Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

The Digital Twin as a Core Component for Industry 4.0 Smart Production Planning

Petr Novák * **Jiří Vyskočil** * **Bernhard Wally** **

* *Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Prague, Czech Republic. (e-mail: {firstname.lastname}@cvut.cz)*
** *Christian Doppler Laboratory Model-Integrated Smart Production, Institute of Business Informatics – Software Engineering, JKU Linz, Linz, Austria. (e-mail: bernhard.wally@jku.at)*

Abstract: Production systems that adhere the Industry 4.0 vision require new ways of control and integration of individual components, such as robots, transportation system shuttles or mobile platforms. This paper proposes a new production system control concept based on closing a feedback loop between a production planning system and a digital twin of the physical production system. The digital twin keeps up-to-date information about the current state of the physical production system and it is combined with the production planner utilizing artificial intelligence methods. Production recipes and concrete process instantiations are planned for each production order on-the-fly, based on the production system state retrieved form the digital twin. This approach provides a high flexibility in terms of ability to add and to remove products as well as production resources. It also enables error recovery by re-planning the production if some failure happens. The proposed approach is tested and evaluated on an internally hosted Industry 4.0 testbed, which confirms its efficiency and flexibility.

Keywords: Production Planning and Control, Intelligent Manufacturing Systems, Flexible and Reconfigurable Manufacturing Systems, Digital Twin

1. INTRODUCTION

Industry 4.0 is an abstract, yet focused, vision preparing for a high degree of flexibility in various aspects, including (mass-)customized products, smart production processes, virtualization, and data transparency. Constructing such a manufacturing system is a complex effort comprising a plethora of tasks involving a great variety of engineering disciplines—it is realized through many iterations during the engineering phase of the system. While it is hard to objectively assess the importance of each single discipline, it can be argued that a key role is played by the engineering of the automation and control system that has to support the flexibility of the production system during runtime.

Typical manufacturing lines, which are geared towards mass production nowadays, have fixed production recipes designed. They are tested and fully maintained by human engineers. With the shift towards Industry 4.0, it is required that emerging manufacturing lines are able to deal with more complex products and recipes, and they usually contain more components being able to cooperate with one another. Such new manufacturing lines are ready for producing a wide variety of goods according to dynamically changing customer demands and production goals.

This paper introduces a compact concept of how digital twins can be perceived and utilized for controlling and planning of flexible, goal-oriented, and reactive Industry 4.0 ready production systems. The proposed usage of such a digital twin reveals the following benefits:

- On-the-fly production planning
- Automated error recovery
- Readiness for formal verification on the level of production plans
- Faster and more energy-efficient by finding optimal recipes and eliminating initial setups

The remainder of the paper is structured as follows. The related work is summarized in Sec. 2. The main contribution of this paper, the smart production planning with a digital twin, is discussed in Sec. 3. The proposed approach is shown on a practical use-case described in Sec. 4. Finally, Sec. 5 concludes and specifies future work.

2. RELATED WORK

2.1 Digital Twins

A digital twin is a common term used for a digital replica of a physical system. We are sticking to this term in this paper, even though we believe that it is highly misleading in many cases: in a real life, a twin is a 1:1 copy of the other twin, but only at time t_0. From then on, the two twins are autonomous entities that might stay closely connected, but that might diverge from each other in a way that an external viewer would not be able to call them "twins" anymore. In general, we believe that a term such as "digital shadow" would better reflect the characteristics of a digital twin (Dong et al. (2009)) in the Industry 4.0 context. Nevertheless, the first concept of a digital

twin was proposed by Dr. Michael Grieves in 2002 [1] , and since that time, the concept has been massively in use in the context of Industry 4.0, where the term is however frequently mistaken for the term *simulation*.

A distinction between simulations and digital twins is made in Kritzinger et al. (2018). According to the level of integration, digital system approximations are classified into three categories: (i) digital models running either independently on the physical objects or just with a manual data exchange, (ii) digital shadows equipped within automatic data flow from physical object to the digital object, and (iii) digital twins featured by both-directional data flow between physical and digital objects. The paper also includes literature review on known approaches to digitization and just a minority of them are categorized as digital twins according to the proposed distinction. The approach proposed in our paper uses bi-directional data flows and thus it belongs to the category digital twin specified in Kritzinger et al. (2018). It means that we are not misusing this term in our paper.

