Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 24 (2018) 100–106

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

4th International Conference on System-Integrated Intelligence

Communication Concept of DeConSim: a Decentralized Control Simulator for Production Systems

R. Davide D'Amico[a,*], Georg Egger[a], AndreaGiusti[a], Erwin Rauch[b], Michael Riedl[a], Dominik T. Matt[a,b]

[a]*Fraunhofer Italia Research, Via A. Volta 13 A, 39100 Bolzano (BZ), Italy*
[b]*Free University of Bozen-Bolzano, Piazza Università, 1, 39100 Bolzano (BZ), Italy*

Abstract

In recent years, the price pressure and demand for personalised products in manufacturing increased continuously. As a reaction, technologies from the Information and Communication Technology (ICT) sector are applied by the industry, this trend is also known as digitalisation or Industry 4.0. Digitalisation might be effectively supported by simulation of production lines and will become crucial for manufacturing companies who want to benefit from the advances in ICT technologies, thus improving efficiency and competitiveness. Many agent-based simulators exist, nevertheless, most of them were developed for biological and social sciences whose use and specifications differ significantly from industrial applications. The commercial ones are expensive in terms of licensing, support and require specific training. We propose a communication concept for the simulator that we are developing called DeConSim, which aims at decentralized systems and is composed of several modules (representing resources, each with its own task and intelligence) and a core. DeConSim is capable of operating in two different modes: simulation mode and real-time mode. The former is used to achieve pure simulation results as fast as possible. The latter operates in real-time enabling DeConSim, in contrast to other existing simulators, to become the operational core of the control communication infrastructure of a real production line.

Keywords: simulation, decentralized architecture, cyber-physical-systems (CPS), multi-agent systems (MAS), industry 4.0, RAMI 4.0

1. Introduction

The increasing need for customized products and the shorter product life cycles require today's production to be rapidly adaptable to different products. A promising approach to face the above-mentioned challenge is to design flexible and reconfigurable production systems [16]. In fact, the adaptability of a production line to a high variety

* Corresponding author. Tel.: +39 0471 1966900
E-mail address: davide.damico@fraunhofer.it

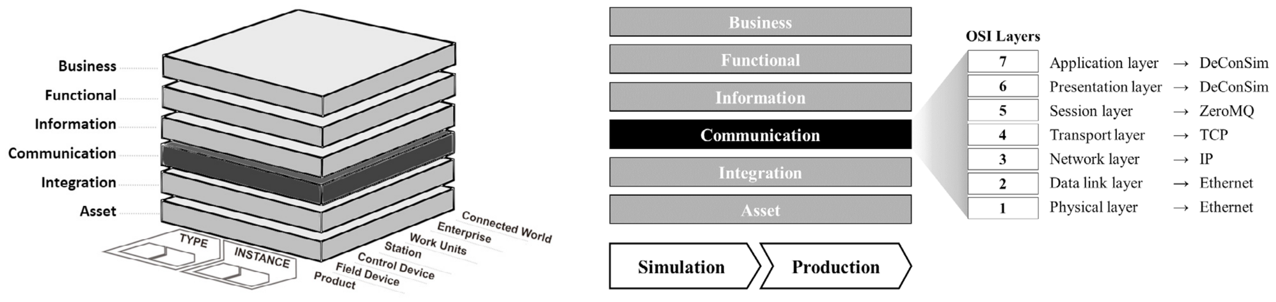

Fig. 1. On the left the RAMI 4.0 three dimensional model [12], on the right the model in two dimensions (business processes and product life cycle) - communication layer is highlighted and expanded in the seven OSI layers, where DeConSim and ZeroMQ have been placed.

of products within a similar product group depends largely on reconfigurability, flexibility in scheduling tasks, and flexibility in accepting and performing new tasks. Over the last few years, numerous projects explored and developed solutions based on reconfigurable manufacturing systems for highly customizable products, which reveal advantages, such as the possibility to reuse the same line to produce different variants, and disadvantages, such as the high complexity of these systems [5, 14].

