In this way, functionalities, such as knowledge management and process control can be performed closer to the industrial equipment. Moreover, this research work discusses the potential and challenges for performing distributed reasoning in the private local automation cloud. In addition, the article describes main aspects of the system architecture and the behaviour of the networked embedded devices in the cloud. The research work results will be used as a high-level roadmap for further system implementation.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

The industrial automation has evolved passing through several generations of embedded devices, which are used for process control at the factory floor. This evolution has resulted in the development of modern Web Service (WS) enabled controllers, such as the S1000 from Inico Technologies,¹ which allows the implementation of the Service Oriented Architecture (SOA) paradigm [1] in modern production lines, as described in [2].

During last decades, industrial control systems have employed different standards for implementing multiple solutions. For example, the IEC-61131-3 [3] and the IEC-61499 [4] can be used for modelling and information exchange in distributed control systems. Actually, one of the main challenges in modern industrial control systems is the management of a large amount of data that is exchanged between system components. One solution is the employment of IEC-61499 for modelling automation systems within Function Blocks (FB) networks [5].

In addition to standards, the industrial domain has included the use of Knowledge Representation (KR) for implementing

knowledge-based systems. These systems are beneficial in the industrial field for:

- overcoming the interoperability issues between system modules within data mappings [6,7]
- monitoring system status at runtime [8], and
- implementing knowledge-driven approaches, in which ontology-based KR enables the modelling and decision-making support in systems [9,10]

Recent research in the industrial automation domain present system architectures in which the management and location of the knowledge is performed away from the shop floor physical machines. As an example, the eScop project² (Embedded systems Service-based Control for Open manufacturing Project) proposes a possible architecture in [11]. The implementation of the project architecture and principles is described in [12]. Similar to the ISA-95 standard automation pyramid,³ the eScop architecture presents that machines reside at the lowest level (i.e. device or shop floor level) of a hierarchical multi-layer structure. Then, the Industrial machinery is isolated from the process of information, which is performed by production management level systems; e.g., the

* Corresponding author.

E-mail addresses: borja.ramisferrer@tut.fi (B. Ramis Ferrer), jose.lastra@tut.fi (J.L. Martinez Lastra).

¹ <http://www.inicotech.com/>.

² <http://www.escop-project.eu/>.

³ <http://isa-95.com/>.

Manufacturing Execution System (MES). Once the information is processed, the upper layers orchestrate the invocation of operations that are described in Remote Terminal Units (RTUs), which are embedded devices physically connected to the machinery. Therefore, RTUs facilitate the implementation of the Cyber-Physical Systems (CPS) concept, which is defined as the integration of computation (cyber systems) and controlled equipment (physical systems) [13]. Basically, RTUs act as a gateway for allowing cross-layer communication between higher level systems and the industrial equipment which is located at the device level. A representation of the pyramidal structure is illustrated in Fig. 1.

As depicted in the above diagram, industrial enterprise systems are distributed in different levels; i.e. factory shop floor, supervisory control, production management and business management. RTUs (and also PLCs) are the interface between the device level and control level. The flow of information can be (i) horizontal, i.e. exchange of information between systems residing on same level or vertical (ii) cross-layer communication between systems located at different levels.

This research is motivated by the fact that the computational resources of the embedded devices used in the industrial domain have increased. Therefore, now it is possible to add more functionality that currently reside in higher levels of the automation pyramid into the devices which are located at the factory shop floor level. Moreover, this research work is motivated for the reduction of the high cost of integration in the industrial automation field. The interconnection of autonomous, flexible and configurable devices should result in a decrement of integration efforts because systems that currently reside on higher levels of the automation pyramid will be replaced by functionalities hosted in the embedded devices located at the factory shop floor.

This research proposes a new architecture for the hierarchical structure implemented in industrial automation enterprises, in which embedded devices will be computational nodes that cooperate for making decisions, controlling industrial processes and other functionalities that are currently managed and coordinated away from the device level. In fact, the proposed distributed system network that is composed by embedded devices creates a private local automation cloud, which is capable of receiving and solving external requests about services and computation resources. On the other hand, the presented approach describes the potential and challenges of using embedded devices that integrate their local resources in order to accomplish distributed reasoning. Therefore, the contribution of this research is not only to propose of an alternative for the conventional architecture in industrial enterprises,

but also to provide the design of the private automation cloud, ontological model and behaviour between devices for exchanging information.

The rest of the paper is structured as follows: Section 2 presents literature and industrial practices of aspects and areas of interest for the presented architecture to be developed and implemented in the field of factory automation. Section 3 introduces the principles of the private automation cloud and describes the encapsulated knowledge and behaviour of devices for solving incoming requests. Then, Section 4 discusses the potential and challenges of the presented approach. Finally, Section 5 presents the conclusion and further work of the research.

2. Literature and industrial practices

2.1. Cloud computing definitions and some concepts

Cloud computing (CC) is a promising paradigm that is currently applied in several domains, such as industrial automation domain. In a nutshell, CC allows abstracting and storing computational resources, as well as providing on-demand access to those resources. CC is related to other research areas as high-performance, utility and grid computing. As an example [14], offers a detailed comparison between grid and CC. Fundamentally, CC can be defined as a computing model composed by networked elements, which control their own resources. The computing resources are hosted locally by each element, i.e. a server. Nevertheless, the cloud can be understood from the outside as a unique element that contains a large amount of cloud services that are used for responding to incoming service requests. In addition, cloud services are a type of WS [15].

There are three different service models (or categories) of CC: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Usually, IaaS, e.g. Google Compute Engine (GCE),⁴ utilizes virtual computer resources that are outsourced by an organization [16]. IaaS users can remotely access and manage the data. Thus, IaaS is also known as HaaS (Hardware-as-a-Service). Secondly, PaaS is a service category that permits developers to implement their applications on a cloud. For example, Microsoft Azure⁵ provides both IaaS and PaaS. Finally, the SaaS, e.g. Google Gmail,⁶ service model allows users to access vendors' applications within the web. It should be noted that way of accessing to PaaS and SaaS is similar [17].

On the other hand, CC can be deployed mainly as a private, public or hybrid clouds, which are compared in [18]. A private cloud is a cloud that can be hosted internally or externally but operated only for a unique organization. Then, private cloud services are accessible solely for users and third-party applications, which belong to the organization that owns the cloud. On the other hand, cloud services of a public cloud are accessible by any user. This is possible because service access is provided through a public network. Finally, hybrid clouds are composed by more than one cloud. This fact does not exclude the composition of private and public clouds in the same hybrid cloud. For example, hybrid cloud is beneficial when an organization requires more computing resources that the ones offered by its private cloud. Then, the creation of a hybrid cloud, by incorporating e.g. a public cloud with accessible computing resources, is a possible approach for meeting the requirements. The deployment of hybrid clouds implies developing federated capabilities, which is needed for linking distributed resources.

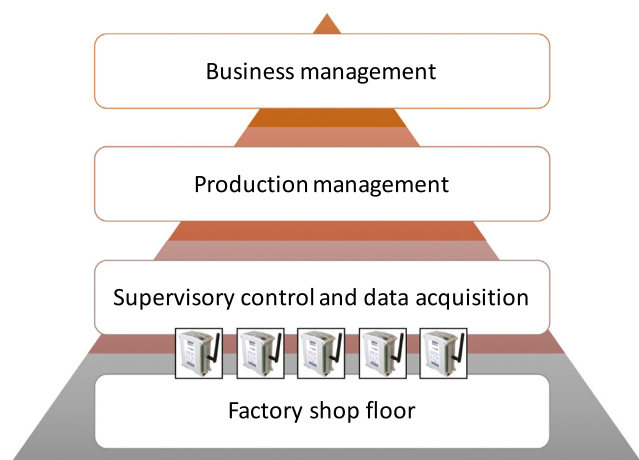


Fig. 1. A representation of the ISA-95 automation pyramid, including WS enabled RTUs.

⁴ <https://cloud.google.com/compute/>.

⁵ <http://azure.microsoft.com/en-us/>.

⁶ <https://mail.google.com/>.

2.2. Integration of cyber and physical systems

The integration of cyber and physical systems is known as CPS [13]. Recent advances on CPS permit cross-domain integration and Machine-to-Machine (M2M) communication in the industrial automation field. Moreover, according to [19], the emergence of CC in the industry, and its combination with CPS, will make it possible to achieve some advantages as scalability and flexibility of resources, among other features in future manufacturing systems. This is possible due to the success of SOA implementation in factory automation. For example [2,20], demonstrated that functionalities of shop floor equipment can be exposed outside through WS. Then, by the deployment of a cloud with access to the shop floor services, integration of equipment and the cloud will permit accessing to physical resources through cloud services.

The relevance of CPS is discussed e.g., in [21]. Aforementioned research work introduces the next generation of manufacturing industry, referred as the Industry 4.0, which is to be built by CPS. Fundamentally, any physical “thing” related to industrial processes will be connected to the Internet, realizing the principles of the Internet of Things (IoT), so it will be also integrated to cyber-systems. A formal overview of the standards and patents for the IoT, presented as a key enabler for the Industry 4.0 can be found in [22]. Furthermore, an example of architecture for implementing CPS on Industry 4.0-based manufacturing systems is presented in [23].

2.3. Knowledge representation in factory automation

Knowledge Representation (KR) is a part of the artificial intelligence concerned on describing the world and allowing computer systems to make decisions by using such descriptions [24]. Moreover, a Knowledge Base (KB) can be defined as an engineering artefact used to keep the knowledge representation. There is a large amount of KR formalisms e.g., ontologies, production rules or frames. A large amount of factory automation developments has used ontology-based KR for both system and product knowledge modelling as e.g., the research works presented in [25–29]. T. Gruber defined ontology as “an explicit specification of a conceptualization” [30]. Then, ontologies contain formal description about certain domain [31].

Although there are many semantic languages for modelling manufacturing systems [32], the Ontology Web Language (OWL)⁷ is preferred for such descriptions due to its high degree of expressivity. Truly, OWL is a vocabulary extension of the Resource Description Framework (RDF)⁸. Then, it is possible to use RDF-based query languages such as the SPARQL Protocol and RDF Query Language (SPARQL)⁹ and the SPARQL Update¹⁰ for retrieving and updating ontological model data, respectively. This is an important fact for integration of computer systems because whenever a system needs some data described in the model, it can access it by querying the model. For example, the research work described in [33] presents a set of queries used for retrieving data from a production line ontological model. As it can be seen in aforementioned research work, OWL-S is an OWL-based model for service description.

On the other hand, reasoning is defined in [24] as “the formal manipulation of the symbols representing a collection of believed propositions to produce representation of new ones”. This process is performed by semantic reasoners or reasoning engines operating on OWL models. A comparison between existing reasoning engines can be found in [34]. This is beneficial for systems implementing

ontology-based KR because reasoners can deduce dynamically data sets that are not defined as a fact when the model is created. For instance, data mappings and classification of system instances in the model are tasks easily performable by semantic reasoning engines [35]. It should be noted that for supporting reasoner inferences, the use of the Semantic Web Rule Language (SWRL) allows the definition of rules on top of OWL models [8]. Moreover, the utilization of reasoning engines allows checking the inconsistency of ontological models, resolving conflicts and reducing redundancy of data. In this scope, a recent research work presented an ontological framework that supports decision making in industrial scenarios within the use of semantic rules created for domain-specific ontologies [10].

Several research works in the factory automation domain present solutions that implement KR in different kinds of architectures. For instance, one of the previous cited research works [33], combines SOA and KR. In fact, the same research team describe how to dynamically retrieve OWL-S descriptions of services in order to find the ones required to perform industrial process actions [36]. Besides its integration with SOA, KR is also implemented in different kind of architectures, such as event driven architecture as recently presented in [37].

2.4. Distributed reasoning

In distributed systems, networked computer systems coordinate their actions within the exchange of messages. In fact, the coordination of actions is done for solving incoming requests to the system. It should be noted that, in this case, resources are understood as the KB data of a computer system. Then, distributed reasoning is defined as a process performed by several computer systems, which exchange messages for integrating their resources.

The principal objective of distributed reasoning is to reason integrated KB data of different computer systems. Fundamentally, taking into account that each computer system owns a different local KB, the integration of several computer systems data is required for responding to requests that cannot be responded using only the resources of a single computer system.

Therefore, it can be stated that the implementation of this behaviour is a requirement for future private and local automation clouds allowing embedded devices to cooperate for solving problems jointly. An employable mechanism for such need has been already described in [38]. Aforementioned work presents an approach for managing distributed knowledge, which is encapsulated in embedded devices, aiming the integration of resources for solving incoming requests.

3. A private automation cloud built by CPS

This research work proposes the creation of a private automation cloud composed by embedded devices, which encapsulate knowledge and functionalities that nowadays are nested and managed above the device level. This cloud is private because is only accessible by allowed elements of a unique organization and resides locally in the same organization. The proposed approach is motivated by the evolution of devices currently available and used at shop floors for process control.

Nowadays, embedded devices have much more computational resources than previously. Hence, the performance of additional tasks beyond direct process control is now feasible. Some of these tasks, as the encapsulation of system knowledge, are directly downgraded from, in this case, production management to shop floor level. Nevertheless, the inclusion of new duties also implies the development of non-existing ones, as e.g., the implementation of algorithms that devices must be capable to execute for managing

⁷ <https://www.w3.org/TR/owl-ref/>.

⁸ <https://www.w3.org/RDF/>.

⁹ <https://www.w3.org/TR/rdf-sparql-query/>.

¹⁰ <https://www.w3.org/TR/sparql11-update/>.

knowledge encapsulation and knowledge integration, which is needed for distributed reasoning [38].

3.1. Private cloud architecture

The private automation cloud includes interconnected embedded devices which control production line processes. This means that the private cloud is directly connected to the physical equipment, integrating physical resources with cyber systems. A basic architecture representation of the system built by CPS is shown in following Fig. 2.

As it is explained in [2], the SOA paradigm can be implemented in actual production lines by the use of certain RTUs, e.g. S1000 from Inico Technologies shown in Fig. 2. This is possible using the Device Profile for Web Services (DPWS) stack that allows implementing WS in resource-constrained devices. Through this technology, embedded devices can be discovered and it is also possible to invoke described services [39], among other functionalities.

Then, the architecture shown in Fig. 2 represents how the shop floor equipment is connected to embedded devices, which control the machine processes. It should be noted that the connection machine-device depends on the machine that is controlled e.g., a conveyor can be connected within digital I/Os and robots via RS232. Aiming the illustration of how this is implemented in a real scenario, Fig. 3 shows RTUs employed by the FASTory line, which is an assembly line located at Tampere University of Technology facilities, for controlling processes. Fig. 3 shows two different views of FASTory line cells, which are capable to draw 729 different variants of mobile phones (FASTory line detailed description is included in [12]).

The objective of showing two different perspectives of FASTory cells is to show i) the major components of each cell (i.e. conveyor and SCARA robot) and ii) the connected RTUs. As it can be seen at the bottom right side of Fig. 3, the three S1000 controllers (that are identifiable by their blue led lights) are connected to the conveyor segment and robot of the corresponding cell. Moreover, one extra RTU is used as an energy meter to manage and monitor the energy

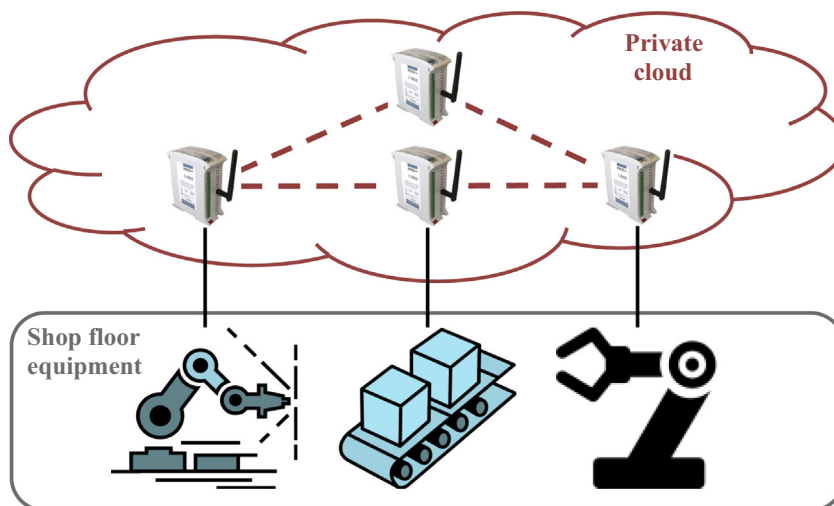


Fig. 2. The private automation cloud basic architecture.

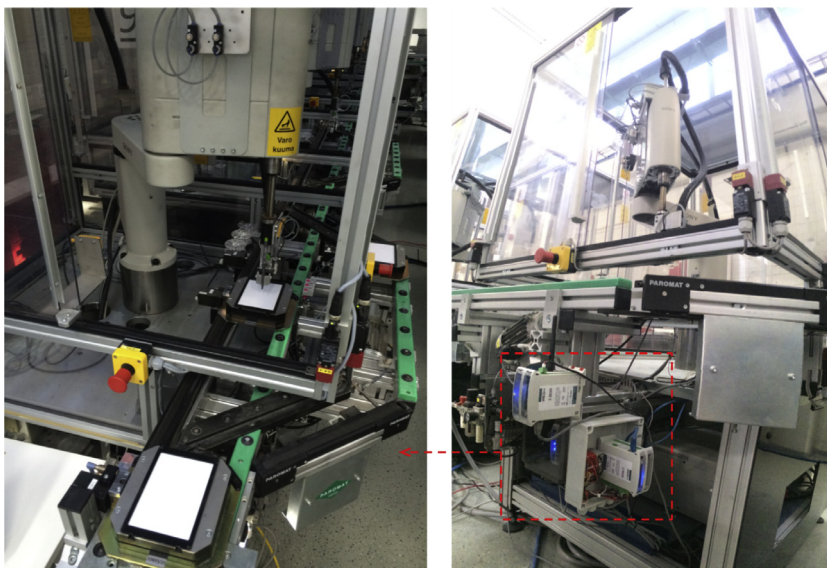


Fig. 3. Three industrial controllers are used for controlling processes of a FASTory line cell.

Table 1
Benchmarking the implementation of the approach in the FASTory line.

| | Original | Private local cloud |
|-------------------------------------|-----------------------------------|-----------------------------------|
| Production type | Mass | Reconfigurable |
| Communication protocols | Open and Proprietary | Open |
| IT infrastructure | Poorly reusable | Highly reusable |
| Control architecture | Centralized | Distributed |
| Control logic | Scan based | Event based |
| Modularity | At manufacturing cell level | At device level |
| Addressability | Hierarchical | Direct (query) |
| Processing layer | Fixed | Customizable |
| Processing of data | At central unit | At device level |
| Number of computational nodes | 1 (centralized architecture) | Many (decentralized architecture) |
| Plug and play | No | Yes |
| Output | Signal data | Information |
| Data format | Heterogeneous | Homogeneous |
| Message overhead | No | Yes |
| Message readability by humans | No | Yes |
| Horizontal and vertical integration | No | Yes |
| Remote monitoring | Additional middleware is required | From cloud and/or from device |
| Cyber security | Obscurity (i.e. system isolation) | Defence-in-depth |

consumption of the equipment. It should be noted that this research work is not restricted to the use of certain RTUs. For example, some experiments for controlling processes within Raspberry Pi¹¹ have been already tested [38].

On the other hand, the devices that inhabit in the private cloud need to encapsulate knowledge of the system, which will permit devices to decide and control the production line. In fact, [9,11] describe the use of ontology-based knowledge representation in knowledge-driven approaches in actual production lines.

However, aforementioned research works implement the knowledge localization and manipulation far away from where data is generated (i.e. physical equipment). This research work proposes lowering the knowledge representation to the device level. Actually, an approach for decentralising the knowledge used for knowledge-driven systems within the eScop project solution is described in [40].

The approach discussed in this article goes one step forward because proposes not only the decentralisation and lowering the knowledge to the device level; but also the implementation and control of higher-level functions on devices. In order to benchmark the proposed approach, Table 1 shows different aspects of the original FASTory line configuration that will be improved within further implementation. Some of the aspects have been contrasted with the ones of the research work presented in [2].

As shown in Table 1, the first direct benefit for the FASTory line is the re-configurability of production type. Meanwhile the original line configuration permits the production of one product, the deployment of embedded devices that are capable of controlling and linking different FASTory line resources will permit customizing and ordering remotely products at run time.

On the other hand, the implementation of the cloud brings new features as remote monitoring, addressability of resources, and horizontal and vertical integration. However, the message overhead will be increased due to the fact that the messages being exchanged contain additional information than the signal data created at different locations of the production line. Despite on this, and thanks to the representation of knowledge, the data becomes more readable and understandable for humans. Moreover, as the data format is homogenized the configuration will demand less effort.

Finally, the fact of adding decentralized computational nodes that are linked to physical resources, causes changes to the operation and architecture of the system:

- The control architecture becomes distributed because the nodes will exchange messages in order to solve requests
- Data will be processed at device level, instead of having a central unit for data processing
- The control logic will be event based because nodes will react to notifications related to changes on machine status
- The amount of computational nodes increases and will be scalable. In fact, the implemented SW and HW will be highly reusable and new devices will be deployable at run time
- In contrast to recent approaches in the same production line [12] the knowledge of the system will be decentralized

3.2. Main concepts to be included in device knowledge models

The KB encapsulated in embedded devices is one of the most important element of the proposed architecture because contains the information used by each cloud device to interact and make decisions with the rest of peers. In this research work, such knowledge container is to be implemented within ontologies. More precisely, OWL is the selected language to describe the KB, which will be inspected and updated within SPARQL-based language queries.

Obviously, each KB has different type of information that is based on each device role in the cloud. Nevertheless, there are a set of concepts that must be similarly described in each KB i.e. *Network*, *Service* and *Device*. In this way, the only different data type contained in KBs is the one e.g., used for controlling specific equipment to which the embedded device is connected.

Following Fig. 4, presents the main concepts that each device hosts as local resources in an ontological-based model.

As explained, Fig. 4 illustrates the principal concepts to be described in the ontological model, which is hosted in the embedded devices forming the private cloud. Such kind of information is required to be included in each device because this research work proposes to handle requests and decision-making fully in the private automation cloud itself. Therefore, devices must know the status, location and type of peers that inhabit in the same network. The specific reason for each concept to be included in the ontology is described below:

- *Network*: It contains information about the network or, in this case, the private cloud. It includes the addresses of other devices in the network. In this manner, a device knows other peers' endpoints to send any type of request.
- *Service*: It includes information about the services described in the device as, for example, service and message types and operations. To design such description, OWL-S specification¹² can be used. Through this concept, device capabilities are exposed to other peers, which is a requirement because the functionality description of each device must be accessible in the private cloud. Basically, anything that a device is capable to perform, or knows, must be shared with other devices residing in the same network.
- *Device*: It contains general information of the device as e.g., its type, address, operation system and firmware version. This data is important because a *Device X* can access to *Device Y* basic information. For instance, a *Device X* may employ this information to update its *Network* concept information with the *Device Y* address. Therefore, this supports devices to decide in which situation and how to access other peers.

¹¹ <http://www.raspberrypi.org/>.

¹² <https://www.w3.org/Submission/OWL-S/>.

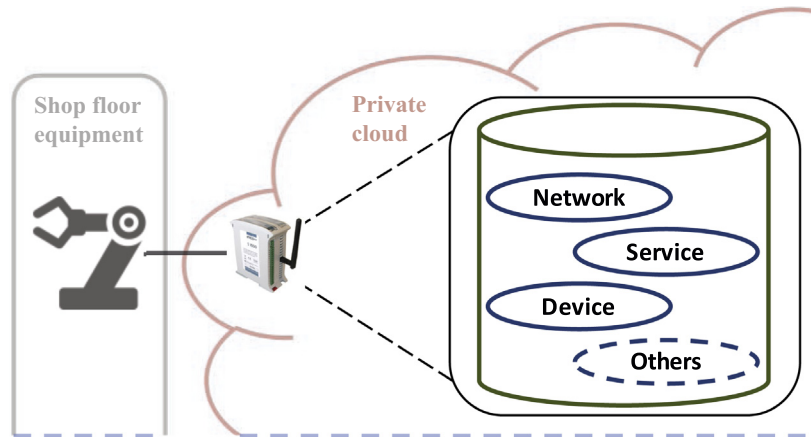


Fig. 4. Main concepts of the ontological-based knowledge representation.

- *Others*: It includes data instances related to the specific role, purpose, operations, etc. that the device is responsible of. Then, *Others* will be a conjunct of classes related to the device type. It should be noted that the *Others* concept of Fig. 4 is a mere representation so that in reality a device can include more than one concept needed for such description in the ontology. For example, if a *Device Z* is controlling an entire cell of an assembly line, the *Others* concept might be implemented as a set of concepts for describing components and processes of that specific cell. On the other hand, if a

Device T is used for calculating Key Performance Indicators (KPIs), the *Others* concept might consist in a set of concepts for describing the type of data and equations to be calculated.

Basically, the “other” concepts included in devices describe information of the specific tasks that each device performs in any controlled system that uses this approach.

Aiming an exemplification of the above described device's ontology main concepts, Fig. 5 shows a possible class hierarchy

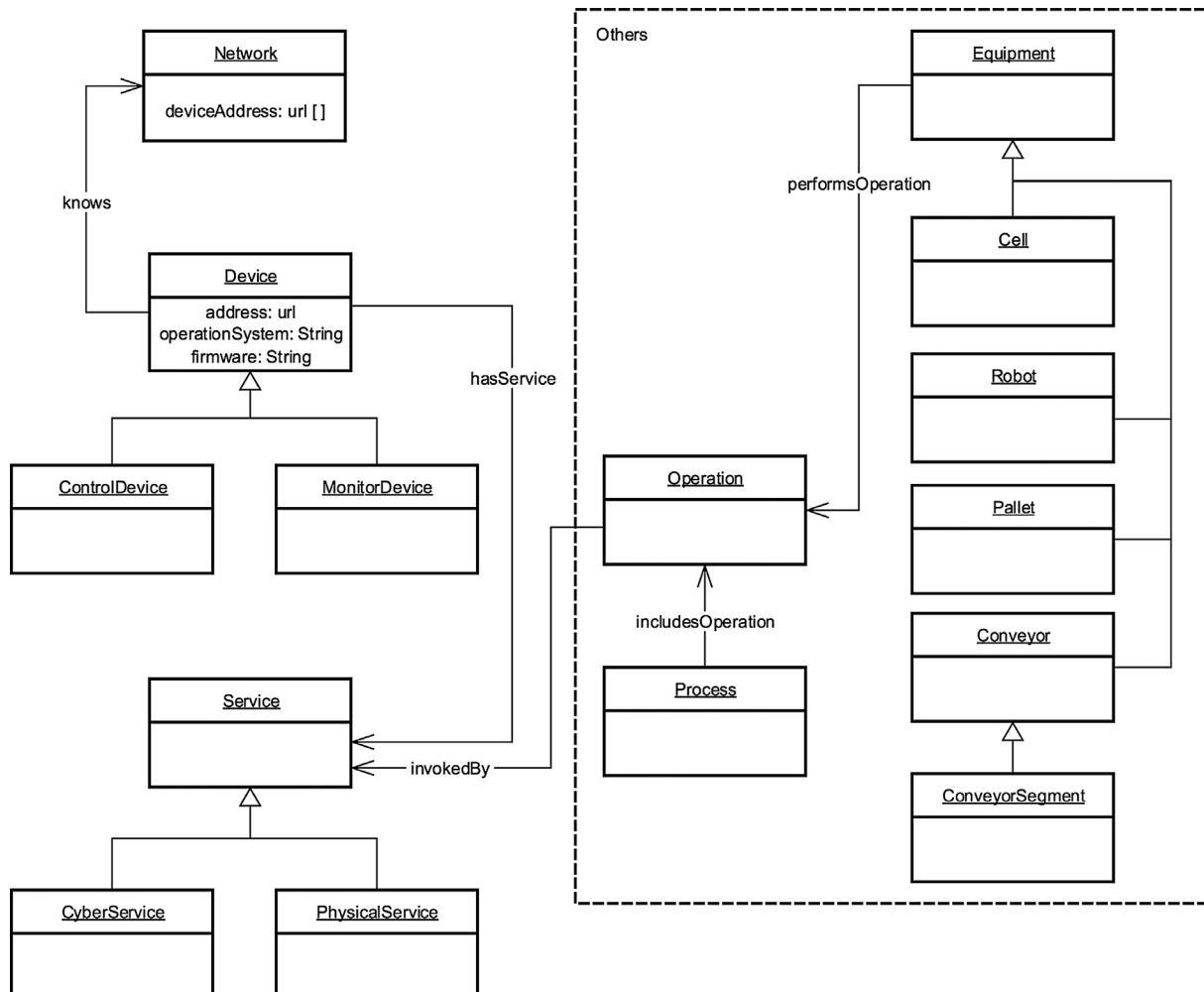


Fig. 5. Class diagram example of a device ontology used for describing and controlling a FASTory line cell.

