3rd Conference on Production Systems and Logistics

Modular Software Architecture For Interfacing Online Scheduling Agents With Assembly Planning And Control Systems

Amon Göppert[1], Jonas Rachner[1], Lea Kaven[1], Robert H. Schmitt[1,2]

[1]*Laboratory for Machine Tools and Production Engineering (WZL) of RWTH Aachen University, Aachen, Germany*
[2]*Fraunhofer Institute for Production Technology IPT, Aachen, Germany*

Abstract

Production systems must be more resilient and adaptive due to mass customization and increasingly external disturbances, such as supply chain disruptions or changing policies. As the last chain in the production value stream, assembly systems are especially prone to fluctuations, leading to alternative and more flexible assembly system designs. Online scheduling is a crucial component for dynamically controlling a flexible assembly system.

This work presents a modular software architecture that interfaces between online scheduling agents and control systems. A standardized data model of the assembly system allows for exchanging different scheduling agents during the planning or operation phase. Applications are benchmarking competing algorithms, validating scheduling results by comparison, and seamlessly substituting or updating scheduling algorithms. The standardized data model and interface on the assembly system side facilitate the transition between planning and operation. A simulation model can be interchanged with a control system without extra effort to integrate the control system's scheduling agents. Additionally, the modular architecture enables production-parallel simulation to optimize the running system by evaluating and executing alternative scenarios.

The long-term assembly system performance can profit from the modular architecture by updating the agent during production with advances in online scheduling algorithms (e.g., machine learning). Furthermore, the modular architecture enables the required resilience and adaptability by fast switching from simulation to real control systems and supporting system optimizations during operation.

Keywords

Production control; software architecture; adaptive assembly; online scheduling; simulation

1. Motivation

Various global trends in production technology push producing companies towards being more resilient and adaptive: Customer’s demands are increasingly volatile and individualization is an ongoing change driver. [1,2] Additionally, the recent pandemic still disrupts globalized supply chains and causes a shortage in semiconductors for production systems. [3] As a consequence, production systems and especially assembly systems, as the vulnerable last piece in the production chain, need to be adaptive and resilient to changes. [4,5] A solution are alternative assembly systems that break with the traditional concept of takt time and linear transfer in classical assembly lines. Examples of such concepts are matrix assembly systems [6,7], modular assembly systems [8], agile assembly systems [9], or dynamically interconnected assembly systems [10]. The break-up of linear transfer in those flexible assembly systems allows job routes that can be determined individually for each job. The allocation of processes at work stations for the jobs along the job

route can occur dynamically and with short reactions times. A consequence of the greater flexibility with job routes and the more dynamical character is a greater complexity in the landscape of software components for planning and control of such assembly systems. Also, in those systems changes in the product mix or disturbances, due to rework, missing components or worker shortages, are occurring in high frequency. [11,12] Therefore, online scheduling is a relevant task in such assembly systems. Machining systems are not considered, due to the significantly longer process time, less disturbances, and consequently less need for frequent re-scheduling or online scheduling.

Following the complexity and the demands from online scheduling, multiple specific challenges for interfacing the different software components arise: During production in dynamical assembly systems with many disturbances, online scheduling algorithms must ensure the efficient job allocation. [13] These algorithms must cope with the complexity and temporal variability of boundary conditions. Therefore, traditional rule-based heuristics are not applicable and machine learning (ML) algorithms need to be applied. [14] These machine learning algorithms for online scheduling need to be connected to the control system that requests solutions for job allocation problems. The machine learning models are trained with simulation models that enable the generation of large datasets, required for machine learning training. Due to the significant computational time needed for most machine learning models they are trained before production [14] or with delay until deployment during production [15]. Consequently, in both cases an interface needs to enable the communication between the required simulation models and the online scheduling algorithms. Additionally, during training, a benchmark of different algorithmic approaches supports the choice over an algorithm. During production, alternative parametrizations or algorithms can be tested and deployed for the application with the control system. This demand for exchangeable online scheduling algorithms also results in an interface between the algorithms and the control system or simulation model.

A possible solution for these complexity-driven challenges could be a modular and standardized software architecture that seamlessly connects the online scheduling algorithms with the simulation and control components during planning and production. In the next chapter 2 the state of the art in scheduling and control architectures, the derived research question and in brief the applied methodology are described.