So far, there are basically two approaches for the control topologies of production systems: traditional centralized architectures that are widely used in industrial practice, and decentralized architectures as an advanced approach but as far on a lower TRL (Technology Readiness Level). Centralised topologies, are typically designed for the production of large batches and can be optimized for specific products. Recent results regarding decentralized production systems promise to be more flexible, especially regarding unforeseen changes, and dynamically reconfigurable for small and medium batch sizes [3]. Modern production systems are highly complex and difficult to model. To address this challenge a reference architecture has been proposed in [12] and is called *Reference Architectural Model Industrie 4.0* (RAMI 4.0) (see Fig. 1). The aim of this paper is to explain in detail the communication concept adopted by the simulator of decentralized production systems we are developing.

Modelling & Simulation tools

Simulation and analysis tools simplify the design of production systems and allow one to optimize and tailor to specific needs this kind of complex systems. In fact, new concepts or systems that meet the expectation of modern manufacturing can be tested and validated before implementing them. [13].

Even though several open source platforms support the simulation of multi-agent topology [2], they are not directly applicable for modelling distributed production systems. In fact, they are for example focused on social [6], biological [20] science, thus, they lack some important capabilities such as communication between agents and distributed decision strategies. There are also commercial solutions [4] available, but on the other hand, they are expensive in terms of licensing, support and require specific training. It was concluded by the authors that implementation with existing simulators represents at least as much effort and risks as rewriting one from the scratch, particularly because of the lack of real-time capabilities [15, 1].

Simulation tools for decentralized systems need reliable communication ways combined with fast and efficient scheduling strategies. A modular and scalable structure is another desired tool, motivated by the need of always on varying manufacturing processes and flexible systems. The simulator we are developing to address the above-mentioned challenges is called DeConSim. DeConSim is composed of several modules (representing resources, each with its own task and intelligence) and a core. The core mainly monitors the status of the linked modules. The modules talk with each other via an event-based communication mechanism. Moreover, the simulator shall take information from a previous design step and store those information in a database.

DeConSim in RAMI 4.0 model

As shown on the left side of Fig. 1, the RAMI 4.0 architecture has been represented as a three-dimensional model where a typical environment of industry 4.0 can be modelled. RAMI 4.0 is one of the main reference architecture, which companies can implement to follow the trend of industry 4.0 [22]. The three axes from the left represent respectively: (i) the layers of an industry from the *asset* (that identify the real things) to the *business* processes, (ii) the product life-cycle and value steam (where the type stands for the development part or simulation and instance stands for all the real production phase and maintenance), and (iii) the hierarchy level from the product to the connected world [17]. The simulator shall meet the requirements of RAMI 4.0 which is gaining importance in *Industry 4.0* environments. As shown in Fig. 1, from the RAMI 4.0 model we consider, in this paper, the *communication layer*. This layer takes care of unifying the mechanism of interaction, defining the data format, providing control and managing services of underlying integration layer. The communication layer can be further divided into the seven (Open Systems Interconnection) OSI layers [11], described in Fig. 1, where the simulator covers the last two layers.

2. Requirements

Among the fundamental requirements are: the possibility of modelling a modular production environment, the possibility for modules to have their own intelligence and communication interface, and that the simulator shall be capable of two modes, *real-time mode*, and *simulation mode*. A central part of the control is considered, to handle orders, record the status and informs the user of potential problems. Those tasks are assigned to the core. When in simulation mode the core also buffers all outgoing communications of the modules and handles them in a sequential order. In the following subsections, the simulation mode and real-time mode are described in more detail.

2.1. *Simulation mode*

With this mode, we can obtain simulation results as fast as possible. That is the way in which the overall production system can be tested before the actual operation. The core divides the overall simulation duration in short time samples and sequentially calls the modules. The latter shall execute all the processing related to that time step, which includes reading from and reacting to any events. Outgoing communication to other modules or to the core shall be returned to the core at the conclusion of the processing. Moreover, the core needs to buffer the messages until the next time step, in order to ensure coherent timings (see Fig. 3 - left).

2.2. *Real-Time mode*

The simulator in real-time mode shall work either as controller of the physical production system or as hardware-in-the-loop testing of physical devices. In the first case, the simulator acts as the controller of the system, distribute the messages and checks the status of each module connected to it. In the second case, a module of the simulator can be used to replace one or more devices in the system. In the case in which one or more devices are not ready or available to work, a module shall replace those and run the system as well. For this purpose, a real-time mode has been integrated.

The major difference with respect to the simulation mode is that in the real-time mode the core does not control the timings of the modules. Instead, each module reacts to I/O events immediately and may trigger any resulting action at any time. The core will forward the messages as they arrive (see Fig. 3 - right).

