Towards a Non-disruptive System for Dynamic Orchestration of the Shop Floor

Milan Pisarić, Vladimir Dimitrieski, Marko Vještica, Goran Krajoski

Towards a Non-Disruptive System for Dynamic Orchestration of the Shop Floor

Milan Pisarić[1][0000-0002-0764-4453], Vladimir Dimitrieski[2][0000-0003-3234-6543], Marko Vještica[2][0000-0003-2368-5818] and Goran Krajoski[1][0000-0002-2619-5796]

[1] Industrial Automation, KEBA AG, Linz, Austria
{pisa, krgo}@keba.com
[2] University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia
{dimitrieski, marko.vjestica}@uns.ac.rs

Abstract. One of the main challenges of Industry 4.0 is the adaptation of existing production lines. Robots are substituting human workers in modern smart factories, as they are much more suitable for repetitive tasks. In contrast to that, Industry 4.0 predicts a high rise in product customization. The total disruption of the current factories, although the easiest solution, is not welcomed by the traditional industry stakeholders. To offer adaptation rather than disruption, and to promote man-machine collaboration rather than complete substitution of the human workforce, we present a Digital Factory solution capable of orchestrating different types of resources —humans, machines, robots —according to their capabilities. The core component of the solution is a real-time Orchestrator that orchestrates factory resources in order to produce the desired product. Orchestrator is a complex, modular, highly scalable, and pluggable software, responsible for dynamical matching, scheduling, and executing of production steps, allowing high customization and lot-size-one production.

Keywords: Orchestration, Digital Factory, Smart Manufacturing, Flexible manufacturing systems, Production Planning, Industry 4.0

1 Introduction

The industrial world is evolving. Although some researchers suggest a revolution or a total disruption of the current industrial automation setting, existing industry stakeholders are rather hoping for stabile transition and adaptation to the wave of Industry 4.0 proposals. Highly frequent repetitive tasks are a common playground for robots in production, but a trend in Industry 4.0 is extensive product customization rather than a huge number in lot-size of products [1]. The substitution of the human workforce with robots is not always motivated by the cost or the efficiency of the production itself, but rather a social trend —as new generations seem less willing to work as low-skilled operators [2]. Industry stakeholders also face the problem of preservation of current knowledge and skills of experienced workers who have nobody to transfer it to [3]. In contrast to that, in the current, rigid production environment, robots are still not fit to do some of the high complexity tasks that mid- and high-skilled human workers do, nor they are fit for high customization of the products because of the lengthy time required for conversion to other product. Robots are not going to completely replace human workers soon, but a proportion of collaboration between a man and a machine will increase in the near future [2].

Our approach, in opposition to a total disruption, is to enable the integration of the existing participants in the production into a flexible manufacturing system of the next generation. Legacy equipment and information systems will be adapted by using a specific hardware-software component in order to create a corresponding proxy, able to coexist, and cooperate with the proposed system. In this paper, we present a Digital Factory solution with an intelligent, real-time Orchestrator as its core component. The Orchestrator is a complex system that orchestrates a safe collaboration of all factory resources —humans, machines, robots — with the goal of producing a requested product. Formally described production processes are fed to the Orchestrator which in turn is responsible for the orchestration —matching, scheduling, and enriching the process —and for the execution on final production steps in a smart factory Digital Twin (DT). Execution commands are propagated from the DT to its corresponding smart device where they are realized at a shop floor level while allowing for lot-size-one production.

2 Related work

Digitalization of the complete production process is a crucial step in reaching any of the Industry 4.0 goals —including automated shop floor reconfiguration and service or task-oriented programming of machines. Therefore, not only products but also processes and resources need to have digital descriptions [4]. Based on these descriptions, modern industrial software systems orchestrate delegation of instructions and their assignment to smart resources [5]. Based on the desired product and by matching required resources with available smart resources and their offered capabilities, these systems can act as self-organizing systems and execute the complete production process [6, 7]. This matching is enabled by the introduction of semantic knowledge of the manufacturing environment, which formalizes the rules and the relationships between the participating objects [8].

