Research Article

Marko Vještica, Vladimir Dimitrieski*, Milan Pisarić, Slavica Kordić, Sonja Ristić, and Ivan Luković

Towards a Formal Specification of Production Processes Suitable for Automatic Execution

https://doi.org/10.1515/comp-2020-0200
Received Mar 21, 2020; accepted May 31, 2020

Abstract: Technological advances and increasing customer need for highly customized products have triggered a fourth industrial revolution. A digital revolution in the manufacturing industry is enforced by introducing smart devices and knowledge bases to form intelligent manufacturing information systems. One of the goals of the digital revolution is to allow flexibility of smart factories by automating shop floor changes based on the changes in input production processes and ordered products. In order to make this possible, a formal language to describe production processes is needed, together with a code generator for its models and an engine to execute the code on smart devices. Existing process modeling languages are not usually tailored to model production processes, especially if models are needed for automatic code generation. In this paper we propose a research on Industry 4.0 manufacturing using a Domain-Specific Modeling Language (DSML) within a Model-Driven Software Development (MDSD) approach to model production processes. The models would be used to generate instructions to smart devices and human workers, and gather a feedback from them during the process execution. A pilot comparative analysis of three modeling languages that are commonly used for process modeling is given with the goal of identifying supported modeling concepts, good practices and usage patterns.

Keywords: Industry 4.0, Production Processes, Knowledge Bases, Formal Languages, Model-Driven Software Development

1 Introduction

Manufacturing has been the driving factor behind the development of the human race since the humans discovered the first tools.[1] Over time, manufacturing became more and more sophisticated. Significant paradigm changes were triggered by inventions that have highly influenced the whole manufacturing process. Until the end of the 20th Century machines were still not fully independent and self-adjustable to variations in the manufacturing processes.

A fourth industrial revolution has been triggered by technological advances and increasing customer need for highly customized products. This manufacturing revolution is called Industry 4.0 in Germany and Smart Manufacturing in the United States. Both countries, and many others like Japan, China and Korea, have established national programs and research projects in the domain of smart manufacturing in order to define and implement its concepts [43].

In traditional manufacturing systems, switching between different products or different product variants requires either stopping the production to reconfigure equipment, or having different production lines for each product [5] which infers additional costs to the manufacturer. On the other hand, one of the goals of Industry 4.0 is to provide automatic and flexible production thus enabling mass customization of products [20]. Such production flexibility would allow different products or product variants to be produced simultaneously without the need to stop production while performing the reconfiguration [5]. In order to enable such flexibility, the entire reconfiguration of the production line has to be done "on the fly" by the automated mechanisms of the smart factory and based on the production process specification. Thus, production process specifications need to be formally described and machine readable.

***Corresponding Author: Vladimir Dimitrieski:** University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia; Email: dimitrieski@uns.ac.rs
Marko Vještica: University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia; Email: marko.vjestica@uns.ac.rs
Milan Pisarić: Industrial Automation, KEBA AG Linz, Austria; Email: pisa@keba.com
Slavica Kordić: University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia; Email: slavica@uns.ac.rs
Sonja Ristić: University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia; Email: sdristic@uns.ac.rs
Ivan Luković: University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia; Email: ivan@uns.ac.rs

1 This paper is an extended version of a conference paper [47].

Production processes need to be specified in the form of production process models by using a Domain-Specific Modeling Language (DSML) that includes concepts for representing materials, products, services, devices, human workers, communication between them and all process steps and tasks needed to create a product. Implementing a DSML in a formal way increases: (i) consistency during modeling, (ii) model quality and (iii) modeling speed. Also, DSMLs could increase human comprehension of models by supporting users to use familiar modeling concepts. In this particular case, process designers would use a DSML – a formal language that provides concepts of a production domain, to model production processes that would be machine readable, but also simple enough for human comprehension.

Due to the increase of production process complexity in Industry 4.0, production process models will help process designers to think about production processes at a higher abstraction level and be more focused on modeling production process steps in order to minimize errors during processes specification. On the other hand, mechanisms for automating the production coupled with mobile robots and workbenches will enable automatic reconfiguration based on the instructions stemming from the process models.

