Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 52 (2020) 127–132

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

5th International Conference on System-Integrated Intelligence

A deployment-friendly decentralized scheduling approach for cooperative multi-agent systems in production systems

Georg Egger[a,*], Dmitry Chaltsev[a], Andrea Giusti[a], Dominik T. Matt[a,b]

[a] *Fraunhofer Italia, via A. Volta 13A, 39100 Bolzano, Italy*
[b] *Free University of Bozen-Bolzano, piazza Università 1, 39100 Bolzano, Italy*

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

Abstract

Decentralized control paradigms are becoming more and more attractive in an ever-changing commercial environment, where there is a strong trend towards smaller production lot sizes. Whereas centralized scheduling might find a global throughput optimum (even at high computational and implementation cost), decentralized scheduling decisions in a multi-agent system are much more manageable and agents are more robust to handle any interruptions that might take place on the production floor. Compared to a centralised architecture, the development, testing and commissioning is definitely more complex, as it requires the availability of the physical units. Yet these aspects are not visited frequently by research activities.

This paper details a novel implementation approach of a multi-agent based production control, that was developed for a lab-contained production environment that serves as test-bed for decentralized scheduling algorithms, with both a nominal operational mode and a simulation mode. The latter one is introduced to ease up the deployment process of the system. The description of the new approach is illustrated with different examples.

Keywords: Production execution system; multi-agent system; decentralized scheduling; distributed process control; Industry 4.0

1. Introduction

Modern economic trends show a tendency to smaller lot sizes produced in industry. Advances in the field of Information and Communication Technology (ICT) help us to achieve this goal by allowing implementation of computationally intensive production organization methods at low cost [2]. Also, demand for custom-made-products has been increasing as customers are looking for products that meet exactly their specific needs and preferences. These factors put high pressure on manufacturers to produce a small number of made-to-order products where the production process of one item is somewhat different from the production process of another item [3,4,5]. In turn, the dynamics of orchestrating such production environments become more and more challenging. A typically used production control architecture is defined e.g. by ANSI/ISA 95 and IEC 62264 [25,26] respectively (see also Fig. 2). While these hierarchical control strategies work well with static mass production demands, where the production process is defined well ahead of time and stays unchanged over a long period of time, a decentralized control system can be more flexible when handling regular changes in the production process [7,9,8,6]. Such a decentralized system can be implemented by autonomous, cooperative agents. Whereas in a hierarchical control paradigm lower-level entities are expected to simply execute what they've been commanded by a higher-level entity, the entities of a multi-agent system communicate on par to perform tasks in a cooperative way, negotiating with other agents towards mutually acceptable agreement on the basis of a predefined optimization criteria [1]. As an example of a highly-flexible environment, the management of airport resources and facilities is described in [8].

Scheduling in Multi-Agent Production Systems. A Multi-Agent System (MAS) is defined as a group of agents organized according to specific, and precisely defined principles (e.g. architecture, messaging style, communication protocols) [16].

10.1016/j.promfg.2020.11.023

In Fig. 2, the functionality of components of centralized and decentralized models are listed. A MAS can achieve near-optimum efficiency, as agents have to base their decision on a subset of the production data, and hence find local optimums. However, finding the best possible schedule is also challenging and time-consuming for a centralized system, both from a development point of view (requires updates as the production changes), and also due to the computational complexity (belongs to the class of NP problems [10,9]). In the production of custom-made items, investing in pursuing the optimum throughput can be considered of lower priority with respect to the ability to reconfigure the production environment quickly to changing needs. Calculating an optimal schedule might be even more time consuming than manufacturing an item with a sub-optimal schedule. A scheduler may not find a global optimum within the available limited computation resources and communication time, rather it will find a feasible sub-optimal schedule [11].

When a workflow is planned in a decentralized environment, only a part of the overall production plan might be taken into consideration by each agent (Fig. 1), thus the amount of constraints and dependencies that are considered is typically much smaller compared to the overall production process [12]. This makes scheduling a job for any agent manageable [13] and agents are more robust to handle any disruptions that might take place on the production floor [9,17].