Production flexibility is achieved not only on the execution level but also in the production process modeling phase. Domain-specific modeling language can be designed for this purpose —not only to enable process customization but also to enable visualization of the process monitoring and graphical simulation of the production process model [9]. Even though modern, flexible manufacturing relies on computer systems, many production lines still lack the possibility of simulating the production process itself. The development of simulators, as generic as possible, that can mimic any give shop floor of the production process is still a promising approach within the industry [10]. Simulation is to act as a digital twin of the production system, that is able to simulate individual process steps or complete production. Production process simulation has been addressed in several solutions with a focus on assembling —to validate the result of assembly planning, to detect the errors of the product design, and to make appropriate modifications to the generated execution plan [5, 7]. Simulation in our solution is practically a digital twin of the product being produced. It can be used as a visualization of the executed production while collecting adequate data for in- or post-production analysis. Otherwise, it can be alienated from the production process as a pure simulation. By simulating the complete process, potential production failures are reduced, resource consumption is optimized and the safety of participating human workers is enhanced. Various forms of cooperation between machine and a human have already been addressed in the area of modern production systems [11]. Vernim et al. have already proposed the integration of human workers as a special type of a factory resource in a capability-based production planning [12]. In addition to this work, we address the orchestration of a human worker, by sending generic instructions for the final execution. A human operator is basically used as Human as a service, with a focus still being on collaboration with machines in this adaptation of existing shop floors.

3 Architecture

In the presented solution (Fig. 1), **Orchestrator** denotes software that can be used to orchestrate factory resources with a goal of producing a requested product. On one hand, it can be run as a self-contained black box (e.g. on an industrial PC). On the other hand, several core elements and external parts of the Orchestrator are pluggable and offer the possibility of adapting the solution to the factory or use-case specific requirements e.g. Dashboard or Analytics. Pluggable elements are denoted with a plus symbol in Fig. 1. The red line depicts the orchestration flow starting from the user interface and ending with the execution of production. The overall architecture is modular and is implemented as a basic **Container and Agent infrastructure**. With the actors as the software components packaged within the containers, it is possible to realize smart, modular, and pluggable software infrastructure. An actor is an individual entity with some intelligence attached to the software components. Software containers also enable high scalability and running on cross-platform with minimum time to setup. This is achieved by introducing de-coupled containers that start, stop, and run independently and can be seen as complete and isolated services. These containers can be run on-demand, based on a multitude of factors such are the number of requests or the load on the existing container instances.

Customer interface (1) offers flexible and intelligent ways to interact with the production facility. In a generic pluggable fashion, customers can express individual products in various formats which are later parsed or transformed into a generic product model or digital product description (such as Computer-Aided Design —CAD, or Computer-Aided Manufacturing —CAM diagrams). A user-defined product description is an input for the orchestrator. **Orchestration Agent** (2) is an essential component that operates the complete orchestration process. It is a state machine that runs each user's product order through various states —matching to execution —in order to create the desired product. **Process Reasoner** (3) uses the digital product description as an input, to reason upon required capabilities, steps, and order of the sequences and generates the digital process description as an output. The process description is a technological description without any resource allocation in it. It also holds the information about the bill of materials, quantities, timing constraints, acceptance, and completion criteria.

Process Reasoner is also implemented in a pluggable fashion to answer to use-case specific requirements. Once the production process specification is inferred, it can be adapted by using the provided process modeler. This tool allows not only altering the existing production processes, to fine-tune them for production, but also creating them from scratch and thus enriching or substituting the customer interface. This creation of process specifications is of utmost importance as some product specifications are not provided in a machine-readable form.