3. Concept and design of DeConSim simulator

The following section explains the process from the concept to the design of DeConSim. It starts from the adopted architecture to continue with a detailed analysis of the mechanism of communication chosen for the simulator.

3.1. Architecture

The architecture of DeConSim consists of a master-slave model of communication. Such model handles all configurations, control starts and shut down of modules, and is in control of time steps.

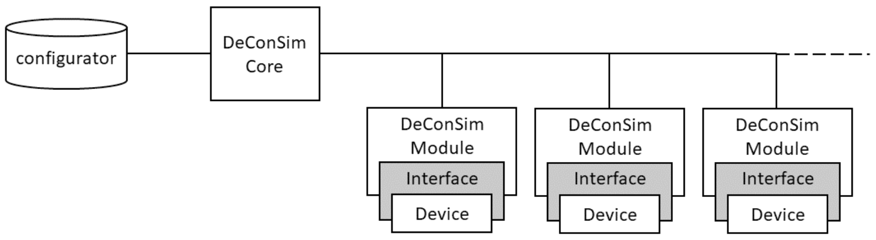

Fig. 2. Overall structure of DeConSim. The core takes the configuration from the configurator. On the other hand, modules, which represent physical devices in the system, are connected to the core.

With reference to Fig. 2, *DeConSim Core* is the principal component of the system, which includes configuration parameters and a list of all modules that the system will include. The core and the modules communicate through IP-based connections, whereas a protocol defined in this document governs the content and the sequence of the messages. The core takes the information of the configuration of the system (such as the port, the address, the operation mode, etc.) from the configuration block. A *DeConSim Module* represents a module or function within the DeConSim framework. For example, a physical device could be connected to our system through an appropriate interface that can communicate on one side with the device (even with its proprietary communication) and on the other side with the simulator with the adopted communication protocol (See Fig. 2).

3.2. Messaging protocol

Due to the modular architecture of DeConSim, particular attention must be given to software interfaces. Even in simulation mode, when all modules might run on the same machine, a mechanism for communication between modules must be specified. The authors concluded that Inter-Process Communication (IPC) mechanism like queues or pipes were too limiting when it comes to distributing the system over several platforms, as required by the real-time mode. Hence, a mechanism based on the Internet Protocol (IP) has been chosen. Of course, IP protocol offers a vast number of communication protocols. Possibilities include direct socket-based communication on transport layer protocols (most prominently UDP or TCP), or on generic message exchange protocols in application layer like MQTT [21] or XMPP [7], up to specialized protocol suites like OPC UA [9] or Unide [8]. When looking at the top-level requirements of DeConSim (see section 2) it becomes clear that one of the major tasks of internal communication will be the orchestration of the modules. In real-time mode, any of the above can be adopted but the simulation mode requires strict control over timings. Among the possibilities, one stood out: *ZeroMQ* [10], for the reasons detailed next.

As introduced above, the communication between core and modules is IP based, allowing communication on the local machine but also to other machines. ZeroMQ has been chosen because it supports both asynchronous and synchronous behaviour, the latter is of interest for simulation mode. It has binary protocol for superior speed as opposed to i.e. HTTP- or XML-based methods, ensure reliable operations, i.e. does not require server to be started first, - Does not rely on IP addresses for identification of participants, so IP addresses can be dynamic and it does not depends from other libraries. Fig. 1 (right) shows the role of ZeroMQ in the communication of the RAMI 4.0 architecture. ZeroMQ allows furthermore maximum flexibility in terms of platforms, distribution over machines and programming languages. ZeroMQ allows several built-in patterns, such as request-reply, publish-subscribe, pipeline and exclusive pair [10]. The explanation of how every pattern works exceeds the aim of this paper and we redirect the interested reader to [10] for further details. The *request-reply* pattern has been chosen for our purpose. The envelope in ZeroMQ is a way of safely packaging up data with an address, without touching the data itself. By separating