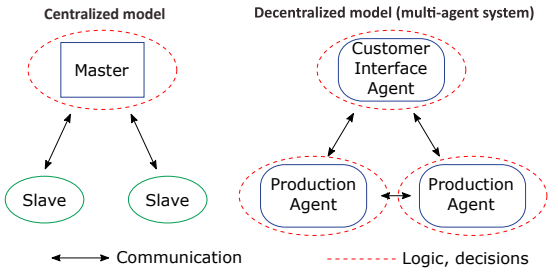

Fig. 1. Centralized vs. decentralized control schemes compared. In the centralized model decisions are made only within the master, whereas in the decentralized model decisions are taken within local resources dependent of both local events or measurements and information exchange with other resources.

An agent is characterized, among other things, by autonomy (can independently carry out complex tasks and make decisions), proactivity (can take the initiative to perform a given task), and communicativeness (can interact with other agents to help with accomplishing their own goals as well as goals of others) [20]. When an agent communicates with another agent, it can request information or request the accomplishment of a task. Also, an agent can negotiate with another agent when making a decision, but it cannot command another agent. While taking optimization decisions, constraints of other agents (within its limited visibility) are considered for the benefit of the whole system. A cooperative agent, when it has made a commitment to do some task, puts all its resources to accomplish this task and withdraws from the completion of the task only if there are unavoidable circumstances that prevent it from accomplishing the task. If this is the case, then this agent communicates with other agents notifying about the current situation. This approach allows the dissemination of information pertaining to changes in the system in a timely manner [13,1]. An agent consists of a software program with its logic, communication capability, and decision-making capabilities. In many cases, the agents can be connected to a physical machine or a robot.

We refer to a production plan as a set of mutual constraints that are in place during the production of an item, where an item is the final product that a customer has ordered. The production of an item consists of several tasks, some of them might be performed in parallel. A production plan might be represented in the form of a connected acyclic graph (tree-graph) with the root node being the last task of production of the item, and children-nodes on this graph are dependent upon tasks that have to be completed before a parent-node task can be worked on [18] (Fig. 3). Scheduling is an assignment of tasks from the production plan to different agents in control of physical objects like machines or robots, with a specific timeslot of starting and finishing time of actual execution of a given task. Each agent in this model manages its own resource reservations by assigning a timeslot to the execution of a certain task.

Description of the approach. This paper proposes an approach where a multi-agent production system is capable of two different modes: the first is a regular *production mode* and the second one is a *simulation mode* where the messages exchange (requests, responses, and negotiations between agents) are entirely compliant to the regular production mode, but the actual production steps with mechanical work such as cutting, drilling, welding can be omitted. This mode brings several practical benefits, as it allows to perform testing, tuning and debugging of the MAS before switching to the production mode. This is possible because the interactions between the agents are exactly identical in both modes, and so is the programming code. Additionally, the simulation mode provides a way to test communication between agents and test scheduling capability before the actual machines are installed in a production line. As soon as cognitive and communicative part of each agent is ready, the MAS system is tested for inter-communication, coherence, and cohesion. Previous work [19] is extended by implementing a communication scheme of negotiation protocols between agents, in turn this is accomplished based on the Foundation for Intelligent Physical Agents (FIPA) contract net and request interaction protocol [24]. The system architecture includes a lightweight communication core responsible for transparently and efficiently route messages to the destination agents. In deployment phase, the core can be tapped easily for centralised debugging and monitoring. Section 2.1 provides more details on the simulation mode and the communication aspects.

Another aspect intrinsic to our system is the ability to schedule tasks without a dedicated schedule-supervisor-agent. Instead, all agents that are involved in the production process negotiate with each other regarding the assignment of tasks among themselves. The same communication pattern of negotiations is used during unexpected situations in the production process. This aspects are covered in detail in section 2.2.