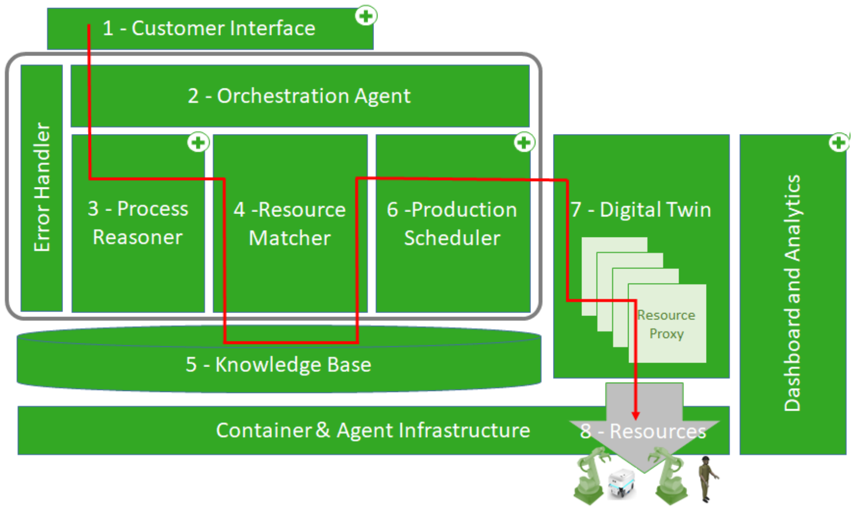

Fig. 1. – Orchestrator Architecture

Resource Matcher (4) takes up the responsibility to match all the required capabilities, constraints, acceptance and completion criteria on the input, with the available resources, based on their offered capabilities stored in **Knowledge Base** (5), to the processes and the required capabilities. The result of the matching process is a set of possible production processes with allocated **Resources** (8). This is represented as a set of directed graphs where each graph represents a single process —practically a variety of production process description. The next step is the enrichment of generated processes. It is done gradually in a sequence of customizable enrichers. The result is production process variations improved in resource level enrichment —which includes the preparation steps for a product order, and in logistic enrichment —which describes the material flow during the production process. The result of this phase is another set of directed graphs, where each graph represents an enriched variation of a process description.

Knowledge Base (KB) is a storage facility for the semantic data models of resources, capabilities, constraints, factory logistics, collaboration models, and interface contracts. It also stores semantics of products and processes and plays a key role in the orchestration of production. Any new version of the process, even when a previous version exists, is stored in the KB. KB offers production knowledge in a machine-readable form and is based on well-defined and formal meta-models. It also holds information on physical and logical connections between devices and required physical connections in the system. A topology inside KB contains information on which device can interact with which other devices to create an appropriate flow of operations based on the inputted processes. It also includes all the spare resources on the shop floor (e.g. machines that are currently being maintained). The Discovery mechanism enables additional resources to enter the shop floor dynamically when they are inserted in the KB in a plug-and-produce fashion. KB also supports inference and query mechanism that can dynamically find connections between all resources. This enables Orchestrator to search for semantic connections and orchestrate the production.

Production Scheduler (6) receives a graph of all possible process descriptions with resources and offered capabilities, process steps, and material allocations. It then performs a scheduling algorithm with the variants of cost functions. It is done in a multilayered sequence of matching-scheduling graders that lean on information stored in KB. Different optimization functions act like graders for the scheduler and enable an optimization via a specific combination of criteria, such as reduced times and energy consumption or optimized material consumption and preferred technological steps used. According to the pluggable architecture, these criteria are organized in factory-specific graders and every graph is then ranked according to the desired set of graders. The result of this phase is an enriched and scheduled list of production process steps ready to be executed.

Digital Twin (DT) (7) is the component of the Digital Factory that takes an executor role and oversees the final execution of commands. DT is a faithful representation of the factory shop floor where each digital element represents an actual resource that it can communicate with. This enables the usage of DT representation as a Simulation only, as well as a simultaneous execution and visual representation of the production process. For each process step, DT sends the appropriate command to a targeted element, which then propagates the execution command to its resource via a resource Proxy. Proxies generate machine-specific commands and are the functional and behavioral interface between DT and the physical assets of the Digital Factory. The hardware platform is still open, and although the structure of execution commands may be common among all the types of resources, the proxies are factory specific in the end. A **resource** is an asset involved in the production process. In the proposed architecture, humans, robots, and production machines are considered as resources that are differentiated by their capabilities, interaction, and interface description. A **human worker** is modeled with its capabilities (i.e. inspection, pick-n-place, turning…) that are not necessarily different from robot capabilities (i.e. pick-n-place, milling, drilling…) but are differentiated by their constraints. Robots can operate with heavier loads or with less fatigue, while human workers will still be of greater trust in some type of jobs (e.g. inspection) for some time in the future. Also, there are still some competences and man-machine interactions that are hardly going to be replaced by a machine until a production completely based on Artificial Intelligence (AI). Although the executor sends generic commands to any resource, a human worker interacts via a human-machine interface (HMI) while machines communicate via a standardized interface. Interaction with non-standard resources, especially legacy non-smart machines, is to be enriched by their appropriate and use-case specific interfaces.