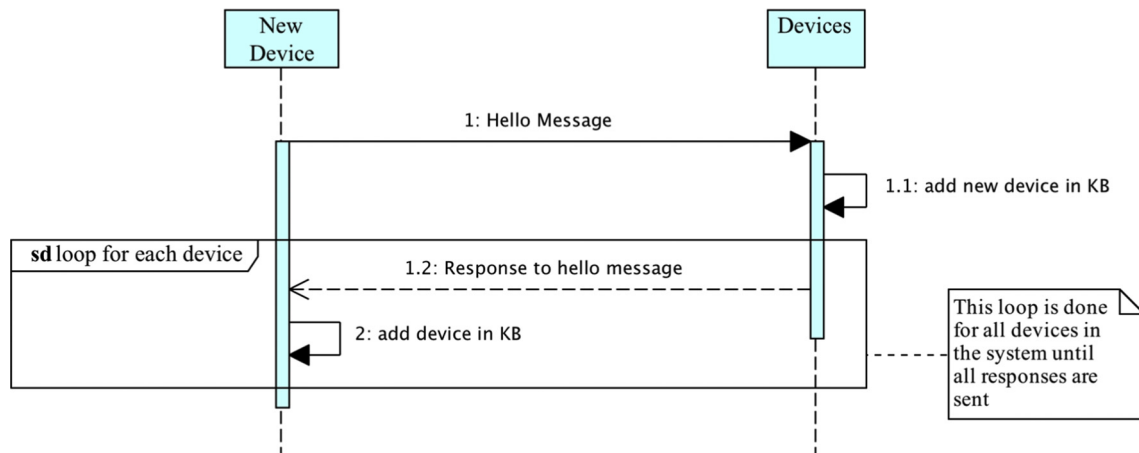


Fig. 6. Sequence diagram for registering a new device [38].

of a device, which is used to control a manufacturing cell of the FASTory line. Then, the presented class diagram demonstrates how the concepts can be implemented in a real-case scenario.

First, the presented class diagram shows that the instance of the *Network* class will handle an array, labelled as *deviceAddress*. This attribute is an array which contains all the addresses of the devices of the network. As described in next Section 3.3, once a device is added to the cloud it shares its address so it can be accessed by other devices. Then, each time that a new device enters in the network, the *deviceAddress* array of each device is updated.

On the other hand, the *Device* class is the one to include the device instance. Besides having data type properties, or attributes (as e.g. *address*, *operationSystem* and *firmware*), *Device* is a superclass of *ControlDevice* and *MonitorDevice*, which define the device type. Meanwhile *ControlDevice* type indicates that the device is used for controlling processes with physical consequences in the environment; *MonitorDevice* types includes the device instance if it is just used for operations that does not change directly the physical world (e.g., calculations, checking values of sensors and data manipulation). In fact, a device can be classified in both types if it is used for both types of tasks. This classification depends directly on the types of services that the device supports to the private cloud.

Any device's KB of the private cloud includes a description of the services that can be of two types i.e., *CyberService* and *PhysicalService*. Services belonging to *CyberService* are those that does not perform any change in the physical environment as e.g. data request services. On the other hand, services belonging to *PhysicalService* are those which imply physical changes as e.g., move, clamp or draw operations.

Finally, to show how *Others* concept is really implemented in a real-case scenario, Fig. 5 shows a set of classes that are needed for describing the physical equipment and process to execute in FASTory cells. Then, as the device of which the KB's class diagram is presented is responsible of the control of FASTory cell operations must include the description of (i) physical equipment, (ii) operations to execute and (iii) processes to perform. Such information, by order, is included in the ontology within *Equipment* superclass and other subclasses (i.e. *Cell*, *Robot*, *Pallet*, *Conveyor* and *ConveyorSegment*), *Operation* and *Process* classes. Then, certain processes that can be performed in the cell are divided into operations (linked within the *includesOperation* object property) that can be invoked by services described in *Service* class (linked within the *invokedBy* object property). It should be noted that

Fig. 5 shows just an example of a device that is controlling operations of a manufacturing cell. Therefore, for a different type of device, the *Others* set of classes will be different. However, *Network*, *Device* and *Service* concepts will have the same structure, following the described approach.

Furthermore, as the presented diagram is just the KB of one device of the private cloud, the KB of the whole cloud would be the integration of all devices' KBs. Then, devices are capable of accessing and sharing these kind of information for accomplishing any distributed reasoning task, which is required for controlling processes. One of the benefits of decentralizing the system knowledge description in the cloud is that the knowledge does not have to reside only in one critical device, which (i) if it fails, the system will fail and (ii) most probably, it will not have enough resources for hosting and managing the model of a large production line. Basically, the entire model is decentralized and hosted by several devices, which cooperate, finding, integrating and reasoning the required information, for solving requests. It should be noted that this knowledge is accessible by other devices using the SPARQL 1.1. Graph Store HTTP Protocol¹³.

3.3. Behaviour of embedded devices

Besides the ontological model encapsulation in embedded devices, the behaviour of such devices in the cloud must be also designed and implemented. The exchange of messages between embedded devices will be performed within a set of algorithms that allow devices to cooperate in the cloud. It is absolutely essential that cloud devices include *device initialization* and *reasoning process performance* algorithms, presented in [38]. In order to clarify how these algorithms are to be executed, this section includes a description of each one.

Firstly, the *device initialization* algorithm permits that a new device makes itself discoverable by other peers as shown in the UML diagram of Fig. 6.

Any cloud device will be capable to access to the resources of the new device because the address of new devices is broadcasted to all peers in the cloud in the first step of the *device initialization* algorithm within the "Hello Message". Hence, this algorithm must be executed when a device enters in the private cloud. Successfully, the new device will receive a response including the address of the corresponding devices that are answering to the "Hello Message". In this way, the new device will populate the *Network* class

¹³ <https://www.w3.org/TR/sparql11-http-rdf-update/>.

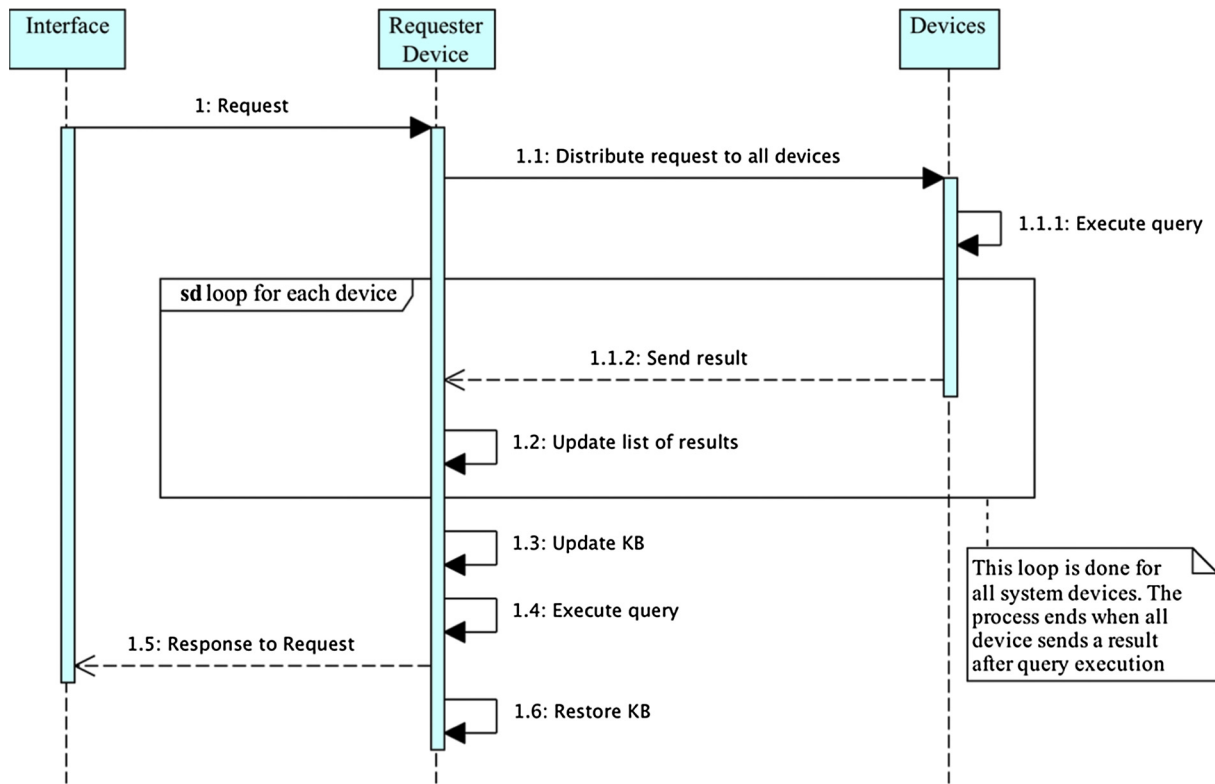


Fig. 7. Sequence diagram of a reasoning process performance [38].

of its own KB with the addresses of peers that inhabit in the same cloud.

On the other hand, the *reasoning process performance* algorithm defines the sequence of messages required for exchanging information in a reasoning process as shown in Fig. 7. The reasoning process is required for reasoning the entire system model, which is formed by all resources hosted and managed by the embedded devices.

As it can be seen in Fig. 7, the *reasoning process performance* algorithm permits cloud devices requesting data that is needed for responding to an incoming query. The request is sent by a client to the system through the cloud interface. In fact, one of the devices of the cloud will take the lead of the reasoning process. As shown in Fig. 7, this device is known as the *requester device*. The requester device will be selected by the interface according to the available resources that devices have to compute the incoming queries. The task of the requester device is to handle the reasoning process, which implies (i) receiving the request from the interface, (ii) distributing the request to all devices, (iii) collecting the results from devices, (iv) integrating all knowledge, (v) executing the query and (v) sending a response to the interface. Moreover, as it can be seen in the last step of the sequence diagram, the requester device will restore its own KB after handling a reasoning process. This last action is executed in order to remove redundant knowledge that is already hosted by other devices but needed in the knowledge integration.

The *reasoning process performance algorithm* is a process that requires the cooperation of all devices for integrating all the knowledge related to the request and creating a valid response that must satisfy the client. The incoming queries will request (i) information for monitoring resources, (ii) computation of cloud resources or (iii) order a product that can be performed by the manufacturing system being controlled by the private cloud.

Therefore, solving requests will frequently imply the execution of services hosted by different embedded devices.

3.4. Implementing ontologies for embedded devices and exemplifying distributed reasoning in the FASTory line

The objective of this section is to show an example of executing distributed reasoning in a concrete manufacturing scenario, i.e. the FASTory line.

Firstly, an ontological model that follows the class hierarchy shown in Fig. 5 is created within an ontology editor. For this experiment, Protégé¹⁴ has been used as the tool for editing the model. Once the ontology is created, it is populated with the instances being controlled by corresponding embedded devices. Thereby, if the *device 1* is hosting *model 1*, the instances of *model 1* will represent the industrial components being controlled by *device 1*. Fig. 8 shows the ontology of the device that is connected and controls the components of the FASTory cell 1.

As it can be seen in Fig. 8, the ontology includes all the classes shown in the ontology model presented in Fig. 5. In addition, Fig. 8 shows, as an example, that the *Robot 1* (controlled by the *device 1*) has a set of operations that can be performed: *DrawFrameA*, *DrawFrameB*, *DrawKeyboardA* and *DrawKeyboardB*. These operations will be executed when the linked services through the *hasService* object property are invoked.

After the design and implementation of each device ontology, the models are deployed into corresponding devices. The deployment of device models will differ according to the type of embedded devices, which will employ different platforms. Nevertheless, this task will imply always two major steps: (i) to place the

¹⁴ <http://protege.stanford.edu/>.

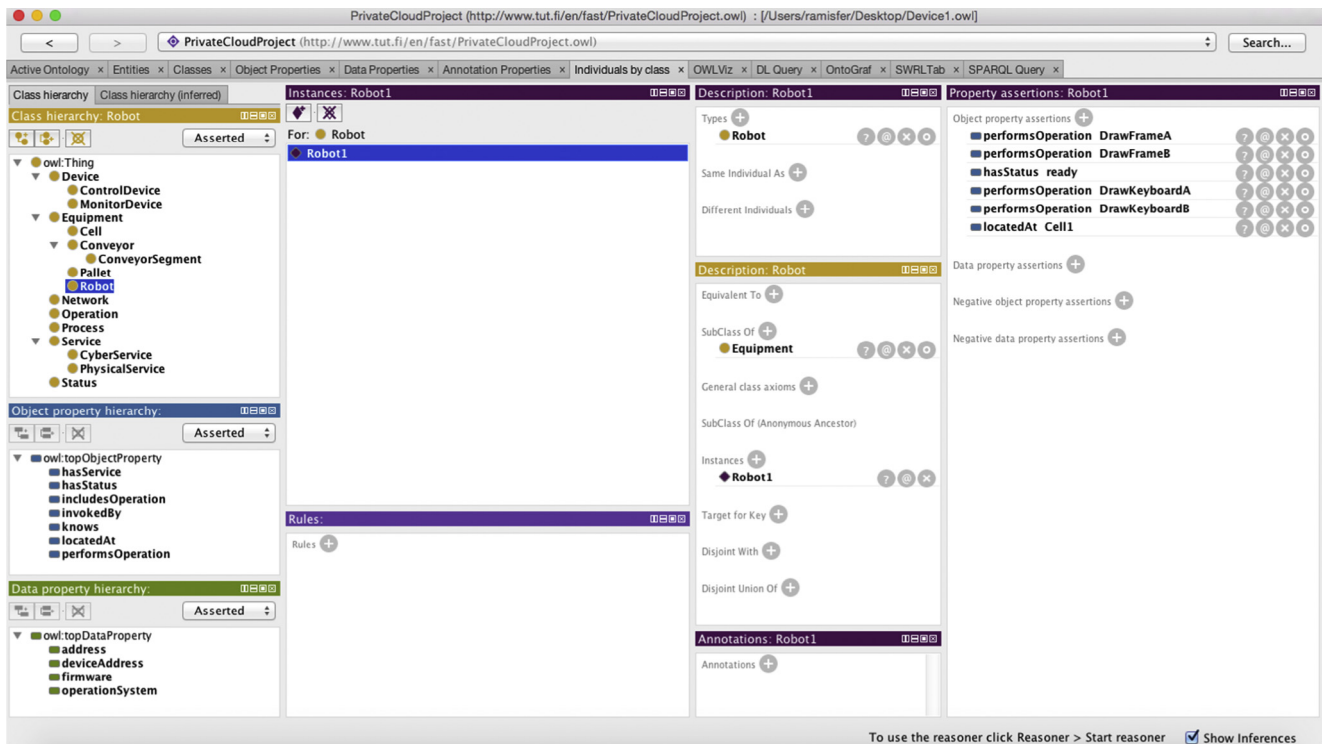


Fig. 8. OWL model hosted by device 1.

```

PREFIX dr:<http://www.tut.fi/en/fast/PrivateCloudProject.owl#>
SELECT ?Robot ?RobotStatus ?Cell ?Operation
WHERE {
  ?Robot dr:hasStatus ?RobotStatus.
  ?Robot dr:locatedAt ?Cell.
  ?Robot dr:performsOperation ?Operation.
}
  
```

Fig. 9. Example of SPARQL query to be solved by the private cloud within the distributed reasoning process.

ontology file into the device and (ii) to implement an interface that permits the interaction with the model. For example, the ontology model can be placed into the SD card of a Raspberry Pi and the interface can be implemented within Apache Jena,¹⁵ which is a framework that permits the management of RDF models. In fact, the *Apache Jena Fuseki* is a server that can run as an operating system service providing the SPARQL 1.1 and the SPARQL 1.1 Graph Store HTTP protocols to query, store and update RDF-based models.

For the FASTory line, incoming queries to the system can be simple monitoring requests, such as asking the status of a robot or complex requests, e.g., order the production of a mobile phone. In any case, the system will react in the same way to requests, i.e. executing the distributed reasoning process. Fundamentally, the execution of the aforementioned algorithm permits the integration of knowledge that is located at different computational nodes to ensure that the validation of the query is done according to the whole model information.

As the research proposes the use of RDF-based models, the ontology has been implemented in OWL syntax, which can be queried within SPARQL queries. In order to present an example of distributed reasoning in the FASTory line, Fig. 9 shows a moni-

toring query to be sent and executed by the private automation cloud.

The query shown in Fig. 9 can be used for requesting a list of robots including their status, location and operations that can perform. As previously said, each embedded device of the FASTory line hosts an ontology with similar class hierarchy but populated with the specific industrial equipment that each peer controls. As each device is used for controlling a different cell, a unique device is not capable of answering the incoming query shown in Fig. 9 including the status of all assembly line robots. Basically, as an example, *device 1* will be capable of answering with the status, location and operations of *Robot 1* but not with the ones of, e.g., *Robot 2*, as shown in Fig. 10.

Nevertheless, the execution of the distributed reasoning process will permit the cloud to achieve a valid answer to be returned to the cloud client. As it can be seen in the sequence diagram of Fig. 7, the incoming query will be broadcasted from the requester device to all cloud devices. Then, each device having any knowledge instance of the type of any query element will be returned to the requester device. Afterwards, the requester device will execute the query with all collected knowledge and conclude the answer, which will be redirected to the cloud interface. Assuming that (i) the integration model is created after the execution of the

¹⁵ <https://jena.apache.org>.

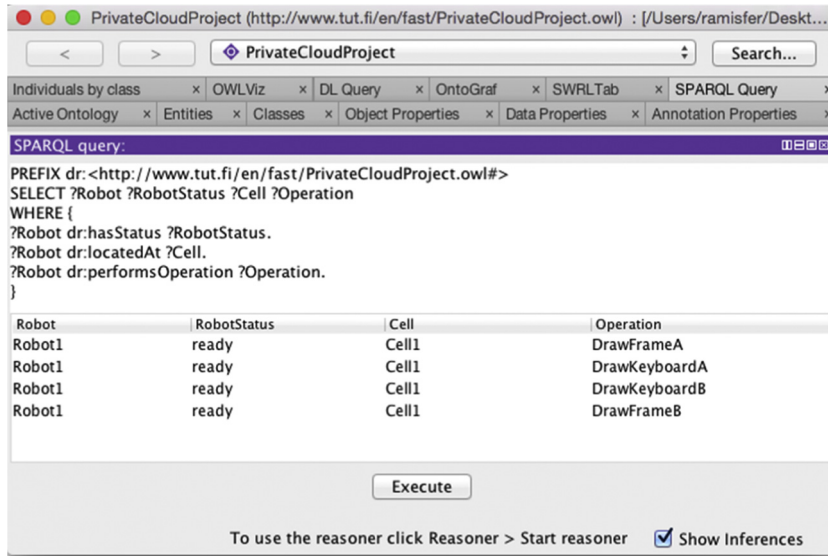


Fig. 10. Result of query execution in device 1.

distributed reasoning process and that (ii) this is tested in a reduced version of the FASTory line with 4 cells: the requester device for the query shown in Fig. 9 obtain the result depicted in Fig. 11.

As it can be seen in the result depicted by Fig. 11, the requester device will be capable of answering the incoming query including all robot status, location and operations of the line. The result is achieved due to previous integration of each device knowledge related to the incoming query.

Another example to discuss the execution of the distributed reasoning process can be a request for routing a pallet through the assembly line. In order to control the transport of pallets in the FASTory line, services describing conveyor operations must

be executed in certain order. For example, assuming that the client requests for a valid path to route the pallet from point A to point B, the cloud needs first to conclude the layout of the line. Then, assuming that those points are manufacturing cells and that the query asks about locations, all devices will share the knowledge instances related to location type; i.e., robots, conveyors and manufacturing cells. Then, the requester device will be capable of shaping the line because the integrated model will contain all cell physical connections.

In fact, when an incoming query requests the production of a mobile phone, the locations of each robot that can perform required operations for the order must be located. As shown in Fig. 11, the current FASTory line configuration has no single cell

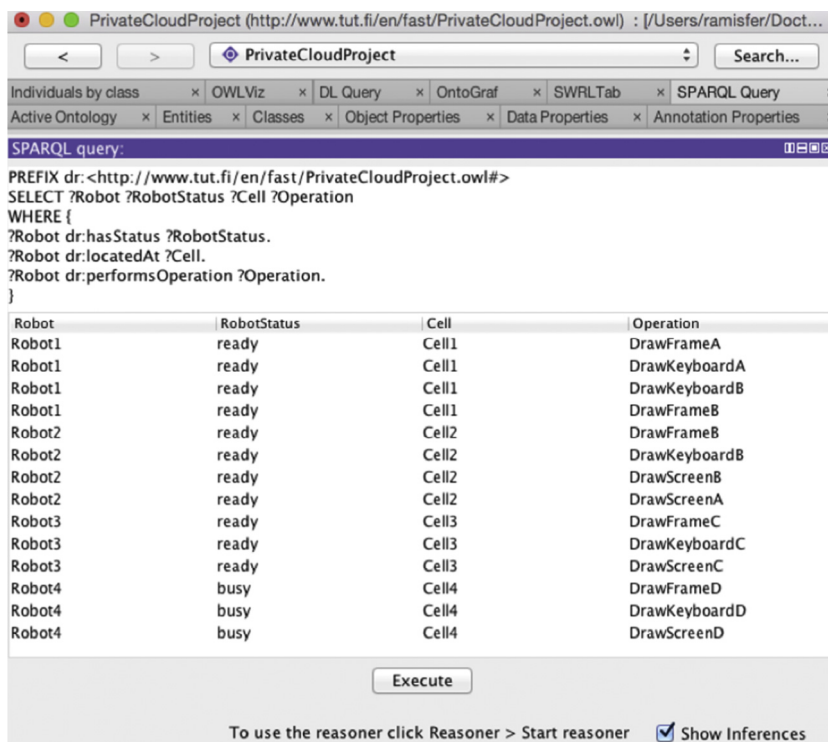


Fig. 11. Query result after distributed reasoning process performance.

capable of assembling all possible component types of a mobile phone (i.e. same type of keyboard, screen and frame). Then, the execution of required operations implies to conclude first the location and status of the equipment capable of performing such operations.

4. Discussion

Once the principles of the private cloud architecture and requirements of embedded devices have been described, potential and challenges of the approach are presented in this section.

4.1. Potential of the approach

The proposed research looks forward the creation of a private automation cloud formed by embedded devices that handle functionality and knowledge management, nowadays done centrally and away from where the data is generated (e.g., industrial equipment, machines, sensors, actuators). Moreover, the presented approach proposes to create a solution within resource-constrained devices, which are capable to cooperate for performing distributed reasoning. The main potential of the presented approach is:

- *Network*: The implementation of such approach will be useful for evaluating the limits of using embedded devices not only for data collection and shop floor process control, but also i) to process the data and transform it into usable information and ii) to cooperate with other peers in the cloud for performing the required distributed reasoning, based on demand of private cloud resources. Hence, the development of the proposed distributed system will be useful to test that the new generation of embedded devices are ready to operate and fulfil the requirements of industrial systems, which are complex and dynamic environments. In addition, this research work has a direct impact on future awareness, flexibility and re-configurability of manufacturing systems [41] in the monitoring and supervisory control area.
- *Reduction of high integration cost*: One of the motivations of this research is the high-cost of integration in the industrial automation field, which can be reduced by the use of distributed intelligence [42]. Then, this research can support such reduction by the automation of the embedded systems configurability in factory floors through reasoning, using the whole range of computational resources available in the proposed cloud.
- *Decentralisation of knowledge-driven management solutions*: This approach follows some principles described in novel knowledge-driven approaches, which are presented in [40] as an alternative for conventional architecture and operation of MES [11]. The CPS integration permits the development of the proposed cloud that control system processes. Potentially, this fact directly contributes to improvement of efficiency and productivity of industrial automation processes.
- *Integration with external systems*: The industry is already employing internet technologies and web-based standards for semantic modelling, linking data and requesting resources. This allows entities to describe, request and manipulate system's KB for decision-making and execution of operations. As such technologies and standards are the ones to be used in presented approach, the interoperability of the cloud with actual external systems (e.g. legacy systems) is achievable. Then, if a third party is interested in demanding computational resources to the cloud, it can do it remotely and without communication issues but, obviously, with prior agreement between the cloud owner and interested organisations.
- *Multi-cloud or interconnection of clouds*: A multi-cloud collaboration between similar clouds owned by the same organization at different geolocations is possible. This connection can be used for sharing and, hence, increasing computational resources of a unique cloud. In this way, different clouds can be specialized in different computation of data, besides controlling the processes of connected equipment. Actually, clouds that are smaller in terms of computation power can consume e.g. data processed with complex and expensive algorithms. Then, scheduling plans, datasets, KPI calculations, among other sort of data can be requested and used between different clouds that manages similar processes in a same organization.

4.2. Challenges of the approach

For making possible the realization of the presented approach, some challenges become, at the end, requirements that must be addressed. Karnouskos et al. describes in [43] their vision for cloud-based industrial CPS, presenting trends and challenges on this fairly new research topic. In fact, some of the already presented challenges in [43] are also applicable in this research. The main challenges of the presented approach are:

- *Implementation of device behaviour*: Explained algorithms in Section 3.3 and others e.g., dynamic discovery of devices must be implemented and evaluated. The importance of these algorithms relies on the need of developing robust and efficient device intercommunication in the cloud.
- *Development of messages*: Previous challenge implies the need of designing message types and structures for information exchange. This point is important so that devices can interact between themselves or/and external users of the cloud. These messages can be understood as queries or request that are different depending on the situation in which devices will be involved. Then, besides the technology or standard within messages are written, different types of messages and corresponding content must be defined. In fact, a possible outcome of this challenge is the creation of a communication protocol for devices.
- *Horizontal scalability*: One of the features of CC the horizontal scalability i.e. increasing cloud capacity by connecting multiple entities so that all entities work as a single one. Although this research work contemplates the incorporation of similar devices with different roles in the system, the main idea of the private automation cloud is the cooperation of devices to work as a unique system. Simply, when a device is added to the cloud, it includes additional resources to be employed by the rest of devices. Therefore, from this perspective, the horizontal scalability is a challenge of the approach.
- *Fault tolerance*: Communications are vulnerable to faults. Thus, in cloud-based systems, *fault tolerance* is an aspect to take into consideration for ensuring that services are available and accessible. In order to guarantee a correct execution of cloud services, specific mechanisms must be provided to solve or, in best case scenario, anticipate to faults in the system. Conceptually, this challenge is about addressing robustness and reliability of distributed system communications.
- *Orchestration and/or choreography service execution*: Independently on device behaviour, cloud devices might need the incorporation of software that will allow the orchestration or choreography for setting the order of service operation execution. In addition, this challenge implies the development of a goal-oriented mechanism for integration of computational resources because service operations of a same process can be encapsulated in different embedded devices.

- *Cloud security*: The private cloud to be implemented will require an investment on data security. Fundamentally, the cloud is meant to be private. Then, it must be accessible only by the allowed organization, users and/or third party applications.
- *Others*: Some challenges addressed in [43] that must be considered as: cross-layer collaboration by CPS, migration and impact of CPS to existing industrial automation approaches, and semantic-driven interaction.