Digital twins create living digital models that update and change as their physical counterparts change. To do so, digital twins continuously learn and update themselves from multiple sources to represent their near real-time states, working conditions or positions. In this paper, a digital twin is not only automatic duplication of the actual state of the physical production system, but also a basis for automated production planning. In other words, the system operation strongly relies on the digital twin.

2.2 AI Planning and Scheduling with PDDL

Artificial intelligence (AI) planning (Ghallab et al. (2016)) is a branch of AI that solves a problem of transforming an initial system state into an envisioned goal state by finding a plan. Such a plan is a sequence of actions or action graphs (typically for execution by smart control systems, robots, or various connected devices/autonomous agents) for a given domain, where allowed actions and related constraints are formally specified.

In fully specified environments with complete domain models available, planning can be done off-line. Plans can be found and evaluated prior to their execution. In dynamic environments (such as industrial production lines), the plans may have to be revised at runtime. The process of planning is usually realized with iterative trial and path finding/branching, as it is commonly seen in artificial intelligence methods. AI planning incorporates techniques such as machine learning, dynamic programming, and combinatorial optimization. Although planning itself is not brand new and it has been investigated for more than fifty years, the recent advances in AI and planning algorithms make this approach feasible for industrial-scaled systems.

Planning refers to establishing a plan of actions, whereas scheduling (sometimes called capacity planning) is less concerned with what is being done and why, but more with when and where. A plan may (e.g., temporal planning) or may not (e.g., classical planning) incorporate times and dates associated to it, whereas a schedule most certainly will. Scheduling is concerned with mathematical formulations and solution methods of problems of optimal ordering and coordination in time of certain operations. Scheduling includes questions on the development of optimal schedules (Gantt charts, graphs) for performing finite (or repetitive) sets of operations, frequently related to capacities and long-term requirements. The problems that scheduling deals with can be formulated as optimization problems for processing a finite set of actions/jobs in a system with limited/constrained resources. In scheduling, the time of arrival for each action into the system is specified. Within the system, each action has to pass several processing stages, depending on the conditions of the problem. For each stage, feasible sets of resources are given, as well as processing times depending on the resources used. Constraints on the processing sequence and actions are usually described by transitive anti-reflexive binary relations.

Given a description of the initial state of the system, a description of the goal state, and a formal specification of a set of possible actions, the planning task is to synthesize a plan that is guaranteed to generate a state, which at the end satisfies all postulated goal conditions.

For specifying planning tasks, several languages have been developed. *Planning Domain Definition Language* (PDDL) is supported by most of the state-of-art planners and we are using it also in this paper. In PDDL, the planning task is explicitly split into two files:

(1) *Domain description*
The planning domain specification defines all allowed *actions*, which potentially change the state space along with their input parameters, *preconditions* (condition that must hold before the action starts) and *effects* (description of changes on state-space immediately after the action is finished).
(2) *Problem description*
The specific planning problem instance holding information about the initial state as well as the required goal-state conditions.

A *solution* for a PDDL problem (specified by its domain and problem description) is a *plan*, a sequence of actions that are to be sequentially applied starting from the initial state of the problem. After application of all actions, the goal-state conditions of the problem are satisfied.

The latest version of the language is PDDL 3.1 (Kovacs (2011)), but there exist numerous variants/extensions that support various features like ontologies, probabilistic effects, numbers and goal-achieved fluents, durative actions (temporal/parallel planning), explicit problem decomposition (multi-agent planning) and others.

A selection of the suitable PDDL extension including the explanation of techniques in successful solvers is provided in Sousa and Tavares (2013). A collection of simple prototypical industrial problems with their formalization in PDDL is presented in Rogalla et al. (2018). Compared to Rogalla et al. (2018), the approach proposed in this paper is much more oriented to a real system of industrial scale, and we are using PDDL not only for isolated planning but tightly integrated with a digital twin (cf. Sec. 3) as one smart production planning and execution system.

[1] cf. https://research.fit.edu/media/site-specific/researchfitedu/camid/documents/Origin-and-Types-of-the-Digital-Twin.pdf

A first attempt to goal-oriented manufacturing execution system (MES) based on PDDL planning is discussed in Novák et al. (2019). It already includes an idea of integrating a digital twin, however, this concept is not described there in details.

