

Towards Flexibility in Future Industrial Manufacturing: A Global Framework for Self-organization of Production Cells

S. Azaiez, M. Boc, L. Cudennec, M. da Silva Simoes, J. Haupert, S. Kchir, X. Klinge, W. Labidi, K. Nahhal, J. Pfrommer, et al.

▶ **To cite this version:**

S. Azaiez, M. Boc, L. Cudennec, M. da Silva Simoes, J. Haupert, et al.. Towards Flexibility in Future Industrial Manufacturing: A Global Framework for Self-organization of Production Cells. 7th International Conference on Ambient Systems, Networks and Technologies / 6th International Conference on Sustainable Energy Information Technology / Affiliated Workshops, May 2016, Madrid, Spain. pp.1268-1273, 10.1016/j.procs.2016.04.264 . cea-01831554

HAL Id: cea-01831554
https://cea.hal.science/cea-01831554

Submitted on 6 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 1268 – 1273

Procedia Computer Science

The 2nd International Workshop on Recent Advances on Machine-to-Machine Communication (RAMCOM 2016)

Towards Flexibility in Future Industrial Manufacturing: A Global Framework for Self-Organization of Production Cells

Selma Azaiez[a,*], Michael Boc[a], Loïc Cudennec[a], Max Da Silva Simoes[a,d], Jens Haupert[b], Selma Kchir[a], Xenia Klinge[b], Wael Labidi[a], Karima Nahhal[a], Julius Pfrommer[c], Miriam Schleipen[c], Christian Schulz[b], Thibaud Tortech[a]

[a]*CEA, LIST, Saclay, France*
[b]*DFKI GmbH, Berlin, Germany*
[c]*Fraunhofer IOSB, Karlsruhe, Germany*
[d]*SYBOT Industries, Orsay, France*

Abstract

The future of manufacturing leads to flexible industrial facilities in which production lines or systems are composed by several production cells. Production cells can be reorganized and reconfigured by introducing new devices, equipment, functionalities or even by re-configuring the communication network. In this context, machine-to-machine communication does not only provide a transport layer for monitoring and control, but also provide a high-level distributed service framework and data management system. In this contribution, the authors address the challenge to manage the self-organization of production cells by means of a global framework. This framework bases on the following technologies: RobotML for the scenario description, OPC UA for service orchestration, object memories for distributed data sharing, Frama-C/Para-C for code verification and SDN for network reconfiguration. This framework has been deployed within a use case involving the SYBOT collaborative robot and a reconfigurable Raspberry-Pi based camera to enhance human operator safety. Experiments show that from a high-level description of the scenario, it was possible to automatically orchestrate at the OPC UA level the different reconfigurations of the production cell.

© 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of the Conference Program Chairs

Keywords: Reconfigurable Manufacturing, Machine-to-machine Communication, Orchestration Framework

1. Introduction

As for different application domains, the manufacturing is facing a bend where traditional production processes and work methods have to be evolved in order to be more flexible and adaptive to a quick changing context. Such an evolution involves the adoption of Cyber Physical Production Systems (CPPS) that include technologies related to communication and data processing between software layers and physical manufacturing devices[1]. In the CPPS

* Selma Azaiez. Tel.: +33-01-69-08-21-36.
E-mail address: selma.azaiez@cea.fr

1877-0509 © 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.04.264

context, production systems are self-organized and composed by networked equipment that are context-sensitive, autonomous and cooperative. A component in a CPPS has to be able to control its tasks and interact with other components or human operators via interfaces[2]. Here, the machine-to-machine communication layer has to include more than a simple transport layer. A CPPS has to be aware about the services and functionalities provided by each device and has to orchestrate such services in order to safely execute a high-level global service.

In this paper, we introduce a software framework that orchestrates services of the autonomous devices included in a production cell. The framework ensures the real-time monitoring of devices, establishes network connections if requested, deploys new applications onto the devices to offer new functionalities, and checks the correctness of the source code of applications that will be deployed. The framework was experimented on a real use case that connect a SYBOT collaborative robot and a reconfigurable Raspberry-Pi-based camera. The Sybot is a collaborative robot dedicated to different industrial tasks such as polishing. The Sybot can detect a human presence by collision. When the Sybot is equipped by a sharp object, this can cause a danger for a human operator. In our use case scenario, we aim to enhance the human operator safety by deploying a new functionality that consists on a distant detection of the human intrusion. For this purpose, we connect the Sybot to an external camera.