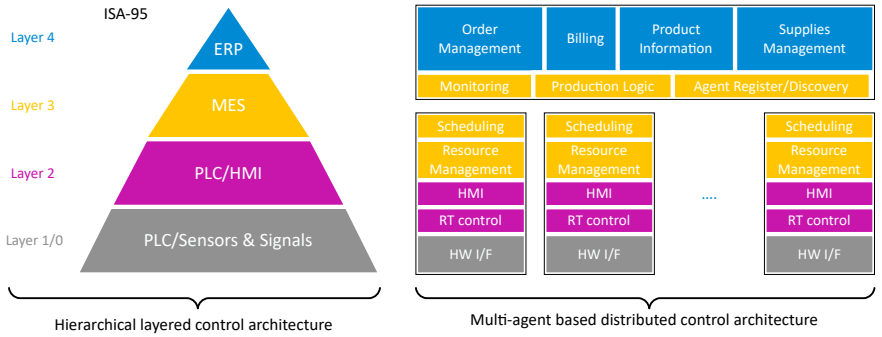

Fig. 2. Centralized vs. decentralized control schemes compared

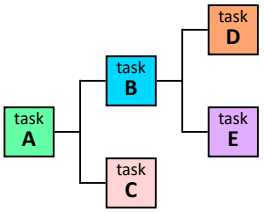

Fig. 3. Production plan with inter-dependencies of tasks.

Finally section 3 describes the results of the experiments that were conducted with the prototype software. Finally, section 4 and section 5 conclude the paper with a critical review and outlines of future work.

2. Decentralized Scheduling for Simulation and Production

2.1. Simulation Mode

In [9] Paulo Leitão describes the difficulty of implementing MAS in the industry. The author identifies the difficulty of a transition from a laboratory experiment environment to a real production shop floor. Our proposed simulation mode can be used to resolve this issue.

Core and Communications between Agents. In order to improve simplicity of deployment, we introduce a core, that acts as a transparent communication channel with receipent-specific forwarding. The communication link is implemented on top of TCP/IP and ZeroMQ, a high-performance messaging library[23]. Compared to plain TCP sockets the benefits are increased reliability due to built-in buffering and multipoint capability. ZeroMQ integrates easily with a lot of programming languages. The messages are encoded in JSON, with the content (fields) following FIPA definitions. The core implements functions of the FIPA Message Transport Service (MTS), and an Agent Management System (AMS) with the Directory Facilitator (DF) [24].

The core is responsible for dispatching incoming messages to the destination agents. In production mode, the core does this on a ad-hoc basis, whereas in simulation mode the time is divided into discrete intervals, and the core buffers incoming messages and forwards them to their individual recipients only after each agent has acknowledged the completion of the individual time step [19]. This approach ensures coherent timing across the agents and simulates the parallel processing of events happening at the physical agents. Any difference in agent communication and computation speed is neglected. As all messages are concentrated in one point, monitoring the activity and communication of agents is easily achieved. Such approach allows e.g. to filter communication belonging to a specific agent and to assess the correctness of its reasoning and actions for debugging purposes.

With this setup, an agent can receive more than one message at a time. The agent then processes all the incoming messages in the order it is listed, then an agent reasons, makes decisions, prepares answers and sends it to the core to be forwarded to its recipients. If any given agent does not have outgoing messages, it notifies the core with a heart-beat message, notifying the core that it is ready to perform any task, but is vacant at this moment of time.

As the agents run in separate processes, it is possible to run several agents on one machine or to distribute them across many machines depending on the workload or processing power of the machines.

Time-scaling Capability of the Simulation Mode. Another aspect of the simulation mode is that it allows to fast-forward a production process that usually takes a long time to complete. Instead of binding the time step number to actual clock, the core proceeds to the next time step as soon as the last agent has completed its operations of the current time step, independently if the actual duration of the time step has expired or not. For example, one time slot corresponds to the duration of 10 seconds. When the core forwards initial messages, it announces that it is time step number one. For all agents, it means that they can assume that they have completed all physical operations that were planned for the first 10 seconds of the production process. It is possible to do it in the simulation mode since there are no physical machines connected to the logical agents. When the core forwards the second batch of messages, it announces that it is time step number two and this gives information to all agents that it is assumed that 20 seconds of the production

have passed since the beginning of the simulation mode process. This process continues until all agents finish their tasks and some items have been produced. This way it is possible to simulate long-duration processes in shorter time periods and to monitor agents' interactions and cooperation during these processes. The disadvantage is that any operation involving interaction between agents lasts at least 2 time steps in simulation mode, whereas in production mode the interaction could be completed in a glimpse of a second. It is the responsibility of the user to ensure that the meaningfulness of the results are not impaired by improper selection of the time step.