5. Conclusion

Recent developments in CPS permit the integration of semantic technologies and embedded devices, which propound a new scenario with many potential and challenges. Thus, this article presents the principles, challenges and potential of a private local automation cloud that is built by CPS. This research considers and rethinks the automation pyramid shown in ISA-95 and other central knowledge management solutions because the process of system information is done at the device level. Thus, one possible outcome of the presented approach is the redefinition of the business model for actual industrial systems.

Advances on CPS in recent research works and its impact on recent industrial system are presented in [11,44]. First, the employment of ontological-based knowledge representation as e.g. presented in early approach description of the eScop project in [11], permits the knowledge-driven solutions as a promising solution in the field. On the other hand [44], presents directly recent advances and trends of CPS in industrial informatics. Nevertheless, aforementioned research works do not consider that embedded devices can take the role of processing information and making decisions for production lines process control, which is the main objective of the presented approach.

Then, this article presents an alternative approach for the use of CPS for building a cloud that can implement other research works solutions but in the location where data is generated. Possibly, this approach will decrease industrial system integration time and costs because it will reduce cross-layer communication. In addition, the article presents an opportunity for evaluating the limitations of using a private cloud composed by embedded devices in the industrial automation domain.

Further work will include the development of models, algorithms and experimental cases for evaluating the cloud performance. Indeed, the challenges that are described in Section 4.2 must be taken into account when developing the proposed cloud. Then, this article can be used as a high-level roadmap for described development.

Acknowledgements

The authors gratefully acknowledge the support of the graduate school funding of Tampere University of Technology in carrying out this work.

References

- [1] M. Qusay H., "SOA and Web Services", 2005. [Online]. Available: <<http://www.oracle.com/technetwork/articles/javase/soa-142870.html>> (accessed: 02-March-2016).
- [2] L.E.G. Moctezuma, J. Jokinen, C. Postelnicu, J.L.M. Lastra, Retrofitting a factory automation system to address market needs and societal changes, in: 2012 10th IEEE International Conference on Industrial Informatics (INDIN), 2012, pp. 413–418.
- [3] I. International Electrotechnical Commission, "IEC 61131-3:2013 | IEC Webstore," 2013. [Online]. Available: <<https://webstore.iec.ch/publication/4552>> (accessed: 02-March-2016).
- [4] I. International Electrotechnical Commission, "IEC 61499-1:2012 | IEC Webstore", 2012. [Online]. Available: <<https://webstore.iec.ch/publication/5506>> (accessed: 02-March-2016).
- [5] V. Vyatkin, IEC 61499 as enabler of distributed and intelligent automation: state-of-the-art review, *IEEE Trans. Ind. Inform.* 7 (4) (2011) 768–781.
- [6] C. Popescu, J.L.M. Lastra, On ontology mapping in factory automation domain, in: IEEE Conference on Emerging Technologies and Factory Automation, 2007, ETFA, 2007, pp. 288–292.
- [7] P. Novák, E. Serral, R. Mordinyi, R. Šindelář, Integrating heterogeneous engineering knowledge and tools for efficient industrial simulation model support, *Adv. Eng. Inform.* 29 (3) (2015) 575–590.
- [8] J. Puttonen, A. Lobov, J.L.M. Lastra, Maintaining a dynamic view of semantic web services representing factory automation systems, in: 2013 IEEE 20th International Conference on Web Services (ICWS), 2013, pp. 419–426.
- [9] B. Ramis et al., Knowledge-based web service integration for industrial automation, in: 2014 12th IEEE International Conference on Industrial Informatics (INDIN), 2014, pp. 733–739.
- [10] L. Petnga, M. Austin, An ontological framework for knowledge modeling and decision support in cyber-physical systems, *Adv. Eng. Inform.* 30 (1) (2016) 77–94.
- [11] G. Marco, Open Automation of Manufacturing Systems through Integration of Ontology and Web Services, 2013, pp. 198–203.
- [12] S. Iarovyĭ, W.M. Mohammed, A. Lobov, B.R. Ferrer, J.L.M. Lastra, Cyber-physical systems for open-knowledge-driven manufacturing execution systems, *Proc. IEEE PP* (99) (2016) 1–13.
- [13] E.A. Lee, Cyber physical systems: design challenges, in: 2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008, pp. 363–369.
- [14] S.M. Hashemi, A.K. Bardsiri, Cloud computing vs. grid computing, *ARNP J. Syst. Softw.* 2 (5) (2012) 188–194.
- [15] D.K. Barry, Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's Guide, Newnes, 2012.
- [16] K. Keahey, M. Tsugawa, A. Matsunaga, J. Fortes, Sky computing, *IEEE Internet Comput.* 13 (5) (2009) 43–51.
- [17] G. Lawton, Developing software online with platform-as-a-service technology, *Computer* 41 (6) (2008) 13–15.
- [18] S. Goyal, Public vs private vs hybrid vs community – cloud computing: a critical review, *Int. J. Comput. Netw. Inf. Secur.* 6 (3) (2014) 20–29.
- [19] A.W. Colombo et al. (Eds.), *Industrial Cloud-Based Cyber-Physical Systems*, Springer International Publishing, Cham, 2014.
- [20] I.M. Delamer, J.L.M. Lastra, Loosely-coupled automation systems using device-level SOA, 2007 5th IEEE International Conference on Industrial Informatics, vol. 2, 2007, pp. 743–748.
- [21] A.W. Colombo, S. Karnouskos, T. Bangemann, Towards the next generation of industrial cyber-physical systems, in: A.W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, J.L. Lastra (Eds.), *Industrial Cloud-Based Cyber-Physical Systems*, Springer International Publishing, Cham, 2014, pp. 1–22.
- [22] A.J.C. Trappey, C.V. Trappey, U. Hareesh Govindarajan, A.C. Chuang, J.J. Sun, A Review of Essential Standards and Patent Landscapes for the Internet of Things: A Key Enabler for Industry 4.0, *Adv. Eng. Inform.*
- [23] J. Lee, B. Bagheri, H.-A. Kao, A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems, *Manuf. Lett.* 3 (2015) 18–23.
- [24] R.J. Brachman, H.J. Levesque, *Knowledge Representation and Reasoning*, Morgan Kaufmann, 2004.
- [25] V. Premkumar, S. Krishnamurthy, J.C. Wileden, I.R. Grosse, A semantic knowledge management system for laminated composites, *Adv. Eng. Inform.* 28 (1) (2014) 91–101.
- [26] F. Song, G. Zacharewicz, D. Chen, An ontology-driven framework towards building enterprise semantic information layer, *Adv. Eng. Inform.* 27 (1) (2013) 38–50.
- [27] A. Lobov, F.U. Lopez, V.V. Herrera, J. Puttonen, J.L.M. Lastra, Semantic Web Services framework for manufacturing industries, in: IEEE International Conference on Robotics and Biomimetics, 2008, ROBIO 2008, 2009, pp. 2104–2108.
- [28] W. Lu et al., Selecting a semantic similarity measure for concepts in two different CAD model data ontologies, *Adv. Eng. Inform.* 30 (3) (2016) 449–466.
- [29] H. Liu, M. Lu, M. Al-Hussein, Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry, *Adv. Eng. Inform.* 30 (2) (2016) 190–207.
- [30] T.R. Gruber, A translation approach to portable ontology specifications, *Knowl. Acquis.* 5 (2) (1993) 199–220.
- [31] J.L.M. Lastra, I.M. Delamer, F. Ubis, *Domain Ontologies for Reasoning Machines in Factory Automation*, ISA, 2010.
- [32] E. Negri, L. Fumagalli, M. Garetti, L. Tanca, A review of semantic languages for the conceptual modelling of the manufacturing domain, in: Proceedings of the XIX Summer School Francesco Turco, 2014, Senigallia, Ancona, 2014, pp. 1–8.
- [33] J. Puttonen, A. Lobov, M.A. Cavia Soto, J.L. Martínez Lastra, Planning-based semantic web service composition in factory automation, *Adv. Eng. Inform.* 29 (4) (2015) 1041–1054.
- [34] S. Abburu, A survey on ontology reasoners and comparison, *Int. J. Comput. Appl.* 57 (17) (2012).
- [35] B. Ramis Ferrer, B. Ahmad, D. Vera, A. Lobov, R. Harrison, J.L. Martínez Lastra, Product, process and resource model coupling for knowledge-driven assembly automation, *Autom.* 64 (3) (2016).
- [36] J. Puttonen, A. Lobov, M.A.C. Soto, J.L.M. Lastra, A Semantic Web Services-based approach for production systems control, *Adv. Eng. Inform.* 24 (3) (2010) 285–299.

- [37] Y. Evchina, J.L. Martinez Lastra, Hybrid approach for selective delivery of information streams in data-intensive monitoring systems, *Adv. Eng. Inform.* 30 (3) (2016) 537–552.
- [38] B.R. Ferrer, S. Iarovy, L. Gonzalez, A. Lobov, J.L.M. Lastra, Management of distributed knowledge encapsulated in embedded devices, *Int. J. Prod. Res.* (2015) 1–18.
- [39] M.J.A.G. Izaguirre, A. Lobov, J.L.M. Lastra, OPC-UA and DPWS interoperability for factory floor monitoring using complex event processing, in: 2011 9th IEEE International Conference on Industrial Informatics (INDIN), 2011, pp. 205–211.
- [40] Borja Ramis Ferrer, A proposal of decentralized architecture for OKD-MES, in: S. Strzelczak, P. Balda, M. Garetti, A. Lobov (Eds.), *Open Knowledge-Driven Manufacturing & Logistics, the ESCOP Approach*, Warsaw University of Technology Publishing House, Warsaw, 2015, pp. 331–340.
- [41] V.V. Vyatkin, J.H. Christensen, J.L.M. Lastra, OOONEIDA: an open, object-oriented knowledge economy for intelligent industrial automation, *IEEE Trans. Ind. Inform.* 1 (1) (2005) 4–17.
- [42] I.M. Delamer, J.L.M. Lastra, Service-oriented architecture for distributed publish/subscribe middleware in electronics production, *IEEE Trans. Ind. Inform.* 2 (4) (2006) 281–294.
- [43] S. Karnouskos, A.W. Colombo, T. Bangemann, Trends and challenges for cloud-based industrial cyber-physical systems, in: A.W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, J.L. Lastra (Eds.), *Industrial Cloud-Based Cyber-Physical Systems*, Springer International Publishing, Cham, 2014, pp. 231–240.
- [44] B.B. Jay Lee, A cyber-physical systems architecture for industry 40-based manufacturing systems, *SME Manuf. Lett.* (2014).

VII

MANAGEMENT OF DISTRIBUTED KNOWLEDGE ENCAPSULATED IN EMBEDDED DEVICES

by

Borja Ramis Ferrer, Sergii Iarovy, Luis Gonzalez, Andrei Lobov, Jose L Martinez Lastra, September 2016

International Journal of Production Research. Volume: 54, Number: 18, pp 5434-5451

Reproduced with permission: 'Management of distributed knowledge encapsulated in embedded devices' by Borja Ramis Ferrer, Sergii Iarovy, Luis Gonzalez, Andrei Lobov & Jose L. Martinez Lastra International Journal of Production Research Vol 54:18 pp. 5434-5451 (2016). This is the authors accepted manuscript of an article published as the version of record in International Journal of Production Research on 17th December 2015. www.tandfonline.com/ <http://dx.doi.org/10.1080/00207543.2015.1120902>



Management of distributed knowledge encapsulated in embedded devices

Borja Ramis Ferrer, Sergii Iarovy, Luis Gonzalez, Andrei Lobov & Jose L. Martinez Lastra

To cite this article: Borja Ramis Ferrer, Sergii Iarovy, Luis Gonzalez, Andrei Lobov & Jose L. Martinez Lastra (2015): Management of distributed knowledge encapsulated in embedded devices, International Journal of Production Research

To link to this article: <http://dx.doi.org/10.1080/00207543.2015.1120902>



Published online: 17 Dec 2015.



Submit your article to this journal [↗](#)



View related articles [↗](#)



View Crossmark data [↗](#)

Management of distributed knowledge encapsulated in embedded devices

Borja Ramis Ferrer*, Sergii Iarovy, Luis Gonzalez, Andrei Lobov and Jose L. Martinez Lastra

Factory Automation Systems and Technologies Laboratory (FAST-Lab.), Tampere University of Technology, Tampere, Finland

(Received 10 April 2015; accepted 9 November 2015)

Embedded electronic devices are now to be found everywhere. In general, they can be used to collect different sorts of data (e.g. on temperature, humidity, illumination and locations). In some specific domains, such as industrial automation, embedded devices are used for process control. The devices may have a programme that can respond immediately to environmental changes perceived through sensors. In the control of large sites, where there are many devices, higher level decisions are made or processed in dedicated computers far away from the sources (devices) where the initial data are collected. This article shows how it is possible to manage portions of distributed knowledge, hosted in embedded devices, making it possible for each embedded device to hold and manage its piece of knowledge. In addition, presented approach allows keeping locus of control at the embedded device level, where the embedded device can make decisions knowing the status of the rest of the world, device contributions and their effects in the overall distributed system knowledge base.

Keywords: knowledge-based systems; ontologies; distributed knowledge bases; distributed query processing; industrial automation

1. Introduction

Embedded devices used as controllers on the factory floor are beginning to have sufficient computational resources to perform additional tasks besides their direct control functionality. This includes, for example, the handling of communications. Traditional industrial communication technology, such as Profibus,¹ has a data rate of about 500 kbps. With the advent of Internet technologies, the data rates went up to 100 Mbps and more. As the Internet-based technologies become widespread, the cost of implementing them in devices has also dropped, making it possible to adapt Internet-inspired protocols for use in industrial communication. The problem of communication overheads, which generally have to be minimised, particularly for industrial communications, has become less important since the data rate has grown more than 200 times (if we compare data rates of 500 kbps with 100 Mbps), thus in principle allowing larger overheads for the same or even greater communication performance. At the same time, the adoption of the Internet-based protocols has reduced the effort needed to create distributed networked applications, making it possible to deliver a message sent by an industrial controller connected to the equipment at the factory floor to any place in the global network (Alexander et al. 2013; Puttonen, Lobov, and Lastra 2013b).

In addition to the growing communication and networking capabilities, the processing power of CPUs at industrial controllers has also grown. This has made it possible to build an affordable device with sufficient resources to perform deterministic control and networking functions. First such commercial devices are already available on the market.²

In parallel with the growth of computational and networking performance, new standards for knowledge representation and reasoning have been developed (Brachman and Levesque 2004). World wide web consortium (W3C)³ published, among others, a specification for a knowledge representation language called web ontology language (OWL). The introduction of a knowledge representation language allows rising of abstraction for application development. It becomes possible for the developer not to think in terms of concrete hardware, where the application should be running, but to focus on what the application should do. In principle, any programming language appearing above the machine code contributes to the same goal. The difference from knowledge representation language such as OWL is that OWL makes it possible to extend the system concepts with time and adapt to the changes in the controlled environment during runtime. OWL realises *Open World Assumption* where things outside of the knowledge base (KB) are treated as unknown and not as false as in the case of *Closed World Assumption* (Yang and Lee-Kwang 2000). Furthermore, the new concepts can be derived through the reasoning process (RP).

*Corresponding author. Email: borja.ramisferrer@tut.fi

The idea of extendibility allows the reorganisation of manufacturing systems during runtime. The knowledge of a machine added to the production line is encapsulated with the controller device in charge of the machine. Thus, once the new equipment is introduced, the overall system KB is extended. In order to reduce system integration effort, the knowledge models and reasoning capabilities are best kept at the device level or as close as possible to the represented and controlled object (e.g. machine).

This article proposes a knowledge distribution and reasoning approach that can be implemented within the synergy of cyber-physical systems (CPS) which, in combination with service-oriented architectures, the industrial internet and cloud computing among other concepts, will establish the complex ground of the fourth generation of the industry (Colombo, Karnouskos and Bangemann 2014). One of the major benefits of approaches like that presented in this article is to make possible the dynamic integration and configuration of devices inhabiting the same networks. Therefore, the solution proposed may be applied to dynamically integrate and change the configuration of manufacturing systems. The rest of the article is structured as follows: Section 2 outlines in detail the research background. Then Section 3 describes the approach for distributed reasoning. The results are described in Section 4. The discussion in Section 5 compares the approach proposed and the results obtained with other works in the field to highlight the achievements of the research work in question. Finally, Section 6 concludes.

2. Research background

Functional classification of the components of factory systems is often represented within a hierarchical structure that forms an automation pyramid, shown in (Harjunoski, Nyström, and Horch 2009). This structure is described in the international standard ISO 62264 (based on ANSI/ISA-95), which defines the concepts required for the integration of dissimilar factory systems. At the top of the hierarchy is the enterprise resource planning (ERP) systems level or the fourth level according to the standardised classification. The systems which provide factory wide planning and other high-level business functions form an ERP system level. Then, Level 3, usually called the manufacturing execution system (MES) level, is located below the ERP level of the automation pyramid. On the MES level the systems provide advanced control over factory, including inventory, resource and energy management, shop floor scheduling, document control, performance analysis and others. The most detailed specification of the MES functions is provided by MESA⁴ in (Kletti 2007). At the bottom of the pyramid are the remaining levels from 0 to 2. These levels correspond to supervisory control and data acquisition (SCADA) and control systems functionalities. Named functionalities of levels 0–2 are tightly bound to shop floor hardware and usually presented within the same layer in the automation pyramid, also described in (Harjunoski, Nyström, and Horch 2009). While in ANSI/ISA-95 the function hierarchy of factory systems is dedicated to support such systems integration, the task remains non-trivial due to the ambiguity of the standards and concepts employed on different factory levels (Alexander et al. 2013; Harjunoski, Nyström, and Horch 2009; Jarovyi, Garcia, and Martinez Lastra 2013; Power and Bahri 2005; Ramis, Garcia, and Martinez Lastra 2013).

The capabilities and availability of computing hardware have in general been constantly increasing following Moore's law. In particular, computer embedded systems are now used in shop floor hardware for controlling production line processes. These embedded devices are also referenced as CPS because they integrate computational and physical resources. Hence, recent advances in CPS allow cross-domain integration and machine-to-machine communication. In fact, the combination of CPS, service-oriented architecture (SOA) and the emergence of cloud computing will bring valuable benefits to the next generation of the Industry, known as Industry 4.0 (Colombo, Karnouskos, and Bangemann 2014).

In the last decade, the computation power of embedded devices has dramatically increased (Wolf 2007), whereas their monetary cost has been contained. This enhancement together with the quantitative improvement of the performance of shop floor-related operations may also make feasible the introduction of additional functionalities, enhancing the connectivity of the systems at the bottom of the automation pyramid and moving the locus of control to embedded systems. More abstract information may be processed and hence more complex decisions may be made in closer proximity to industrial hardware, making the factory system less centralised and as a result more robust and flexible (Colombo et al. 2014; Lastra and Delamer 2006; Zuehlke 2010). Even more benefits of robustness and flexibility may be provided, as 'smart' devices are able to communicate among themselves in order to solve problems. This may be achieved by following the concept of distributed systems (Coulouris, Dollimore, and Kindberg 2005).

As the device functions become more complicated the question of knowledge management gains importance. There are several knowledge representation formalisms such as ontologies, semantic nets, frames and production rules, all introduced in (Brachman and Levesque 2004). Semantic nets are widely employed in the form of linked data to describe semantic web. Ontology-based knowledge representation provides the most diverse toolset for knowledge modelling, combining some properties of other formalisms.

Several languages have been developed to represent ontologies. In fact, (Agyapong-Kodua et al. 2013; Islam, Abbasi, and Shaikh 2010; Kiritsis 2013; Vrba et al. 2011) are studies presenting a review of semantic languages, methodologies and ontology-based technologies for modelling industrial automation systems. These languages are generally grouped as traditional and web standard-based ontology languages. According to several recent research works (Chungoora et al. 2013; Lin et al. 2011; Shen, Wang, and Sun 2012; Zhang et al. 2009), it can be stated that web standard-based ontology languages are dominant for ontology representation in general and in the field of semantic web in particular. Web standard-based ontology languages follow a sublanguages model where the basic language is resource description framework (RDF), which provides a foundation for others. The building block of RDF is a triple, depicting the relation between two concepts. RDF Schema (RDFS) and OWL provide more expressivity for the knowledge model and in the case of some OWL sublanguages even support reasoning (Grau et al. 2008). In order to manipulate the RDF-based ontology querying languages may be employed. SPARQL Protocol and RDF Query Language (SPARQL) is used to extract data from RDF, while SPARQL Update language provides the capability to modify the RDF ontology. Taking into account that RDFS and OWL are based on RDF, RDFS and OWL ontologies may also be queried employing named query languages. Using named standards, it is possible to represent knowledge and access it within different application domains. RDF, OWL, SPARQL and their extensions are standards developed by the W3C.

Based on the knowledge representation concepts and standards described above the research on applicability of this framework for industrial automation was done in (Ramis et al. 2014). The article presents a service-oriented production line. Ontology-based representation of this line is stored and updated in a KB, and service orchestration flow is controlled by the status of the production line model. SPARQL is employed for KB management. The authors claim that the approach may provide benefits of runtime configurability and may be applied to dissimilar industrial systems.

As the functionality of centralised industrial automation systems may be distributed and encapsulated in embedded devices, in some cases it may be reasonable to distribute knowledge together with functions. For low-level processes this may allow closing of the execution loop within the embedded device, as all the required data are placed within the device together with algorithms using the data. This may be also beneficial in dynamically changed systems, which in centralised architecture may need changes at all levels of hierarchy to adapt the changes on one of the levels. Nevertheless, such systems will require a generic approach to access data in the devices. This leads to a need for distributed query processing. The research in querying of distributed sources is performed in related fields such as semantic web (Kumar, Singh, and Verma 2010; Paret et al. 2011; Kurita et al. 2007). Moreover, concepts of Federated queries were provided in SPARQL 1.1 (Buil-Aranda et al. 2013).

The present work contributes to the production research with an approach that illustrates how the encapsulated knowledge in embedded devices can be managed, reasoned and distributed throughout a network, which is implemented as a private cloud of such devices that control the equipment of manufacturing systems. All this is achieved by the utilisation of semantic technologies, which are now employed in production lines for service and system descriptions.

3. Approach

This research presents an approach to managing distributed knowledge encapsulated in embedded devices located in the lowest layer of the ANSI/ISA-95 automation pyramid. The approach is based on a goal-oriented mechanism for the integration of computational resources. The proposal describes how to utilise and merge the overall system knowledge that is distributed among different devices. Thus, this solution is beneficial in some specific domains, such as industrial automation where the devices use system knowledge representation for control processing. This section describes the approach, including diagrams on communication, architecture and the behaviour of devices which can be used to reproduce the results of the work.

3.1 Description of a distributed system

In order to design a generic approach and provide more robustness in the solution, it is considered that the distributed system by default has no hierarchy. This means that initially there is no master device working as an entry point of the system. In other words, no device works as a system gateway. Moreover, all the devices have the same type of communication channels. Thus every device can communicate with any other networked device. Conceptually the way networked elements are interconnected is defined as topology. There are many network topologies as described in (Sosinsky 2009).

From a logical topology perspective, this approach follows a mesh architecture because each device can communicate with other devices, permitting the distribution of data. In fact, such a topology provides robustness for the system, because if any of the devices fails, the communication between others remains unaffected. Nevertheless, this approach is

also achievable using bus topology in which all devices communicate through a common backbone. The same advantage of failure robustness is experimented within this topology. Therefore, the decision on working with mesh or bus topologies will mostly be determined by the technology utilised in the communicating devices.

In some specific domains, such as industrial automation, it is necessary to know the current state of the system for controlling and monitoring processes. In distributed systems, the state data are spread among networked devices. In knowledge-based driven implementations, knowledge is retrieved by querying system KBs. Therefore, the applications in charge of controlling and monitoring tasks must issue, e.g. SPARQL queries in order to retrieve relevant data from the devices (Ramis et al. 2014).

In this approach, an external actor, which can be a person or a third party application, can interact with the distributed system to control or monitor the data of the system. This interaction is achieved through an element called the operator interface (OI). The OI can be hosted in any server, or even nested in an embedded device. Moreover, the OI is able to communicate with every networked device of the distributed system. The main task of the OI is to allow the insertion of SPARQL queries that will be executed by distributed system devices. The algorithm implemented on the devices which allows the teamwork for executing queries is described Section 3.3.

3.2 Device architecture

Devices are capable of interacting with the external world by means of sensors and actuators. Using sensors, relevant data are acquired and stored. On the other hand, actuators allow the device to alter the system, performing physical actions. It is important to note that the concept of embedded devices implies that these units are constrained in computational resources. Therefore, this approach focuses on designing and implementing lightweight components and internal interactions to avoid overloading the computational resources of the device.

Internally, three modules compose the embedded device: *Sensor & Actuator Processor*, *Ontology Service* and the *Device & Operator Interface Processor*. The device architecture is shown in Figure 1.

The *Sensor & Actuator* (SA) processor is in charge of collecting the device sensors' information. In addition, it requests the ontology service to store data in the device KB. To achieve this, the SA processor reads the sensors' values and transforms them into SPARQL Update queries. These queries are sent to the *Ontology Service*. Moreover, the SA processor can also be used for the device to send an action to activate actuators.

The *Ontology Service* is responsible for two main tasks: (1) holding and managing the device KB and (2) exposing the KB to other device modules. The *Request Processor*, which is an endpoint of the *Ontology Service*, has the task of handling SPARQL queries over HTTP ("SPARQL 1.1 Graph Store HTTP Protocol" 2013), so that they can be executed using the device KB. Therefore, using this endpoint, the SA processor is capable of mapping the world state perceived by the sensors and described in the device KB. The present approach proposes the use of KBs, which are described in RDF-based languages.

Other networked devices, third party applications or individuals can request the device knowledge through a module called *Device & Operation Interface Processor*. This module works as another network endpoint, which also handles SPARQL queries over HTTP. Incoming queries arriving at this endpoint may come from other devices or from the OI, as explained in the preceding section.

3.3 Management of distributed knowledge

In order to control and monitor system processes, pieces of system knowledge are represented in separate KBs, which are hosted in embedded devices. This article shows how it is possible to manage distributed knowledge, making it possible for each embedded device to manipulate its respective piece of knowledge. The management of local information must allow devices to support processes performed by all available devices as a unique system.

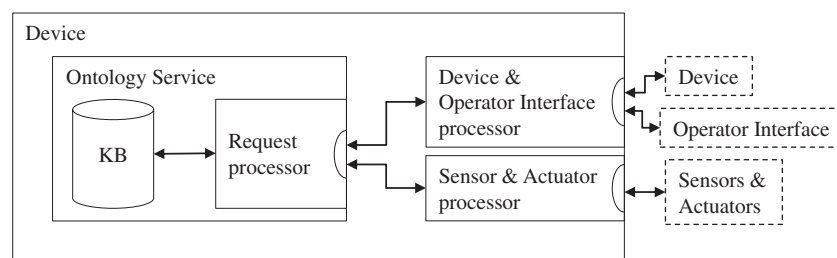


Figure 1. Architecture of the device.

To achieve this purpose, the essential behaviour of devices for facing client requests must be determined. This conduct is presented in this article through three UML⁵ sequence diagrams.

3.3.1 Device initialization

Devices must be able to introduce themselves to other devices in the network. In fact, devices include, as part of local resources, information on all system devices. The reason for this is that addresses and identifiers of networked devices are required for query distribution. The inclusion of device information is done when a new device is connected to the distributed system. Figure 2 presents a sequence diagram showing the conduct of a new device when it joins the network.

Fundamentally, when a new device enters in the system, it sends its address and identifier. Therefore, new devices can be reachable by other peers. Once any device is initialized in the network and reachable by the OI, it can participate in client requests.

3.3.2 RP performance

In the approach proposed, a RP is defined as a series of actions to produce representation of new statements by manipulating symbols of believed propositions. In other words, RP is a process whereby a system uses its KB to respond to queries. Since the approach deals with pieces of all the knowledge encapsulated (or hosted) in devices, a single RP is divided into subprocesses, which are performed by different devices. A subprocess of a RP is known as a reasoning process portion (RPP). In RPPs, devices execute incoming queries using their own KBs. Hence, this approach can also be defined also as a teamwork-based mechanism, by which an incoming query is distributed and then solved as a set of RPPs.