All system components communicate with each other through an internal communication layer based on a highly-scalable communication backbone (in Zero Messaging

Queue). An intra-module communication is established through a broker implementation. Every component provides logging and tracing data to the logging-bus module. It is a rudiment for the rule-based **Error handling** which is triggered according to the information provided in logs and provides event-based error mechanisms for all the components including matching and scheduling. Additional pluggable components, that also lean heavily on the communication layer are **Dashboard and Analytics** modules of various kinds. The proposed infrastructure and architecture are foundations for high-scalability and a large flow of information. System components are modular, independent, and scalable, which enables both horizontal and vertical scalability. The current proposal is also technology agnostic with the possibility to run on a single PC, distributed on several industrial PCs or in a cloud.

4 Use case (proof of concept)

Several use-cases were used while analyzing and testing the architecture flow presented on Fig. 1. One assembly use-case was used to set up a laboratory conditions testbed, alongside a corresponding simulation. Test arena comprises of several elements — smart shelves and material area, smart assembly tables, an autonomously guided vehicle (AGV) with an industrial robot mounted on it, 3D printer, LEGO bricks of various sizes and colors and additional industrial robot fixed on the assembly table. Simulation encompasses real-world device simulation models and a graphical preview of the simulated system and the Digital Twin (7). The simulation is created using ROS environment because of the ease of usage and understanding to the non-robotics community, in a way to resemble the desired test arena as much as possible. The human operator is also a part of the testbed, depending on the use-case variation.

There are two mobile applications available for human participants —one is used as a communicator to the human operator, and the second is a starting point of interaction with the Digital Factory. This interface (1) allows the upload of a digital description of the product. In the particular case, there is a GUI set up on a tablet device, that enables a user to design a custom flag. The desired flag is to be assembled out of LEGO bricks, using the resources existing in the Knowledge Base (5) and without any additional coding. All the participants of the testbed - smart areas, inventory, bricks, and assembly table —are regarded as resources (8). After the user designs a flag, the Orchestration Agent (2) initiates the orchestration mechanisms and the flag is then being processed. The Process Reasoner (3) generates the production process specification from the inputted digital description of the product, and stores it in the KB (5). In this use-case, the raster, shape, dimensions, and number of bricks to be used are extracted from the input, and a digital specification of the production process is generated as an output of this element. All the production steps are deduced according to the factory-specific topology and use-case. It is then possible to make changes or additions to the generated process in the dedicated Process Tool, with the optimization being the main goal. Additionally, the execution process can be reviewed in the tool, thus acting as a secondary DT. The digital description is a recipe that is used for further orchestration of all the resources towards the product being delivered. The Resource Matcher (4) uses the semantic information stored in KB to match the existing resources (8), their capabilities and constraints, and offers an optimal match for every production step described in the generated production process. In this use-case, the available resources capable of assembling are limited to two robots and a human. Matcher first produces a production process variation in which it assigns these resources to their dedicated production steps (e.g. AGV is to pick a targeted brick, the robot is to assemble a targeted brick on the targeted table; human is to prepare a targeted brick, etc.). This production variation is later enriched in several layers of enrichment. This re-matching or deeper matching refinement is done according to the availability of the resources in the testbed, current topology of the machines, physical limitations of the collaboration between resources, etc. In this use-case, there is a safety limitation of human being too close to the industrial robots, and this information is therefore additionally calculated in the matching process.

The Production Scheduler (6) is scheduling the execution order of all the matched production steps to find an optimal schedule. Time was the only optimization grader used in this use-case, intending to assemble the LEGO flag as quickly as possible. This led to bricks being transferred from a material area to the assembly table in higher volume and not brick by brick. After the matching and scheduling phases are done, the participating resources receive the commands via dedicated Resource Proxies. These proxies are elements of Digital Twin (7) which is practically acting as an executor. Generic commands (e.g. Pick blue brick of size 2x2 from location A) received as a result of previous phases are transformed in a set of factory-specific instructions and are then propagated to the designated resources. In the described use-case, commands are sent via ROS bridge for robots, and a tablet or smartwatch for human workers, who receive instructions and send feedback when their activity is finished. The human worker provides additional bricks that are out of stock or delivers a brick created additionally in the 3D printer. If a brick is to be created or to be delivered by a human worker, the AGV reaches the targeted material area where the human operator and the 3D Printer are, and where the collaboration is safe. In the end, the fixed robot assembles the flag vertically by placing LEGO bricks one by one on the assembly table.