Switching to the Production Mode. Once a MAS is tuned and tested in the simulation mode, it can be switched to the production mode. In this mode, the core forwards incoming messages to its recipients as soon as it receives them. Another difference from the simulation mode is that all agents refer to their internal clock as time base instead of the division into time steps.

2.2. Decentralized Scheduling Approach

The initial request for manufacturing comes to the agent that can perform a root-node task (that is the final agent delivering the completed product). Then an iterative process starts where each agent upon receiving a request to perform a certain task does one of the following actions depending on the inter-dependency of a task in the hierarchy of the production plan:

- if a task is a leaf-node task, then an agent checks its internal reservations, finds the first possible time slot when a requested task could be scheduled, reserves this time slot, and then answers to the requesting agent with the time of planned completion of the task,
- if a task corresponds to a node with child-nodes, a request is sent to the agents capable of performing the tasks. The own task is scheduled in available time slots, considering the completion times from the dependent tasks, ensuring the precedence constraints from the production plan.

The scheduling process workflow is shown as an example in Fig. 4. The illustration of every interaction is outside the scope of this paper. Note that child-node tasks could be scheduled in any order. In the above example, there is no restriction between tasks D and E. One could be done first and then the other task, or it could be worked on simultaneously since tasks have different type and could be worked on by two different agents.

Communication Protocols. As starting point we consider a scenario, where any type of task might be executed by only a single agent. In this case, an agent sends a request for performing a task to another agent (the request is sent in the format of FIPA request interaction protocol [24]), another agent schedules this task on its own time line and returns a message with the planned completion time of this scheduled task. However, the situation changes when there is more than one agent capable of performing a given task, then a negotiation is taking place in the format of FIPA contract net interaction protocol [24]. The call-for-proposals message is sent to each agent capable of performing the task. These agents send a response back with a proposal to complete this job at a certain time by using a certain amount of resources [22]. Then, the initiator of the call-for-proposals selects a proposal and confirms that it accepts this proposal and declines all other proposals. Different fitness functions could be used to compare the proposed options. It is possible to consider the cost of the production or the earliest time of task completion as a decision criterion, sometimes the fitness function encompasses a mix of both parameters [21]. Additionally, evaluation attributes could contain the following parameters: overall rating of an agent, on-time delivery rating, reliability, maintainability, customer satisfaction, and others [4]. In our production line, the fitness function considers the duration of the production, i.e. the quicker agent is preferred. The result of the scheduling process are consistent reservations across the agents.

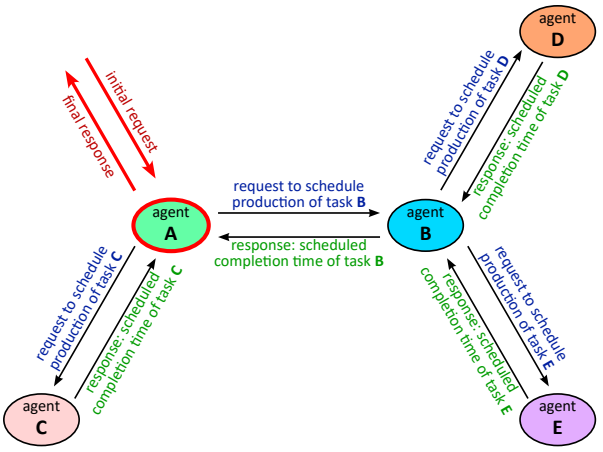

Fig. 4. Scheduling process workflow: Initial request arrives at agent A (task A is the root-node task of the production plan). As task A depends on tasks B and C, agent A sends requests to agents B and C regarding tasks B and C. Task C is a leaf-node, and could be scheduled at any time when agent C is available. Agent C confirms to agent A completion time of task C. Task B depends on tasks D and E, messages are sent to agents D and E. Agent B receives the completion times of tasks D and E, and schedules task B. Agent A receives confirmation of task B. Agent A schedules task A taking into account the completion times of tasks C and B.