Any RP execution needs one device to assume a leadership role. The *requester device* role designs a device that takes responsibility for managing a single RP. The sequence diagram shown in Figure 3 presents how any available device can assume this role.

Whenever a client submits a request in the OI, a demand to take the requester device role is sent to all known system devices. Then those devices that are available to take the leadership role will send a response to the OI. A candidate for this role will take it if it receives a query from the OI, namely the query sent by a client. It should be noted that the requester device is not excluded from participating in the RP, but is the one that will lead the execution of the RP. Devices will respond to a requester device role demand when:

- (1) The device has no requester device role in any ongoing RP execution.
- (2) The device has enough resources (i.e. memory) to handle a RP execution.
- (3) The device responds to the demand before it knows that some other device has been designated as a requester device.

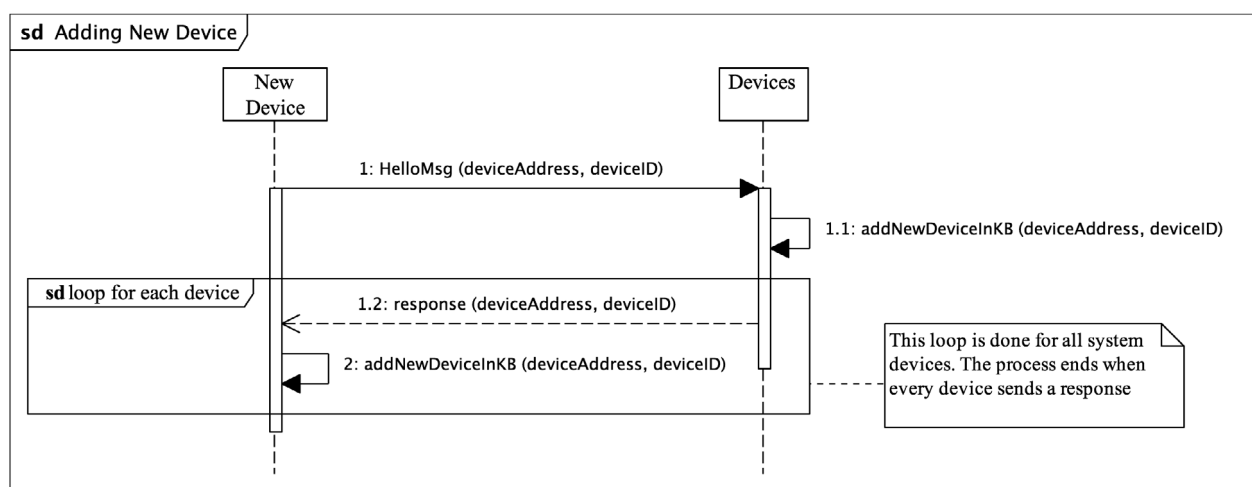


Figure 2. Sequence diagram of adding a new device.

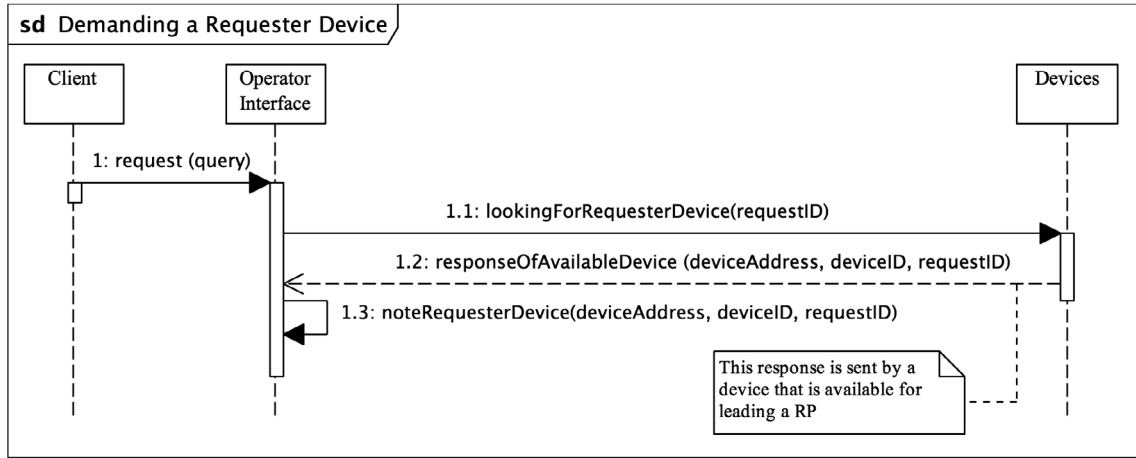


Figure 3. Sequence diagram of demanding a requester device.

Once there is a requester device in the network, an RP can be solved. This means that if a device has correctly assumed the requester device role for a query_x, the RP_x can be started. The sequence for any RP performance is depicted in Figure 4.

Although the beginning of the sequence is the OI query sending to the requester device, it can be considered that the RP is started and finished by the requester device. However, the diagram shows the request emission from OI for a task that is an immediate continuation of the OI behaviour, presented in Figure 3.

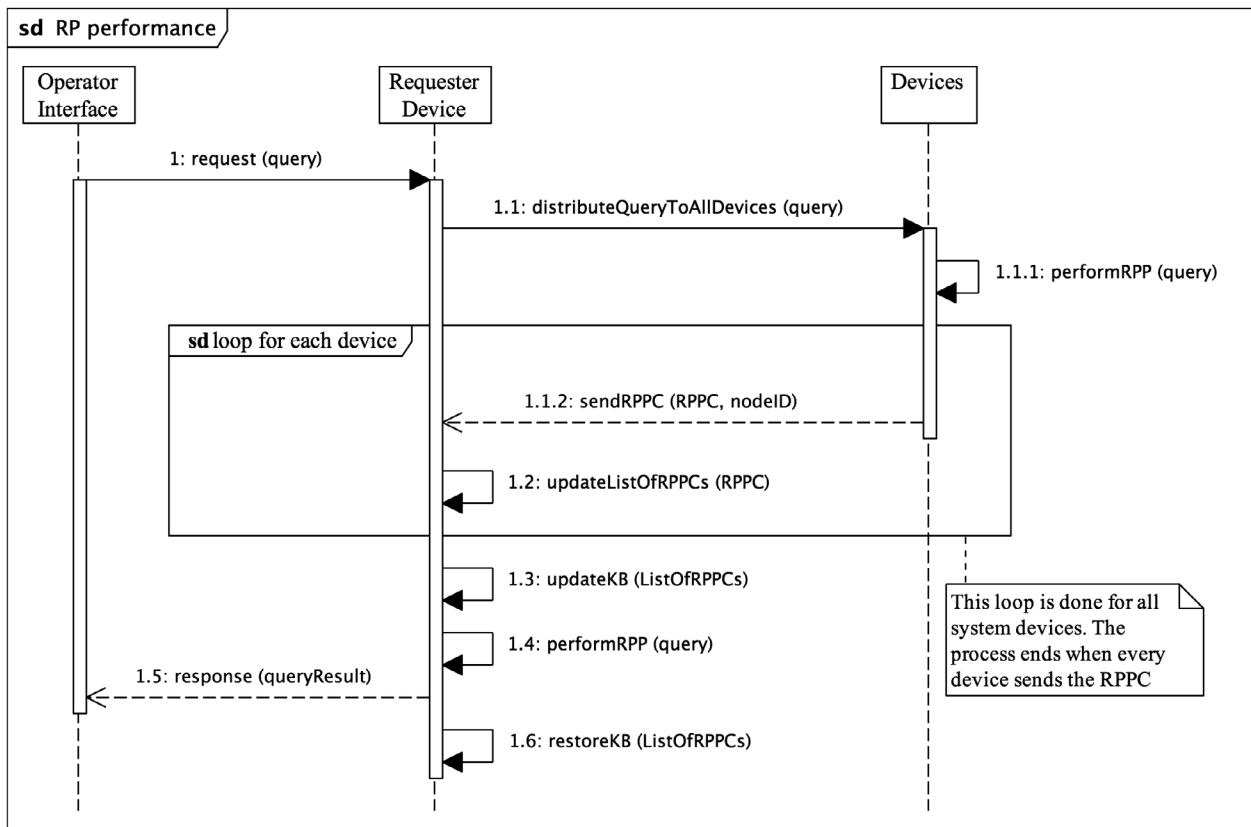


Figure 4. Sequence diagram of an RP performance.

Firstly, the incoming client query is sent to the requester device. Then, the requester device takes the lead in the RP and sends the query to other devices, which will perform an RPP using their own resources. Once RPPs have been performed, each participating device returns its corresponding reasoning process portion conclusion (RPPC) to the requester device. Thus, RPPC is defined as a response that contains the result of a distributed query execution, which is sent from devices to the requester device. Then the requester device updates its own KB with each RPPC that it receives. Once the requester device KB has been updated with all the RPPCs, the last RPP is performed. The result of the last RPP is the final result of the incoming query. Then the requester device sends this result to the OI. Afterwards the requester device will delete the updates done after RPPC collection to restore its KB to the initial state and be ready to participate in the next RP.

One benefit of this approach is that if the requester device does not have the knowledge to solve a certain incoming query it will be able to give a result because the RP finalises by executing the query using a KB, which is updated with RPPCs sent by other network devices. Thus, although system knowledge is encapsulated and only accessible to owners, other devices always receive the information that is required to perform RPs.

3.3.3 Device situations for query execution

The preceding sequence diagram execution ensures that any device which distributes a query to be executed receives useful information in response. For this reason, the correct behaviour of devices permits them to send the maximum amount of information to be used by requesters.

In the approach described devices can be involved in three different situations when executing queries: total support situation (TSS), partial support situation (PSS) and no support situation (NSS). These situations depend on the resources, or local KB, available by each embedded device. Then the manner in which devices act in RPPs execution will determine the type of RPPC returned to the requester device. Each situation is described below:

- Total support situation:
 - Resources in the device are sufficient
 - The device is able to execute the query using its own knowledge
 - The generated RRPC is an update query with statements on the RPP result
- Partial support situation:
 - Resources in the device are insufficient
 - The device has some knowledge but not sufficient to execute the query
 - The generated RRPC is an update query with useful statements from the local KB
- No support situation:
 - Resources in the device are non-existent
 - The device has no knowledge about the query
 - The generated RRPC is a 'no support message'

Each situation name described above reflects the type of support that a device can offer. The three reactions are implemented on devices because different requests may involve the same device in different situations. Then embedded devices must be able to send the RPPC type that corresponds to the device situation.

A TSS situation occurs when a device is able to execute the incoming query, obtaining a complete result using its own KB. A TSS does not imply that the result provided by one device is the final result of an entire RP. A device may achieve a result when it executes a query with local resources that is different for the overall system KB. This is because when the requester device receives all RPPCs, it executes the distributed query, but with all the system information.

PSS and NSS occur when the device is not able to produce a result for the distributed query. It should be noted that devices do not identify beforehand in which situation they are involved. First, when a query arrives at a device interface, the query is processed and executed. Then TSS will be determined if a result is obtained, and RPPC will be performed from the result. Otherwise, devices will analyse the query looking for data that match any term of the query. If the device finds information to be sent, it will determine that is involved in a PSS and will create a RPPC containing useful data for the requester device. On the other hand, if there is nothing to be sent, the device will conclude that is in an NSS and will just send a message communicating its inability to support in the ongoing RP.

3.3.4 From queries to updates: the RPPC execution

According to previous explanations, RPPC formation is important in RP execution because it is the manner of devices to respond to requests. Thus, embedded devices analyse queries and reuse the query syntax to produce RPPCs, which in reality are another type of query. More specifically, devices transform SPARQL queries into SPARQL Update queries. This process requires processing the queries and is performed differently in each situation. The steps to be performed in TSS are:

- (1) Variables or queries are substituted by their value obtained after query execution
- (2) Prefixes of statements are substituted by their value and the required prefixes for datatypes are also added
- (3) If a variable is a class type, an extra statement is created to also insert its type
- (4) The SELECT clause is changed to the INSERT clause
- (5) Functions not allowed by the SPARQL 1.1 Update standard are removed, for instance, FILTER.

As a representative example, Figure 5 shows the transformation of a SELECT query into an INSERT query. It should be noted that the query transformation could be performed after executing the incoming query since the value of the variable is required. The reason is that INSERT query statements do not include unknown variables because they contain only valuable data to be inserted.

On the other hand, the steps to be performed in PSS are:

- (1) Different SELECT queries are created and executed, one per statement
- (2) Steps *i* to *iv* of the TSS case are performed for each executed SELECT query that gives any result.

It should be noted that the devices would conclude that they are in NSS when the first step of PSS in query transformation is performed. This is because the device will not find any result when created SELECT queries are executed, meaning that the device KB does not contain any useful information to be included in the RPPC. Hence, in the absence of results, the device will create a non-support message and send it in response to the requester.

Summing up the above steps, accomplishing the steps that correspond to a device situation, RPPCs are created from incoming SPARQL queries. It should be noted that the approach proposed is limited to transforming SPARQL SELECT queries into SPARQL Update INSERT queries.

4. Results

4.1 Test case set-up

For testing purposes, the system presented in Figure 6 was implemented. This small-scale system implementation makes it possible to prove the concept of the approach proposed. The implementation presented shows a restriction with respect to the approach proposed: the OI is connected to only one device. This means that the decision on the requester

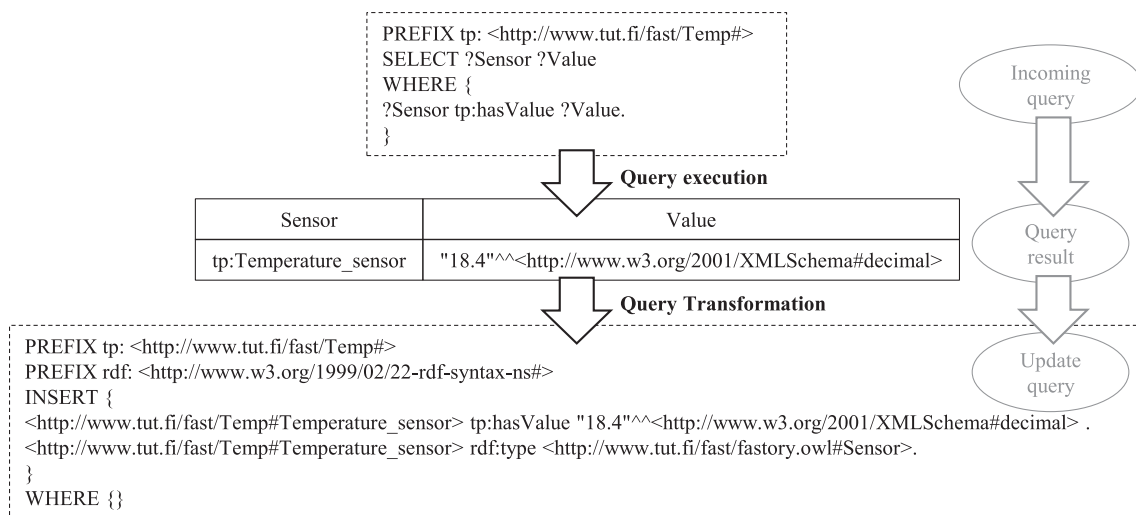


Figure 5. Transformation of queries.

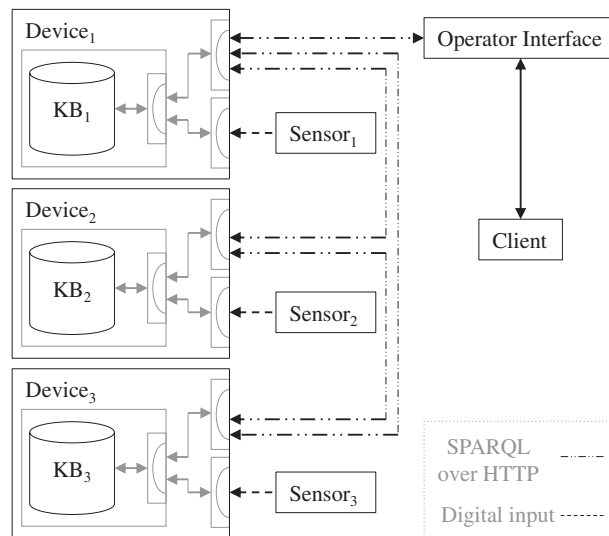


Figure 6. Testing case implementation.

device is pre-defined. Hence, the sequence shown on Figure 3 is not illustrated, but rather focuses on other aspects of the approach. This limitation does not affect the main objective of the approach, which is the management of distributed KBs encapsulated in embedded devices. Thus, the sequence for RP performance shown in Figure 4 has been successfully implemented.

The testing implementation shows how a client is connected to the OI. This link permits queries sent by a user to be executed by the distributed system. Then the interconnection between the OI and *Device₁* is used for redirecting the incoming query from client to devices. It should be observed that for the implemented case, *Device₁* will be always the requester device.

Devices follow the architecture presented in Figure 1. Thus, they are capable of handling device-to-device connections (implemented via SPARQL over HTTP) and device-to-sensor or device-to-actuator connections (through digital/analogue I/Os). Moreover, each device contains its own KB, which can be updated when there are changes in the statuses of connected components. The sensors are connected to devices.

Raspberry Pi⁶ has been selected as the embedded device for testing the approach. This was chosen mainly because it offers a fast prototyping stage. More precisely, the selected model is Raspberry PI B, which includes a 512 MB of RAM memory and a 700 MHz processor. Moreover, the devices have been loaded with Raspbian,⁷ which is a light-weight instance of Linux. Note that the sensors are connected to the devices using the general-purpose I/O (GPIO) of Raspberry PI and using resistor components to limit the current drain.

4.2 Proving the concept

With the objective of showing representative cases of use, the following subsections describe two different experiments. Each test is situated in a different scenario, in which several sensors are used for obtaining measurements. In each test case, the devices host a different ontological model (implemented within protégé⁸ ontology editor), describing the domain in which the system devices reside.

Each test case is described separately. Furthermore, models are presented and important aspects of each experiment are detailed so that the tests can be reproduced. Moreover, besides the final result of the query execution process, the inner process steps, i.e. the queries and transformations used, are also shown for each case. All the queries used in the following research experiments are shown in Figure 7, in which each query is preceded by a commented line containing its corresponding title.

4.2.1 First experiment: measuring the overall room temperature

The first experiment is a simple situation in which a large room is equipped with three temperature sensors. Moreover, the room has a display showing the temperature of the room. Most probably the temperature given by each sensor does

Query 1 - Client request for the first experiment:

```
PREFIX tp: <http://www.tut.fi/fast/Temp#>
SELECT ?Sensor (AVG (?value) AS ?Temperature_average)
WHERE {
  ?Sensor tp:hasValue ?value.}
GROUP BY ?Sensor
```

Query 2 - Generated RPPC query by $Device_2$ in the first experiment:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX tp: <http://www.tut.fi/fast/Temp#>
INSERT {
  <http://www.tut.fi/fast/Temp#Temperature_sensor> tp:hasValue "18.0"^^<http://www.w3.org/2001/XMLSchema#decimal>.
  <http://www.tut.fi/fast/Temp#Temperature_sensor> rdf:type <http://www.tut.fi/fast/Temp#Sensor>. }
WHERE { }
```

Query 3 - Client request for the second experiment:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX winery: <http://www.tut.fi/fast/Storage#>
SELECT ?Temperature ?Light ?Humidity
WHERE {
  ?MeasTemp rdf:type winery:Temperature.
  ?MeasTemp winery:hasValue ?Temperature.
  ?MeasLight rdf:type winery:Light.
  ?MeasLight winery:hasValue ?Light.
  ?MeasHumidity rdf:type winery:Humidity.
  ?MeasHumidity winery:hasValue ?Humidity}
```