2. State of the Art and Methodology

Literature presenting research regarding online scheduling primarily concentrates on the performance improvement or evaluation of scheduling algorithms, for instance, with the application of deep reinforcement learning techniques, but not on the integration in dynamical assembly system planning and control. [16–19] The existing literature on architectures for production planning and control for flexible production systems focuses on adaptive architectures as multi-agent systems in different heterarchical or hierarchical structures [20–22] or cloud manufacturing paradigms. [23,24] In those architectures, an agent is defined as a computer system embedded in an environment, which has the ability to perform autonomous actions to achieve predefined goals. [25] Online scheduling algorithms are included as agents connected to the shopfloor environment (e.g. [26]). Other potentially suitable architectures present modular control systems that incorporate approaches for online scheduling. [27] The presented control architectures typically focus on integrating online scheduling algorithms in the production phase and not on the application of the agents in the planning phase through connection to simulation environments. However, as explained above, simulations are indispensable e.g. for training ML algorithms. Therefore, a large body of literature integrating online scheduling algorithms into simulations exists (e.g. [28,29]), but neglects the integration of the proposed algorithms into existing control systems.

As a summary, existing architectures contain agent-based approaches that enable a modularity for the scheduling agents and they provide interfaces for the superordinate planning or control systems. But, the reviewed architectural approaches either focus on the application of online scheduling in connection with

simulation models that run independently from real production systems. Or they only focus on the application in multi-agent systems that are connected to the shopfloor during production. Therefore, the deficit can be concluded that online scheduling agents cannot be connected to simulation models during planning or ML training and not to control systems in a multi-agent system approach during production with the same interface in current architectures. In other words, in the analysed literature, no dedicated software component for independently managing online scheduling requests and decisions. The following research question can be derived from this deficit:

> *How can the seamless connection of various online scheduling agents with planning and control systems be enabled?*

To answer this research question the developments focus on newly overall new architecture, the required components and the necessary standardized interfaces. The well-established multi-agent system paradigms, including modular scheduling agents [20–22], already existing interface data models for transferring data in online scheduling scenarios [27–29] and principles of service-oriented architectures [30,31] are incorporated and used for the new architecture concept.

The following chapter 3 presents the modular software architecture to address the above-stated deficit on a conceptual level. Two representations are used for this. First, a software architecture oriented overview on the components and their interfaces is given and described in detail. Second, an incorporation of the components in the layers of the Internet of Production reference framework is presented to provide a different perspective from production technology. The subsequent chapter 4 presents an implementation and testing of the conceptual architecture. The overarching structure of chapters 3 and 4 is based on the methodology of software and model development with the phased of formalization (conceptual model), implementation (executable model) and experimentation (testing results). [32]

3. Conceptual Modular Software Architecture

Figure 1 presents an overview of the modular software architecture. The proposed architecture is divided into two phases and two layers. In the system layer, the simulation module and the control system represent the digital version of the assembly shopfloor. The simulation module is applied to train machine learning models for online scheduling. Furthermore, independently of online scheduling, the simulation can evaluate assembly system alternatives, which is applied mostly during planning, but also during production to optimize the system. The control system, or manufacturing execution system, is responsible in the production phase to monitor and execute assembly or auxiliary processes at work stations, buffers, or the transport system. In a flexible alternative flexible assembly system as explained in the motivation, a job route needs to be scheduled during production respectively during simulation run time. After a process step, the job might have a variety of next process-station combinations as potential alternatives.

The simulation or control system gathers the required information in a data model for decision-making. This data model needs to comprise all relevant static parameters describing the overall system configuration and dynamic variables representing the current system state. The parameters and variables need to fully represent the three system categories products (e.g. jobs allocated at processes and work stations), processes (e.g. progress, sequence, durations) and resources (e.g. transportation system, buffers, work stations states). The data model is standardized to allow interoperability. The ontology-based definition enables a standardized meta-model of digital twins as a connected data model. (cf. [33,34]) The interface methods are implemented continuously through the planning or production phase, i.e. in simulation and control systems. The methods are responsible for connecting to the scheduling layer. A scheduling request sends the data model of the assembly system.

The scheduling server, as a crucial part of the architecture, builds the interface between the system layer and the scheduling agents. It processes the scheduling request and forwards it with interfacing methods to one or multiple scheduling agents. Due to their independent decision-making and potential ability to learn in exchange with the environment, the term scheduling agent is chosen.