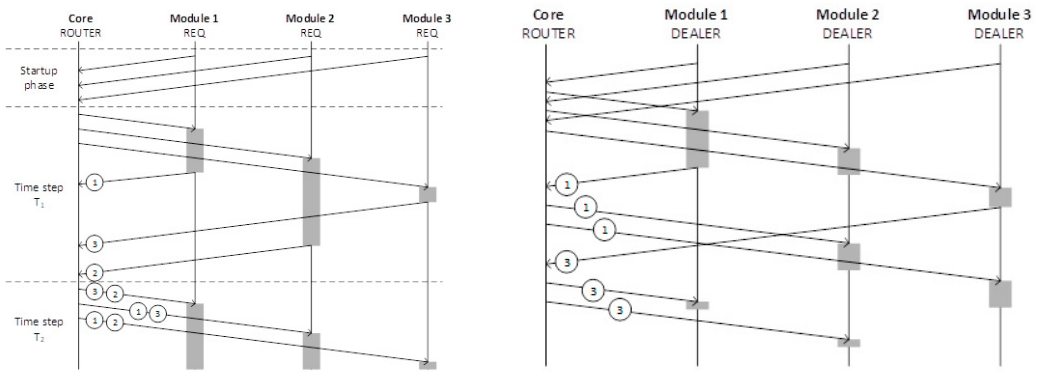

Fig. 3. Two examples of how the simulator works in *Simulation mode* (left), and in *Real-time mode* (right).

addresses into an envelope we make it possible to write general purpose intermediaries such as APIs and proxies that create, read, and remove addresses no matter what the message payload or structure is [10].

Two different patterns have been chosen for the two different modalities (simulation and real-time mode). *req-to-router* for the simulation mode and *dealer-to-router* for the real-time mode [10]. They will be described in detail in the following paragraphs.

Simulation mode in ZeroMQ

In simulation mode, the pattern *req-to-router* has been chosen. It gives us an asynchronous server that can talk to multiple *req* clients at the same time. The core shall bind to a ZeroMQ port, which is defined in the configuration file, configured as *router* socket, while the modules will connect to the core via the *req* socket. The reason why we chose a router socket is that creates identities for its connections, and passes these identities to the caller as the first frame in any received message, as defined in the above-mentioned envelope, and because it allows asynchronous communications. This means that receiving messages from the core does block modules, which is required in this mode and ensure that modules wait for an answer of the core every time step. Recall that the *req-to-router* pattern is a one to many version of the main request-reply (*req-rep*) pattern, the latter being an exchange of messages in strictly alternating directions, i.e. first a request, then a response, all over again. When the core sends data to the module, it's released from the interlock and may process the incoming data, read I/O, etc. The module may take its time to react to those inputs and upon conclusion, the module might sends the potential data back to the core and goes into listening mode again. The core buffers all outgoing messages from the modules and waits for the conclusion of the slowest module before triggering the next time step. This behaviour allows for the strict distinction of the various time steps and ensures coherent timing among the modules.

In Fig. 3 (left), an example of the simulation mode is given. After the start-up phase, where all the linked modules send the message to the core that they are ready to work, the core can send the respective job (the grey rectangle) for each module. Only once all the modules have completed the respective job, and send back to the core the message that they have completed the job, the core might send the new job to each module.

Real-Time mode in ZeroMQ

In real-time mode, the pattern *dealer-to-router* has been chosen. It gives us asynchronous clients talking to asynchronous servers, where both sides have full control over the message formats. Indeed, unlike the simulation mode, in the real-time mode, the modules connect to the core via the *dealer* socket. This means that the strict alternation of message directions can be broken up and receive does no more block the modules. The core still receives the outgoing messages and forwards them to the other modules as needed. In Fig. 3 (right), an example of the real-time mode is given. In that particular case, after the asynchronous start-up phase, module 1 wants to send a job for module 2 and 3, thus, the core, working as a sort block, redirect the message from module 1 to module 2 and 3. Then the analogue happens for module 3 who wants to talk with module 1 and 2.

4. Conclusions

We presented a communication concept for a novel simulation approach for production systems that can support decentralized architectures. To the best of the authors' knowledge, it does not exist an open source simulator, which might work either as controller of the physical system or as hardware-in-the-loop testing or as the simulator to simulate the system. From the DeConSim point of view, it does not matter if a module is a physical device or a simulated one. This allows us to start using our laboratory set up without having all the machines installed in advanced.

Additionally, over the past two decades, several researchers fostered agent-based technology to solve manufacturing scheduling problems [19]. Even agent-based solutions are recognised as promising, however, benchmarks where different approaches are tested are not available yet [18]. DeConSim could be used to test such different approaches, either centralized or agent-based. Indeed, the scheduler can work as a module connected to the simulator environment.