Although there is a lot of invested research effort, many research questions are left unanswered and challenges which are not yet addressed. One of the important research topics not fully addressed is the identification of concepts that lead to the creation of a formal language to describe production processes, which in turn allows easier specification of process models and supports production of highly customized products.

To create production process models, a digital twin of a physical process is needed that will enable process monitoring, real-time decision making and control [56]. A digital twin represents a virtual model of a physical object which can simulate the object behavior [34]. In that way it is possible to simulate production steps and to predict their impact on the product [49]. These simulations are highly utilized in Industry 4.0 to simulate products, robots and humans in order to reduce failures and optimize resource consumption [45], *e.g.* energy consumption.

Without simulations, a production failure would happen more often due to the complexity of the production process. Also, resource consumption could be higher than it should be because a certain product can be produced using different methods and simulations help to find an optimal production process for a specific product. Having a formal and machine-readable production process model will allow for more precise and domain-specific simulations and enable easier creation and execution of simulations [36].

Besides the production simulations and manufacturing flexibility, having formal production process models would provide means for better integration of humans in production processes in a way that is prescribed by lean production principles [16]. A visualization of production process models will be necessary for humans to either supervise the production or participate in the production [10]. The visualized production process models could be used by: (i) human supervisors for process monitoring, and (ii) human workers as a manual to execute production tasks. The visualization of production process models could provide human workers with better understanding of processes which will minimize errors during process execution.

In this paper we establish a proposal for a research on creation of a framework for the formal description and automatic execution of production processes. The framework is centered on a production process modeling language that needs to be designed and implemented. The language can be categorized as a DSML aimed to specify production processes, as existing process modeling languages are not tailored to model production processes. Most of existing languages lack a capability to model a material flow, some of them can only specify sequential processes and others cannot describe smart resources that should execute process steps. We discuss these issues further in Section 4 and 5.

In addition to the content of our conference paper [47], different modeling languages are compared in this paper in order to determine which production process concepts can be modeled by the languages and are they able to: (i) specify human-machine collaboration, (ii) support generation of human-readable and machine-readable instructions automatically, and (iii) improve the production of highly customized products and product variants.

A production process example is constructed to cover most important concepts and it is modeled by means of three process modeling languages: Business Process Model and Notation (BPMN) [12], Unified Modeling Language (UML) [13] and Petri nets [30]. Many researchers apply these languages and language extensions in the production domain, as is presented in Section 5. Thus, we chose these languages to model identified production process concepts. However, even if BPMN extensions or UML profiles were created, it is still hard to model human-machine integration in the smart factory and to model production processes with all the details needed for automatic code generation.

The human-machine integration, automatic code generation and execution of generated instructions could provide basis for automatic management of the production

and thus highly customized products could be produced at the lower cost. Due to domain-specific challenges in modeling production processes, researchers are trying to apply domain-specific modeling [52]. Some of the domain-specific challenges are modeling production process steps with all the details substantial for automatic process execution; smart resources that will execute each process step; and the human-machine collaboration. In order to cope with these challenges we have proposed a usage of a DSML to model production processes.

The main goal of the proposed research is to define a methodological approach and a software solution in which the Model-Driven Software Development (MDSD) principles and a DSML will be used to formally specify production processes, to automatically generate instructions for smart machines or human workers, and to gather a feedback from them during the process execution.

Apart from this section, this paper is organized as follows. In Section 2 an overview of a smart factory is presented. Requirements for modeling production processes in Industry 4.0 and an MDSD based system for automatic execution of code generated from models are described in Section 3. In Section 4 different process modeling languages and their capability to model production processes are compared. Section 5 represents a related research to the one presented within this paper. Conclusions and future work on implementation of the DSML related to production processes in Industry 4.0 are presented in Section 6.