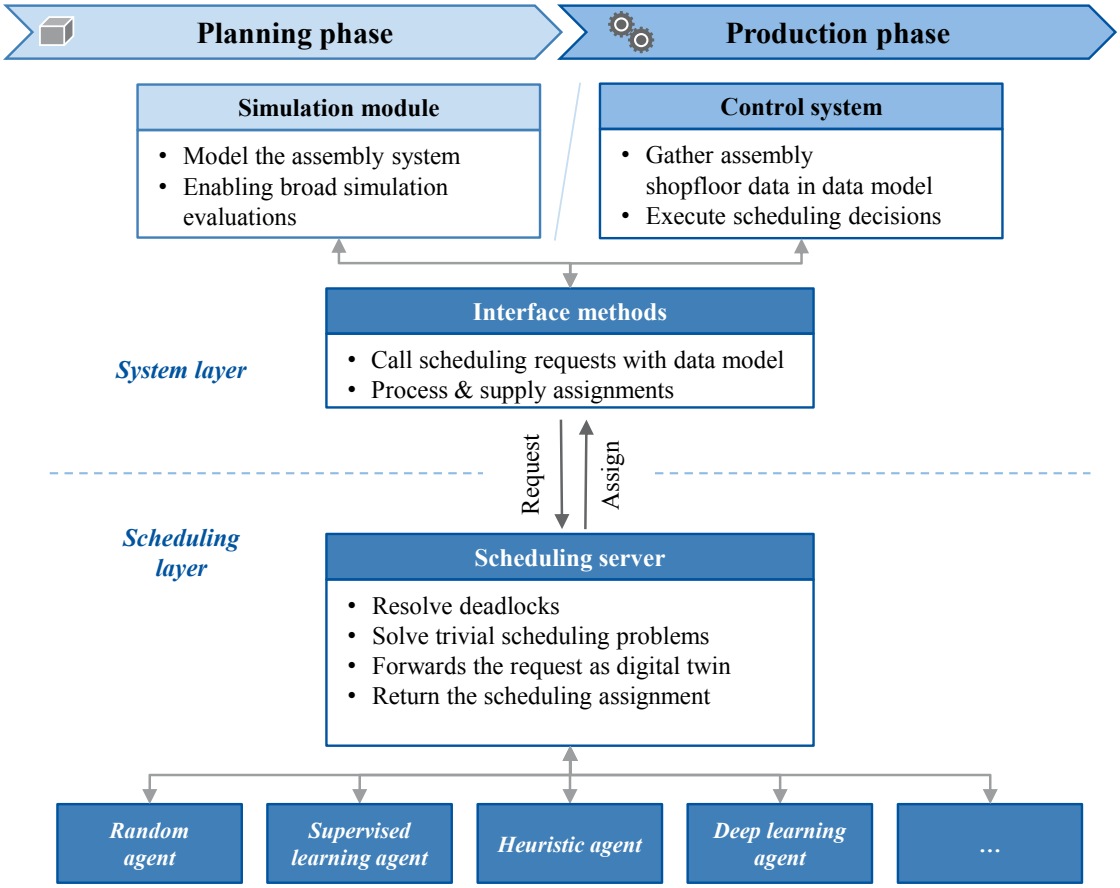

Figure 1: Overview of the modular software architecture for online scheduling.

In addition to communication functionalities, the scheduling server abstracts repetitive and simple tasks from the scheduling agents to reduce time-to-response. In case of trivial scheduling problems with only one station-process combination, the scheduling server can directly respond to the system layer with a scheduling assignment without addressing the scheduling agents. Another encapsulated task in the scheduling server is deadlock resolution. Deadlocks in online scheduling occur, for instance, when two jobs cross request the current station of the respective other job. In such cases, the scheduling server must force a scheduling decision or send a job to an intermediate buffer to free a station. As deadlock resolution is not a specific scheduling problem, but more a general production control issue, the abstraction avoids unnecessary communication and duplicate implementation in the scheduling agents.

The decision-making to provide a scheduling assignment occurs in the scheduling agents. As described in the motivation, various scheduling agents can be deployed: A random agent as a random performance baseline for algorithm comparison, a heuristic agent with rule-based logic, or agents applying machine learning techniques such as deep reinforcement, unsupervised or supervised learning. The type of machine learning technique is independent for the scheduling server as it communicates via the request and assignment messages. The scheduling agents can be seen as suppliers of scheduling-as-a-service in the modular architecture.

The scheduling decision is a process-station combination. The scheduling agent returns it to the scheduling server, which replies to the interface methods that supply the results to the simulation or control system. There, the results are used to execute simulated or real processes or actions. For instance, during production, the scheduling decision triggers a handling unit and a transport system to move the job to the next work station.

As an alternative representation, Figure 2 shows the modular software components from Figure 1 in the layers of the Internet of Production framework. [35] In comparison with Figure 1, this representation aims at the allocation of components in an existing reference framework. The underlying intention is to prepare a wider application of single components beyond the current online scheduling application. Examples would be a usage of the simulation model and the interface methods for the optimization of factory layouts or application in intralogistics as described in more detail in the last chapter 5.