Production Process Initiators. An actual manufacturing tasks is triggered either by:

- an internal timer, when the task corresponds to a leaf-node on the production plan
- a message informing the agent about the change of state, when the task corresponds to a node with child-nodes

This approach makes it possible to handle communication during delays. If any agent is confident that the current task will be completed later than a scheduled completion time (for example, due to the absence of materials), then this agent communicates with the agent responsible for the parent-node task.

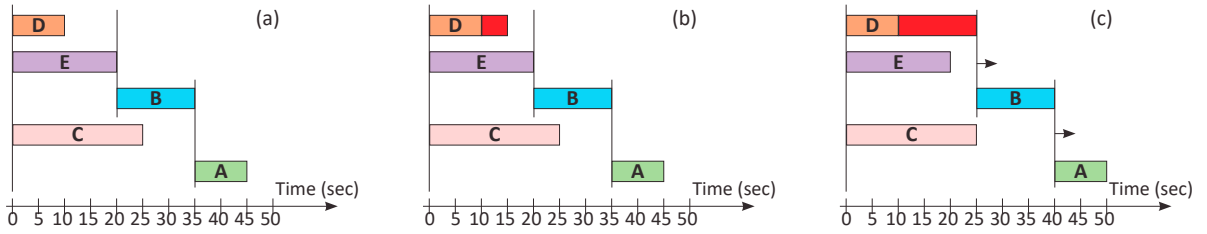

Fig. 5. Agent time lines. (a) shows the initial schedule. In (b) minor delay during the completion of task D occurs, that is covered by the longer duration of E. In (c) the major delay during the completion of task D compromises the schedule of other tasks.

Delays, in general, affect the completion time of the following items and the final product in particular, or it might not affect it at all, i.e. when the delay does not occur in the critical place. Examples of such situations are presented in Fig. 5. We consider the production plan from Fig. 3 and two possible scenarios of delays for task D. In Fig. 5(a), time lines of agents at the moment of the scheduling process are illustrated. If there is a minor delay during an actual completion of task D, task B, that depends on the completion of task D, still starts on time since it must wait for the completion of task E as well, Fig. 5(b). However, if there is a long delay with the completion of the task D, then task B is moved down the time line and consequently, the production time of task A is affected as well, Fig. 5(c).

3. Numerical Results

While experimenting with the software platform, six different scenarios were tested. In the reference scenarios, nine different types of agents capable of certain type of job are considered. A production plan has a tree-structure that is similar to a plan on Fig. 3, but instead this production plan contains 39 distinctive tasks.

The first scenario contained *one physical agent* for each type of task, i.e. there were nine different agents on the production line. When a new order arrived for manufacturing a product that consists of 39 tasks, a schedule compilation for this product took 78 messages to send/receive between agents (one message with a request to perform a task and then one message with the agreement confirmation), this process of scheduling was done in less than 5 seconds.

In all the following scenarios, there was more than one agent per task type (in the second scenario there were two agents per task type, then tree agents per task type and so on up to six agents per task type). In this case, a negotiation took place for each task from the production plan and the number of messages between agents increased substantially. In the Fig. 6, there are results of these six experiments.

It is important to note that the number of messages exchanged between agents has a non-linear dependency from the number of redundant agents (Fig. 6). At the same time, the duration of scheduling process depending on the number of the redundant agents is close to linear dependency (Fig. 6). We regard the result to the distribution of the decision making process among agents and not concentrating it in one entity.

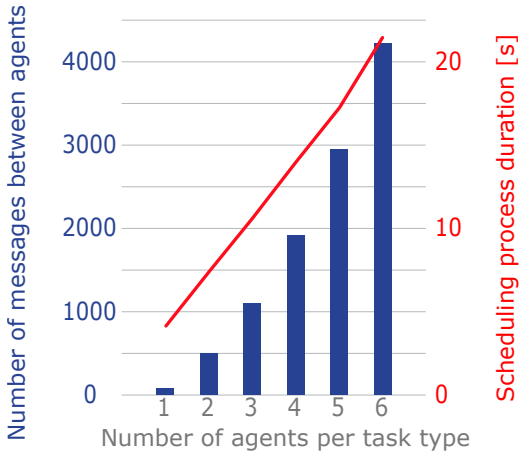

Fig. 6. The number of messages exchanged in a scheduling process and the duration of scheduling process depending on the number of the redundant agents.