2 An Overview of a Smart Factory

To create a smart factory, advanced technologies like Cyber-Physical System (CPS), Internet of Things (IoT) and Wireless Sensor Network (WSN) are introduced in the manufacturing process [50, 55]. CPS is the core element of Industry 4.0 and production processes. It is used to denote physical objects with embedded software and computing power, which transforms products into smart ones and enables a smart and decentralized production [2]. This transformation provides significant resource and cost advantages compared with traditional production systems [54]. CPSs are the connection between physical and virtual manufacturing worlds [27], and they are aimed at enhancing process control by synchronizing these two worlds and by hiding the feedback from sensors and actuators state changes [37]. An exchange of the feedback and the communication between CPSs in manufacturing systems are provided by IoT and WSN. The smart products and machines have to be fully networked and integrated with the smart factory in order to require minimal manual interventions [7].

The smart factory is based on advanced technologies, smart resources, smart materials and smart storages that are commonly used in Industry 4.0. In this context the smart resource is a service provider, the one that will process operations, and the smart material is a service consumer, the one that will be processed [2]. The smart factory production system needs to have smart resources of different types, like robots, machines or human workers. In order to produce highly customized products with many different products involved in the production, a routing of smart resources should be reconfigured dynamically to enable easy switching between different smart products, which is very hard to achieve in traditional manufacturing. To enable dynamic routing, all these smart resources and smart products need to collaborate and communicate with each other. Modeling of smart resources and their collaboration and communication by means of a process modeling language can be of great help for process designers in the Industry 4.0 environment.

In the rest of this section, we provide discussion on: (i) production logistics in the smart factory environment, (ii) disturbance detection and error handling (iii) integration of humans and machines within the smart factory, (iv) potentials of a cluster of industrial computers named Orchestrator to manage the production, (v) gathering feedback from the production and data analysis, (vi) an example of an assembly smart factory and (vii) the summary.

2.1 Production Logistics with Automated Guided Vehicles

Automated Guided Vehicles (AGVs) are intelligent automatic vehicles used for transport and dynamic routing. AGVs are important resources in smart factories and intelligent manufacturing because they improve production by increasing automation and efficiency, reducing labor costs and helping make production fast and safe [49]. Dorofeev *et al.* [5] presented an example of a rich transportation network of seven AGVs used for creation of different product variants. The transportation of smart materials and smart products can use most of the production time [33], so modeling transportation activities and management of transport vehicles are very important.

In order to transport smart materials and smart products or execute different operations on them, AGVs can be equipped with Radio Frequency IDentification (RFID) readers to read or write to the smart products' or smart materials' RFID tags and to communicate with other smart resources

via IoT [35, 50]. RFID tags enable a wireless identification and localization of smart products and smart materials, and event generation to track the production process, *e.g.* the material is picked up or placed [23, 41]. Besides smart resources and smart materials, there are also smart storages that can be equipped with different sensors in order to track their state or inventory. This is important information to know during the production as smart resources need to transport smart materials to available smart storages that lack specific materials for the production.

2.2 Error Handling

Error handling represents a set of activities to reduce negative effects of an error after its occurrence and to avoid a failure scenario [8]. As AGVs or other robots can read products' or materials' states, they can detect if any disturbance occurs due to automatic manufacturing process or production logistics activities in order to manage the error, *e.g.* send a manual to a human worker to manually fix the problem [10]. A detection of any disturbance during the production requires the error handling, which is very important as errors can occur in any step of the process and they need to be carefully managed and modeled in order to clearly define errors and identify their boundaries [42].

Industrial systems are becoming more complex and expensive so there is less tolerance for any kind of fault. Consequently, it is very important to respond quickly to detected and identified faults or disturbances in order to minimize performance degradation, productivity decrease and safety hazards [9]. However, error handling activities usually are not integrated within physical processes due to lack of defined workflows [8], and thus production process models should also include error handling activities.

2.3 A Human-Machine Collaboration

Besides machines and robots, human workers may also be considered smart resources. They are integrated within the smart factory and they work on tasks in which robots are not applicable or they work as problem solvers. Human workers interact with the smart factory using smart phones, smart watches, tablets or smart glasses to send or receive different kinds of information [10, 43]. In comparison to human workers, autonomous robots can work on different tasks more precisely and in places that human workers are not allowed to work [45]. Also, in some countries, laws limit human workers from working some jobs, *e.g.* to lift heavy objects. In the context of smart manufacturing, where the collaboration between human workers and robots is important, cobots are being used. Cobots are designed to work and collaborate with human workers and to ensure their safety during production [43].