In the following, the allocation and purposes of the software components in the different Internet of Production layers are described briefly. The system level provides the scheduling problem and the shopfloor data. The interface methods act as a middleware+ that is capable of connecting the different software components. The scheduling server and the data model reside at the integration layer to provide multi-modal access to the online scheduling agents. They are responsible for decision-making and autonomous actions to provide the scheduling assignments in the smart expert layer.

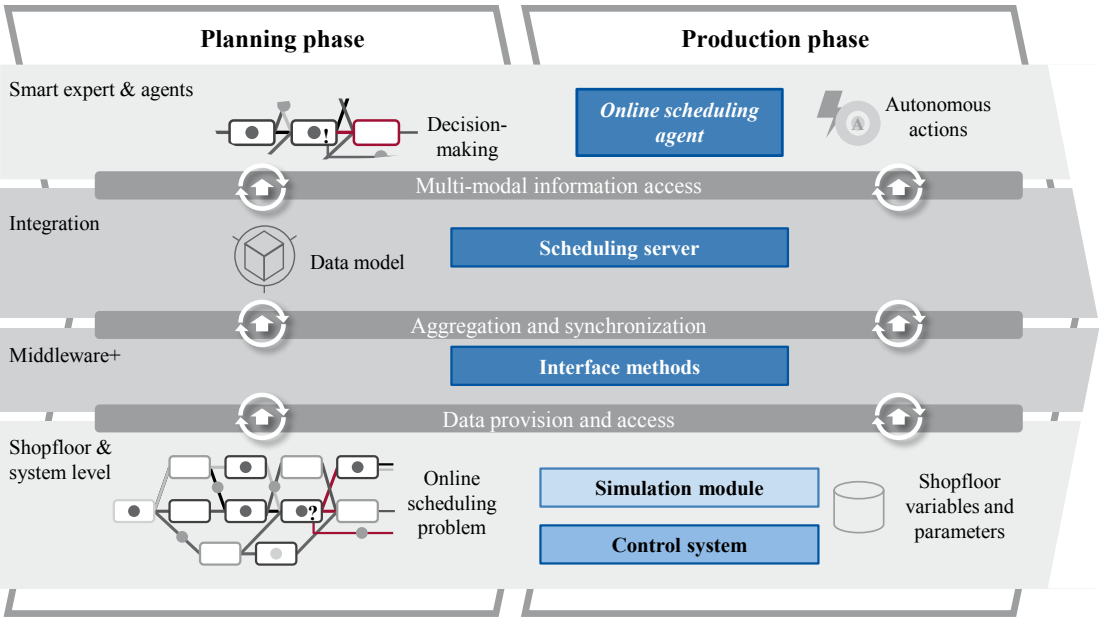

Figure 2: Incorporation of the modular software components in the reference framework Internet of Production for the production technology perspective (based on [35]).

4. Implementation and Functional Testing

The components and interface methods from the conceptual architecture presented in Figure 1 are implemented in infrastructure of the machine hall of the Laboratory for Machine Tools and Production Engineering (WZL) of RWTH Aachen University. The machine hall is equipped with several robots, manual work stations and automated guided vehicles. The software infrastructure consists of a Robot Operating System (ROS) middleware and the Message Queuing Telemetry Transport (MQTT) protocol for distributed communication. The control system COPE, developed by the Fraunhofer Institute for Production Technology (IPT) [27] is connected to the MQTT communication broker.

In Figure 3 in the top right corner, a section of the machine hall and the COPE user interface are visualized. The machine hall is represented in a discrete-event simulation model, created with Tecnomatix Plant Simulation, as shown on the top left.

Both, the simulation model and the COPE control system integrate the interface methods. The interface is set up with the Flask micro-framework, offering standardized RESTful Hypertext Transfer Protocol (HTTP) commands for transferring JavaScript Object Notation (JSON) files for requesting and assigning scheduling decisions with the HTTP methods POST and GET, respectively. The scheduling server was created with the high-level Python web framework Django, offering classes and methods for setup up the addressable web server. Within the scheduling server the functionalities of resolving deadlocks by enforcing sub-optimal scheduling decisions and solving trivial scheduling problems are also implemented in Python. In Figure 3 two excerpts of the content of request and assignment JSON files are shown. For the request, the file contains the name of the to-be-scheduled job as an identifier and the NextStation and NextProcess variables with the value -1, indicating a scheduling demand. Not listed in the excerpt is the current information about the assembly system with all current status of resources (robots, transport, manual work stations, etc.), other jobs and processes. In addition to the live date, static data that contains the overall system setup, such as locations of work stations, is included in the request. The assignment reply consists of the job name (COM-PRD-002) and the station-process combination (COM-ROB-004 and COM-PRC-015). The nomenclature of these strings follows a JSON scheme, i.e. three letters or words and predefined letters, e.g. for the robot station. This scheme is defined the meta model, to allow standardized data formatting (according to [33]).