4. Known Limitations

According to [9] many MAS research initiatives are carried out only emulating the physical devices. There seems to be a big gap between MAS as they are used in the scientific research and potential practical implementation by commercial manufacturers and production companies. Our research suggests an approach where the same communication protocols are used during the simulation and the production modes with the difference of sub-partitioning communications between agents in manageable subsets for debugging purposes.

The proposed approach has been developed for implementation of a academic use case in a lab environment, hence no operational experiences could have been made. Yet the fundamental workings of the developed framework and the simulation mode have been validated successfully. This includes mainly the scheduling aspects, that represent the most complexity from an inter-agent interaction perspective. The contract net protocol is however known for exponentially growing communication demand, as the number of agents involved in a negotiation grows.

There are however also some functions, that are depending on the physical machines and their real-time behaviour. In order to develop the simulation mode for such a machine, the behaviour must be modelled. Often this can not be done in a time-

efficient manner, and using the physical machine is considered the better approach.

Due to the current implementation of the production mode and the simulation mode, every agent in the system must be put in the same mode. This possibility is missing when a still missing module should be replaced by a software emulation of it's behaviour. The simplicity of the simulation mode becomes a blocker in this scenario, as the timestamps are not compatible.

5. Conclusion

In this work, we have described a novel approach that is used in the implementation of a multi-agent production system with a simulation mode. This mode allows to fine tune communication and cooperation between agents that are installed in the production system prior to the deployment of physical production devices. Our proposed approach makes it possible to partition all agent activities and agent communications into manageable time steps, in this way all actions and logic of an agent could be tested and fine-tuned before switching to the production mode.

Another aspect that was covered in this work is distributed scheduling process that is dispersed among machine-agents that are involved in the physical manufacturing process on the shop floor. The new approach empowers the system to handle delays during the production process with the same tactic as scheduling since all agents that are involved in the production process (machine-agents) use the same pattern of communication (request protocol and contract net protocol) to handle possible delays during the manufacturing process.

In the future, we are planning to test this MAS in the environment when a new additional agent joins a production system. This will trigger rescheduling of all production orders that are not in the production phase yet with the goal to include newcomer agent and by doing so to decrease the production time of an item. Also, the implementation of other methods of distributed control is planned, that will allow benchmarking the results relative to each other.

References

[1] Valckenaers P., Van Brussel H., 2005. Holonic Manufacturing Execution Systems. CIRP Annals, vol. 54, 2005, pp. 427-432.

[2] Rojas R., Rauch E., Vidoni R., Matt D.T., 2017. Enabling Connectivity of Cyber-physical Production Systems: A Conceptual Framework. Procedia Manufacturing 11, pp. 822-829.

[3] Scholz-Reiter B., Görges M., Philipp T., 2009. Autonomously Controlled Production Systems – Influence of Autonomous Control Level on Logistic Performance. CIRP Annals – Manufacturing Technology, vol. 58. 1, pp. 359-398.

[4] Zhang Y., Qian C., Lv J., Liu Y., 2017. Agent and Cyber-Physical System Based Self-Organizing and Self-Adaptive Intelligent Shopfloor. Journal IEEE Transactions on Industrial Informatics, vol. 13. no. 2, 2017, pp. 737-747.

[5] Monizza G.P., Rauch E., Matt D.T., 2017. Parametric and Generative Design Techniques for Mass-Customization in Building Industry: A Case Study for Glued-Laminated Timber. Procedia CIRP 60, pp.392-397.

[6] Dallasega P., Marcher C., Marengo E., Rauch E., Matt D.T., Nutt W., 2016. A Decentralized and Pull-Based Control Loop for On-Demand Delivery in ETO Construction Supply Chains. Proceedings of the 24th Annual Conference of the International Group for Lean Construction, pp. 33–42.