Query 4 - Generated RPPC query by $Device_2$ in the second experiment:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX winery: <http://www.tut.fi/fast/Storage#>
INSERT {
  <http://www.tut.fi/fast/Storage#Value2> rdf:type <http://www.tut.fi/fast/Storage#Humidity>.
  <http://www.tut.fi/fast/Storage#Sensor2> winery:hasValue "60.0"^^<http://www.w3.org/2001/XMLSchema#decimal>.}
WHERE { }
```

Figure 7. Queries used and generated in the experiments.

not match. Thus, to display the room temperature, the system must use the values of each device and conclude an overall result. It should be noted that for mapping the components of Figure 6 to this example, the display is the system client and each temperature sensor is controlled by one device.

In the first scenario, each KB hosted by a device contains an ontology model with the class and instance distribution shown in Figure 8. An instance (or individual) is understood as a specific realisation of a class (or object). The ontology model includes a class named *Sensor*. Each model contains a defined *Sensor* class instance named *Temperature_sensor*, which is the sensor connected to the device. Finally, the model has defined a datatype property named *hasValue*. This datatype property is the one that is updated by the devices when the sensor temperature changes. In fact, each device is connected to a different temperature sensor, which means that ontology models will be updated with different sources of data. Table 1 shows the sensor values at the moment of doing the experiment and indicates the resources on devices for executing a query to retrieve a temperature value (shown in *Query 1* of Figure 7). For such a query and situation, all the devices have sufficient knowledge resources. Thus, each device will be involved in a TSS, meaning that they can support the RP with RPPCs created from a complete result after the query execution. Note that the performance of this test follows the sequence diagram presented on Figure 4.

The process starts when the system client sends the first SPARQL query (*Query 1*) shown in Figure 7 to the OI. The query is used to request an average of temperature values. Once the query arrives at the OI, it is redirected to *Device₁*, which is the requester device for the RP to be performed.

Then the RP begins when *Device₁* forwards the query to the other devices. Each device will be requested to participate in the RP. Because each device executes the query for different KBs, each device will create different RPPCs. Note that the RPPC performance depends on the device situation. As an example, the second query of Figure 7 (*Query 2*) is the RPPC created in *Device₂*.

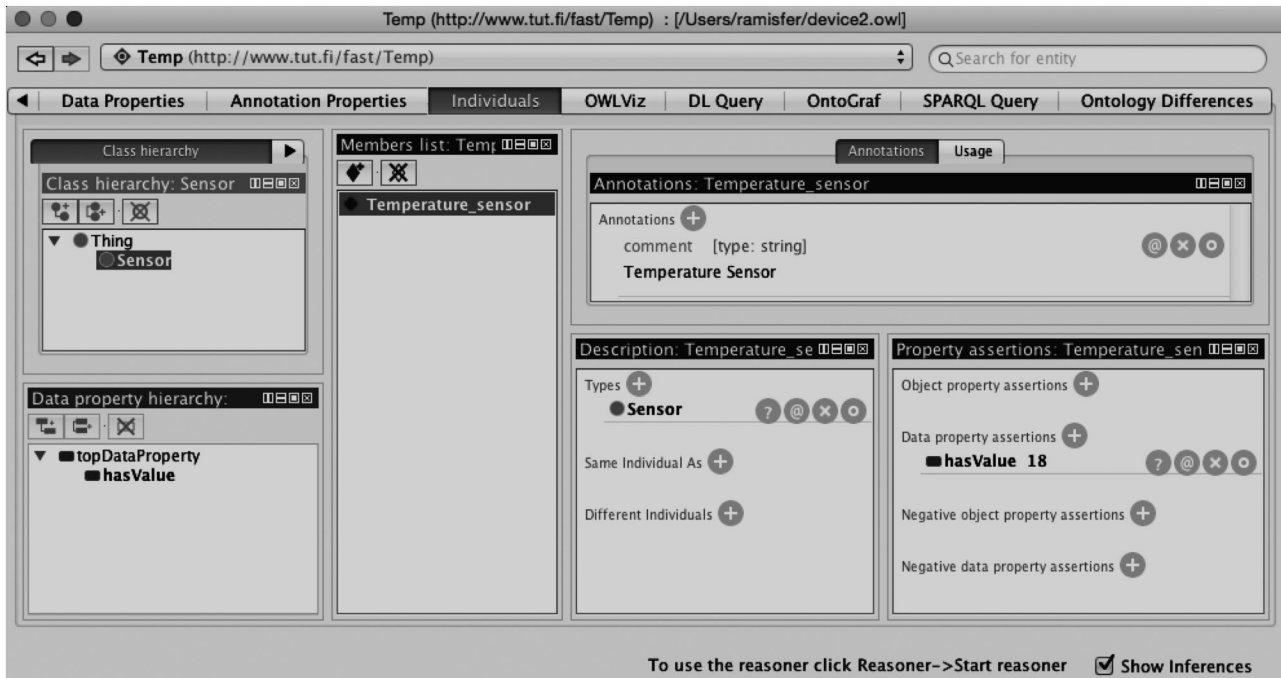


Figure 8. Ontology model implementation in the first experiment.

Table 1. Resources available and sensor values of devices in the first experiment.

| Device | Resource in device | <i>hasValue</i> data type property value |
|---------------------|--------------------|--|
| Device ₁ | Sufficient | "19.1"^^<http://www.w3.org/2001/XMLSchema#decimal> |
| Device ₂ | Sufficient | "18.0"^^<http://www.w3.org/2001/XMLSchema#decimal> |
| Device ₃ | Sufficient | "18.4"^^<http://www.w3.org/2001/XMLSchema#decimal> |

Afterwards, *Device₂* and *Device₃* send the created RPPCs to *Device₁*. Then the requester device performs the last RPP and achieves the final result, which is shown in Table 2. The values presented compose the result obtained in the requester device when the RP is complete. The final result of the temperature is equal to 18.5. The result is correct because $(19.1 + 18.0 + 18.4) / 3 = 18.5$. Hence, it is concluded that the process has ended successfully.

4.2.2 Second experiment: monitoring the conditions in a wine cellar

The second experiment is conducted in a more specific environment, a wine cellar. Fine wines need long-term storage, allowing them to improve. However, for a wine to acquire certain properties that enhance its quality, it requires special storage conditions. Efficient storage of wine demands controlling at least three factors of the space: temperature, humidity and light. Hence, the approach proposed can be useful for monitoring the measurements of different sensors which detect changes in storage conditions. This experiment proposes having several sensors that update corresponding instances of KBs, encapsulated in devices each time that there is a change in the value being measured.

Table 2. Final query result of the first experiment.

| Sensor | Temperature_average |
|-----------------------|---|
| tp:Temperature_sensor | "18.5" <http://www.w3.org/2001/XMLSchema#decimal> |

To describe this scenario using the approach proposed, each device hosts an ontological model with the hierarchical distribution of classes, which is illustrated in Figures 9 and 10. In addition, both figures show that the model defines two different properties. First, an object property called *hasMeasurement* links *Sensor* with *Measurement* object instances. Second, a data property named *hasValue* is used to describe the value of any *Measurement* class instance.

Just as shown in Figure 8, each sensor is represented by an instance. In this experiment, each sensor individual is defined in the *Sensor* subclass that corresponds to its sensor type. For instance, a temperature sensor will be represented in the model as an instance called *Sensor1* defined as an individual of the *TemperatureSensor* class. On the other hand, this model has a different distribution of properties from the first experiment case. In this test there is another individual in the *Measurement* class, which represents a measurement performed by a sensor. In the example, the ontological model hosted in the device that controls the temperature includes an individual under the *Measurement* class named *Value1*. The complete list of individuals described in all system devices is presented in Table 3. It should be noted that the table also presents the *hasMeasurement* datatype values and their type at the moment of conducting the second experiment.

It is important to consider that the model proposed could be reduced on each device. Fundamentally, each device could delete the concepts that are not in use. Nevertheless, Figure 9 presents the complete domain model including all the modules. For example, the model hosted by a device which controls the light sensor can reduce the model by deleting the *TemperatureSensor*, *HumiditySensor*, *Temperature* and *Humidity* classes. This model reduction does not affect the RP performance because the devices only update instances of their own KB.

The reason why the classes' distribution, instances definition and properties relations have been designed as described is also to test the research with a reasoner. Hence, to ensure that the approach proposed works correctly with inferred data, an axiom has been defined (shown in *Description* of *TemperatureSensor* class in Figure 9). This axiom is included in each *Sensor* subclass in the following form:

The screenshot displays the Protégé ontology editor interface. The main window title is "Storage (http://www.tut.fi/fast/Storage) : [/Users/ramisfer/device1.owl]". The top navigation bar includes tabs for "Entities", "Classes", "Object Properties", "Data Properties", "Annotation Properties", "Individuals", "OWL Viz", "DL Query", "OntoGraf", "Ontology Differences", and "SPARQL Query".

On the left side, there are three panels:

- Class hierarchy:** Shows a tree structure starting with "Thing", containing "Measurement", "Humidity", "Light", "Temperature", "Sensor", "HumiditySensor", "LightSensor", and "TemperatureSensor".
- Object property hierarchy:** Shows "topObjectProperty" with "hasMeasurement".
- Data property hierarchy:** Shows "topDataProperty" with "hasValue".

The central pane, titled "Members list: Sens", shows a list of individuals, with "Sensor1" selected.

On the right side, there are several panes:

- Annotations: Sensor1:** Shows an annotation "comment [type: string] Temperature Sensor".
- Description: Sensor1:** Shows "Types" as "TemperatureSensor", "Same Individual As", and "Different Individuals".
- Description: TemperatureSensor:** Shows "Equivalent To" as "hasMeasurement only Temperature", "SubClass Of" as "Sensor", "General class axioms", "SubClass Of (Anonymous Ancestor)", "Members" as "Sensor1", "Target for Key", "Disjoint With", and "Disjoint Union Of".
- Property assertions: Sensor1:** Shows "Object property assertions" as "hasMeasurement Value1", "Data property assertions", "Negative object property assertions", and "Negative data property assertions".

At the bottom right, there are checkboxes for "Reasoner active" and "Show Inferences".

Figure 9. Ontology model implementation on the second experiment showing the *Sensor1* instance.

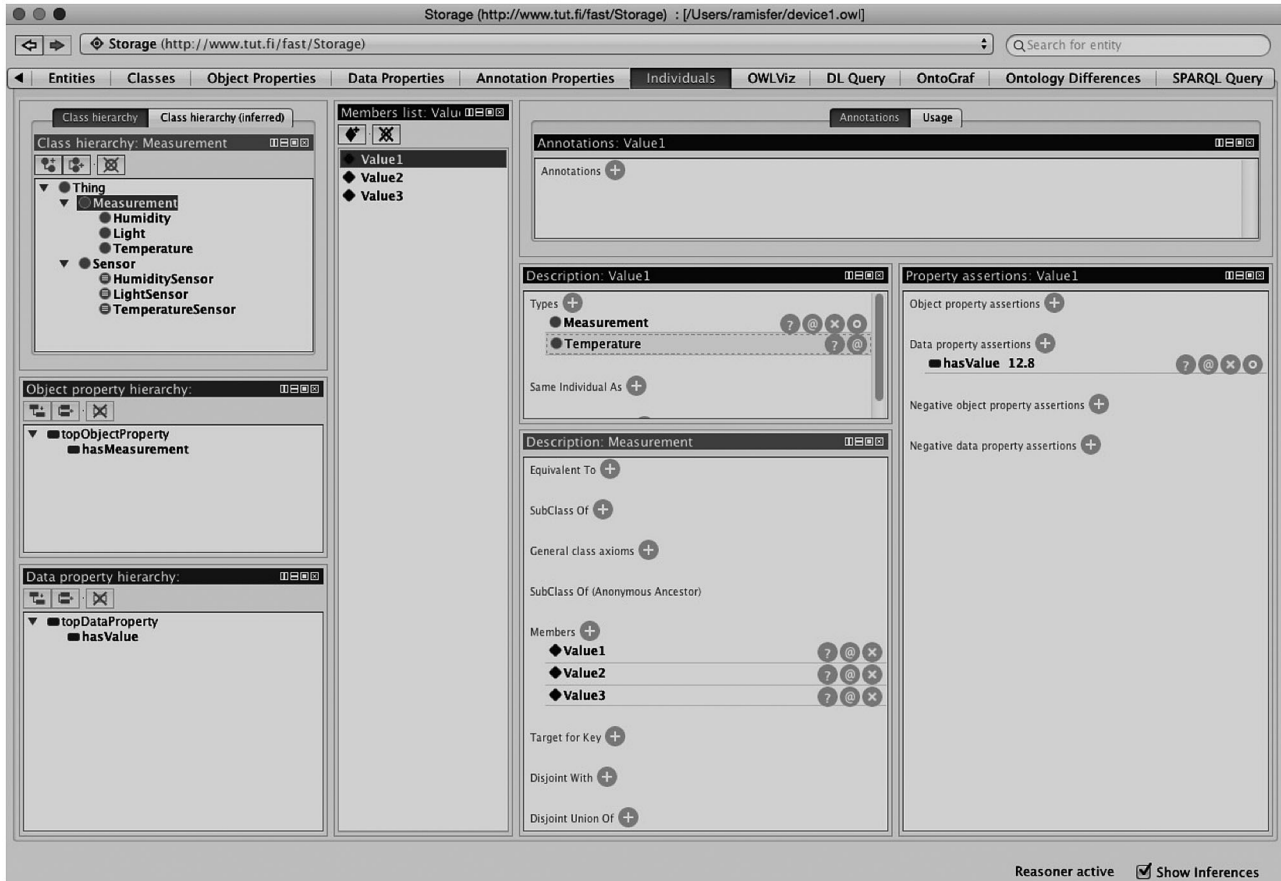


Figure 10. Ontology model implementation in the second experiment showing the *Value1* instance.

Table 3. List of ontology instances hosted by each device and their associated measurement value in the second experiment.

| Device | Classes | | hasMeasurement datatype value and its type | | |
|---------------------|---------------------|--------------------|--|----------------|-----------------|
| | Sensor | Measurement | Temperature (type) | Light (type) | Humidity (type) |
| Device ₁ | Sensor ₁ | Value ₁ | 12.8 (decimal) | – | – |
| Device ₂ | Sensor ₂ | Value ₂ | – | 60.0 (decimal) | – |
| Device ₃ | Sensor ₃ | Value ₃ | – | – | 76.2 (decimal) |

ObjectA hasMeasurement only *ObjectB*, where *ObjectA* determines a subclass of *Sensor* and *ObjectB* is a subclass of *Measurement*. For example, *LightSensor* hasMeasurement only *Light* states that a light sensor can only measure light values. Hence, a reasoner will infer that if e.g. *Sensor1* instance of *TemperatureSensor* object hasMeasurement *Value1* instance of *Measurement* object, the measurement *Value1* is a *Temperature* instance type. This example is depicted in *Value1* Description in Figure 10. The reasoning engine used for this experiment was *Pellet* (Sirin et al. 2007).

Table 4 presents the knowledge that devices have for certain query execution (shown in *Query 4* of Figure 7), which retrieves at one time the measurement values of temperature, humidity and light. Therefore, each device is involved in a PSS, meaning that RPPCs are not created from a complete result after query execution. In turn, RPPCs are created after collecting useful statements and putting them together in the corresponding INSERT query, as explained in Section 3.3.4. It should be noted that the performance of the second experiment also follows the sequence diagram presented on Figure 4.

The experimental RP process starts when a client sends the third SPARQL query shown in Figure 7 (*Query 3*). The query, which is inserted in the OI, is used to request temperature, humidity and light measurement values. Once the query arrives at the OI, it is redirected to *Device1*, which is the requester device for the RP that is performed.

Table 4. Resources available and sensor values of devices in the second experiment.

| Device | Resource in device | <i>hasValue</i> datatype property value |
|---------------------|--------------------|--|
| Device ₁ | Insufficient | “12.8”^^<http://www.w3.org/2001/XMLSchema#decimal> |
| Device ₂ | Insufficient | “60.0”^^<http://www.w3.org/2001/XMLSchema#decimal> |
| Device ₃ | Insufficient | “76.2”^^<http://www.w3.org/2001/XMLSchema#decimal> |

The RP begins when *Device*₁ sends the query to the other devices. Afterwards, each device performs an RPP to support an RPPC. As the RPPC performance depends on the device situation, the fourth query of Figure 7 (*Query 4*) exemplifies the RPPC created in *Device*₂. Any INSERT query created by a device involved in a PSS must follow the corresponding steps presented in Section 3.3.4.

In the same way, other devices contribute to the RP with RPPCs containing the value and type of the corresponding controlled sensor. The final RPP done by the *Device*₁ is performed within a model that includes all the useful information hosted in *Device*₂ and *Device*₃. Hence, in the second experiment the result for the incoming query is achieved, thanks to the contribution of all devices involved in PSS. Finally, the values that compose the result given by the requester device (or *Device*₁) once the query has been executed are presented in Table 5. Afterwards, this result is delivered to the OI so the client can receive the response to its request.

5. Discussion

The approach demonstrates how to manage and process distributed knowledge that is encapsulated among devices inhabiting the same system. Although the research background positions the work and potential benefits in the field of automation, the approach might also fit other domains requiring distributed information management. Once the experiments have been presented, it can be concluded that the paper describes a method for managing knowledge that is distributed among networked devices for solving SPARQL requests.

Note that the approach does not require any computational infrastructure apart from the CPUs of the embedded devices. The alternative could be to use cloud computing infrastructure (Camp and Lobov 2014), where the computational resources that may not be available on the factory floor could be outsourced to the cloud. However, this approach still necessitates maintaining the infrastructure. The novelty of the approach proposed is that the whole computational infrastructure is just the embedded devices with knowledge processing capabilities in addition to the ability to perform the deterministic control for the equipment.

A review of the network topology could determine which configuration is more efficient for implementing the mechanism presented. In principle, the research work proposes implementing mesh or bus topologies because of the robustness of these network shapes. Moreover, distributed systems could afford losing a device because the remaining ones might be able to execute the requested task. However, is debatable if this feature justifies the high cost of these configuration types. As an alternative, a ring topology could be beneficial for reducing the data traffic in the network. In fact, the network knowledge representation hosted by each device would be minor because less description is required, thereby decreasing the use of memory in resource-constrained embedded devices. Nevertheless, ring configuration could be problematic if a connection device-to-device fails so that RPs, for example, would immediately stop working.

On the other hand, although the experiment does not exploit other features and capabilities of ontologies, this research work targets new trends in the field of industrial automation. As described, e.g. in (Puttonen et al. 2010) and in (Puttonen, Lobov and Lastra 2013a), the area tends to develop knowledge-driven solutions which manage large data-set models for monitoring and controlling system status. These models are being developed as ontologies. Hence, the approach presented here also offers an alternative to on-going research work in the field of industrial automation (Ramis

Table 5. Final query result of the second experiment.

| Temperature | Light | Humidity |
|--|--|--|
| “12.8”^^<http://www.w3.org/2001/XMLSchema#decimal> | “60.0”^^<http://www.w3.org/2001/XMLSchema#decimal> | “76.2”^^<http://www.w3.org/2001/XMLSchema#decimal> |

Ferrer 2015), which proposes the decentralisation of the knowledge representation that in most of the current MES implementations is kept as a central component.

Further, there are several approaches to distributed knowledge management already available in the literature. Unfortunately, such approaches do not satisfy the needs of the dynamic, heterogeneous and resource constrained environments. The other approaches often provide a different but complementary perspective on knowledge-driven manufacturing systems.

There is an interesting study presenting a distributed query processing method that proposes the partition and distribution of data to several computation nodes (Kurita et al. 2007). It claims that distributed query processing can be optimised by evenly partitioning the data to which the query is applied. Even though the research works with Extensible Markup Language (XML), it can be linked to our research because the ontology models represented in OWL or RDF are, in fact, XML-based. The study by (Kurita et al. 2007) shows interesting results on the partition of data. However, the method is restricted to the industrial automation domain for several reasons. Firstly, in case of manufacturing systems, the data kept in devices is attached to controlled mechanisms. This means that the data used for controlling, e.g. an actuator, is located in the device that is connected to it. Secondly, uniform distribution can be problematic using resource-constrained distributed devices because not all devices will have same amount of resources available. Finally, the dynamic and heterogeneous nature of manufacturing system devices significantly limits the possibility of predefining knowledge distribution efficiently. Hence, the approach proposed by Kurita et al. is in general complementary to that proposed here, but requires further study to address the named issues. Thus, the distributed approach presented in this article can be proposed as an option for dealing with distributed knowledge residing in embedded devices.

Another study that could be compared to the work presented here is that described in (Paret et al. 2011). There the researchers propose using metadata indexing for optimising the query distribution. The main problem is that metadata indexing is described to be one-time performed. The dynamism in industrial automation systems would have tangible problems with this feature since models are frequently updated. We therefore take the view that our proposal is the way to go for dynamic environments in which embedded devices are used for managing distributed KBs.

Finally, the research presented in (Kumar, Singh, and Verma 2010) investigates distributed query processing from the perspective of reducing communication in the system. The authors suggest applying genetic algorithms in order to analyse data available in the system and to optimise the query-processing strategy. The approach is feasible for powerful devices, in which communication is indeed a most scarce resource, generally arranged in stable and static systems. Instead, this research addresses the distributed and dynamic system of resource-constrained embedded devices. Thus, it could be concluded from our approach that the efficiency of distributed query processing is secondary to the robustness of the query result. Moreover, as the study deals with a distributed system using resource-constrained devices, it can be stated that the approach is restricted by the nature of the type of system devices.

6. Conclusions

The research work described an approach in which the locus of control is maintained at embedded device level. One of the direct benefits that this feature brings to the system is the reduction of time in controlling processes. At present many devices managing control processes compose industrial automation systems. Usually, the decisions are managed by higher level components. The problem is that cross-layer communication, e.g. between different layers of the ANSI/ISA-95 automation pyramid, consumes more time than the exchange of data among same-level components. This means that this approach enables systems to make decisions and solve problems by accomplishing horizontal communication. Thus, the readiness of higher level components is increased because the number of information transactions pushed from the bottom to the top layers is reduced.

The work presented here describes a mechanism for reasoning in distributed devices. However, an analysis of the limitation of resources for RP performance should be performed to avoid data overflow. Moreover, a scalability approach should be tested for providing a mechanism that would in principle allow using large networks of distributed devices.

According to the approach described in (Buil-Aranda et al. 2013), the use of SPARQL SERVICE operator ("SPARQL 1.1 Federated Query." 2013) makes it possible to directly define the ports to which queries are sent and later executed. This characteristic would reduce the complexity of the implementation at device level because the endpoint definition is nested in the query.

A further task for the research presented is to optimise the communications between the devices in a reasoning network, following existing technologies. Moreover, an iteration of the RP will be tested. This experiment will be useful to test and evaluate more complex situations not explored by this research.

Acknowledgement

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n°332946 and from the Finnish Funding Agency for Technology and Innovation (TEKES), correspondent to the project shortly entitled eScop,⁹ Embedded systems for service-based control of open manufacturing and process automation.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

ARTEMIS Joint Undertaking [Grant Number 332946]; Tekes.

Notes

1. <http://www.profibus.com/>
2. <http://www.inicotech.com/>
3. <http://www.w3.org/>
4. <http://www.mesa.org/en/index.asp>
5. <http://www.uml.org/>
6. <http://www.raspberrypi.org/>
7. <http://www.raspbian.org/>
8. <http://protege.stanford.edu/>

References

- Agyapong-Kodua, K., Niels Lohse, Robert Darlington, and Svetan Ratchev. 2013. "Review of Semantic Modelling Technologies in Support of Virtual Factory Design." *International Journal of Production Research* 51 (14): 4388–4404. doi:10.1080/00207543.2013.778433.
- Alexander, Dennert, Jakob Krause, Jorge Garcia, Andres Jorge, Stefan Hesse, Jose Luis Martinez Lastra, and Martin Wollschlaeger. 2013. "Advanced Concepts for Flexible Data Integration in Heterogeneous Production Environments." In *11th IFAC Workshop on Intelligent Manufacturing Systems*, edited by Tsuzuki Marcos, 348–353. doi:10.3182/20130522-3-BR-4036.00047.
- Brachman, Ronald J., and Hector J. Levesque. 2004. *Knowledge Representation and Reasoning*. San Francisco, CA: Morgan Kaufmann.
- Buil-Aranda, Carlos, Marcelo Arenas, Oscar Corcho, and Axel Polleres. 2013. "Federating Queries in SPARQL 1.1: Syntax, Semantics and Evaluation." *Web Semantics: Science, Services and Agents on the World Wide Web* 18 (1): 1–17. doi:10.1016/j.websem.2012.10.001.
- Camp, Roberto, and Andrei Lobov. 2014. "Implementing Circulating Oil Lubrication Systems Based on the IMC-AESOP Architecture." In *Industrial Cloud-based Cyber-physical Systems*, edited by Armando W. Colombo, Thomas Bangemann, Stamatis Karnouskos, Jerker Delsing, Petr Stluka, Robert Harrison, Francois Jammes, and Jose L. Lastra, 183–202. Cham: Springer International Publishing. http://link.springer.com/10.1007/978-3-319-05624-1_8.
- Chungoora, N., A.-F. Cutting-Decelle, R. I. M. Young, G. Gunendran, Z. Usman, J. A. Harding, and K. Case. 2013. "Towards the Ontology-based Consolidation of Production-centric Standards." *International Journal of Production Research* 51 (2): 327–345. doi:10.1080/00207543.2011.627885.
- Colombo, Armando W., Thomas Bangemann, Stamatis Karnouskos, Jerker Delsing, Petr Stluka, Robert Harrison, Francois Jammes, and Jose L. Lastra, eds. 2014. *Industrial Cloud-based Cyber-physical Systems*. Cham: Springer International Publishing. <http://link.springer.com/10.1007/978-3-319-05624-1>.
- Colombo, Armando W., Stamatis Karnouskos, and Thomas Bangemann. 2014. "Towards the Next Generation of Industrial Cyber-physical Systems." In *Industrial Cloud-based Cyber-physical Systems*, edited by Armando W. Colombo, Thomas Bangemann, Stamatis Karnouskos, Jerker Delsing, Petr Stluka, Robert Harrison, Francois Jammes, and Jose L. Lastra, 1–22. Cham: Springer International Publishing. http://link.springer.com/10.1007/978-3-319-05624-1_1.
- Coulouris, G. F., J. Dollimore, and T. Kindberg. 2005. *Distributed Systems: Concepts and Design*. Addison-Wesley. <http://books.google.fi/books?id=d63sQPvBezqC>.
- Elien, Paret, and William Van Woensel. 2011. "Efficient Mobile Querying of Distributed RDF Sources."
- Grau, Bernardo Cuenca, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. 2008. "OWL 2: The Next Step for OWL." *Web Semantics: Science, Services and Agents on the World Wide Web* 6 (4): 309–322.
- Harjunkoski, Iiro, Rasmus Nyström, and Alexander Horch. 2009. "Integration of Scheduling and Control – Theory or Practice?" *Computers & Chemical Engineering, FOCAP0 2008 – Selected Papers from the Fifth International Conference on Foundations of Computer-aided Process Operations* 33 (12): 1909–1918. doi:10.1016/j.compchemeng.2009.06.016.

- Iarovyi, Sergii, Jorge Garcia, and Jose L. Martinez Lastra. 2013. "An Approach for OSGi and DPWS Interoperability: Bridging Enterprise Application with Shop-floor." In *11th IEEE International Conference on Industrial Informatics (INDIN), 2013*, 200–205. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6622882.
- Islam, N., A. Z. Abbasi, and Z. A. Shaikh. 2010. "Semantic Web: Choosing the Right Methodologies, Tools and Standards." In *2010 International Conference on Information and Emerging Technologies (ICIET)*, Karachi, Pakistan. 1–5. doi:10.1109/ICIET.2010.5625736.
- Kiritsis, Dimitris. 2013. "Semantic Technologies for Engineering Asset Life Cycle Management." *International Journal of Production Research* 51 (23–24): 7345–7371. doi:10.1080/00207543.2012.761364.
- Kletti, Jürgen, ed. 2007. *Manufacturing Execution Systems – MES*. Berlin, Heidelberg: Springer Berlin Heidelberg. <http://www.springer.com/gp/book/9783540497431>.
- Kumar, T. V. V., V. Singh, and A. K. Verma. 2010. "Generating Distributed Query Processing Plans Using Genetic Algorithm." In *2010 International Conference on Data Storage and Data Engineering (DSDE)*, 173–177. doi:10.1109/DSDE.2010.56.
- Kurita, H., K. Hatano, J. Miyazaki, and S. Uemura. 2007. "Efficient Query Processing for Large XML Data in Distributed Environments." In *21st International Conference on Advanced Information Networking and Applications*, Ontario, Canada. AINA '07, 317–322. doi:10.1109/AINA.2007.64.
- Lastra, J. L. M., and I. M. Delamer. 2006. "Semantic Web Services in Factory Automation: Fundamental Insights and Research Roadmap." *IEEE Transactions on Industrial Informatics* 2 (1): 1–11. doi:10.1109/TII.2005.862144.
- Lin, L. F., W. Y. Zhang, Y. C. Lou, C. Y. Chu, and M. Cai. 2011. "Developing Manufacturing Ontologies for Knowledge Reuse in Distributed Manufacturing Environment." *International Journal of Production Research* 49 (2): 343–359. doi:10.1080/00207540903349021.
- Paret, E., W. Van Woensel, S. Casteleyn, B. Signer, and O. De Troyer. 2011. "Efficient Querying of Distributed RDF Sources in Mobile Settings based on a Source Index Model." *Procedia Computer Science* 5: 554–561. doi:10.1016/j.procs.2011.07.072.
- Power, Yvonne, and Parisa A. Bahri. 2005. "Integration Techniques in Intelligent Operational Management: A Review." *Knowledge-based Systems*. 18 (2–3): 89–97. doi:10.1016/j.knsys.2004.04.009.
- Power, Yvonne, and Parisa A. Bahri. 2005. "Integration Techniques in Intelligent Operational Management: A Review." *Knowledge-based Systems*. 18 (2–3): 89–97. doi:10.1016/j.knsys.2004.04.009.
- Puttonen, Juha, Andrei Lobov, Maria A. Cavia Soto, and Jose L. Martinez Lastra. 2010. "A Semantic Web Services-based Approach for Production Systems Control." *Advanced Engineering Informatics* 24: 285–299. doi:10.1016/j.aei.2010.05.012.
- Puttonen, J., A. Lobov, and J. L. M. Lastra. 2013a. "Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems." In *2013 IEEE 20th International Conference on Web Services (ICWS)*, 419–426. doi:10.1109/ICWS.2013.63.
- Puttonen, J., A. Lobov, and J. L. M. Lastra. 2013b. "Semantics-based Composition of Factory Automation Processes Encapsulated by Web Services." *IEEE Transactions on Industrial Informatics* 9 (4): 2349–2359. doi:10.1109/TII.2012.2220554.
- Ramis Ferrer, Borja. 2015. "A Proposal of Decentralized Architecture for OKD-MES." In *Open Knowledge Driven Manufacturing and Logistics – the ESCop Approach*, edited by Stanislaw Strzelczak, Pavel Balda, Marco Garetti, and Andrei Lobov, 331–340. Warsaw: Warsaw University of Technology Publishing House.
- Ramis, B., J. Garcia, and J. L. Martinez Lastra. 2013. "Assessment of IEC-61499 and CDL for Function Block Composition in Factory-wide System Integration." In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, 212–217. doi:10.1109/INDIN.2013.6622884.
- Ramis, Borja, Luis Gonzalez, Sergii Iarovyi, Andrei Lobov, Jose L. Martinez Lastra, Valeriy Vyatkin, and William Dai. 2014. "Knowledge-based Web Service Integration for Industrial Automation." In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 733–739. doi:10.1109/INDIN.2014.6945604.
- Shen, Jin, Liya Wang, and Yiwen Sun. 2012. "Configuration of Product Extension Services in Servitisation Using an Ontology-based Approach." *International Journal of Production Research* 50 (22): 6469–6488. doi:10.1080/00207543.2011.652744.
- Sirin, Evren, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. 2007. "Pellet: A Practical OWL-DL Reasoner." *Web Semantics: Science, Services and Agents on the World Wide Web* 5 (2): 51–53. doi:10.1016/j.websem.2007.03.004.
- Sosinsky, B. 2009. *Networking Bible*. Wiley. <http://books.google.fi/books?id=3DOReqRZejcC>.
- "SPARQL 1.1 Federated Query." 2013. <http://www.w3.org/TR/sparql11-federated-query/>.
- "SPARQL 1.1 Graph Store HTTP Protocol". 2013. <http://www.w3.org/TR/sparql11-http-rdf-update/>.
- Vrba, Pavel, Miloslav Radaković, Marek Obitko, and Vladimír Mařík. 2011. "Semantic Technologies: Latest Advances in Agent-based Manufacturing Control Systems." *International Journal of Production Research* 49 (5): 1483–1496. doi:10.1080/00207543.2010.518746.
- Wolf, Wayne. 2007. "Chapter 1 – Embedded Computing." In *High-performance Embedded Computing*, edited by Wayne Wolf, 1–63. San Francisco, CA: Morgan Kaufmann. <http://www.sciencedirect.com/science/article/pii/B9780123694850500026>.
- Yang, Jae Dong, and H. Lee-Kwang. 2000. "11 – Treating Uncertain Knowledge-based Databases." In *Knowledge-Based Systems*, edited by Cornelius T. Leondes, 327–351. San Diego, CA: Academic Press. <http://www.sciencedirect.com/science/article/pii/B9780124438750500124>.

- Zhang, W. Y., M. Cai, J. Qiu, and J. W. Yin. 2009. "Managing Distributed Manufacturing Knowledge through Multi-perspective Modelling for Semantic Web Applications." *International Journal of Production Research* 47 (23): 6525–6542. doi:[10.1080/00207540802311114](https://doi.org/10.1080/00207540802311114).
- Zuehlke, Detlef. 2010. "SmartFactory – Towards a Factory-of-Things." *Annual Reviews in Control* 34 (1): 129–138. doi:[10.1016/j.ar-control.2010.02.008](https://doi.org/10.1016/j.ar-control.2010.02.008).

VIII

AN ARCHITECTURE FOR IMPLEMENTING PRIVATE LOCAL AUTOMATION CLOUDS BUILT BY CPS

by

Borja Ramis Ferrer, Jose Luis Martinez Lastra, October 2017

43rd Annual Conference on IEEE Industrial Electronics Society (IECON)

2017 IEEE. Reprinted, with permission, from Borja Ramis Ferrer, José L Martinez Lastra, An Architecture for Implementing Private Local Automation Clouds Built by CPS, 43rd Annual Conference on IEEE Industrial Electronics Society (IECON), November 2017.

An Architecture for Implementing Private Local Automation Clouds Built by CPS

Borja Ramis Ferrer, José Luis Martínez Lastra
Tampere University of Technology, Laboratory of Automation and Hydraulics
Tampere, Finland
{borja.ramisferrer, jose.lastra}@tut.fi

Abstract—The connectivity of industrial automation domain systems has been enhanced by the employment of information and communication technologies. This permits the implementation of systems that are aligned with the vision of the fourth industrial revolution, or Industry 4.0. In this scope, Cyber-Physical Systems (CPS), i.e., integration of cyber and physical systems, enables the control and monitoring of modern production lines. Previous research work has proposed the implementation of a Private Local Automation Cloud (PLAC) that is composed by a set of networked embedded devices that are connected to industrial equipment. Such devices implement the service oriented architecture paradigm within the encapsulation of a knowledge model that describe the industrial system and the web service operations to be physically executed at the factory shop floor level. Aspects of such approach as the collaborative behavior of devices, decentralization of system knowledge or the interface between industrial equipment and devices have been already proved. However, a common architecture for implementing similar solutions is still missing. This paper aims to present an architecture for implementing PLACs which are built by CPS. In order to provide a formal design, the architecture is shown and discussed through a set of different and concurrent views following the known “4+1” view model. In addition, the manuscript reviews relevant qualitative attributes of several CPS-based architectures, research works and solutions that have been published during the last years.

Keywords—*cyber-physical systems; distributed reasoning; architecture; industrial automation*

I. INTRODUCTION

Advances on Information and Communication Technologies (ICT) permit the exchange of information remotely and without large delays. Thus, industrial domain engineers are constantly employing ICT-based solutions in order to enhance the connectivity of factories and integrate them in smart ecosystems [1]. In this context, the implementation of the Internet of Things (IoT) [2], [3], and the Cyber-Physical Systems (CPS) [4] emerge in the industrial domain to realize the envisioned requirements of the so-called Industry 4.0 (I4.0) [5].

Besides the adoption of ICT-based solutions, I4.0-inspired factories employ new generation of physical systems, such as industrial controllers, machines and industrial IoT devices. One of the focus of this research is the enhancement of the

industrial IoT devices’ capabilities, such as storage and computation of data.

There are recent research works that propose the deployment of web service (WS) enabled devices to act as a gateway between different layers of manufacturing systems [6]–[8]. In addition, previous works claim that the collaboration of devices for achieving smart solutions to control and monitor industrial systems can be now handled at device level [9]–[11]. For example, the work presented in [12] proposes the encapsulation of available services and system descriptions in decentralized repositories that are managed by industrial IoT devices. Although aforementioned works have already shown early-stage proofs, there is a need of a reference architecture that can be followed for implementing clouds built by CPS for controlling and monitoring industrial processes.

This article aims to present an architecture that might be useful for i) permitting the implementation of a private local automation cloud (PLAC) built by CPS [12] and ii) enabling the distributed reasoning of semantic resources that are hosted in collaborative embedded devices [13]. In addition, this article also aims the comparison of the presented architecture with existing CPS-based solutions and architectures. To present a formal design, different and concurrent views of the architecture are presented following the known “4+1” view model of software architecture, which is shown in following Fig. 1 and described in [14]. The employment of such view model allows to address independent concerns of different stakeholders of the solution to be implemented.

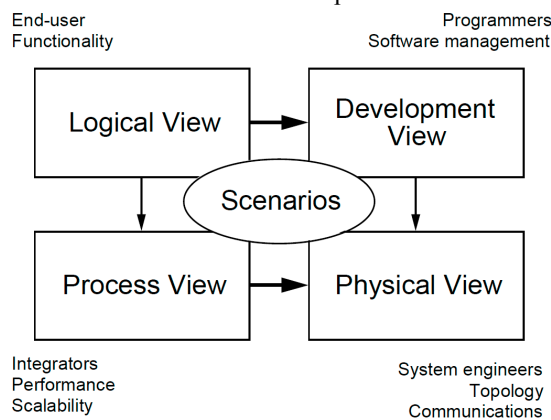


Fig. 1: The 4+1 View Model [14]

The rest of the article is structured as follows: Section II includes previous and related work performed in the scope of this research. Afterwards, Section III presents the proposed architecture throughout five subsections i.e., one per each view of the “4+1” view model. Then, Section IV compares the presented approach and other architectures in order to discuss their alignment with the principal characteristics of CPS. Finally, Section V concludes the article.

II. PREVIOUS AND RELATED WORK

The I4.0 seeks for smart, reliable and autonomous solutions that meet the requirements for implementing CPS [4], [15]. Conceptually, the implementation of CPS and its deployment in the industrial domain is the main aspect of the fourth industrial revolution that will enable the interconnection of heterogeneous and remote systems [1]. This can be achievable within the implementation of the service-oriented architecture paradigm [16], the employment of new ICT-based solutions and the realization of IoT-based concepts for industrial systems. In this context, previous research to the presented work has been performed in order to proof some of the concepts that are required for implementing such kind of systems.

The combination of knowledge-based systems and the WS integration permitted the development of a solution that permits the orchestration of semantically described WS operations to be executed in modern production lines [17]. The research performed in [17] demonstrated how industrial domain ontologies can be used as the system Knowledge Base (KB) for other components, such as orchestrator engines, to check or update the actual status of industrial machines. One of the advantages of similar approaches is the encapsulation of functionality in WS that may reside locally or even remotely i.e., in the cloud. In this scope, recent projects, such as the Cloud Collaborative Manufacturing Networks (C2NET) project¹ proposes the implementation of a cloud-based platform and its deployment in the supply chain. This will permit i) the interaction of remote systems involved in the same value chain and ii) the provision of cloud services that may support the supply network optimization of manufacturing and logistic resources.