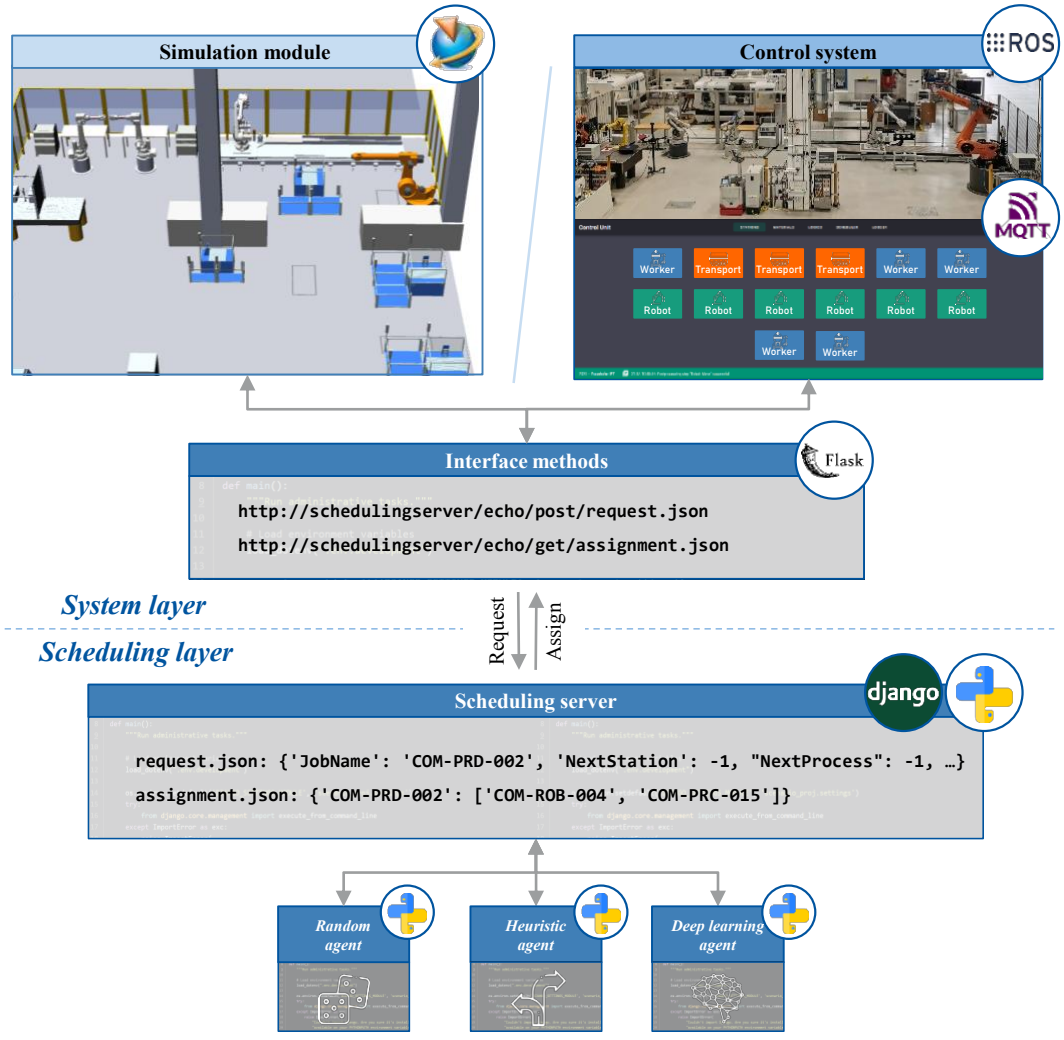

Figure 3: Overview of the implemented software components, following the modular software architecture.

The implemented scheduling agents can be chosen with a parameter in the scheduling request and the scheduling server is forwarding the request to the respective scheduling agent. The deep learning online scheduling agent is implemented according to the descriptions in Göppert et al. (2020, 2021). [36,37] The random agent chooses scheduling assignments randomly and the heuristic agent (also explained in [36]) minimizes the waiting and transportation time for the next station-process combination. The random and heuristic agents serve as reference agents to evaluate the scheduling performance of the deep learning agent. All scheduling agents were implemented in Python using open libraries (e.g., NumPy, scikit-learn, Keras).