Furthermore, DeConSim is not limited to a particular programming language or platform. This is important for the flexibility and the scalability of the manufacturing system and ensures the ease of implementation of any production or resource module. DeConSim does not have a graphic user interface and is not yet user-friendly. A stable version of DeConSim will be tested and run in a realistic production environment, a laboratory set-up with several devices including robot arms, laser machines, and a transport system, among others.

Acknowledgements

This research is part of the project DeConPro (Decentralized control of production processes in the factory of the future) and has received funding from the European Regional Development Fund ERDF 2014-2020 under the grant agreement with project code FESR1017 (CUP B56J16001730001).

References

[1] Barbosa, J., Leitao, P., . Simulation of multi-agent manufacturing systems using agent-based modelling platforms, in: 2011 9th IEEE International Conference on Industrial Informatics, pp. 477–482.
[2] Berryman, M.J., Angus, S.D., 2010. Tutorials on agent-based modelling with netlogo and network analysis with pajek, in: Complex physical, biophysical and econophysical systems. World Scientific, pp. 351–375.
[3] Bousbia, S., Trentesaux, D., 2002. Self-organization in distributed manufacturing control: state-of-the-art and future trends, in: IEEE International Conference on Systems, Man and Cybernetics.
[4] Critical, 2014. Simulators comparison. URL: `https://paginas.fe.up.pt/~ei09035/thesis/SimulatorsComparison_V1.pdf`.
[5] Egger, G., Rauch, E., Matt, D.T., Brown, C.A., 2017. (re-) design of a demonstration model for a flexible and decentralized cyber-physical production system (cpps), in: MATEC Web of Conferences, EDP Sciences.
[6] Epstein, J., Axtell, R., 1996. Growing Artificial Societies. Social Science from the Bottom Up. MIT Press.
[7] Foundation, J.S., 2001. Xmpp. URL: `https://xmpp.org/`.
[8] Fundation, E., . Unide. URL: `https://www.eclipse.org/unide/`.
[9] Fundation, O., 2008. Opc ua. URL: `https://opcfoundation.org/about/opc-technologies/opc-ua/`.
[10] Hintjens, P., 2007. Zeromq. URL: `http://zguide.zeromq.org/page:all`.
[11] ISO/IEC 7498-l, 1994. Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Mode. Standard. International Organization for Standardization. Geneva, CH.
[12] ITCOM, VDMA, ZVEI, 2015. Reference architectural model industrie 4.0 (rami 4.0). URL: `https://www.plattform-i40.de`.
[13] Mourtzis, D., Doukas, M., Bernidaki, D., 2014. Simulation in manufacturing: Review and challenges. Procedia CIRP 25, 213 – 229. 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution.
[14] Rauch, E., Matt, D.T., 2017. Designing assembly lines for mass customization production systems. in Modrak V. (Ed.) Mass Customized Manufacturing Theoretical Concepts and Practical Approaches. pp. 15-35. 1 ed., CRC Press, Boca Raton.
[15] Sallez, Y., Berger, T., Tahon, C., 2004. Simulating intelligent routing in flexible manufacturing systems using netlogo, in: Industrial Technology, 2004. IEEE ICIT '04. 2004 IEEE International Conference on, pp. 1072–1077 Vol. 2.
[16] Scheifele, S., Friedrich, J., Lechler, A., Verl, A., 2014. Flexible, self-configuring control system for a modular production system. Procedia Technology 15, 398 – 405. 2nd International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering.
[17] Schulte, D., Colombo, A.W., 2017. Rami 4.0 based digitalization of an industrial plate extruder system: Technical and infrastructural challenges, in: IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, pp. 3506–3511.
[18] Shen, W., 2002. Distributed manufacturing scheduling using intelligent agents. IEEE Intelligent Systems 17, 88–94.

[19] Shen, W., Wang, L., Hao, Q., 2006. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 36, 563–577.
[20] Spencer, S.L., Gerety, R.A., Pienta, K.J., Forrest, S., 2006. Modeling somatic evolution in tumorigenesis. PLOS Computational Biology 2, 1–9.
[21] Stanford-Clark, A., Nipper, A., 1999. Mqtt. URL: http://mqtt.org/.
[22] Zezulka, F., Marcon, P., Vesely, I., Sajdl, O., 2016. Industry 4.0 an introduction in the phenomenon. IFAC-PapersOnLine 49, 8 – 12. 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.