A closer coupling of production system models and PDDL has been realized in Wally et al. (2019a), where a rule set for the conversion of production system features into planning domain and problem entities has been provided. Initial experiments have shown that fully automated production (re-)planning from reasonably detailed production system models is feasible. This approach was making use of formal abstractions using methods, techniques and tooling taken from model-driven software engineering. In Wally et al. (2019b), this approach has been further extended by utilizing more advanced and computationally expensive features of PDDL, most notably *durative actions*. While the time and memory demands for finding production plans have sky-rocketed, the computed solutions (if they were found) have been of excellent quality, as they inherently take into account possible parallelism of multiple manufacturing tasks. We are building our integrated system that is presented in this paper on top of the findings gathered in these previous works and build a prototypical MES that includes production-system-to-PDDL-coupling just as one aspect of its architecture.

3. PRODUCTION PLANNING WITH A DIGITAL TWIN

A production line's digital twin holds the required knowledge about the current state of the entire production line, which in this context includes (excerpt):

- Involved components and their states: ready, running, under maintenance, or out of order.
- Positions of robotic arms including the information what is held by their tools/grippers.
- Positions/states of all components (such as automated guided vehicles (AGVs) or pallets) in transportation system and what content is carried.
- Information about quantities and locations of material in warehouses and buffers.
- Information from various sensors attached (e.g., temperature, humidity, distance, location).
- States of human-machine interfaces for interactions (control panels, buttons, visual indicators).

From the planning domain point of view, such a digital twin needs to accommodate the following properties and requirements (streamlined towards PDDL in our use case):

- External control signals need to be translated into PDDL actions that can be processed only under well specified conditions (PDDL preconditions) and that can have some effects on the internal state of the digital twin (PDDL effect on state-space).
- Interactions among digital twin components can be simulated by PDDL actions as well.
- The current PDDL state-space can access relevant sensor signals from digital twin sensors and received values from digital twin components.

The major problem with creation of a digital twin, according to the previous points, is to translate such external control signals into PDDL actions (or vice versa). Sometimes PDDL actions need more information (as arguments) than the provided external control signals contain. The digital twin then have to check all the preconditions of all actions that are continuously sent to it. In case of any action inconsistency, the digital twin does not apply the effect on such action and it reports an error back to MES.

Such a PDDL-enhanced digital twin can be used for the following application scenarios:

- Recomputation of a new plan from the current state of the production line in case of failure or in case of any modification of the production line.
- Visualization of the current state of the line.
- Global overview that can support centralized, consistent, and formalized (computer readable) data source for further processing in related systems (e.g. ERP, predictive maintenance, etc.)

The proposed way of integration of the digital twin with the rest of the automation system is depicted in Fig. 1. Both twins are controlled by the MES, which executes PDDL actions. The physical twin is controlled via OPC UA standard protocol (onto which the utilized PDDL actions are mapped), whereas the digital twin is updated via PDDL actions at the same time when these actions are sent via OPC UA to the physical twin. Orders are coming from the enterprise resource planning (ERP) system (on top) and it is the MES which is responsible for invoking the production planning and scheduling. For each order, a technology/production plan is automatically planned by a PDDL solver and it is returned back to the MES. In other words, the MES works in a 2-step manner, first processing an order by requesting planning of the production process, and second by interpreting it in a form of a *LispPlan* [2] and executing all actions.

A continuously running digital twin is a key enabler for production re-planning in case of failures. Any production operation such as robotic manipulation or transport shuttle movement can result into a failure in the physical system, e.g., due to a collision or loosing/destroying a component. Since the digital twin is keeping the up-to-date information about the started and finished PDDL actions, i.e., successful and unfinished production operations, the planner can use this state information as a basis for re-planning the production process to achieve the production goals with limited resources. After identification and isolation of the failure and modifying the PDDL problem description, a new production plan will be found if the remaining available resources can substitute the original failed ones. This continuous support for planning and re-planning is not supported by other available tools and we call this MES feature "reactivity".

3.1 Integration of PDDL and MES

The *production planner & scheduler* subsystem needs to blend into the overall production control system in order to provide up-to-date information that can be used to create control statements for the MES system. The three main components of this subsystem fulfill specific tasks, as explained below.