This paper is organised as follows. We first introduce some challenges of manufacturing system reconfiguration and we introduce our integrated framework. Then we describe each framework's building block and finally, we introduce the use case we used to experiment the use of the framework.

2. The Challenge of Reconfiguring a Flexible Manufacturing System

Different methods have been proposed to upgrade traditional Dedicated Manufacturing Lines (DML) to flexible systems such as Flexible Manufacturing Systems (FMS) and Reconfigurable Manufacturing System (RMS). FMS introduces a highly flexible machines adjustable to different production patterns. But FMS are highly costly and studies show that the technical success provided by FMS is not a guarantee of a business success [3]. Then, RMS were introduced to cope with capacity, functionality and cost. With RMS, technical innovations concern both hardware and software[4], with a decentralized control of intelligent entities in a distributed infra-structure[5]. Mechanical and power interfaces allow hardware flexibility and connectivity. But there is still a lack in communication flexibility that makes complex the implementation of reconfiguration in any existing application software. These limitations could be overcome by integrating the CPS and IoT concepts [6].Hence, more recently, the CPPS concept was introduced and the trends move to a more complex system: open software and hardware, assorted communication technologies, miniaturization of the hardware, reduction of cost, etc. This vision enhances the challenges. CPPS are enormous systems that have to be self-organized and self-, real-time controlled, efficient, safe and correct. CPPS leaves the scope of simply establishing a machine-to-machine communication in the terms of data transportation and includes organization, context-awareness, self-evolution in a robust way. The communication layer have been enhanced by service-oriented architecture in which each device is identified by the services it can provide or ask for. In the Industry 4.0 program, OPC-UA has been selected as a standard for heterogeneous machine interoperability. However, even if the OPC-UA provides discovery mechanisms and basic services that facilitate devices communication, the dynamic orchestration of several heteregenous devices is still challenging. In this work, we aim at providing a methodology for complex orchestration with a limited modification of the existing production lines. We used OPC-UA to implement the following new internal services:

- Initial configuration: the structure and functionalities of each equipment in the production cell are modeled using RobotML, and possible (re)-configuration are described,
- Service orchestration: the orchestration is handled by a specific device and can be managed through interfaces such as a PC or any wireless connected devices,
- Applications: the software implementing new functionalities can to be easily and safely deployed,
- Network configuration: connections between devices can be established in an automatic way,
- Monitoring: devices can be real-time monitored by any of the other devices to take actions.

We introduce these different aspects of flexibility within a single framework that integrates technical building blocks provided by the different research institutes: (1) RobotML for modeling initial configuration is provided by CEA, (2)

OPC-UA for service orchestration is provided by Fraunhofer, (3) Frama-C/Para-C for static analysis of application source code is provided by CEA, (4) OMS for monitoring is provided by DFKI and (5) the SDN controller for automatic network configuration is provided by CEA. The following figure shows how we used our integrated framework in the use case.

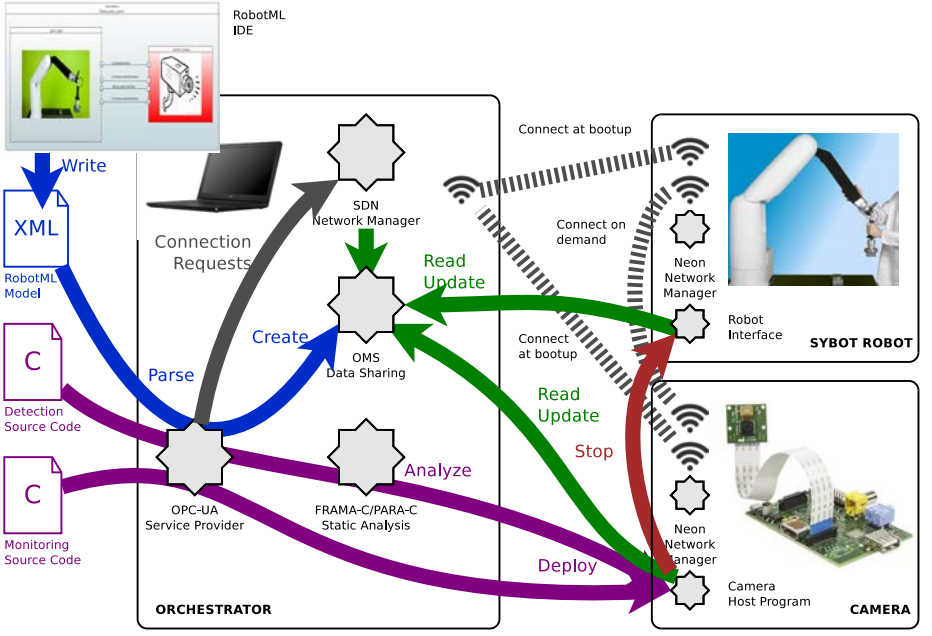


Fig. 1. *Use-Case Big Picture. Reconfiguration of a camera for intrusion detection within a robot environment.*