The functionality of all components and interfaces was tested with qualitative validation measures, i.e. single error handling checks and large-scale simulation experiments. The qualitative validation results with corresponding inputs and functions are presented in Table 1.

Table 1: Qualitative validation of interfaces and scheduling server functionality.

Input description	Function	Result
Data model content not schema-compliant in request (e.g. 'COM-PRODUCT-…')	Scheduling server request consistency check	Scheduling server error message: "Data model incorrect"
Missing data model content in request (e.g. no NextProcess value given)	Scheduling server request consistency check	Scheduling server error message: "Data model incorrect"
Sending scheduling request with one possible station-process combination	Scheduling server functionality check	Assignment is consistent with the possible station-process combination, no deviation could be observed
Response message from scheduling agent incorrect	Scheduling server agent message check	Scheduling server error message: "Scheduling agent error"
Sending the same scheduling requests from a simulation model and control system	Interface function consistency check	Identical scheduling results for simulation and control system
Not existing algorithm name requested (e.g. "test_agent")	Scheduling server request consistency check	Scheduling server error message: "Algorithm incorrent"

In addition to the qualitative testing, a large number of test scenarios with the simulation model was executed as a large-scale validation. As shown in Göppert et al. (2020) 10,944 simulation scenarios, i.e. full simulation runs with 180 scheduling decision points per scenario were generated with the automated scenario analysis tool. [36] In total, the 1.97×10^6 scheduling requests were processed by the scheduling server and the scheduling agents. The scheduling assignments were successfully retrieved by the simulation leading to validated scenario outputs in the form or reasonable performance indicators.

5. Conclusion and Outlook

In this last chapter, a conclusion on the research question with the key findings and an outlook on potential applications and research opportunities is provided. The research question – *How can the seamless connection of various online scheduling agents with planning and control systems be enabled?* – can be answered as follows: The standardized interfaces, the dedicated scheduling server and the overall validated modular software architecture enable the planning and control with various online scheduling agents.

From the presented conceptual architecture and implementation, the key findings and additions to the knowledge base are:

1. The dedicated scheduling server software module enables the encapsulation of scheduling decisions and externalization from simulation or control systems to, for instance, enable independent scheduling agent development.
2. Advantages of the scheduling server are the central deadlock resolution to avoid repeated implementation in the scheduling agents for trivial scheduling task solution and to reduce unnecessary communication.
3. Web-based interfaces allow for the system-agnostic (simulation or control) handling of scheduling requests and assignments and, therefore, enable switching seamlessly from simulating during planning to the control of a running system.

4. Applying data modelling standards such as schemes and meta-models facilitates the interoperable communication via data models and the modularization of components.
5. The mentioned open-source libraries for web-server development give researches and practitioners from production technology potential support in creating a comparable modular scheduling architecture.

Besides these benefits and learnings, also drawbacks of the presented architecture exist. A downside of the modularity are the additional communications efforts between the system layer, the scheduling server, and the scheduling agents. The time for the additional communication results in an overhead waiting time for each scheduling request. For simple scheduling requests, the communication overhead might reduce the performance, especially for discrete-event simulations that seek to simulate large numbers of scenarios quickly. Extra external communication efforts for each scheduling decision, can significantly increase the total simulation time. Also, the development and implementation costs of the dedicated scheduling server and the interface methods have to be considered. Proprietary software for simulation or control might also not allow the integration of the interface methods.

Further potential applications of the proposed software architecture and its components can be found outside assembly systems, wherever online scheduling is applied. The data model for resembling the real system must be adapted to the application, but the functionality of the components is independent of that. A field with high potential is intralogistics, due to the short transport times and ad-hoc request of transportation, which cannot be scheduled ahead in large and complex systems. Further exemplary fields outside the manufacturing domain are airplane terminal allocation, scheduling of patients to hospital resources and assigning data computing tasks to processing units.

Further research opportunities, besides the broader application in the above-mentioned fields, comprise the measurement of these external communication times depending on the various communication protocols or web server frameworks. With these results, the impact of external communication in contrast to the benefits of modular scheduling agents can be investigated. Also, the communication protocol that supplies the fastest response times can be chosen. In general, the long-term application of the modular scheduling architecture in real production with a control system could lead to more insights about potential interface improvements and could validate the postulated benefits. Furthermore, the long-term application would increase the architecture's maturity. Eventually, standardization committees could develop a specific reference architecture that follows domain-wide definitions of interfaces and data models.