Human workers: (i) supervise robots' work, (ii) collaborate with robots during production, and (iii) work on the tasks that robots are not capable of doing. Collaboration and exchange of information between a human and a machine is not the only type of collaboration in existence. It is also important to model collaboration between humans and between machines. In the context of Industry 4.0, most of the communication will be between various machines [35, 48].

Most of the described communications and collaborations between machines and human workers need to be modeled within the production processes in order to specify exchange of messages between them and their work in parallel, which will make a production of highly personalized products faster. Otherwise the production can only be sequential and without significant optimization.

2.4 Orchestrating the Production

In order to connect and control smart resources, materials and storages, an automatic management of production processes requires all the resources to work together and to be connected with a decision-making system [35]. This system could be represented by Orchestrator which can: (i) delegate instructions to different smart resources, (ii) manage transportation of materials and products between storages, and (iii) detect and configure new smart resources or reconfigure existing ones [5, 31]. Examples of a production orchestration can be found in the work of Loskyll [18] and Keddis [15].

To orchestrate the production, a knowledge of the smart factory topology is needed. This knowledge is composed of: (i) smart resources and their set of skills, (ii) smart materials and their properties such as dimensions, shape and mass, (iii) production processes and their steps, skills needed for the execution of process steps and different constraints that need to be included during the process execution, and (iv) the production logistics. For example, smart material mass is an important property as not all smart resources are capable of transporting and processing heavy materials [44], and Orchestrator needs to find all of the capable smart resources and match them with process steps that require any work with the material.

The presented knowledge is similar to the manufacturing capability conceptual model in [21]. Orchestrator communicates with the knowledge base that stores all such

knowledge and a reasoning mechanism must be implemented. Similar to Orchestrator, the smart machine agent module [55] is described alongside a real-time data perception, a pool of knowledge and rules, a reasoning mechanism and an executor.

2.5 Gathering Feedback and Data Analysis

During production, smart resources, materials and storages generate data from their sensors and that feedback data is needed for data analysis in order to optimize production processes. Generated feedback data requires a lot of storage memory, so the data is sent to a big data storage [50]. Every RFID sensor will send data like the storages' status, smart resources' or smart materials' states and process steps' completion status during the production. An Enterprise Resource Planning (ERP) system gathers the feedback data from RFID sensors through smart resources that can read the data and store it in a feedback data storage. This will allow data analysis and process monitoring to increase production quality, optimize energy consummation, reduce production costs and detect when maintenance is required [35, 49].

An asynchronous communication between smart resources and the ERP system, and between smart resources and Orchestrator is needed to gather a feedback and to delegate instructions. Communication protocols and platform-independent interfaces must be integrated with an intelligent information system of the smart factory. The asynchronous communication could be established with REpresentational State Transfer (REST) Application Programming Interfaces (API) [37] and the communication with different smart resources through sensors, actuators and mobile devices could be established with industrial protocol Open Platform Communications Unified Architecture (OPC UA) [10, 37]. REST APIs and the OPC UA protocol will be used to send instructions to smart resources generated from production process models, which will enable process execution without manual interventions.

2.6 A Smart Factory Example

To summarize previously described aspects of the smart factory, we present an example of smart assembling in Figure 1.

Smart materials are stored in a global storage. If some of the materials are needed for the production, an AGV will bring all required materials to a smart shelf. During production, another AGV will bring materials from the smart shelf to a smart material area. A human worker and a cobot will work together or work in parallel to create a product using materials from the smart material area. The product is then created at a smart table. A finished product will be transported to a finished product area using another AGV.