2.1. Design of the Reconfigurable Manufacturing System Using RobotML

RobotML (Robot Modeling Language)[7] is based on the Papyrus [1] modeling tool. RobotML intends to facilitate the development of robotics applications and to ease exchanges between roboticists and end users. It allows the specification of a component-based architecture. Low-level details have been hidden behind parametrized and easier to manage abstractions proposed through a graphical interface integrated to Eclipse environment. A sizable amount of low-level programming knowledge has been put into code generation transformations. RobotML generators allow generating source code from models to several target platforms.

Our approach to configure the production scene rely on a modeling step within RobotML. At the modeling level, each component properties and ports are configured. Properties are related to the internal configuration of the component (e.g: Degrees of freedom of the Sybot). They can also indicate whether the component is critical or not for safety analysis purposes. This is useful to indicate if the application to be deployed has to be checked for some safety properties. Components can also provide and require a set of services through ports. These services are defined in interfaces representing the API of each component so that, we can ask for a new service in this component. Also, to allow the dynamic reconfiguration of the network, we have defined a communication interface where we can ask components to connect to another ones. Once the components are configured, we define the architecture of the system by connecting these components. RobotML then generates an initial configuration that will initialize the service orchestrator. In our case, OPC-UA is used for this purpose. The mapping between RobotML and OPC-UA service orchestrator is based on mapping between concepts used within RobotML and OPC-UA languages. Transformation

[1] Papyrus, `https://eclipse.org/papyrus/`

rules have been implemented in Acceleo [2] language to provide an XML file which lists all the components, their properties, their ports and the connections between them in addition to interfaces. This file is the entry point of the OPC-UA orchestrator.

2.2. Machine-to-Machine Communication and Orchestration Using OPC UA

The OPC Unified Architecture (OPC UA)[8] is an industrial machine to machine communication protocol based on a platform independent service-oriented architecture. In the present application, OPC UA is used as a generic interface and framework for the orchestration of different production components (hardware or software) and their skills and encapsulated methods such as code analysis, network reconfiguration and deployment of new functionalities. Fraunhofer IOSB integrated an open source OPC UA SDK[9] (open62541) written in C99 into the Lua programming language [3]. The combined OPC UA server and client called OPC UA orchestration server was designed and developed based on this combination. It can run on a regular computer, as well as on mobile platforms and even on a low-end Raspberry Pi computer.

The OPC UA orchestration server receives the description of the whole production scene from RobotML using an XML representation. This scene description includes information about possible new functionalities and possible reconfigurations of production cells. For example, in this scenario it is possible to ask each production cell to reconfigure its network, to perform source code analysis, to switch the camera from monitoring to detection and to start and stop the robot. Additionally, a visual dashboard has been developed to show an overview of the scene and to display the content of the OPC UA servers for the operator. The dashboard is based on an existing visualization system, ProVis.Visu [4], which runs on Windows operating system and includes an OPC UA client based on the Unified Automation C++ UA Toolkit [5]. Other standard OPC UA clients can be used as well to display information of OPC UA server. These can run on mobile platforms as well.

2.3. Real-Time Monitoring and Data Sharing Using OMS

The Object Memory Server (OMS)[10] provides access to Digital Object Memories as described by the W3C Object Memory Modeling Incubator Group[11]. Such memories are data repositories linked to corresponding physical artifacts which can be viewed and manipulated by different agents, like humans, machines or other artifacts. Changes made to the physical object can be logged over time and are digitally represented via various interfaces.

Within the developed framework, the OMS server is tightly coupled to the orchestration layer and provides monitoring functions as well as a real-time representation of the system. On startup it creates object memories for each production cell, according to the high-level RobotML scenario description. These object memories can host generic and specific blocks. Generic blocks contain information that is relevant for all production cells, such as the production cell identifier or the network interface status. Specific blocks are related to the production cell type and the functionalities that have been deployed. In the considered scenario, the camera can push the intrusion detection information each time such an event occurs, while periodically pushing its processing status (e.g. frame per seconds). These information are described within the RobotML model and can therefore be linked to other production cell models for on-site access. The OMS interface relies on either regular OPC UA or representational state transfer (REST) protocol for access. The latter has made the implementation and deployment of the experimentation quite straightforward and technologically agnostic.