Acknowledgements

We would like to thank the BMWi, DLR-PT and our partners for their kind support. This work is part of the research project "AIMFREE", which is funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) within the "Directive on a joint funding initiative to support research and development in the field of electro mobility" (funding code: 01MV19002A) and supported by the project management agency German Aerospace Center (DLR- PT). The authors are responsible for the content. More information about the current research activities within the project AIMFREE at www.aimfree.wzl.rwth-aachen.de.

References

[1] Lanza, G., Nyhuis, P., Fisel, J., Jacob, A., Nielsen, L., Schmidt, M., Stricker, N., 2018. Wandlungsfähige, menschzentrierte Strukturen in Fabriken und Netzwerken der Industrie 4.0. acatech Studie) München," Herbert Utz Verlage.

[2] Rauch, E., 2013. Konzept eines wandlungsfähigen und modularen Produktionssystems für Franchising-Modelle. Fraunhofer-Verlag, Stuttgart.

[3] Belhadi A., Kamble, S. Jabbour, C., Gunasekaran, A., Ndubisi, N.O., Venkatesh, M., 2021. Manufacturing and service supply chain resilience to the COVID-19 outbreak: Lessons learned from the automobile and airline industries. Technological Forecasting and Social Change 163, 120447.

[4] Hees, A., 2016. System zur Produktionsplanung von rekonfugrierbaren Produktionssystemen. Dissertation, München.

[5] Nyström, M., Jouffray, J.-B., Norström, A.V., Crona, B., Jørgensen, P.S., Carpenter, SR, Bodin, Ö., Galaz, V., Folke, C., 2019. Anatomy and resilience of the global production ecosystem. nature 575 (7781), 98–108.

[6] Greschke, P.I., 2015. Matrix-Produktion als Konzept einer taktunabhängigen Fließfertigung. Dissertation, Braunschweig.

[7] Schönemann, M., Herrmann, C., Greschke, P., Thiede, S., 2015. Simulation of matrix-structured manufacturing systems. Journal of Manufacturing Systems 37, 104–112.

[8] Kern, W., Rusitschka, F., Bauernhansl, T., 2016. Planning of Workstations in a Modular Automotive Assembly System. Procedia CIRP 57, 327–332.

[9] Ramsauer, C., Rabitsch, C., 2015. Agile Produktion - Ein Produktionskonzept für gesteigerten Unternehmenserfolg in volatilen Zeiten, in: , Industrial Engineering und Management. Springer Fachmedien Wiesbaden, pp. 63–81.

[10] Hüttemann, G., Göppert, A., Lettmann, P., Schmitt, R.H., 2017. Dynamically Interconnected Assembly Systems, in: Schmitt, R., Schuh, G. (Eds.), 7. WGP-Jahreskongress Aachen, 5.-6. Oktober 2017. Apprimus Wissenschaftsverlag, Aachen, pp. 261–268.

[11] Heilala, J., 2022. Modeling and Simulation for Decision Making in Sustainable and Resilient Assembly System Selection. Scandinavian Simulation Society, 180–188.

[12] Wu, X., Zhao, J., Tong, Y., 2018. Big Data analysis and scheduling optimization system oriented assembly process for complex equipment. IEEE Access 6, 36479–36486.

[13] Pinedo, M.L., 2014. Scheduling: Theory, Algorithms, and Systems, 5th ed. Springer.

[14] Usuga Cadavid, J.P., Lamouri, S., Grabot, B., Pellerin, R., Fortin, A., 2020. Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. Journal of Intelligent Manufacturing 31 (6), 1531–1558.

[15] Morariu, C., Morariu, O., Răileanu, S., Borangiu, T., 2020. Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems. Computers in Industry 120, 103244.

[16] Chaudhry, I.A., Khan, A.A., 2016. A research survey: review of flexible job shop scheduling techniques. International Transactions in Operational Research 23 (3), 551–591.

[17] Fang, Y., Peng, C., Lou, P., Zhou, Z., Hu, J., Yan, J., 2019. Digital-twin-based job shop scheduling toward smart manufacturing. IEEE Transactions on Industrial Informatics 15 (12), 6425–6435.

[18] Minguillon, F.E., Stricker, N., 2020. Robust predictive-reactive scheduling and its effect on machine disturbance mitigation. CIRP Annals 69 (1), 401–404.

[19] Rinciog, A., Mieth, C., Scheikl, P.M., Meyer, A., 2020. Sheet-Metal Production Scheduling Using AlphaGo Zero. Conference On Production Systems and Logistics, 1–10.

[20] Dorri, A., Kanhere, S.S., Jurdak, R., 2018. Multi-Agent Systems: A Survey. IEEE Access 6, 28573–28593.