[2] LispPlan is explained in details later in Sec. 3.1

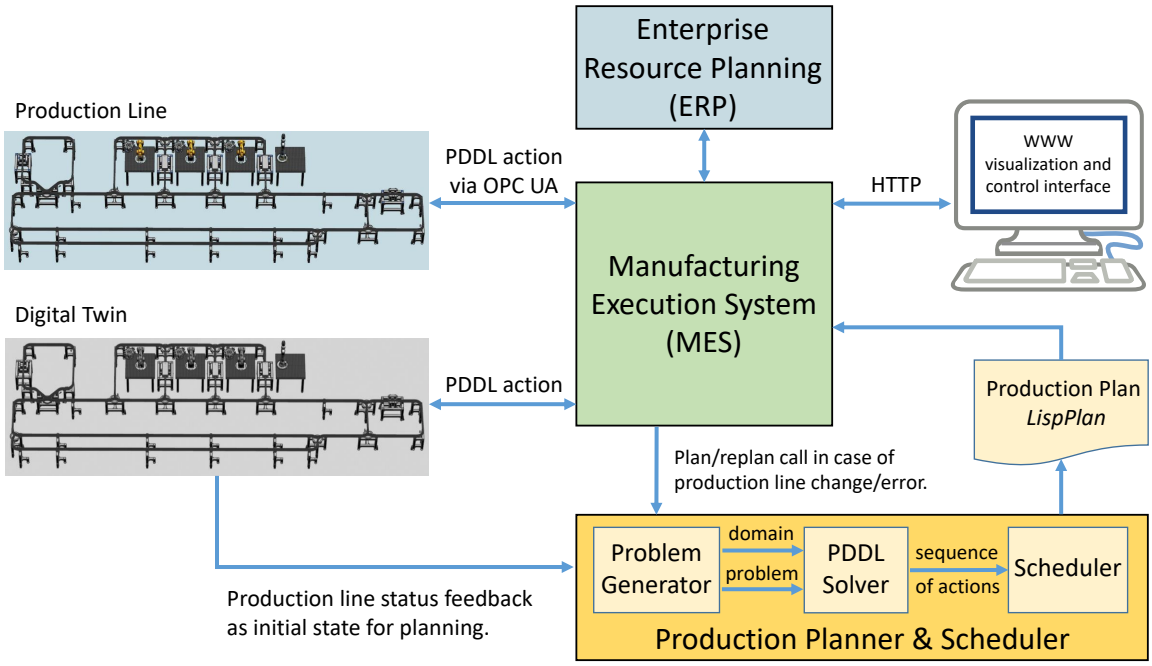

Fig. 1. Production planning with a PDDL-enhanced digital twin.

The problem generator module (cf. Fig. 1) executes its task based on the following inputs:

- Current state of the production line captured in the digital twin module
- Production goals from the MES module
- Capabilities of the production line (what operations are available)

The output of the problem generator module is a standard PDDL domain and problem specification, which is ready to be used by the PDDL solver module (currently realized by using the Fast Downward planner [3]). The output of the PDDL solver module is a sequence of actions, each including a complete specification of its parameters.

The task of the scheduler module is to read the output plan from the PDDL solver module, analyze it, and finally create a more detailed and tailor-made schedule for the MES module. For that purpose, we have developed a special format called *LispPlan* with the following features:

LISP-like syntax that is human and computer readable. It can be quickly enriched with new features or translated into another format, such as XML.
Task and sub-task definitions – tasks can be recursively divided into sub-tasks.
Locations – description of the location of the required resources for each task.
Actions – description of target operations in a PDDL-compatible format.
Requirements – description of dependencies among tasks (tasks can be executed in parallel).

[3] cf. `http://www.fast-downward.org/` with the following heuristics search switch on: `--search "lazy_wastar([ff(),blind(),hmax()], bound=100, boost=0)"`

The goal of a *LispPlan* interpreter (in our use case the MES) is to complete all tasks that are specified by respective task definitions.

- A task is completed if its action or all of its sub-tasks are completed.
- Some task X can start being processed if (i) its parent task has already been started (or task X does not have any parent task) and (ii) all tasks that are specified in the requirements section of task X have already been completed (or task X has no requirements).

Because of recursive nature of task and sub-task specifications, *LispPlan* differentiates between two types of identifiers:

- Absolute: identifier starts with "@"
- Local: identifier does not start with "@" and the real name (that is used in *LispPlan* interpretation) is composed by sequence of local identifiers joined by "." from the nearest higher absolute identifier (in definition hierarchy) or from the root local identifier (in case there is no nearest higher absolute identifier used). The order is similar to that of the Internet domain name system: the most local identifier is the first one and the root identifier is the last one.