On the other hand, the Embedded systems for Service-based control of Embedded systems for Service-based Control of Open manufacturing and Process automation (eScop) project² presented a framework, i.e., the Open Knowledge Driven Manufacturing Execution System (OKD-MES) framework, that proposes the combination of the MES modularity [18] with the knowledge-driven approach [19]. In fact, [11] recently proposed the decentralization of the storage and retrieval of OKD-MES semantic knowledge due to the advances on the capabilities of the embedded devices which are used for controlling and monitoring manufacturing processes. In this context, the research work presented in [12], discusses the potentials and challenges of distributed reasoning in a PLAC. In fact, first implementations on the behavior of devices following a PLAC-based approach has been presented

¹ <http://c2net-project.eu/>

² <http://www.tut.fi/escop/>

in [13]. However, there is a need for a formal architecture that might be used for building PLACs within CPS.

III. THE PROPOSED ARCHITECTURE

This section presents the proposed architecture for implementing PLACs built by CPS. As previously mentioned, the architecture is designed within the “4+1” view model [14]. Then, each subsection presents a different view of such model.

A. Logical architectural view

The first architectural view is used for presenting mechanisms and design elements across the various parts of the system to be implemented [14]. This architectural view is represented in Fig. 2 through a UML class diagram that shows the principal objects related to the PLAC.

As it can be seen, Fig. 2 depicts that the different types of users of the system (i.e., managers and clients) are capable to interact with the PLAC through a web interface. As the technologies to be used in the implementation of the system are web-based, the PLAC web interface is accessible remotely through the internet by using a web browser. User credentials are requested and verified before any possibility of interaction between managers and clients with the PLAC. This is important for security reasons [20].

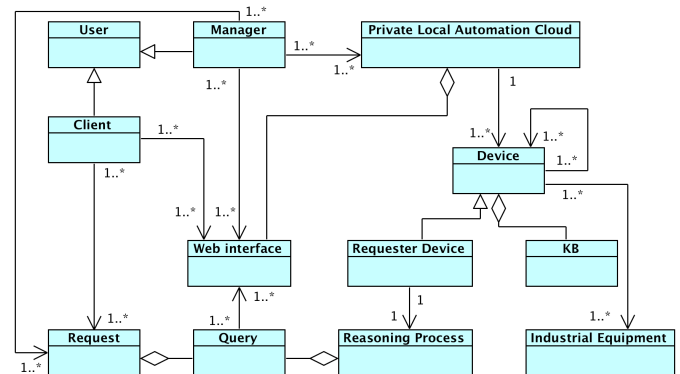


Fig. 2: Logical architectural view within an UML class diagram

The system executes incoming requests within a process known as Reasoning Process (RP) that i) is discussed in following subsection as a part of the process view and ii) has been previously described in [13]. In addition, requests are transformed into queries, as the principal message type that the PLAC devices will exchange in order to execute operations and share information. Thus, both managers and clients are connected to the request object, which, in turn, is connected to the query class. Moreover, as described in [13], each query is firstly handled by a device having the role of *Requester Device*, in a separated RP. Put briefly, whenever a query is sent to the PLAC, the devices that have sufficient resources for handling a RP will be eligible to be a Requester Device and, thus, lead such process.

The instances belonging to the *Device* class are the devices that are deployed in the PLAC. These devices host a KB which is implemented within ontologies as described in [12]. Basically, such semantic repositories permit the devices to encapsulate knowledge descriptions that will be queried and managed whenever any RP is executed. Depending on the

resources of devices, the size of semantic models will be different. Nevertheless, as the devices are connected to specific industrial equipment in order to control operations, each device will host and manage, at least, the information required for controlling and monitoring interconnected equipment. The content of devices' KB is detailed in [12].

B. Process architectural view

The second architectural view is used for presenting how the main objects from the logical view fit within the process architecture. Then, through a set of UML activity diagrams, this view shows the main aspects of the interaction between system components. Presented diagrams are revised ones from the sequence diagrams shown in [13].

The Fig. 3 shows the process of registering a new device that is deployed in the PLAC. To increase the scalability of the system to be implemented, the PLAC should permit *plug-and-play* for new devices. In fact, devices only need the addresses of other peers in order to establish communication and start requesting information. The first action to be carried out when a device is incorporated to the PLAC is to allow networked peers to discover new devices and vice versa. Thus, whenever a device is connected to the PLAC, it creates a hello message that is broadcasted to reach all the cloud devices, which will include its address into their KB. In addition, once the new device is registered, each device will create and send a response to the new device for allowing it to update its KB with all peers in the PLAC.

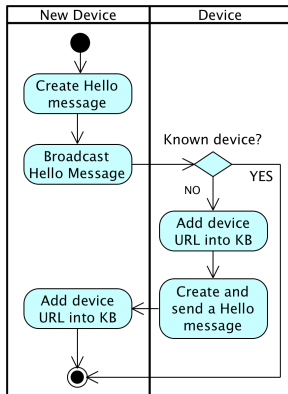


Fig. 3: Process for registering a new device represented within an UML activity diagram

Once the devices are interconnected and can send and receive requests from peers, any RP can be performed. However, as advanced in previous subsection and described in [13], for a RP to start, a *Requester Device* must be elected. The activity diagram depicted in Fig. 4 presents the process of finding such kind of device. First, a user creates and sends a request to the PLAC through the web interface. Then, the interface processes the request and sends it in a query format expecting any available device to respond for leading the RP that will correspond to the incoming request. Whenever a response is received, the interface will add the emitting device as the Requester Device and the RP will be started.

The described selection method of Requester Device is simple because it considers the first response to a demand for leading a RP. However, more complex methods can be implemented and be compatible with the system designed in

this research work. The addition of complexity in the selection process might bring to this operation the benefit of selecting the most suitable device for leading a specific RP. To support such smart selection, the interface should include more capabilities than the ones presented at this moment of the research. For example, it should be capable of buffering different responses for some time about the same Requester Device. Then, a more complex method, that is indeed contemplated in the component diagram of next subsection, could be employed for selecting the best device according to configurable parameters, such as the one with more resources available or the one that is handling less processes.

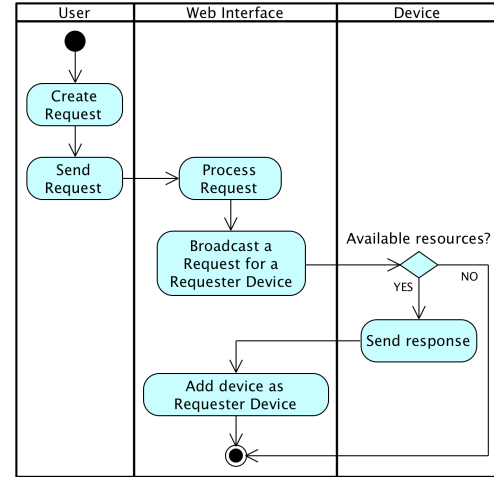


Fig. 4: Process for finding a requester device within an UML activity diagram

Once the leader, i.e., Requester Device, is selected, the RP can start in order to conclude the solution for a query and advance in the decision-making of any operation to be executed. Then, the Fig. 5 shows the process of performing a RP for an incoming query.

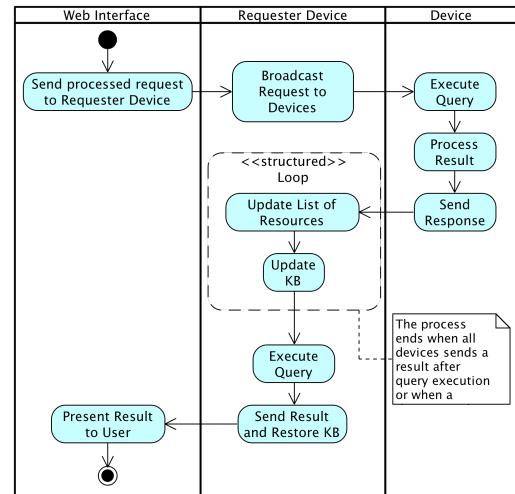


Fig. 5: Process for performing a RP within an UML activity diagram

The RP process starts with the delivery of the processed user's request in form of a query to the Requester Device. Afterwards, as shown in Fig. 5, the RP leader will broadcast such query to all devices that are included in its own KB. Then, each device will execute such query and will redirect the result to the Requester Device. Each time that the leader collects a new result, it integrates such information with its own KB.

Then, when all conclusions are received, the broadcasted query is executed for the last time on the Requester Device KB. The result of such execution will be sent to the user and the Requester Device will remove the integrated information incoming from other devices in order to keep only its own knowledge after the RP is finished.

C. Development architectural view

The third architectural view is used for presenting the implementation of system components. Then, the UML component diagram depicted in Fig. 6 shows the interaction of different software components to be implemented in order to realize the needs of the PLAC.

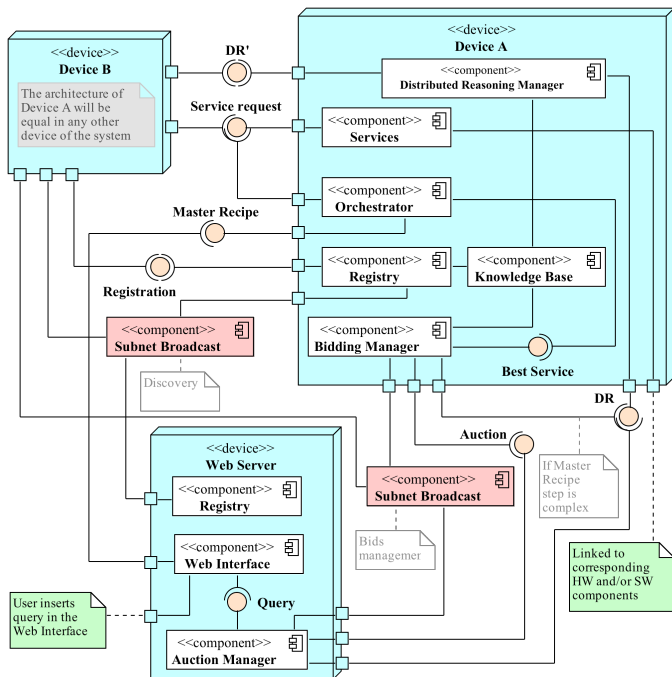


Fig. 6: Development architectural view within an UML component diagram

As it can be seen in previous diagram, each PLAC device encapsulates a set of software components that will permit i) the exchange of information between other peers, ii) the web server and execution of incoming requests in form of queries, iii) the description of services and iv) the capability of orchestrating service operations.

The main components for solving the reasoning process in the PLAC device-side are the Knowledge Base (KB) and the Distributed Reasoning Manager (DRM). Firstly, the KB component is concerned about the semantic description of any resource that the device might need not only to respond queries but also to perform other functions, such as orchestration of services, registering of PLAC information or service management. Secondly, the DRM is the component that controls the RP process. The device will employ the DRM according to its role for a specific RP. In other words, if the device is leading a RP, the DRM must allow it to handle the RP management. Otherwise, if the RP is processed by another peer, the DRM will exclusively be in charge of retrieving the requested information from the KB. Meanwhile DR stands for the Distributed Reasoning service that is requested from the server in order to lead a RP; the DR' is a similar service but

requested from another device that is managing a RP. Besides, Fig. 6 represents an interface that is present at each device for connecting industrial equipment or other software components. The physical architectural view presented in the next subsection describes some of the procedures that are needed when connecting devices to industrial equipment. Although the focus of this research work is to describe how to implement a CPS for distributed reasoning within a formal architecture, Fig. 6 includes a component for managing the orchestration of services, i.e., the Orchestrator. In the context of having WS-enabled devices that are in charge of controlling and executing industrial operations, the Orchestrator is in charge of sequencing such operations and controlling its execution in order to meet a concrete process. Relevant previous work on this matter has been already presented in [21], [22].

Moreover, the PLAC Subnet Broadcast (SB) component is in charge of managing the emission of messages that must be reached by all PLAC devices. Then, this component has a critical task in any of the processes depicted in Fig. 3, Fig. 4 and Fig. 5.

Finally, Fig. 6 shows that the web server includes an Auction Manager (AM) component. In principle, this component is in charge of routing an incoming query to its corresponding Requester Device. However, as mentioned in the process architectural view description, more complex methods could be implemented than just send que query to the first device which answers to the demand of a RP leader. For example, the AM might be configured with a set of parameters that would be required in order to earn the leadership of a RP. Then, each device would provide required information within the Bidding Manager (BM) component, which is included in the devices. Then, the BM is an optional component for making a more complex decision on which device should lead a RP. For this method to be implemented, the requested parameters would be also handled by the SB.

D. Physical architectural view

The fourth architectural view is used for representing the PLAC and interconnected entities from a system engineer's point of view. Then, the UML deployment diagram depicted in Fig. 7 shows the main physical and software components that to be interconnected.

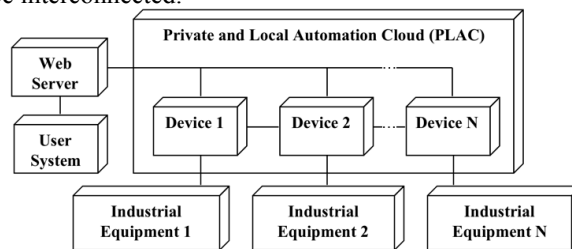


Fig. 7: Physical architectural view within an UML deployment diagram

As it can be seen in previous diagram, the PLAC is composed by a set of devices which, in turn, are connected to the industrial equipment. To interact with the PLAC, users (i.e., people or software) may use a web interface throughout any kind of system with internet connection. Then, users can send requests and receive corresponding solutions through such communication channel.

The physical connections between devices and the industrial equipment will depend on the specific interfaces of

machines. For example, the research work presented in [23] describes the an approach to retrofit an assembly line within WS-enabled devices that describes operations to be executed. In aforementioned work, different types of connections, such as DeviceNet, Modbus/TCP, RS232 and Ethernet networks are connected to Inico Technologies Ltd.³ S1000 industrial controllers. In the case of selecting non-industrial IoT devices for the implementation of the presented architecture, it is possible to attach commercial shields, add-ons, extension boards and/or adapters. This permits the upgrade of such devices for adding extra interfaces, such as Ethernet ports or analog and digital I/Os. For example, the research works [24]–[26] show some industrial applications with Raspberry Pi⁴ and Arduino⁵ that requires the utilization of additional ports.

E. Scenario

The last part of the “4+1” view model, which can be understood as the fifth view, is concerned on the representation of one or a set of scenarios that allow the identification of the required interactions between objects and processes. Then, this subsection describes representative activities to be performed by the PLAC that link the different views and components described along previous subsections. The selected activities are depicted within the UML use case diagram shown in Fig. 8.

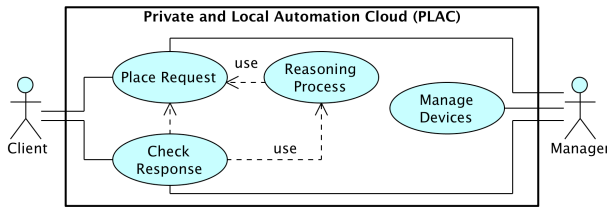


Fig. 8: Scenario represented within an UML use case diagram

Fig. 8 contemplates four main activities: *Place Request*, *Check Response*, *Reasoning Process* and *Manage Devices*. The first two activities can be performed by both type of users that are also represented in the logical architectural view shown in Fig. 2. In fact, it should be noted that “Client” might refer to both humans or cyber systems, such as third-party application or even a PLAC device willing to request information from another peer. Then, the Place Request activity consists on the placement of an order that will be translated in form of query and sent to the web interface, which, in turn, will start the process shown in Fig. 4. On the other hand, the Check Response activity is performed by any user to consult the answer of a specific request. Furthermore, as shown in Fig. 8, the Reasoning Process activity will use the request sent by any user for performing the PLAC distributed reasoning process. This will be accomplished within the process shown in Fig. 5. Finally, the activity Manage Devices can be only performed by the manager of the PLAC. Basically, the manager will be authorized to check and, if needed, modify the configuration of PLAC devices.

IV. DISCUSSION ABOUT CPS ARCHITECTURES

The research and implementation of CPS is highly important for the industrial domain. This will permit the

³ <http://www.inicotech.com/>

⁴ <https://www.raspberrypi.org/>

⁵ <https://www.arduino.cc/>

creation and design of robust, flexible and reliable systems that permit the control, monitoring and connectivity of industrial resources. In fact, the number of research works that have been published from 2006 has been growing dramatically. This is depicted in the Fig. 9, which is a result of searching publications within the keyword “cyber physical systems” in the IEEE Xplore Digital Library⁶. As it can be seen, the column chart shows that the amount of publications targeting the publication of works on CPS has never stopped increasing from 2006, excepting a light drop in 2014.

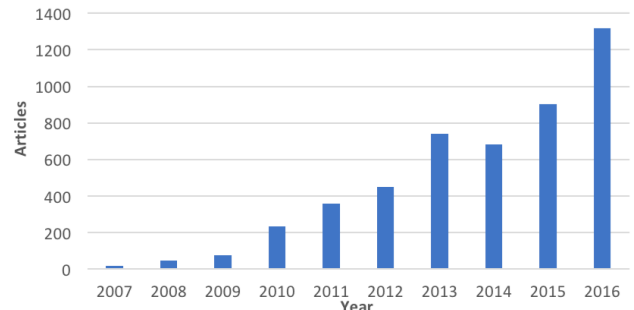


Fig. 9: IEEE Xplore Digital Library published articles found by the keyword “cyber physical systems”

This research work considers official documents that have been published for presenting technical options and challenges for new European Commission projects’ calls (e.g., H2020⁷) in the scope of CPS and the digitalization of industrial systems. More precisely, this research work categorizes different works throughout the descriptions that are included in the ARTEMIS⁸ Strategic Research Agenda (SRA) [27] and in the ECSEL⁹ Multi-Annual Strategic Research and Innovation Agenda (MASRIA) [28], both documents published in 2017 [29].

The objective of this section is twofold. Firstly, an identification of CPS key applications and the classification of different and recent works in the scope of the implementation of CPS is provided in TABLE. I. Secondly, a classification of different CPS architectures and research works depending on the consideration of specific qualitative attributes are presented in TABLE. II.

TABLE. I. MAPPING OF RESEARCH WORKS WITH CPS KEY APPLICATIONS

| CPS Key applications | Description according to SRA and MASRIA [27], [28] | Research work |
|----------------------------|--|--|
| Smart Production | Electronic components and systems and ICT for digitalizing the industry | [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45] |
| Smart Health and Wellbeing | Medical technology supporting patients throughout the phases of the care cycle | [15], [46], [47], [48], [49] |
| Smart Energy | Efficient, reliable and secure management of power generation, transmission, storage and consumption | [50], [51], [52], [53], [54], [55] |
| Smart Mobility | Digitalization of mobility within sensor technologies, embedded software and CPS | [56], [57], [58], [59] |
| Smart Society | Include information collection, exchange and processing for livable environments | [60], [61], [62], [63], [64], [65], [66], [67] |

⁶ <http://ieeexplore.ieee.org/Xplore/home.jsp>

⁷ <https://ec.europa.eu/programmes/horizon2020/>

⁸ <https://artemis-ia.eu/>

⁹ <http://www.ecsel-ju.eu/web/index.php>

Commonly, integration is included in the definition of CPS to declare that such systems bring together cyber and physical domains. Nevertheless, the integration of such type of systems imply non-trivial challenges and requirements which are discussed in several research works [15], [34], [40], [55], [65]. Therefore, there are many features that any architecture must guarantee for implementing CPS. This research work claims four qualitative attributes that any reference architecture should consider for the design and implementation of CPS: interoperability, reusability, scalability and reconfigurability. *Interoperability* is concerned about understandable interfaces between system components that will facilitate their interconnectivity and exchange of data. *Reusability* is a feature of systems' components to be used without modifications in a new configuration of the system. *Scalability* means that the provided solution is capable of meeting the requirements even though the system is scaled up or down. *Reconfigurability* refers to the ability of the system to be configured with other behaviors without affecting the system performance.

TABLE. II. A CLASSIFICATION OF DIFFERENT CPS ARCHITECTURES

| Ref | Qualitative attributes | | | |
|------|------------------------|-------------|-------------|-------------------|
| | Interoperability | Reusability | Scalability | Reconfigurability |
| [30] | • | • | • | • |
| [31] | • | | • | • |
| [32] | • | | • | • |
| [33] | • | | • | |
| [34] | • | • | | • |
| [35] | • | • | • | • |
| [36] | • | | | • |
| [37] | • | • | • | • |
| [38] | | | • | • |
| [39] | • | | | |
| [40] | • | • | • | • |
| [44] | • | • | • | • |
| [41] | • | • | | • |
| [42] | • | | | • |
| [43] | • | | | • |
| [45] | | • | • | |
| [15] | • | | • | • |
| [46] | • | | • | • |
| [47] | • | | • | • |
| [48] | • | | • | • |
| [49] | • | | • | |
| [50] | • | | | |
| [51] | | • | • | |
| [52] | • | | • | |
| [53] | • | | • | |
| [54] | • | | • | • |
| [55] | • | | • | |
| [56] | • | | • | |
| [57] | • | | | |
| [58] | • | | • | |
| [59] | • | | • | |
| [60] | • | | • | • |
| [61] | • | | • | |
| [62] | • | | | |
| [63] | • | | • | |
| [64] | • | | • | • |
| [65] | | • | • | |
| [66] | • | • | | • |
| [67] | | • | • | |

The TABLE. II presents a classification of different CPS-based architectures and solutions depending on the consideration of the aforementioned qualitative attributes. The

symbol “•” is used to denote the research works that provide sufficient means to demonstrate such characteristics. Although most of the research works included in TABLE. II claim that such attributes are required for building CPS, some of them fail in demonstrating that their approaches implement each feature. Therefore, TABLE. II may be used for identifying already published research works that fulfill specific CPS architecture qualitative attributes.

As it can be seen, TABLE. II shows that mostly all considered works provides sufficient description and demonstrates interoperability on their CPS-based architectures and solutions. Conceptually, the main reason for this is the fact that the integration between cyber and physical domains implies the performance of interfaces that permits the interoperation between different type of systems. In fact, in medium and large-scale systems the manipulation of heterogeneous systems and data types is frequent. Thus, means for interoperability must be implemented for ensuring any interaction between components. In fact, interoperability is a key feature in the CPS architecture that is presented in this article because devices are interoperable not only between them but also with the controlled industrial equipment. This permits the collection of data from machines and the exchange of information between peers.

Similarly, the scalability is frequently taken into account due to the need of incrementing the scale of CPS. In other words, most of the works consider the possibility of increasing the number of devices, machines or even virtual models to be handled within the same CPS-based architecture. The architecture presented in this article provides scalability to implemented CPS because the PLAC can be scaled up or down and the solution will meet the requirements. Still, it must be noticed that solutions implemented following the presented architecture tend to perform better with large-scale systems [11].

On the other hand, reusability is a characteristic that is often missing in CPS-inspired research works. This might be justified by the tendency on developing ad hoc solutions. In other words, some approaches present e.g., models that cannot be expanded neither adapted to other scenarios. This research work proposes the design and implementation of ontologies for describing the CPS and their services. These models can be extended and modified, even on runtime. For example, RDF¹⁰ based models can be updated within SPARQL Update¹¹ queries. Furthermore, the reusability is not only considered for the semantic models but also for the software components that are deployed in embedded devices. In fact, each embedded device to be implemented within the presented CPS architecture, is a replica in terms of functionalities and capabilities. Thus, all software modules can be reused, replicated and deployed in any peer of the PLAC.

Finally, reconfiguration is another qualitative attribute that tends to be absent in many considered research works. One of the reasons is because engineers present solutions that need specific configuration before systems are turned on. Nevertheless, as defined, any CPS should have, at some degree, the possibility of reconfigurability. This feature reduces the effort and time consumption on system configuration and

¹⁰ <https://www.w3.org/RDF/>

¹¹ <https://www.w3.org/TR/sparql11-update/>

adds autonomy to the implemented CPS. The characteristic of reconfigurability can be appreciated in the approach presented in this article at the device level. Each device that enters in the PLAC is capable of configure and expand its own KB within the algorithm described in Fig. 3. This configuration can be different depending on the controlled equipment of each device. Then, embedded devices will interact by themselves with no need of manual configuration when they are plugged in the PLAC. Furthermore, each device is capable of act as requester device, which implies a different behavior than the one followed by other peers.

V. CONCLUSIONS

This article presents the architecture that has been designed following the “4+1” model view, described in [14]. This research work will be used as a reference guide for implementing a solution that allows distributed reasoning in PLACs which are built by CPS. One of the main advantages of designing the proposed architecture with the selected view model is that different stakeholders of the solution may understand several functionalities of the solution from different perspectives. Then, the presented architecture will not be only useful for developers but also for end-users, and other interested parties.

The proposed architecture presents a possibility to implement a system that will enable the integration of semantic data that is decentralized through a cloud of specific embedded devices. These devices are capable of solving incoming requests and making decision within concrete behavior diagrams described in the process architectural view.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the graduate school funding of Tampere University of Technology in carrying out this work.

REFERENCES

- [1] B. Sniderman, M. Mahto, and M. Cotteleer, “Industry 4.0 and manufacturing ecosystems,” *DU Press*, Feb-2016. [Online]. Available: <https://dupress.deloitte.com/dup-us-en/focus/industry-4-0/manufacturing-ecosystems-exploring-world-connected-enterprises.html>. [Accessed: 07-Jun-2017].
- [2] R. H. Weber and E. Studer, “Cybersecurity in the Internet of Things: Legal aspects,” *Comput. Law Secur. Rev.*, vol. 32, no. 5, pp. 715–728, Oct. 2016.
- [3] I. T. S. Sector and O. ITU, “Quality of experience requirements for IPTV services,” *ITU-T Recomm. G*, vol. 1080, 2008.
- [4] E. A. Lee, “Cyber physical systems: Design challenges,” in *Object Oriented Real-Time Distributed Computing (ISORC)*, 2008 11th IEEE International Symposium on, 2008, pp. 363–369.
- [5] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0,” *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.
- [6] I. M. Delamer and J. L. M. Lastra, “Loosely-coupled Automation Systems using Device-level SOA,” in *2007 5th IEEE International Conference on Industrial Informatics*, 2007, vol. 2, pp. 743–748.
- [7] R. Harrison *et al.*, “Next Generation of Engineering Methods and Tools for SOA-Based Large-Scale and Distributed Process Applications,” in *Industrial Cloud-Based Cyber-Physical Systems*, A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, and J. L. Lastra, Eds. Cham: Springer International Publishing, 2014, pp. 137–165.
- [8] “Shop-Floor Decentralization: Shop Floor Marketplace Emerges.”
- [9] L. D. Xu, W. He, and S. Li, “Internet of Things in Industries: A Survey,” *IEEE Trans. Ind. Inform.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.
- [10] J. Wan *et al.*, “Software-Defined Industrial Internet of Things in the Context of Industry 4.0,” *IEEE Sens. J.*, vol. 16, no. 20, pp. 7373–7380, Oct. 2016.
- [11] B. R. Ferrer and J. L. M. Lastra, “Towards the encapsulation and decentralisation of OKD-MES services within embedded devices,” *Int. J. Prod. Res.*, vol. 0, no. 0, pp. 1–13, May 2017.
- [12] B. Ramis Ferrer and J. L. Martinez Lastra, “Private local automation clouds built by CPS: Potential and challenges for distributed reasoning,” *Adv. Eng. Inform.*, vol. 32, pp. 113–125, Apr. 2017.
- [13] B. Ramis Ferrer, S. Iarovyi, L. Gonzalez, A. Lobov, and J. L. Martinez Lastra, “Management of distributed knowledge encapsulated in embedded devices,” *Int. J. Prod. Res.*, pp. 1–18, Dec. 2015.
- [14] P. B. Kruchten, “The 4+1 View Model of architecture,” *IEEE Softw.*, vol. 12, no. 6, pp. 42–50, Nov. 1995.
- [15] S. A. Haque, S. M. Aziz, and M. Rahman, “Review of Cyber-Physical System in Healthcare,” *Int. J. Distrib. Sens. Netw.*, vol. 10, no. 4, p. 217415, Apr. 2014.
- [16] I. M. Delamer and J. L. M. Lastra, “Service-Oriented Architecture for Distributed Publish/Subscribe Middleware in Electronics Production,” *IEEE Trans. Ind. Inform.*, vol. 2, no. 4, pp. 281–294, Nov. 2006.
- [17] B. Ramis *et al.*, “Knowledge-based web service integration for industrial automation,” in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 733–739.
- [18] F. K. Johnson, “Future of Manufacturing Execution Systems: The Brave New Modular World of Manufacturing Intelligence,” *Rev. Manag.*, vol. 1, no. 1, p. 4, Jan. 2011.
- [19] S. Iarovyi, W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra, “Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems,” *Proc. IEEE*, vol. 104, no. 5, pp. 1142–1154, May 2016.
- [20] S. O. Afolaranmi, L. E. G. Moctezuma, M. Rak, V. Casola, E. Rios, and J. L. M. Lastra, “Methodology to Obtain the Security Controls in Multi-Cloud Applications,” in *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, 2016, pp. 327–332.
- [21] B. R. Ferrer, B. Ahmad, A. Lobov, D. Vera, J. L. Martinez Lastra, and R. Harrison, “A knowledge-based solution for automatic mapping in component based automation systems,” in *Industrial Informatics (INDIN)*, 2015 IEEE 13th International Conference on, 2015, pp. 262–268.
- [22] J. Puttonen, A. Lobov, M. A. Cavia Soto, and J. L. Martinez Lastra, “Planning-based semantic web service composition in factory automation,” *Adv. Eng. Inform.*, vol. 29, no. 4, pp. 1041–1054, Oct. 2015.
- [23] L. E. G. Moctezuma, J. Jokinen, C. Postelnicu, and J. L. M. Lastra, “Retrofitting a factory automation system to address market needs and societal changes,” in *2012 10th IEEE International Conference on Industrial Informatics (INDIN)*, 2012, pp. 413–418.
- [24] M. V. García, F. Pérez, I. Calvo, and G. Morán, “Building industrial CPS with the IEC 61499 standard on low-cost hardware platforms,” in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–4.
- [25] A. D. Deshmukh and U. B. Shinde, “A low cost environment monitoring system using raspberry Pi and arduino with Zigbee,” in *2016 International Conference on Inventive Computation Technologies (ICICT)*, 2016, vol. 3, pp. 1–6.
- [26] A. Imteaj, T. Rahman, M. K. Hossain, M. S. Alam, and S. A. Rahat, “An IoT based fire alarming and authentication system for workhouse using Raspberry Pi 3,” in *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2017, pp. 899–904.
- [27] European Commission, “The ARTEMIS Strategic Research Agenda 2016- the pathway to digital transformation,” *Digital Single Market*. [Online]. Available: <https://ec.europa.eu/digital-single-market/en/news/artemis-strategic-research-agenda-2016-pathway-digital-transformation>. [Accessed: 05-Jun-2017].

- [28] ECSEL JU, "2017 Multi Annual Strategic Research and Innovation Agenda for ECSEL Joint Undertaking." 2017.
- [29] ARTEMIS, "ARTEMIS - Documents," *artemis-ia-eu*. [Online]. Available: <https://artemis-ia.eu/article/documents-1.html>. [Accessed: 05-Jun-2017].
- [30] R. Harrison, D. Vera, and B. Ahmad, "Engineering Methods and Tools for Cyber-Physical Automation Systems," *Proc. IEEE*, vol. 104, no. 5, pp. 973–985, May 2016.
- [31] P. Leitao, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart Agents in Industrial Cyber-Physical Systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1086–1101, May 2016.
- [32] B. B. ↑ Jay Lee, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," *SME Manuf. Lett.*, 2014.
- [33] C. Liu and P. Jiang, "A Cyber-Physical System Architecture in Shop Floor for Intelligent Manufacturing," *Procedia CIRP*, vol. 56, pp. 372–377, 2016.
- [34] S. Iaroyvi, W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra, "Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems," *Proc. IEEE*, vol. PP, no. 99, pp. 1–13, 2016.
- [35] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Comput. Ind.*, vol. 81, pp. 11–25, Sep. 2016.
- [36] M. Garetti, L. Fumagalli, and E. Negri, "Role of Ontologies for CPS Implementation in Manufacturing," *Manag. Prod. Eng. Rev.*, vol. 6, no. 4, Jan. 2015.
- [37] O. Penas, R. Plateaux, S. Patalano, and M. Hammadi, "Multi-scale approach from mechatronic to Cyber-Physical Systems for the design of manufacturing systems," *Comput. Ind.*, vol. 86, pp. 52–69, Apr. 2017.
- [38] J. Lee, B. Bagheri, and H.-A. Kao, "Recent advances and trends of cyber-physical systems and big data analytics in industrial informatics," in *International Conference on Industrial Informatics (INDIN)*, 2014.
- [39] B. Bordel Sánchez, R. Alcarria, D. Martín, and T. Robles, "TF4SM: A Framework for Developing Traceability Solutions in Small Manufacturing Companies," *Sensors*, vol. 15, no. 12, pp. 29478–29510, Nov. 2015.
- [40] A. W. Colombo, S. Karnouskos, and T. Bangemann, "Towards the Next Generation of Industrial Cyber-Physical Systems," in *Industrial Cloud-Based Cyber-Physical Systems*, A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, and J. L. Lastra, Eds. Cham: Springer International Publishing, 2014, pp. 1–22.
- [41] T. Lu *et al.*, "Cyberphysical Security for Industrial Control Systems Based on Wireless Sensor Networks," *Int. J. Distrib. Sens. Netw.*, vol. 10, no. 6, p. 438350, Jun. 2014.
- [42] M. A. Pisching, F. Junqueira, D. J. dos Santos Filho, and P. E. Miyagi, "An architecture based on IoT and CPS to organize and locate services," in *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*, 2016, pp. 1–4.
- [43] R. Langmann and L. Rojas-Peña, "PLCs as Industry 4.0 Components in Laboratory Applications," *Int. J. Online Eng. IJOE*, vol. 12, no. 07, p. 37, Jul. 2016.
- [44] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, and S. Yin, "Industrial Cyberphysical Systems: A Backbone of the Fourth Industrial Revolution," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 6–16, Mar. 2017.
- [45] B. Ramis *et al.*, "Knowledge-based web service integration for industrial automation," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 733–739.
- [46] J. Wang, H. Abid, S. Lee, L. Shu, and F. Xia, "A secured health care application architecture for cyber-physical systems," *ArXiv Prepr. ArXiv12010213*, 2011.
- [47] Y. Zhang, M. Qiu, C. W. Tsai, M. M. Hassan, and A. Alamri, "Health-CPS: Healthcare Cyber-Physical System Assisted by Cloud and Big Data," *IEEE Syst. J.*, vol. 11, no. 1, pp. 88–95, Mar. 2017.
- [48] I. Lee *et al.*, "Challenges and Research Directions in Medical Cyber-Physical Systems," *Proc. IEEE*, vol. 100, no. 1, pp. 75–90, Jan. 2012.
- [49] E. Sultanovs and A. Romānovs, "Centralized healthcare cyber-physical system's data analysis module development," in *2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, 2016, pp. 1–4.
- [50] A. Florea, C. Postelnicu, B. Zhang, and J. L. M. Lastra, "Ecosystem oriented energy management: An implementation," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012, pp. 912–918.
- [51] S. K. Khaitan and J. D. McCalley, "Cyber physical system approach for design of power grids: A survey," in *2013 IEEE Power Energy Society General Meeting*, 2013, pp. 1–5.
- [52] A. Farahat, A. Florea, J. L. Martinez Lastra, C. Branas Reyes, and F. J. Azcondo, "Energy efficient outdoor lighting: An implementation," in *2014 IEEE 15th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2014, pp. 1–5.
- [53] L. T. T. Phuong, N. T. Hieu, J. Wang, S. Lee, and Y. K. Lee, "Energy Efficiency Based on Quality of Data for Cyber Physical Systems," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, 2011, pp. 232–241.
- [54] C. S. Shih, J. J. Chou, N. Reijers, and T. W. Kuo, "Designing CPS/IoT applications for smart buildings and cities," *IET Cyber-Phys. Syst. Theory Appl.*, vol. 1, no. 1, pp. 3–12, 2016.
- [55] Z. x Chen, Y. j Zhang, Z. x Cai, L. c Li, and P. Liu, "Characteristics and technical challenges in energy Internet cyber-physical system," in *2016 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2016, pp. 1–5.
- [56] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, "From wireless sensor networks towards cyber physical systems," *Pervasive Mob. Comput.*, vol. 7, no. 4, pp. 397–413, Aug. 2011.
- [57] P. González-Nalda, I. Etxeberría-Agiriano, and I. Calvo, "Towards a Generic Architecture for Building Modular CPS as Applied to Mobile Robotics," *Int. J. Online Eng. IJOE*, vol. 12, no. 1, p. 4, Jan. 2016.
- [58] S. Schulte, P. Hoenisch, K. Kipp, D. Burgstahler, S. Abels, and G. Liguori, "A Service Framework for Smart Mobility Scenarios," in *2016 IEEE International Conference on Mobile Services (MS)*, 2016, pp. 17–24.
- [59] G. Fabbri, C. M. Medaglia, A. Pecora, L. Maiolo, and M. Santello, "Cyber Physical Systems and Body Area Sensor Networks in Smart Cities," in *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, 2016, pp. 980–985.
- [60] K. M. Alam and A. El Saddik, "C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems," *IEEE Access*, vol. 5, pp. 2050–2062, 2017.
- [61] C.-Y. Lin, S. Zeadally, T.-S. Chen, and C.-Y. Chang, "Enabling Cyber Physical Systems with Wireless Sensor Networking Technologies," *Int. J. Distrib. Sens. Netw.*, vol. 8, no. 5, p. 489794, May 2012.
- [62] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, "Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber-Physical Systems," *Proc. IEEE*, vol. 100, no. 1, pp. 283–299, Jan. 2012.
- [63] S. Wang, Y. Zhang, Z. Yang, and Y. Chen, "A Graphical Hierarchical CPS Architecture," in *System and Software Reliability (ISSSR), International Symposium on*, 2016, pp. 97–105.
- [64] E. Simmon, S. K. Sowe, and K. Zetsu, "Designing a Cyber-Physical Cloud Computing Architecture," *IT Prof.*, vol. 17, no. 3, pp. 40–45, May 2015.
- [65] X. Yuan, C. J. Anumba, and K. M. Parfitt, "Review of the potential for a cyber-physical system approach to temporary structures monitoring," *Int. J. Archit. Res. ArchNet-IJAR*, vol. 9, no. 3, pp. 26–44, 2015.