5 Conclusions and future work

We have presented the concept of a Digital Factory as a pluggable software system, that automatically extracts instructions from the given input, sets up the factory shop floor accordingly, and executes the production itself. Both the simulation and the real-time testbed set in a laboratory are used as a proof of concept. Not only are the machines orchestrated but a human worker in collaboration with robots as well.

We are continuing the work on the presented solution. Several areas of investigation are still open, including the introduction of AI-based scheduling graders that determine main properties of delivering the optimized execution graphs; decentralization of the orchestrator process to the participating assets; automatic extraction of the production process steps out of the existing CAD/CAM diagrams or recipes; deducing the missing information on process descriptions based on AI previous knowledge, etc.

Acknowledgment

The research in this paper is supported by KEBA AG Linz.

References

1. Lu, Y.: Industry 4.0: A survey on technologies, applications and open research issues. J. Ind. Inf. Integr. 6, 1–10 (2017). https://doi.org/10.1016/j.jii.2017.04.005.
2. The Risk of Automation for Jobs in OECD Countries: A Comparative Analysis. (2016). https://doi.org/10.1787/5jlz9h56dvq7-en.
3. Müller, J.M., Voigt, K.-I.: Sustainable Industrial Value Creation in SMEs: A Comparison between Industry 4.0 and Made in China 2025. Int. J. Precis. Eng. Manuf.-Green Technol. 5, 659–670 (2018). https://doi.org/10.1007/s40684-018-0056-z.
4. Backhaus, J., Reinhart, G.: Digital description of products, processes and resources for task-oriented programming of assembly systems. J. Intell. Manuf. 28, 1787–1800 (2017). https://doi.org/10.1007/s10845-015-1063-3.
5. Keddis, N.: Capability-Based System-Aware Planning and Scheduling of Workflows for Adaptable Manufacturing Systems, (2016).
6. Michniewicz, J., Reinhart, G.: Cyber-Physical-Robotics – Modelling of modular robot cells for automated planning and execution of assembly tasks. Mechatronics. 34, 170–180 (2016). https://doi.org/10.1016/j.mechatronics.2015.04.012.
7. Zhang, Y., Qian, C., Lv, J., Liu, Y.: Agent and Cyber-Physical System Based Self-Organizing and Self-Adaptive Intelligent Shopfloor. IEEE Trans. Ind. Inform. 13, 737–747 (2017). https://doi.org/10.1109/TII.2016.2618892.
8. Alsafi, Y., Vyatkin, V.: Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. Robot. Comput.-Integr. Manuf. 26, 381–391 (2010). https://doi.org/10.1016/j.rcim.2009.12.001.
9. Vještica, M., Dimitrieski, V., Pisarić, M., Kordić, S., Ristić, S., Luković, I.: Towards a formal description and automatic execution of production processes. In: Proceedings of 15th IEEE International Scientific Conference on Informatics (Informatics'2019). pp. 450–455. IEEE, Poprad, Slovakia (2019).
10. Boschert, S., Rosen, R.: Digital Twin—The Simulation Aspect. In: Hehenberger, P. and Bradley, D. (eds.) Mechatronic Futures. pp. 59–74. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-32156-1_5.
11. Romero, D., Stahre, J., Wuest, T., Noran, O., Bernus, P., Fast-Berglund, Å., Gorecky, D.: Towards an Operator 4.0 Typology: A Human-Centric Perspective on the Fourth Industrial Revolution Technologies. In: Proceedings of International Conference on Computers & Industrial Engineering pp. 1–11. Tianjin, China (2016).
12. Vernim, S., Walzel, H., Knoll, A., Reinhart, G.: Towards capability-based worker modelling in a smart factory. In: 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). pp. 1576–1580. IEEE, Singapore (2017). https://doi.org/10.1109/IEEM.2017.8290158.