Absolute and relative naming can be used in requirements as well. The same absolute and relative identification holds for locations.

The current implementation of the scheduler module does not support scheduling tasks for concrete times and dates. As of now, only the accessing of resources is analyzed, in order to produce constraints on the processing sequence of actions. Finally, a directed acyclic graph in the LISP format represented as *LispPlan* is produced.

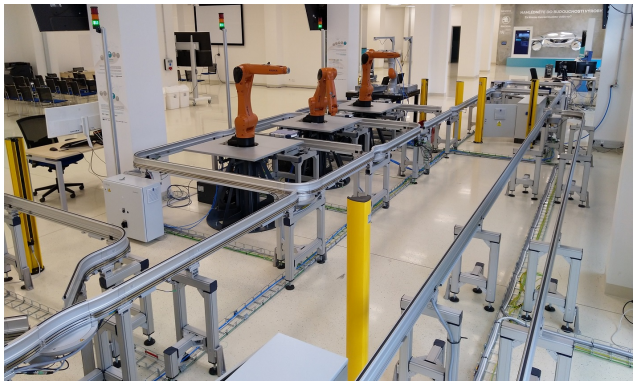

Fig. 2. Industry 4.0 testbed located at the Czech Technical University in Prague, CIIRC.

4. USE CASE: SMART PRODUCTION IN THE INDUSTRY 4.0 TESTBED

The Industry 4.0 testbed (cf. Fig. 2) is a laboratory-scale system for educatory and testing purposes located at the Czech Technical University in Prague, CIIRC [4]. It is equipped with modern infrastructure to test and verify innovative automation and control approaches for final assembly of products. Although the testbed is rather a small-scale environment, the distributed nature of its control system together with the use of standardized industrial components assure that evaluation results can be extrapolated to industrial production line scales.

4.1 Physical Layout of the Production System

The core part of the Industry 4.0 testbed is depicted in Fig. 2. It comprises four industrial robots of two types that are logistically connected via a transportation system from montratec GmbH [5]. This transportation system consists of a monorail track that is assembled from straight segments, curves, and switches, as well as *shuttles* that move on this track and *positioning units* that assure exact positioning of these shuttles in certain locations, such as work cells. The layout of the Industry 4.0 testbed including identifiers of key elements is depicted in Fig. 3. It is important to note that the work cell positioning units are each shared between two robots—this setup welcomes cooperation between robots.

The Industry 4.0 testbed is equipped with the following industrial robots:

- 3x KUKA KR Agilus: Fast industrial 6-axis robots, programmed with KUKA Robot Language (KRL)
- 1x KUKA LBR iiwa: Cooperative 7-axis robot, programmed with Java

The Industry 4.0 testbed provides an environment to be able to find answers for the following issues coming from cooperation with industrial partners:

- How to combine production of goods of different nature in parallel?
- How to recover from errors if a certain resource or a set of resources is not able to finish a production operation (e.g., a specific route of the transportation system is blocked, or a robot becomes unavailable after a collision)?
- How to systematically add and remove production resources at runtime (e.g., adding a new shuttle, or removing a robot)?

For all of the aforementioned problems at the border between research and industrial practice, we found out that the concept of production planning based on digital twin is a suitable and efficient paradigm. The main reason is the declarative nature of the digital twin and of PDDL for planning, as it is possible to declare new pieces of products and resources without the need for re-programming the machinery, which would not be feasible at runtime. Moreover, we can unambiguously analyze and update the state of the production line by means of the digital twin.

4.2 Production of Small Trucks

To test and evaluate the digital twin concept in the Industry 4.0 testbed, we used a set of components for a small truck printed on 3D printers, whose shapes are depicted in Fig. 4. Each such truck can consist of a chassis (only black color and one type), a cabin (4 colors, same shape), and a body (4 colors, 4 shapes). The production goal is to produce an ordered truck and transport it to the station S200 designed for handing over finished goods.

At the very beginning of the truck assembly, the production line and the digital twin have to be in a pre-defined initial state. But after producing a first truck, the line does not need to be returned back to such a pre-defined state, but the last state retrieved from the digital twin can be used for specifying the problem description.