2.4. Verification of Parallel Code Using Frama-C/Para-C

The verification and the validation layer is included on our software framework to ensure the safety and the compatibility of the software being deployed. The user may need to have some information and calculate some metrics on

[2] Acceleo, `http://www.eclipse.org/acceleo/`
[3] Lua, `http://www.lua.org`
[4] ProVis,Visu, `http://www.iosb.fraunhofer.de/servlet/is/35793/`
[5] Unified Automation, `https://www.unified-automation.com/`

the software architecture of the new application. Several kinds of analysis tools can be included in this layer depending on the kind of analysis the user wants to perform on its software. In our use case, we choose to use Frama-C [TBD ref] platform and specific plugin Para-C dedicated to parallel applications analysis. A para-C plugin development is an ongoing project dedicated to parallel code analysis. It is developed for two kinds of purpose. The first one is a re-engineering purpose, where the parallel code is parsed, then its architecture is generated throughout different graphs (i.e. thread control flow graph, data dependency graph etc.). Para-C also calculates metrics that provide information about the software. The second purpose of para-C is to validate parallel software applications and to check whether they are bug-free. It verifies some safety properties according to some parallel patterns. Such properties are absence of deadlocks or race conditions.

2.5. Network Monitoring and Dynamic Reconfiguration Using SDN

The SDN architecture defines three interactions level: the infrastructure level, the SDN controller, and high-level services. At the infrastructure level, one retrieves all communicating devices that will support data traffic exchange and generation. These devices could be the robots, cameras, network switches and routers in the factory. To support the SDN architecture, we provides a specific software called NEON. NEON basically establishes and maintains the communication with the SDN controller, collect all devices network interfaces states and capabilities, and is able to interpret and activate commands coming from the SDN controller. The SDN controller is the link between high-level applications and the infrastructure. Its role is to collect all information from the architecture, to generate an abstracted view of the system to high-level services and to interpret high-level actions into concrete network actions to be pushed to the infrastructure. The high-level services layers has a direct access to the SDN controller through a Northbound API. This API provides a wide range of output information on different format (JSON-based REST API, binary, JSON-based CLI). The output information could be used to monitor the configuration of the network and traffic performance at near to real time. High-level services could also push commands to request a new configuration or advanced information such as the average signal strength on a specific wireless link. It is also possible to high-level services to push network actions directly on infrastructure devices. Such actions are not interpreted by the SDN controller and could target the realization of a specific task related to the device function, i.e., execution of a high-level method (not related to network services) on a specific device. For the latter capability, a plugin system allow to enhance the SDN controller and/or the NEON software with new high-level functionalities.

2.6. Reconfigurable Camera Software Based on User Source Code

As we said previously, the Sybot is a collaborative robot dedicated to help the humain operator in his tasks. The Sybot has 2 main modes: a learning mode where it directly collaborates with the operator to learn how to perform a task. In this mode, the robot records all the information about the trajectory that must be followed and the actions that must be performed. The second mode is an automatic mode which is triggered by the operator at the end of the first mode. When the robot is in this mode it follows the recorded trajectory and performs the recorded actions. Equipped by a sharp object, the Sybot may cause danger to human operator. In our use case scenario, we aim to enhance the human operator safety while reducing hardware cost. We aim to avoid the robot arm replacement. The idea consists on connecting the robot to an external camera and to create a safe zone: when a human is approaching the robot, an alarm is triggered and the robot is stopped.

The camera scenario is composed by two main steps. A first step is to ask the camera to work as a regular monitoring system. The *Orchestrator* copies and remotely compiles a base program source code onto the camera device. This base program streams back the raw video to the *Orchestrator* for monitoring purpose. This program is not critical and does not need to be checked by the analysis tool. Then, once the network has been set up between the camera and the robot at the system level, we ask the camera to reconfigure its software to work as an intrusion detection device. For this purpose, the detection code is provided by the user and is critical for two main reasons. First, the code is related to the security of the persons and goods. Second, this code has been developed by a potentially external actor without a deep knowledge of the relying hardware. Furthermore, the owner of the equipment might require certifications about the code reliability before deploying. Therefore, this second step is triggered once a source code analysis is performed by Frama-C/Para-C, as seen previously. The detection code used in this scenario is a naive OTSU threshold filter[12] with parallel code sections. It discriminates each pixel of the frame into black and white

classes. It then calculates the percentage of black pixels in the result and decides whether an object is detected or not by comparing this percentage with a given threshold percentage. Detection algorithms have been widely studied in the literature. However, for this experimentation we use our own ANSI C, Posix Threads implementation because of the strong requirements on source code that are expected by Frama-C/Para-C at this point. We were able to achieve more than 24 frames per second on the Raspberry Pi 2, using the $900MHz$ ARMv7 quadcore processor and the dedicated 640x480 VGA camera (no use of GPU). Also, the camera reconfiguration time, switching from monitoring to detection with source code analysis, deployment and compilation only takes a few seconds.