[21] Leusin, M.E., Kück, M., Frazzon, E.M., Maldonado, M.U., Freitag, M., 2018. Potential of a Multi-Agent System Approach for Production Control in Smart Factories. IFAC-PapersOnLine 51 (11), 1459–1464.

[22] Park, B., Jeong, J., 2020. A CPS-Based IIoT Architecture Using Level Diagnostics Model for Smart Factory, in: Computational Science and Its Applications - ICCSA 2020. Springer International Publishing, Cham, pp. 577–587.

[23] Helo, P., Phuong, D., Hao, Y., 2019. Cloud manufacturing – Scheduling as a service for sheet metal manufacturing. Computers & Operations Research 110, 208–219.

[24] Oluyisola, O.E., Bhalla, S., Sgarbossa, F., Strandhagen, J.O., 2022. Designing and developing smart production planning and control systems in the industry 4.0 era: a methodology and case study. J Intell Manuf 33 (1), 311–332.

[25] Jennings, N.R., 2000. On agent-based software engineering. Artificial Intelligence 117 (2), 277–296.

[26] Zhou, T., Tang, D., Zhu, H., Zhang, Z., 2021. Multi-agent reinforcement learning for online scheduling in smart factories. Robotics and Computer-Integrated Manufacturing 72, 102202.

[27] Jung, S., Ochs, J., Kulik, M., König, N., Schmitt, R.H., 2018. Highly modular and generic control software for adaptive cell processing on automated production platforms. Procedia CIRP 72, 1245–1250.

[28] Krockert, M., Matthes, M., Munkelt, T., 2021 - 2021. Agent-based Decentral Production Planning and Control: A New Approach for Multi-resource Scheduling, in: Proceedings of the 23rd International Conference on Enterprise Information Systems. 23rd International Conference on Enterprise Information Systems, Online Streaming, --- Select a Country ---. 26.04.2021 - 28.04.2021. SCITEPRESS - Science and Technology Publications, pp. 442–451.

[29] Vieira, M., Moniz, S., Gonçalves, B.S., Pinto-Varela, T., Barbosa-Póvoa, A.P., Neto, P., 2021. A two-level optimisation-simulation method for production planning and scheduling: the industrial case of a human–robot collaborative assembly line. International Journal of Production Research, 1–21.

[30] Aggarwal, S., 2019. Flask Framework Cookbook: Over 80 proven recipes and techniques for Python web development with Flask. Packt Publishing Ltd.

[31] Cerny, T., Donahoo, M.J., Trnka, M., 2018. Contextual understanding of microservice architecture: current and future directions. ACM SIGAPP Applied Computing Review 17 (4), 29–45.

[32] Rabe, M., Spieckermann, S., Wenzel, S., 2008. Verifikation und Validierung für die Simulation in Produktion und Logistik: Vorgehensmodelle und Techniken. Springer Science & Business Media.

[33] Göppert, A., Grahn, L., Rachner, J., Grunert, D., Hort, S., Schmitt, R.H., 2021. Pipeline for ontology-based modeling and automated deployment of digital twins for planning and control of manufacturing systems. Journal of Intelligent Manufacturing.

[34] Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W., 2018. Digital Twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine 51 (11), 1016–1022.

[35] Brecher, C., Klocke, F. Schmitt, R.H., Schuh, G. (Ed.), 2017. Internet of Production für agile Unternehmen: AWK Aachener Werkzeugmaschinen-Kolloquium. Apprimus Verlag.

[36] Göppert, A., Mohring, L., Schmitt, R.H., 2021. Predicting performance indicators with ANNs for AI-based online scheduling in dynamically interconnected assembly systems. Production Engineering (Journal) 15 (5), 619–633.

[37] Göppert, A., Rachner, J., Schmitt, R.H., 2020. Automated scenario analysis of reinforcement learning controlled line-less assembly systems. Procedia CIRP 93, 1091–1096.

Biographies

Amon Göppert (*1992) is head of the department of Model-based Systems at Laboratory of Machine Tools and Production Engineering of RWTH Aachen University (WZL) since 2021. From 2017 to 2021 he was a research associate at WZL.

Jonas Rachner (*1995) is a research associate at the department of Model-based Systems at Laboratory of Machine Tools and Production Engineering WZL of RWTH Aachen University since 2019.

Lea Kaven (born Grahn) (*1995) is research associate at the department of Model-based Systems at Laboratory of Machine Tools and Production Engineering WZL of RWTH Aachen University since 2020.

Prof. Dr.-Ing. Robert H. Schmitt (*1961) is head of the Chair of Production Metrology and Quality Management at Laboratory for Machine Tools and Production Engineering WZL of RWTH Aachen University and board member of the Fraunhofer Institute for Production Technology (IPT).