To illustrate the production, we have ordered a truck having a yellow cabin and a blue stakebed body, see Fig. 5. The formal specification of this goal is depicted in Fig. 6. The planner found a solution expressed in Fig. 7. The digital twin keeps the information that body position at robot R2 warehouse is occupied by a silver tank when the order comes. The first step is to move this component to SHUTTLE2 (see tasks 0 and 1), then to swap shuttles 2 and 4, and afterwards to get rid of a yellow dumper body from SHUTTLE4. The yellow cabin is already on SHUTTLE4, it is not moved and thus not mentioned in the *LispPlan*. Finally, the blue stakebed is moved from SHUTTLE3 to SHUTTLE4 and the ordered truck is transported to the final positioning unit.

4.3 Tests and benchmarks

The proposed approach has been thoroughly tested in the Industry 4.0 testbed for more than 3 months including several hundreds continuous runs. All runs were logged into the XES standard format [6]. The utilization of XES facilitates further analysis of grabbed data, including process mining tools such as ProM [7]. Based on these analyses, we found out that the time period needed for getting production order from the ERP system, retrieving the production

[4] cf. `https://www.ciirc.cvut.cz/`
[5] cf. `https://www.montratec.de/`
[6] XES format is standardized as IEEE 1849-2016 and it is available online: `http://www.xes-standard.org/`
[7] Process Mining Workbench (ProM) is available online: `http://www.promtools.org/`

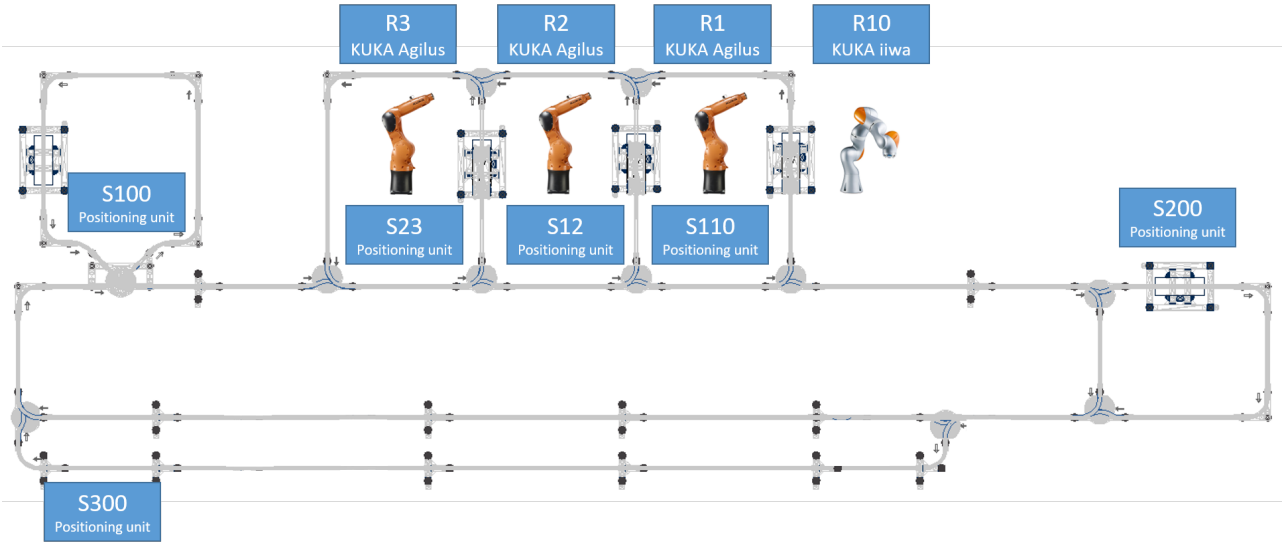

Fig. 3. Topology of the Industry 4.0 testbed including annotations for the key components: robots and positioning units.

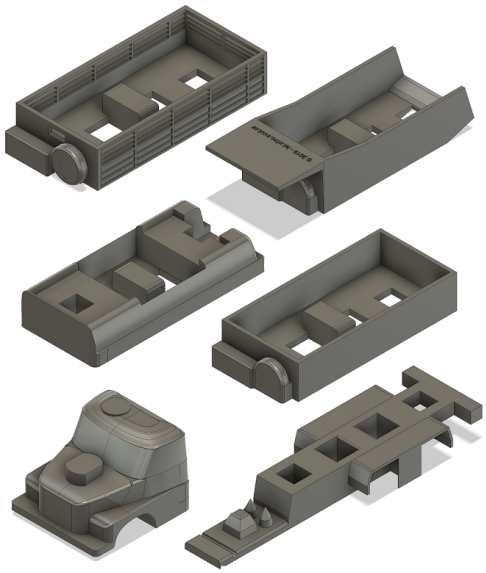

Fig. 4. All truck spare part shapes: chassis, cabin, and a variety of bodies.