- [66] L. Petnga and M. Austin, "An ontological framework for knowledge modeling and decision support in cyber-physical systems," *Adv. Eng. Inform.*, vol. 30, no. 1, pp. 77–94, Jan. 2016.
- [67] T. Sanislav, G. Mois, and L. Miclea, "An approach to model dependability of cyber-physical systems," *Microprocess. Microsyst.*, vol. 41, pp. 67–76, Mar. 2016.

IX

PRINCIPLES AND RISK ASSESSMENT OF MANAGING DISTRIBUTED ONTOLOGIES HOSTED BY EMBEDDED DEVICES FOR CONTROLLING INDUSTRIAL SYSTEMS

by

Borja Ramis Ferrer, Samuel Olaiya Afolaranmi, Jose Luis Martinez Lastra, October 2017

43rd Annual Conference on IEEE Industrial Electronics Society (IECON)

2017 IEEE. Reprinted, with permission, from Borja Ramis Ferrer, Samuel Olaiya Afolaranmi, José L Martinez Lastra, Principles and Risk Assessment of Managing Distributed Ontologies Hosted By Embedded Devices for Controlling Industrial Systems, 43rd Annual Conference on IEEE Industrial Electronics Society (IECON), November 2017.

Principles and risk assessment of managing distributed ontologies hosted by embedded devices for controlling industrial systems

Borja ramis Ferrer, Samuel Olaiya Afolaranmi, Jose Luis Martinez Lastra
FAST Laboratory, Tampere University of Technology
Tampere, Finland
{borja.ramisferrer, samuel.afolaranmi, jose.lastra}@tut.fi

Abstract—Industry is continuously moving towards the employment and exploitation of semantic technologies due to diverse enterprise needs, such as cross-domain interoperability, system modeling, categorization of information, model validation and data reasoning. Therefore, the research community presents new semantic-based approaches for implementing industrial systems that are more flexible, self-descriptive, dynamic and interoperable with other systems that are already deployed in the field. For example, the World Wide Web Consortium standards may be employed for linking remote resources. Currently, researchers claim that there is already a great success on implementing cyber-physical systems and efficient machine-to-machine and machine-to-human interactions through semantics. However, presented solutions are not always validated in terms of security. This article suggests two techniques: threat modeling and risk assessment for protecting solutions from attacks and malicious access. In addition, these techniques permit reconsidering the architecture of common semantic-based solutions by finding requirements not predicted during design phase. Moreover, this research illustrates the application of aforementioned techniques within a case of study. The study case is focused on a semantic-based solution which handles manufacturing processes through cyber-physical systems.

Keywords— *Knowledge representation; distributed ontologies; risk assessment; threat modelling; factory automation.*

I. INTRODUCTION

The research and implementation of semantic technologies for industry has reached its maximum level and it will probably remain alike further. One of the main reason for the high acceptance of semantic technologies in the industrial automation field is the need of interoperability between different industrial systems that generate and/or consume heterogeneous data. Presently, it is not enough to improve the efficiency of production just by improving the efficiency on factory shop floors. This is due to the large number of parameters and factors in the entire product supply chain that affects the overall production process. Therefore, enterprises seek valid solutions that allow cross-domain integration of information. In this way, all data generated and required for systems working in the production process can be distributed and analyzed.

Semantic technologies permit the integration, interpretation, categorization, modeling and extraction of the system

knowledge information. In fact, such data can be represented within diverse languages, which are nowadays standardized and understood by both machines and humans [1]. Furthermore, through automated reasoning, it is possible to inference implicit knowledge of systems for instance, assess unexpected resource relationships or categorization in the design phase or even validate the consistency of data models.

From the first decade of the 2000s until now, many research groups have paid special attention to the implementation of ontologies in the industrial automation field [2]–[4]. Ontologies permit to formally represent specific knowledge of a specific domain [5]. Moreover, ontology models can be queried and updated at runtime, which makes them a powerful asset for any system that needs a model to be dynamically actualized. For example, models can be updated with actual status of the industrial equipment [6].

Nevertheless, the openness and connectivity of recent semantic-based proposed solutions increases the vulnerability of platform resources to different kinds of attacks, which could modify, delete and steal critical data, among other undesired actions. The security of solutions containing semantic knowledge accessed and manipulated only by allowed users (i.e. humans and machines) can be enhanced by performing threat modelling and risk assessment in the design phase of the solution.

The objective of this article is twofold. Firstly, this paper aims to present a semantic-based approach for managing distributed ontologies that form a private automation cloud built by Cyber-Physical Systems (CPS) [7]. This approach is motivated by the fact that current embedded devices have more resources than ever before so that more functionalities (e.g., decision-making and manipulating semantic descriptions) can be handled at the low level: the factory shop floor. On the other hand, the second objective of this research work is to suggest that the application of threat modelling and risk assessment should be a common practice for improving security of semantic-based approaches. This is shown within the application of such techniques in the proposed approach.

The rest of the article is structured as follows: Section II describes related work in the field of industrial automation regarding the implementation of semantic technologies and

security in distributed systems. Section III describes general architectural components in knowledge-based solutions and presents two main techniques to enhance the security of such solutions. As a case of study, Section IV describes a solution to be implemented for handling processes in factory shop floors within embedded devices which are connected to and encapsulate semantic descriptions of industrial equipment. In addition, Section IV describes the application of the techniques for protecting semantic descriptions of a knowledge-based solution. Then, Section V discusses the research work and exemplifies some threats. Finally, Section VI concludes the article.

II. LITERATURE REVIEW AND INDUSTRIAL PRACTICES

A. Distributed systems and semantics in the industry

Distributed systems are those ones in which actions are performed according to exchanged messages among networked systems [8]. In the industrial automation field, one of the next steps after the use of relay controllers was the employment of Programmable Logic Controllers (PLCs) for providing control functions. Then, distributed control systems were deployed in manufacturing systems through a hierarchical network of industrial devices as e.g., PLCs for controlling processes of factories.

Nowadays, Information and Communication Technologies (ICT) implementation permit the accessibility and creation of collaborative networks in the product supply chain [9]. This goes beyond the coordination of PLCs in a factory shop floor because it enables the exchange of information between different systems that work under same or even different organizations in a product life cycle. This implies that information is now to be consumed not only by industrial equipment, but also by other systems at different parts of the enterprise or even by humans as e.g. operators or clients. The distribution of data on service request is doable e.g., implementing Service-Oriented Architectures (SOA) [10] within embedded devices that act as gateways between device level and superior levels of enterprises [11]. In this context, the meaning of data becomes important because stakeholders of different data produced in the same process must be capable of understanding the information being received.

Semantic technologies are currently exploited in the industry because they permit deriving meaning from information. There is a vast amount of employable semantic technologies e.g., Natural-Language Processing, Artificial intelligence (AI) or Data Mining technologies. These technologies permit representing, expressing and/or processing information based on specific domains.

The use of semantics is not limited to the description of models but also the text structuration and data type entailment of messages being sent. Therefore, the use of semantics in the industry becomes a requirement in enterprises because designers of systems must convey a common data format for describing the information to be distributed.

B. Ontologies and the Semantic Web for industrial systems

Knowledge Representation and Reasoning (KR&R) is an AI discipline that permits describing the world in a way that can be understood by humans and machines [1]. The container of semantic descriptions can be referred to as a Knowledge Base (KB). Moreover, the knowledge stored in a KB can be extended and updated according to semantic reasoning, which asserts implicit statements concluded from the represented explicit descriptions [12]. For industrial automation developments that use a KB as a source of system's knowledge, this is a powerful aspect of KR&R because it permits e.g., deriving unexpected facts at system design or operation phases, validation of the consistency of the KB, data mappings or the automatic categorization of incoming data.

Although there are many KR formalisms and semantic languages to implement KB [13], ontologies are prominent in recent developments for the industry [14]–[16]. Conceptually, ontologies contain formal description about certain domain [17]. Probably, the ease to design, query, extend, merge, reuse and reason ontologies, among other features, are the ones that make them an excellent choice for describing domain knowledge. In fact, ontologies abstract users from syntax, logic and programming languages through ontology editors. In addition, ontologies can be integrated with semantic rules, which add to KB new facts about certain domain concepts.

The Semantic Web was defined in 2001 as an extension of the Web wherein the meaning of the information is well-defined to enable the cooperation between people and computers [18]. The Semantic Web technologies are usually presented as a set of stacked technology standards from the World Wide Web Consortium (W3C) that link data on the Web. In fact, the different standards and levels of the Semantic Web Stack [19] have diverse objectives. First, at the bottom of the stack, the Uniform Resource Identifier (URI) is employed to define the identifiers, which are unambiguous names for resources. Then, the eXtensible Markup Language (XML) is used as a surface syntax for the written documents while the Resource Description Framework (RDF) and RDF Schema (RDFS) standards add vocabulary for the data interchange and taxonomy definition respectively. Afterwards, the Ontology Web Language (OWL) is used to add more vocabulary for describing richer ontologies in terms of expressivity. Furthermore, Rule Interchange Format (RIF) or the Semantic Web Rule Language (SWRL) is used for defining rules also on the top of RDFS. Finally, ontologies can be queried within the SPARQL Protocol and RDF Query Language (SPARQL). Actually, the SPARQL Update (SPARUL) is an extension of SPARQL that permits updating the ontology with new statements. It should be noted that the aforementioned standards are recommendations from the W3C, which provides updated specifications and drafts for each one in [20]. Moreover, a well-described manuscript about Semantic Web and its principle can be found in [21].

The concept of the Semantic Web is nowadays adopted in the industry to create a web of linked resources, which can be

implemented within ontologies. The designed map of semantic resources is useful, among other functionalities, e.g., checking dependencies of processes, service discovery or system and service description. There are many research works in the industrial automation community that show how RDF-based models can be used as the KB of systems to use system's knowledge [2], [6], [22].

C. On security of distributed automation systems

The recent advances in communication, networking and internet-based technologies have greatly improved the computational power of embedded systems as they are now better equipped to handle, process and store large datasets of information. This extends to the storage and processing of semantic descriptions as seen in [23] where knowledge bases (KBs) and the algorithms using the KBs were encapsulated within embedded devices located at the factory floor. This attribute has further enhanced the utilization and deployment of embedded systems (i.e. devices) in Distributed Automation Systems (DAS) particularly in Industrial Internet of Things (IIoT) [24] networks because they are easily connected to communicate with one another within the network.

In DAS, embedded devices exist as dispersed independent components, which collaborate in an integrated manner to achieve a common goal. The decentralized model of DAS promotes efficient tasks' management between the embedded devices and also offers excellent advantages such as scalability/extendibility: as devices can be easily added to the network, reliability, availability, accessibility and improved performance [25]. However, decentralization also brings about decentralized control, as there are multiple points of entry to the system. This increases the surface area of the attacks in the devices. In addition, the use of internet-based technologies further widens the attack base. This is due to the open nature of the internet, which makes it easily accessible to everyone including users with malicious intentions. The combination of these two factors make embedded devices vulnerable and exposed to threats that are capable of hampering their performance as well as that of the overall DAS.

In view of the foregoing, the security management of DAS is a key issue. It becomes even more complex as multiple points of entry means multiple points of failures and attacks and therefore multiple points of security protection. It is therefore required that the security management of DAS has to be approached in a holistic manner, which takes into consideration all the points through which the system may be accessed. In essence, security has to be provided in DAS at the following levels: device level, device-to-device communication and overall distributed system. Security has to be taken into consideration throughout the entire lifecycle of the system i.e. from the design phase up to execution phase.

Some concerns such as the security of the communication link and the administering of proper control in the network often come up in distributed systems and as such requires proper management. Therefore, as it relates to this research

work, enforcement of control and communication channel security are vital in order to protect and prevent device KBs from theft, interception and unauthorized modification.

A classification on distributed systems as cluster computing, grid computing, distributed storage system, distributed databases is found in [25]. DAS may be classified as distributed databases. In this study, the authors highlighted Denial of Service (DoS) attacks, snooping, and unauthorized access as some of the main security threats, and lists authentication, integrity check, confidentiality, authorization, access control and multi-level security as security mechanisms to mitigate the threats.

The research work described in [26] conducts a review on the security issues in distributed systems. The authors highlighted eavesdropping, masquerading, message tampering, replaying and denial of service as the general security attacks on distributed systems. The research work also identifies the authentication, access control, cryptographic techniques, quorum based system, trust based model as different security approaches in distributed systems.

In view of the aforementioned, an efficient security management has to be defined and implemented in DAS particularly using technologies that promote and support effective control of accessibility and interactions with/within the system. This begins with performing security assessment, which involves the identification of threat targets as well as possible attacks; the result of which will help in the realization of countermeasures needed to mitigate the threats. This will ensure that semantic descriptions stored in the knowledge bases of the devices are safeguarded as well as secure transfer and exchange of information between devices.

III. MAIN ASSETS AND POSSIBLE THREATS WHEN IMPLEMENTING ACTUAL KNOWLEDGE-DRIVEN SOLUTIONS

A. Knowledge-driven solutions

Knowledge-driven solutions are those ones wherein operations are executed according to the decisions made when computing the knowledge of the system. The industry is moving towards the exploitation of semantic technologies due to their potential for supporting systems in processes of the supply chain. In fact, the implementation and synergy between ICT, SOA and CPS creates a solid basement to transform, host, integrate, transport, monitor and manipulate data. CPS permits the integration between the physical and cyber domain [27]–[29].

The principal component of knowledge-driven solutions is the KB because it contains all the semantic descriptions of the system. In fact, such semantic information not only represent the knowledge of physical constraints of equipment; but also contains data related to service descriptions, data format for messaging, actual status of industrial equipment and much more. Thus, the KB is known also as the model of the system. Examples of models used in knowledge-driven approaches can be found in [2], [6], [29].

Moreover, other important and common components in knowledge-driven solutions are User Interfaces (UIs), service composers, industrial equipment and third-party applications. First, UIs permit the access of users for configuring or monitoring purposes. Then, service composers (i.e. orchestrators) are engines implemented in SOA developments to manage the execution of services [30]. Third, the physical equipment is obviously a required component in the industry for performing the operations at factory shop floors. Finally, third party applications are those that interact remotely with the system from the outside. It should be noted that, with the employment of web standards and emergence of the connectivity to realize the IIoT, knowledge-driven solutions also include web-based user interfaces so that remote users or clients can be connected to the solution. The monitoring of physical equipment can be done within service-enabled embedded devices that can expose functionality and status of machines to the Internet [31].

B. Threat Modelling (TM) and Risk Assessment (RA)

Owing to the complex nature of DAS, it becomes imperative to carry out security assessment and analysis of the DAS right from the design phase. This will involve the security evaluation of all the components making up the DAS in order to identify security issues (risk or threat), specify security requirements, specify (or identify) security controls and countermeasures. To achieve this, the approach of threat modelling and risk assessment is utilized in this research work. Threat modelling is a process which enables effective security analysis of an application [32]. It allows for the recognition, rating and mitigation of threats in an application or system. It brings about systematic addressing of security issues. The result of the threat modelling process is a threat model, which presents the security information of the application or system and enhances the performance of risk analysis of the application.

At the state of the art, there are basically 3 approaches to threat modelling: asset-based, attack-based and system-based [33]. In these approaches, threat modelling is carried out based on vital entities (components) to be protected, attacker's intention and individual system components respectively. In this research, the system based approach shall be utilized. The threat modelling process consists of the following steps:

1. *System component identification*: This step involves splitting up the DAS in order to identify its components or building blocks. This provides information about the system functionalities as well as its mode of interaction with external entities. It involves identification of components, entry points (connection interfaces) and trust levels (access levels).
2. *Component threat identification and ranking*: This step involves the identification of the threats associated with each component of the DAS. This is done using a threat categorization methodology such as STRIDE (*Spoofing, Tampering with data, Repudiation, Information disclosure, Denial of service and Elevation of privileges*) [34], which

helps to identify threats based on the possible actions of an attacker. It is then followed by a risk assessment process which involves determining the risk associated with each identified threat using a qualitative risk model or other models such as DREAD (*Damage, Reproducibility, Exploitability, Affected Users, Discoverability*) [34].

3. *Security requirements specification*: This step involves specifying the security requirements that have been particularly identified as basic requirements for Industrial Automation and Control Systems (IACS). These requirements help to provide a security view uniquely from the perspective of industrial automation and control systems.

4. *Selection of Security Controls*: This is the final step and it involves the identification and selection of the security mechanisms needed to mitigate the identified threats and fulfill or satisfy the security requirement [35] specified in the previous step. This helps to clearly address cybersecurity threats and IACS security requirements. The set of relevant controls are derived from the threat mapping lists in [32] where several mitigation techniques are provided to tackle STRIDE threats, as well as from other relevant standards which fulfills the IACS security requirements.

The result of this process is a threat model document, which provides information about the identified threats per component together with the risks associated with each of the threats. With these, adequate security controls are selected to mitigate the identified threats.

IV. CASE OF STUDY

Once the principles and main assets of knowledge-driven solutions have been described in Section III, this case of study presents a recent approach based on the employment of distributed ontologies encapsulated in embedded devices that control industrial processes. Moreover, this section shows how TM and RA techniques are applied to guide the design of the solution to make it more secure against diverse attack types.

A. Distributed reasoning within embedded devices

Aiming a representation of the described components in this subsection, Fig. 1 shows a simplified High-Level Architecture (HLA) of the solution. This was similarly represented in [36].

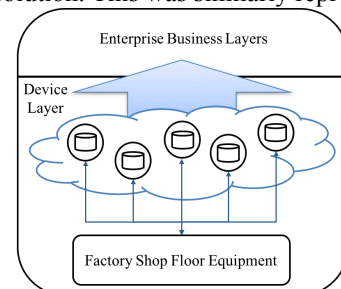


Fig. 1: High level architecture of the proposed solution

The research work to be used as a case of study in this article presents an approach that performs distributed reasoning in a private automation cloud built by embedded devices that aims to solve incoming requests and manage

processes of industrial systems. Embedded devices are used as a gateway that interfaces the factory shop floor equipment and cyber systems from the enterprise business layers, which are currently used for monitoring and managing service operations that are executed at the device layer.

One of the particularities of presented HLA is that some of the functionalities that are commonly done in higher level of enterprises can be lowered to the device level. The embedded devices (seen as circles in Fig. 1) implement functions normally executed by manufacturing execution systems [29].

In the proposed semantic-based approach, as it can be seen in Fig. 1, each device hosts a KB (shown as a can in Fig. 1), which describes certain knowledge from the system being controlled. Each device KB is implemented as an OWL model, which can be accessed by sending SPARQL over HTTP [20] requests to an ontology service. The union of all portions of knowledge hosted by different devices form the whole knowledge model of the system. The principal concepts included in each device KB and their relationship through object properties are shown in Fig. 2.

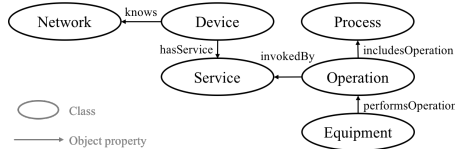


Fig.2: Main class diagram of each device KB

As it can be seen in Fig. 2, the *Device* concept is linked to *Network* and *Service* classes. Meanwhile the *Network* object includes a list of addresses for devices to know where to access other peers in the private cloud, the *Service* contains the services that the device holds and can be invoked. The right side of the class diagram represents objects directly related to operations performed by industrial equipment. *Operation* class includes all the operations that machines as e.g., robots or conveyors can execute. The *Process* object includes processes that can be performed through the manufacturing system. Finally, the *Equipment* class is populated with instances of the real industrial machines located at the factory floor.

Then, when a request comes from higher layers that implies the execution of an operation by certain industrial machine, the request is received by the cloud of devices that will conclude how to proceed through a distributed reasoning process. It should be noted that such process involves exchanging messages between computer systems to integrate their semantic resources. The algorithm to solve incoming requests for this knowledge-driven approach was previously presented in [23] and is illustrated in Fig. 3.

Fundamentally, the distributed reasoning process is started and led by one of the devices that broadcasts an incoming SPARQL query as a request that will be firstly executed locally (i.e., in each device). Once each device obtains the results of the query, they are returned in a form of a SPARUL query that will update the KB of the leading device. Finally, the leading device for the corresponding query will execute the incoming SPARQL query for all integrated knowledge

(within previously executed SPARUL queries) to finally find the solution, which is sent to a dedicated interface.

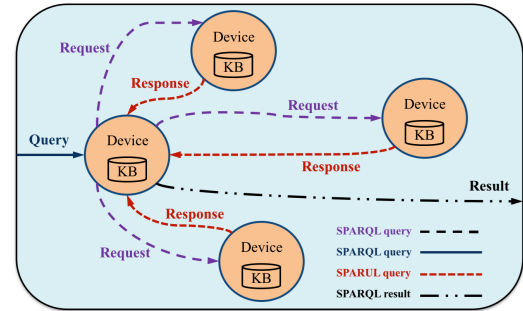


Fig. 3: Representation of a distributed reasoning process

One of the principal objectives of encapsulating semantic descriptions in embedded devices is to utilize the available resources of embedded devices, which are nowadays larger than before. Therefore, the proposed knowledge-driven solution uses devices not only to make the integration of physical and cyber systems and permit cross domain communication, but also to implement functionality that, in the industry, is handled by systems residing in upper layers of the enterprise as e.g., encapsulation and management of semantic descriptions, orchestration of services, scheduling, dispatching and decision making about process execution.

Presented solution includes also two interfaces: admin and user web-based interfaces. These interfaces permit remote access to authorized users to configure or send regular requests. It should be noted that OWL models could be updated within SPARUL queries or by replacing the entire ontology using the interface provided by the graph store. In this case study, a *Fuseki* server is used for such purpose [37].

B. TM and RA performance

The TM and RA steps outlined in the previous section are applied in the use case to identify the threats and determine the appropriate security controls. The application of each step in the use case are described below.

System component identification: The distributed automation system is composed of mainly HMI and field components. The identified components, entry points and trust levels are listed below, and, linked and described in Table 1.

- i. Components: User-Interface, Admin-Interface, devices (hosting the KBs), equipment (e.g., robots, conveyors and sensors) and controllers (PLCs).
- ii. Entry points: HTTP, MODBUS TCP, RS232, IP
- iii. Trust Levels: Operator, Administrator, DAS components

TABLE 1. DAS COMPONENT, ENTRY POINTS AND TRUST LEVEL

| Components | Description | Trust Level | Entry Points |
|--------------------|--|---|-----------------------|
| Operator Interface | WebUI for Operator interaction with DAS | Operator | HTTP Port, IP |
| Admin Interface | WebUI for Administrator interaction with DAS | Administrator | HTTP Port, IP |
| Devices | Stores, processes and encapsulates KBs) | Administrator, Operator, DAS components | HTTP Port, IP |
| Controllers (PLCs) | Processes logic for performing operations | Operator, Devices | MODBUS, IP TCP, RS232 |
| Equipment | Performs operations | Operator, Devices | RS232 |

Component threat identification and ranking: The STRIDE categorization methodology was applied in the use case on all the DAS components to identify the threats likely to affect the system. For each of the identified threats, the *likelihood* and *impact* on each component was estimated in order to calculate the risk. A qualitative risk model was used and the likelihood and impact values were specified on a scale of (0-10). The Risk being the product of likelihood and impact (i.e. Likelihood x Impact) was specified on a scale of (0-100) with 0 as *low risk* and 100 as *high risk*. The result of this step is summarized in Fig. 4. For each threat, the top left value represents the likelihood, the right value is the impact and the highlighted value below them is the risk. It can be seen that all the categories of threats pose different risks to different DAS components. However, the major components which requires high-level protection are the operator interface, admin interface and the devices. A detailed discussion on the results obtained in TM and RA can be found in following Section V.

| Asset Threat | Operator Interface | | Admin Interface | | Devices (KB) | | Controller | | Equipment | |
|-------------------------|--------------------|----|-----------------|----|--------------|----|------------|---|-----------|---|
| | L | I | L | I | L | I | L | I | L | I |
| Spoofing | 8 | 9 | 9 | 10 | 9 | 10 | 4 | 7 | 1 | 7 |
| | 72 | | 90 | | 90 | | 28 | | 7 | |
| Tampering | 3 | 5 | 3 | 5 | 9 | 10 | 7 | 9 | 2 | 7 |
| | 15 | | 15 | | 90 | | 63 | | 14 | |
| Repudiation | 8 | 9 | 8 | 9 | 3 | 9 | 3 | 9 | 1 | 7 |
| | 72 | | 72 | | 27 | | 27 | | 7 | |
| Information disclosure | 3 | 6 | 8 | 9 | 3 | 9 | 3 | 9 | 1 | 7 |
| | 18 | | 72 | | 27 | | 27 | | 7 | |
| Denial of Service | 9 | 10 | 6 | 9 | 5 | 10 | 3 | 9 | 1 | 7 |
| | 90 | | 54 | | 50* | | 27 | | 7 | |
| Elevation of privileges | 8 | 10 | 9 | 10 | 2 | 5 | 3 | 9 | 1 | 7 |
| | 80 | | 90 | | 10 | | 27 | | 7 | |
| Total Risk | 347 | | 393 | | 244 | | 199 | | 49 | |

RISK thresholds

| | |
|-----|--------|
| 100 | High |
| 50 | Medium |
| 0 | Low |

*The risk of DoS attack on Devices depends on the number of devices. (it grows with smaller devices number)
 Fig 4: DAS Threat Identification and Risk Assessment

Security requirements specification: In such specification, a relevant standard, which clearly states security requirements for IACS was utilized. ISA-99 (Security for Industrial Automation and Control Systems) standards was used to identify these requirements. ISA-62443-1-1 [38] specifies the foundational requirements for IACS, which are Identification & Authorization control, Use control, System Integrity, Data Confidentiality, Restricted Data Flow, Timely response to events and Resource availability. ISA-62443-4-2 [39] further provides component requirements and guidelines needed to fulfill the foundational requirements in IACS components.

Selection of Security Controls: For the category of threats and the security requirements identified from the previous steps, the following security controls have been selected to mitigate these threats and to fulfill the security requirements. Mitigation techniques like those presented by OWASP [32] can be applied to tackle STRIDE related threats. Security controls are also selected according to ISA-62443-4-2 [39] to fulfill the security requirements. Table 2 and Table 3 show the selected security controls.

TABLE 2. SELECTION OF SECURITY CONTROLS FROM OWASP [32]

| Threats | Security Control |
|------------------------|---|
| Spoofing | Strong Authentication, Secret data protection |
| Tampering | Integrity, Strong Authorization, Use of digital signatures, Use of Tamper-resistant protocols |
| Repudiation | Use of digital signatures, Audit and Logging |
| Information disclosure | Confidentiality, Authorization, Use of privacy-enhanced protocols, Strong Encryption, Non-storage of passwords in plain texts |
| Denial of Service | Availability, Authorization, Authentication, Validate and filter input |
| Elevation of Privilege | Authorization (Use of Access Control Lists), Use of least privilege service to run processes and access resources |

TABLE 3. SELECTION OF SECURITY CONTROLS FROM ISA-62443-4-2 [39]

| Security Requirements | Security Control |
|---|--|
| Identification and authentication control | Multifactor authentication (for Humans and devices), Use of Strong passwords, PKI certificates and tokens, System use notification |
| User Control | Authorization enforcement, Session control, Session lock, Audit records, digital signatures and timestamps |
| System Integrity | Cryptographic integrity protection, communication link protection, input validation, session integrity |
| Data Confidentiality | Information confidentiality, use of cryptography |
| Restricted Data Flow | Network segmentation, Boundary protection |
| Timely Response to events | Audit log accessibility and continuous monitoring |
| Resource Availability | Denial of Service protection, System Backup |

V. DISCUSSION

The presented solution for controlling industrial processes (at device level) is an opportunity to move down functionality, which is nowadays performed at supervisory, management and business levels. This can minimize the amount of vertical communication and, hence, it might reduce the saturation of the network due to the amount of cross-domain message exchange. Thus the common architecture might be reshaped in production systems that implement hierarchical architectures, such as the ones presented by [2] and [40].

This research work suggests the decentralization of the systems' knowledge by its distribution and encapsulation in embedded devices. Because of the management of semantic descriptions can be done within multiple embedded devices [23], [29], this research work presents a solution with no single access point and location of the KB. Then, there is no single point of failure and the system might operate with some devices down. Nevertheless, a mechanism of backing up semantic descriptions when a device fails should be further considered in order to consider all pieces of information. Moreover, the KB implementation within ontology languages such as OWL presents a rich and efficient technology to handle and query semantic descriptions on system demand. Owing to the possibility of monitoring and updating OWL models at runtime, dynamic environments like industrial automation systems can exploit the features and potential of such knowledge representation formalisms.

On the other hand, this research work also presents a methodology for evaluating certain risks at the design phase of any knowledge-based solutions that are accessible remotely via the Internet. In fact, the threat modeling and risk assessment revealed that spoofing, tampering and DoS attacks pose a high-level risk on the devices and, hence, the hosted KBs. Spoofing attacks on the devices may occur if a user interacts with the DAS using another user's credentials,

thereby claiming to be that user. An example of spoofing attack is man-in-the-middle attack. Security controls like authentication and strong encryption are recommended to mitigate this attack. However, it is very important that users also protect their credentials to prevent them from being stolen and used maliciously. Moreover, tampering attacks are aimed at maliciously modifying data at rest or in transit. In the DAS, this may occur if semantic descriptions stored in KBs or semantics in queries being exchanged between devices are modified. Tampering attacks may occur through SPARQL or SPARUL injections. Tampering attack is illustrated in Fig. 5.

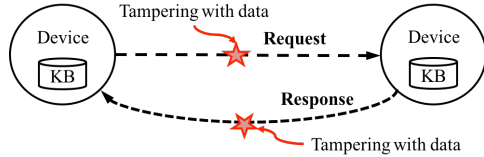


Fig 5: Representation of “Tampering with data” in proposed solution

When tampering with data happens, a wrong (modified) request is sent and which causes the generation of a wrong response. The consequence of this is that the KB gets wrongly updated with this wrong response and thus makes the semantic descriptions in the KBs to be inconsistent.

In order to exemplify a tampering threat, the query shown in Fig. 6 is used by the presented solution for checking the status of the components that must be assembled for specific product. Thus, the response of such query must be a list with the specified product (i.e. *product_1* for query as shown in Fig. 6) and its relation with assembled or not assembled components within different object properties i.e., *hasComponentAssembled* or *hasNotComponentAssembled*.