[7] Scholz-Reiter B., Rekersbrink H., Görges M., 2010. Dynamic Flexible Flow Shop Problems – Scheduling Heuristics vs. Autonomous Control. CIRP Annals – Manufacturing Technology, vol. 59. 1, 2010, pp. 465-468.

[8] García Ansola P., García Higuera A., Otamendi F.J., de las Morenas J., 2014. Agent-Based Distributed Control for Improving Complex Resource Scheduling: Application to Airport Ground Handling Operations. Journal IEEE Systems, vol. 8. 4, pp. 1145-1157.

[9] Leitão P., 2009. Agent-based distributed manufacturing control state-of-the-art survey. Journal of Engineering Applications of Artificial Intelligence, vol. 22. 7, pp. 979-991.

[10] Mirshekarian S., Šormaz D.N., 2016. Correlation of job-shop scheduling problem features with scheduling efficiency. Journal of Expert Systems with Applications, vol. 62, pp. 131-147.

[11] Klima M., Gregor J., Harcuba O., Mařík V., 2017. Agent-Based Shop Floor Scheduling. International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2017, Lyon, France, pp. 24-36.

[12] Trentesaux D., 2009. Distributed control of production systems. Journal of Engineering Applications of Artificial Intelligence, vol. 22. 7, pp. 971-978.

[13] Hermann M., Pentek T., Otto B., 2016. Design Principles for Industrie 4.0 Scenarios. 49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, USA, pp. 3928-3937.

[14] Frazzon E.M., Kück M., Freitag M., 2018. Data-driven production control for complex and dynamic manufacturing systems. CIRP Annals – Manufacturing Technology, vol. 67. 1, pp. 515-518.

[15] Wiendahl H.-P., ElMaraghy H.A., Nyhuis P., Zäh M.F., Wiendahl H.-H., Duffie N., 2007. Changeable Manufacturing – Classification, Design and Operation. Annals of the CIRP, vol. 56. 2, pp. 783-809.

[16] Mařík V., McFarlane D.C., 2005. Industrial Adoption of Agent-Based Technologies. Journal IEEE Intelligent Systems, vol. 20. 1, pp. 27-35.

[17] Pechoucek M., Riha A., Vokřínek J., Marik V., Prazma V., 2002. ExPlanTech: Applying multi-agent systems in production planning. International Journal of Production Research, vol. 40. 15, pp. 3681–3692.

[18] Caux C., Barth M., De Guio R., 2000. Graph Based Tools for Production Flow Analysis. 2nd IFAC Conference on Management and Control of Production and Logistics, MCPL 2000, Grenoble, France, vol. 33. 17, pp. 885-889.

[19] D'Amico D., Egger G., Giusti A., Rauch E., Riedl M., Matt D.T., 2018. Communication Concept of DeConSim: a Decentralized Control Simulator for Production Systems. Procedia Manufacturing, 4th International Conference on System-Integrated Intelligence, SysInt 2018, Hannover, Germany, vol. 24, ScienceDirect, pp. 100-106.

[20] Wooldridge M.J., 2009. An Introduction to Multi-Agent Systems. Wiley Publishing, ISBN 978-0-470-51946-2.

[21] Shen W.M., Wang L.H., Hao Q., 2006. Agent-Based Distributed Manufacturing Process Planning and Scheduling: A State-of-the-Art Survey. Journal IEEE of Transactions on Systems, Man, and Cybernetics, vol. 36. 4, pp. 563-577.

[22] Gerber A., Russ C., 2001. A Holonic Multi-agent Co-ordination Server. Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference, FLAIRS–2001, Key West, USA, pp. 200-204.

[23] ZeroMQ distributed asynchronous messaging library. URL: http://zeromq.org/

[24] FIPA, 2002. FIPA Specifications. URL: http://www.fipa.org/repository/standardspecs.html

[25] ANSI/ISA-95.00.01-2010 Enterprise-Control System Integration - Part 1: Models and Terminology, The International Society of Automation.

[26] IEC 62264-1:2013 Enterprise-control system integration — Part 1: Models and terminology.