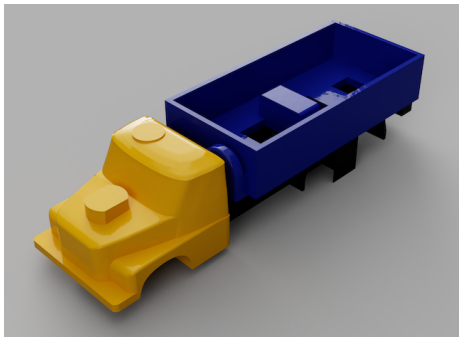

Fig. 5. Small truck consisting of three components (chassis, cabin, and body) as an exemplary product.

```
(:goal
  (and
    (exists (?P1 - POSITION ?P2 - POSITION ?P3 - POSITION ?S - RESOURCE)
      (and
        (RESOURCE_CONTAINS_PART_AT ?S PART-TYPE-T-TRANSPORT
            PART-YELLOW-CABIN ?P1)
        (RESOURCE_CONTAINS_PART_AT ?S PART-TYPE-T-TRANSPORT
            PART-BLUE-STAKEBED ?P2)
        (RESOURCE_CONTAINS_PART_AT ?S PART-TYPE-T-TRANSPORT
            PART-BLACK-CHASSIS ?P3)
        (IS_SHUTTLE ?S)
        (RESOURCE_IN_STATION ?S S200) ) ) ) )
```

Fig. 6. Example of a declarative specification of a goal in PDDL. This goal specifies the following target state of the production line: At the position unit S200 is some shuttle that contains a part-wise specified truck (see Fig. 5).

system state from the digital twin, planning production process up to starting this process in the MES does not exceed 8 seconds. Whereas the planning based on digital twin itself took less than 1 second. For such experiments, we used one computation node from our experimental industrial cluster that is equipped with Intel® Xeon® CPU E5-2630 v2 @ 2.6 GHz, 4 cores, 16 GB RAM, 100 GB SSD storage, and Ubuntu Linux 18.04.2 LTS. We proved that the proposed approach provides enough performance and reliability for continuous run in the system of the testbed size. We have not indicated any problems related to digital twin, production planning system nor MES system. But, for curiosity, instead of solving any problem with our software prototype, we had to fix several hardware related problems namely with one industrial electrical gripper [8] and two robots [9]. Due to the replanning function of the proposed solution all the hardware problems did not affect the goals of the entire production (as the production was re-deployed to other available resources).

[8] Schunk EGP 50-N-N-B
[9] KUKA Agilus (type: KR 10 R1100 sixx) and KUKA LBR iiwa (type: 14 R820)

```
(define (task Building_Truck)
  (:location testbed.ciirc.cvut.cz)
  (define (task 0) (:location R2)
                   (:action (ROBOTIC_PICK R2 R2_TABLE SVR2 X2_Y6_Z7_R0
                            PART-SILVER-TANK PART-TYPE-T-STORAGE)))
  (define (task 1) (:location R2)
                   (:requirements 0)
                   (:action (ROBOTIC_PLACE R2 SHUTTLE2 S23 X0_Y5_Z7_R0
                            PART-SILVER-TANK PART-TYPE-T-TRANSPORT)))
  (define (task 2)
     (:requirements 1)
     (define (task A) (:location SHUTTLE2)
                      (:action (SHUTTLE_MOVE_AND_LOCK SHUTTLE2 S23 S200)))
     (define (task B) (:location SHUTTLE4)
                      (:action (SHUTTLE_MOVE_AND_LOCK SHUTTLE4 S200 S23))))
  (define (task 3) (:location R2)
                   (:requirements 1 2)
                   (:action (ROBOTIC_PICK R2 SHUTTLE4 S23 X0_Y5_Z7_R0
                            PART-YELLOW-DUMPER PART-TYPE-T-TRANSPORT)))
  (define (task 4) (:location R2)
                   (:requirements 0 3)
                   (:action (ROBOTIC_PLACE R2 R2_TABLE SVR2 X2_Y6_Z7_R0
                            PART-YELLOW-DUMPER PART-TYPE-T-STORAGE)))
  (define (task 5) (:location R2)
                   (:requirements 4)
                   (:action (ROBOTIC_PICK R2 SHUTTLE3 S12 X0_Y5_Z7_R0
                            PART-BLUE OPENTOP PART-TYPE-T-TRANSPORT)))
  (define (task 6) (:location R2)
                   (:requirements 3 5)
                   (:action (ROBOTIC_PLACE R2 SHUTTLE4 S23 X0_Y5_Z7_R0
                            PART-BLUE-OPENTOP PART-TYPE-T-TRANSPORT)))
  (define (task 7)
     (:requirements 2 6)
     (define (task A) (:location SHUTTLE2)
                      (:action (SHUTTLE_MOVE_AND_LOCK SHUTTLE2 S200 S23)))
     (define (task B) (:location SHUTTLE4)
                      (:action (SHUTTLE_MOVE_AND_LOCK SHUTTLE4 S23 S200)))))
```