3. Conclusion

Machine-to-machine communications is becoming one key topic for the future of industry. In the CPPS context, where systems have to be context-aware and self-organized, the M2M communications not only provides a transport layer with standard discovery protocols, but also a complete service-oriented infra-structure. In this work we propose a framework that facilitate production cells reconfiguration by easily adding or removing functionalities. By using our framework, the reconfiguration of the production cell does not imply more than a simple push on a button in a GUI deployed on supervision device (e.g. a PC, a tablet, a phone, etc.). Then, the system is automatically reconfigured using OPC-UA orchestrator server according to the RobotML scenario. We were able to deploy a complete reconfigurable system within the Sybot collaborative robot. Online automatic reconfiguration can significantly reduce the exploitation costs (TCO) as it does not require stopping the production for the maintenance and the integration of new equipments. One feedback from the industry about this work is the early design stage of the system, and that the current state of production lines is quite far from integrating such technologies. However, this is a natural and foreseen approach.Several perspectives are considered for our work: first cosolidate the infrastructure by using it in other use cases, then enhance monitoring and verification and validation internal services to be more suitable to industrial practices and finally integrate the Reference Architectural Model Industrie 4.0 (RAMI 4.0) model[13] proposed by the OPC UA foundation.

Acknowledgment

This work was supported by the HII CPS - High Impact Initiative on Cyber Physical Systems of the European EIT Digital organization, as well as the Echord++ - European Clearing House for Open Robotics Development project.

References

1. N.n. securing the future of german manufacturing industry: Recommendations for implementing the strategic initiative industrie 4.0. Final report of the Industrie 4.0 Working Group, acatech; 2013.
2. Monostori, L.. Cyber-physical production systems: Roots, expectations and R & D challenges. PROCEDIA CIRP; 2014.
3. R. P. Parker, A.W.. Manufacturing flexibility: Measures and relationships. *European Journal of Operational Research 118* 1999;:429–449.
4. *Modular design for machine tools*. McGraw-Hill Education; 1 edition; 2008.
5. V. MALHOTRA T. RAJ, A.A.. Excellent techniques of manufacturing systems: Rms and fms. *International Journal of Engineering Science and Technology* 2010;**2(3)**:137–142.
6. Nicoleta-Cristina Gaitan Vasile Gheorghita, I.U.. A survey on the internet of things software arhitecture. *International Journal of Advanced Computer Science and Applications* 2015;**6**(12).
7. Dhouib, S., Kchir, S., Stinckwich, S., Ziadi, T., Ziane, M.. Robotml, a domain-specific language to design, simulate and deploy robotic applications. In: Noda, I., Ando, N., Brugali, D., Kuffner, J., editors. *Simulation, Modeling, and Programming for Autonomous Robots*; vol. 7628 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-642-34326-1; 2012, p. 149–160. doi:\bibinfo{doi}{10.1007/978-3-642-34327-8_16}. URL `http://dx.doi.org/10.1007/978-3-642-34327-8_16`.
8. The opc unified architecture (ua). OPC Foundation; ???? `https://opcfoundation.org/about/opc-technologies/opc-ua/`.
9. open62541: An open source and free c (c99) implementation of opc ua stack. TU Dresden PLT, Fraunhofer IOSB, RWTH-PLT; ???? `http://open62541.org/`.
10. The object memory sever (oms). DFKI GmbH; ???? `http://www.dfki.de/omm-tools/oms.php`.
11. The object memory modeling incubator group. W3C; ???? `https://www.w3.org/2005/Incubator/omm/`.
12. Otsu, N.. A threshold selection method from gray-level histograms. *Systems, Man and Cybernetics, IEEE Transactions on* 1979;**9**(1):62–66. doi:\bibinfo{doi}{10.1109/TSMC.1979.4310076}.
13. The reference architectural model industrie 4.0 (rami 4.0). OPC Foundation; 2015.