```
PREFIX bo:<http://www.tut.fi/FAST/CoSummit2013Demo#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
SELECT ?product ?statusProperty ?component
WHERE {
?product bo:hasComponent ?component. FILTER (?product = bo:product_1)
?statusProperty rdfs:subPropertyOf bo:hasComponentStatus.
?product ?statusProperty ?component.
}
```

Fig 6: Checking the status components to be assembled for a specific product

Then, if *product_1* is composed of *component_A* and *component_B* and both have been assembled, the resulting SPARUL query to be sent to the requester device (after the execution of the SPARQL query shown in Fig. 6) would be the one shown in Fig. 7.

The query shown in Fig. 7 is the one to be sent by a device that can answer totally to the query. However, such knowledge could be distributed among different devices due to the decentralization of knowledge in the solution. In this case, the statements of assembly status for *product_1* would be sent in different SPARUL queries from different devices hosting required semantic statements.

```
PREFIX bo:<http://www.tut.fi/FAST/CoSummit2013Demo#>
INSERT{
bo:product_1 bo:hasComponentAssembled bo:component_A.
bo:product_1 bo:hasComponentAssembled bo:component_B.
}
```

Fig 7: Example of SPARUL query to update the component status

Therefore, a tampering attack could (1) change any element of the SPARQL query request making the response wrong since it would answer to a different query (e.g., *product_1* is replaced by *product_2* in request), or (2) modify the SPARUL query with wrong statements to be updated in the KB of requester device (e.g., *hasComponentAssembled* is replaced by *hasNotComponentAssembled*). In fact, the tampering attack to SPARUL query can be even more severe for example, if the patterns are changed to erase the entire KB of the device.

This attack is easily carried out because SPARQL queries are transferred over HTTP protocol, which transmits the queries in plain text and thus can easily be seen, read and modified by an attacker. An effective control mechanism would be to send SPARQL queries over HTTPS, which encrypts the queries from adversaries and thus prevents modification. Although, the handling of keys and certificates by industrial embedded devices remains a challenge.

Finally, DoS attacks also pose a high risk on the devices and may occur if there is a congestion of the network as a result of high amounts of requests (or queries) sent to the devices. The actual impact of this attack depends on the number of devices because it grows with smaller number of devices in the network. An effective way to guard against this attack is through authentication and authorization (which promotes authorized interaction with the system), through adequate backups and replication (availability), and by validating and filtering all requests in order to ensure that only appropriate, well-formed and valid requests are forwarded to the devices. This may be achieved by deploying a gateway or firewall, which checks, filters and validates all requests before they are passed to the devices.

VI. CONCLUSION

This article presents a solution that implements distributed reasoning of semantic descriptions for controlling processes in industrial systems. An HLA of the solution is shown and main principles of it are described. The approach decentralizes the represented knowledge of the system in distributed KBs, which are encapsulated in embedded devices that realize the CPS concept since it allows integrating both cyber and physical domains. KBs are implemented within OWL, which are queried within SPARQL and SPARUL queries over HTTP.

Moreover, the presented approach is employed as a case of study to argue that security for developments that manage semantic descriptions should be considered at design phase. Fundamentally, the article claims that different type of attacks to assets that are common in semantic-based solutions can be analyzed within TM and RA techniques. In order to achieve an optimal result in TM and RA technique performance, an expert on security should assess the probability of different attacks and the designer of the solution should value the impact.

Further, it will be researched how such techniques could be included in ontologies so that designers would use ontological models to assess automatically the probability, impact and

possible threats to the system. This would reduce time and efforts in performing TM and RA assessments.

ACKNOWLEDGMENT

The project leading to this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n° 644429 correspondent to the project shortly entitled MUSA, Multi-cloud Secure Applications.

REFERENCES

- [1] R. J. Brachman and H. J. Levesque, *Knowledge Representation and Reasoning*. Morgan Kaufmann, 2004.
- [2] L. Fumagalli, S. Pala, M. Garetti, and E. Negri, "Ontology-Based Modeling of Manufacturing and Logistics Systems for a New MES Architecture," in *Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World*, Springer, 2014, pp. 192–200.
- [3] A. Giovannini, A. Aubry, H. Panetto, M. Dassisti, and H. El Haouzi, "Ontology-Based System for supporting Manufacturing Sustainability," *Annu. Rev. Control*, vol. 36, no. 2, pp. 309–317, Dec. 2012.
- [4] J. L. M. Lastra and I. M. Delamer, "Ontologies for Production Automation," in *Advances in Web Semantics I*, T. S. Dillon, E. Chang, R. Meersman, and K. Sycara, Eds. Springer Berlin Heidelberg, 2009, pp. 276–289.
- [5] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.
- [6] J. Puttonen, A. Lobov, and J. L. M. Lastra, "On the Updating of Domain OWL Models at Runtime in Factory Automation Systems," *Int. J. Web Serv. Res.*, vol. 11, no. 2, pp. 46–66, 32 2014.
- [7] E. A. Lee, "Cyber Physical Systems: Design Challenges," in *2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369.
- [8] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*. Addison-Wesley, 2005.
- [9] B. Andres, R. Sanchis, and R. Poler, "A Cloud Platform to support Collaboration in Supply Networks," *Int. J. Prod. Manag. Eng.*, vol. 4, no. 1, pp. 5–13, Jan. 2016.
- [10] D. K. Barry, *Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's Guide*. Newnes, 2012.
- [11] I. M. Delamer and J. L. M. Lastra, "Service-Oriented Architecture for Distributed Publish/Subscribe Middleware in Electronics Production," *IEEE Trans. Ind. Inform.*, vol. 2, no. 4, pp. 281–294, Nov. 2006.
- [12] B. Ramis Ferrer, B. Ahmad, D. Vera, A. Lobov, R. Harrison, and J. L. Martínez Lastra, "Product, process and resource model coupling for knowledge-driven assembly automation," - *Autom.*, vol. 64, no. 3, Jan. 2016.
- [13] E. Negri, L. Fumagalli, M. Garetti, and L. Tanca, "A review of semantic languages for the conceptual modelling of the manufacturing domain," in *Proceedings of the XIX Summer School Francesco Turco, 2014*, Senigallia, Ancona, 2014, pp. 1–8.
- [14] B. Ramis *et al.*, "Knowledge-based web service integration for industrial automation," in *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*, 2014, pp. 733–739.
- [15] M. Baqar Raza and R. Harrison, "Design, Development & Implementation Of Ontological Knowledge Based System For Automotive Assembly Lines," *Int. J. Data Min. Knowl. Manag. Process.*, vol. 1, no. 5, pp. 21–40, Sep. 2011.
- [16] G. H. Lim, I. H. Suh, and H. Suh, "Ontology-Based Unified Robot Knowledge for Service Robots in Indoor Environments," *IEEE Trans. Syst. Man Cybern. - Part Syst. Hum.*, vol. 41, no. 3, pp. 492–509, May 2011.
- [17] J. L. M. Lastra, I. M. Delamer, and F. Ubis, *Domain Ontologies for Reasoning Machines in Factory Automation*. ISA, 2010.
- [18] T. Berners-Lee, J. Hendler, O. Lassila, and others, "The semantic web," *Sci. Am.*, vol. 284, no. 5, pp. 28–37, 2001.
- [19] The World Wide Web Consortium, "October 2006: Semantic Web and Other W3C Technologies to Watch (19)." [Online]. Available: [https://www.w3.org/2006/Talks/1023-sb-W3CTechSemWeb/#\(19\)](https://www.w3.org/2006/Talks/1023-sb-W3CTechSemWeb/#(19)). [Accessed: 11-Jul-2016].
- [20] W3C, "All Standards and Drafts - W3C," 2017. [Online]. Available: <https://www.w3.org/TR/>. [Accessed: 11-Jul-2017].
- [21] E. Friedman-Hill, *JESS in Action*, vol. 46. Manning Greenwich, CT, 2003.
- [22] L. Ramos, "Semantic Web for manufacturing, trends and open issues: Toward a state of the art," *Comput. Ind. Eng.*, vol. 90, pp. 444–460, Dec. 2015.
- [23] B. Ramis Ferrer, S. Iarovyi, L. Gonzalez, A. Lobov, and J. L. Martínez Lastra, "Management of distributed knowledge encapsulated in embedded devices," *Int. J. Prod. Res.*, pp. 1–18, Dec. 2015.
- [24] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Trans. Ind. Inform.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.
- [25] M. Firdhous, "Implementation of Security in Distributed Systems - A Comparative Study," *Int. J. Comput. Inf. Syst.*, vol. 2, no. 2, Feb. 2011.
- [26] V. Prakash and M. Darbari, "A Review on Security Issues in Distributed Systems," *Int. J. Sci. Eng. Res.*, vol. 3, no. 9, 2012.
- [27] J. Lee, B. Bagheri, and H.-A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015.
- [28] A. W. Colombo, S. Karnouskos, and T. Bangemann, "Towards the Next Generation of Industrial Cyber-Physical Systems," in *Industrial Cloud-Based Cyber-Physical Systems*, A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, and J. L. Lastra, Eds. Cham: Springer International Publishing, 2014, pp. 1–22.
- [29] S. Iarovyi, W. M. Mohammed, A. Lobov, B. R. Ferrer, and J. L. M. Lastra, "Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems," *Proc. IEEE*, vol. PP, no. 99, pp. 1–13, 2016.
- [30] J. Puttonen, A. Lobov, and J. L. Martínez Lastra, "Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2349–2359, Nov. 2013.
- [31] L. E. G. Moctezuma, J. Jokinen, C. Postelnicu, and J. L. M. Lastra, "Retrofitting a factory automation system to address market needs and societal changes," in *2012 10th IEEE International Conference on Industrial Informatics (INDIN)*, 2012, pp. 413–418.
- [32] The Open Web Application Security Project, "Application Threat Modeling - OWASP." [Online]. Available: https://www.owasp.org/index.php/Application_Threat_Modeling. [Accessed: 14-Jul-2016].
- [33] R. Schlegel, S. Obermeier, and J. Schneider, "Structured system threat modeling and mitigation analysis for industrial automation systems," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, 2015, pp. 197–203.
- [34] The Open Web Application Security Project, "Threat Risk Modeling - OWASP." [Online]. Available: https://www.owasp.org/index.php/Threat_Risk_Modeling. [Accessed: 14-Jul-2016].
- [35] S. O. Afolaranmi, L. E. G. Moctezuma, M. Rak, V. Casola, E. Rios, and J. L. M. Lastra, "Methodology to Obtain the Security Controls in Multi-cloud Applications," in *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, 2016, pp. 327–332.
- [36] B. Ramis Ferrer and J. L. Martínez Lastra, "Private local automation clouds built by CPS: Potential and challenges for distributed reasoning," *Adv. Eng. Inform.*, vol. 32, pp. 113–125, Apr. 2017.
- [37] The Apache Software Foundation, "Apache Jena - Fuseki: serving RDF data over HTTP." [Online]. Available: https://jena.apache.org/documentation/serving_data/. [Accessed: 13-Jul-2016].
- [38] The International Society of Automation, "ISA99 Committee - WP-1-1." [Online]. Available: <http://isa99.isa.org/ISA99%20Wiki/WP-1-1.aspx>. [Accessed: 14-Jul-2016].
- [39] The International Society of Automation, "ISA99 Committee - WP-4-2." [Online]. Available: <http://isa99.isa.org/ISA99%20Wiki/WP-4-2.aspx>. [Accessed: 14-Jul-2016].
- [40] The International Society of Automation, "ANSI/ISA-95.00.03-2013 Enterprise-Control System Integration - Part 3: Activity Models of Manufacturing Operations Management." [Online]. Available: <https://www.isa.org/store/products/product-detail/?productId=116782>. [Accessed: 15-Jul-2016].

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-4083-7
ISSN 1459-2045