Fig. 7. Example of an automatically generated production plan (LispPlan) for a truck as it is depicted in Fig. 5.

5. CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel approach for industrial production system control based on tightly integrated digital twin and automated AI planning. This way of integration leads to more flexible, efficient, and goal-oriented manufacturing execution processes.

We successfully validated this approach on the Industry 4.0 testbed use-case. The simplicity of adding and removing resources or products makes this approach suitable for flexible production sites oriented on small lots as well as production sites equipped with multiple redundant technologies/components. In small lots, it is possible to automatically find a plan that is compliant with the current capability of the production line. In larger lots, efficiency of the production can be improved compared to fixed resource assignments done by humans, as the utilization of resources/assets is balanced.

In *future work*, we would like to validate the proposed approach on medium- and large-scale industrial production lines. We trust that this approach can be efficiently used for smart grid and microgrid control, and we are going to validate this hypothesis on laboratory-scaled DC microgrid systems. We would like to connect our digital twin to an industrial simulation system (such as Siemens Process Simulate) for more precise validation in terms of energy, cost, and time efficiency.

ACKNOWLEDGEMENTS

This result was funded by the Ministry of Education, Youth and Sport of the Czech Republic within the project Cluster 4.0 reg. no. CZ.02.1.01/0.0/0.0/16_026/0008432. It was also supported by the Flexible AC/DC microgrid for collective housing project, and National Competence Center - Cybernetics and Artificial Intelligence, all funded by the Technology Agency of the Czech Republic. Further, financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged. We thank our colleague Václav Jirkovský for designing, rendering, and 3D printing truck parts.

REFERENCES

Dong, B., Wally, B., and Ferscha, A. (2009). Tokenized interaction architecture. In *Proceedings of the 2nd International Workshop on Pervasive Advertising*.

Ghallab, M., Nau, D.S., and Traverso, P. (2016). *Automated Planning and Acting*. Cambridge University Press.

Kovacs, D.L. (2011). Complete BNF description of PDDL 3.1. Language specification, Department of Measurement and Information Systems, Budapest University of Technology and Economics.

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

Novák, P., Vyskočil, J., and Kadera, P. (2019). Plan executor MES: Manufacturing execution system combined with a planner for Industry 4.0 production systems. In V. Mařík, P. Kadera, G. Rzevski, A. Zoitl, G. Anderst-Kotsis, A.M. Tjoa, and I. Khalil (eds.), *Proceedings of the 9th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS)*, 67–80. Springer.

Rogalla, A., Fay, A., and Niggemann, O. (2018). Improved domain modeling for realistic automated planning and scheduling in discrete manufacturing. In *Proc. 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 464–471.

Sousa, A.R. and Tavares, J.J.P.Z.S. (2013). Toward automated planning algorithms applied to production and logistics. *IFAC Proceedings Volumes*, 46(24), 165–170.

Wally, B., Vyskočil, J., Novák, P., Huemer, C., Šindelář, R., Kadera, P., Mazak, A., and Wimmer, M. (2019a). Flexible production systems: Automated generation of operations plans based on ISA-95 and PDDL. *Robotics and Automation Letters*, 4(4), 4062–4069.

Wally, B., Vyskočil, J., Novák, P., Huemer, C., Šindelář, R., Kadera, P., Mazak, A., and Wimmer, M. (2019b). Production planning with IEC 62264 and PDDL. In *Proceedings of the 17th IEEE International Conference on Industrial Informatics (INDIN)*, 492–499.