All these smart storages, smart materials and smart resources are tagged with an RFID so they can generate dif-

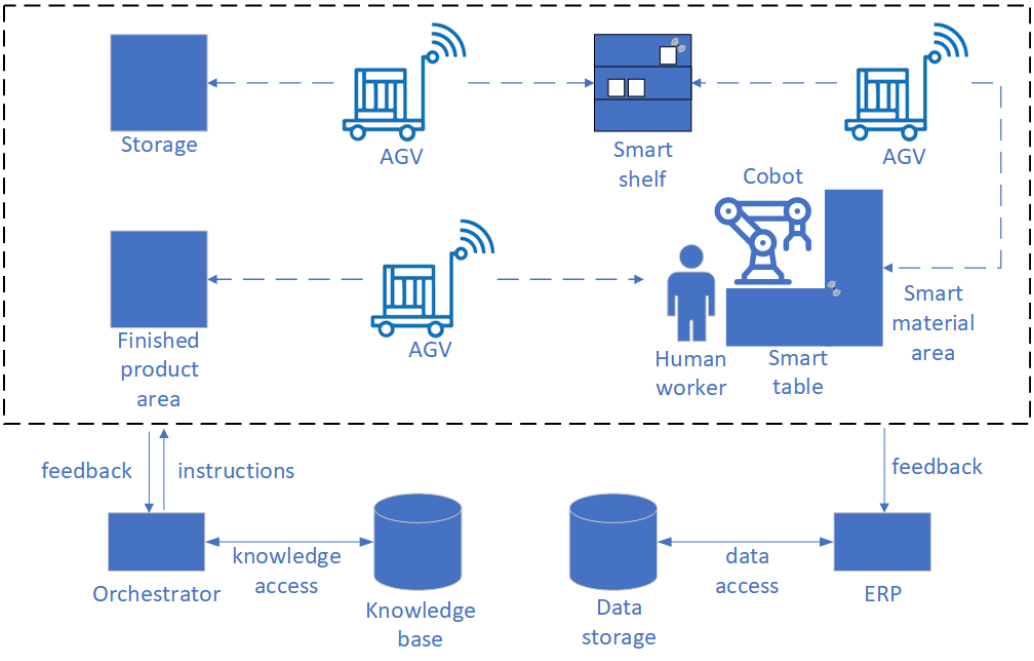

Figure 1: An example of assembly smart manufacturing.

ferent information, and smart resources are equipped with an RFID reader to read the information. The human worker has a mobile device to send or receive different information. This smart factory is managed by Orchestrator which communicates with a knowledge base to gather all necessary knowledge about the production and the smart factory. Also, whenever a production disturbance is detected, Orchestrator should handle the error and manage a process recovery. A feedback from RFID tags is gathered by an ERP and stored in a data storage for data analysis and optimization. The data storage will have various data that is gathered from different sources and a large volume of data must be stored [34]. An example of a reconfigurable assembly system that uses two industrial robots and RFID tags to generate events can be found in the work of Makris *et al.* [23].

2.7 Summary

To cover all aspects of the presented smart factory, we identified production process modeling concepts needed to fully describe all process steps, smart resources, smart materials, smart products, the error handling, the production logistics and the collaboration between smart resources. These concepts are needed to model production processes and generate instructions from them to smart resources. A modeling language is needed to cover all the presented aspects of production processes. This will make a bridge between process specification and process execution. The language also needs to provide easier production process specification done by process designers. As a result, the process specification will have fewer faults and will be more precise, and the process will be optimized.

3 A Production Process Modeling Language within an MDSD Approach

Within the context of a smart factory as described in the previous section, we propose the creation of a DSML aimed at production process modeling that will be used within an MDSD approach to generate and automatically execute code on smart resources. In this section we present basic concepts of the DSML identified within the smart factory and a system for automatic execution of production process models that supports the MDSD approach.

3.1 An Overview of the Core Language Concepts

A production process model needs to be presented as a workflow that is composed of different activities, *i.e.* tasks that need to be executed in a defined order [53]. An order of activities can be specified as follows: (i) a sequence – a strict order in which activities must be executed, *e.g.* drilling must be done before sanding, (ii) a parallelism – two or more activities that can be executed in parallel, *e.g.* two holes can be drilled in parallel, (iii) a flow control – depending on the specified parameter values, different activities are executed, *e.g.* sand a drilled hole if it has a diameter of the specified value, otherwise dismiss the material, and (iv) an iteration – one or more activities executed a certain number of times until a specified condition is fulfilled, *e.g.* heat a metal plate five times with a duration of three seconds. There are also activities that are unordered – they cannot be executed in parallel, but the order of execution is not important, *e.g.* part *A* and part *B* needs to be assembled with part *C* and the order of assembly activities is not important, but because of parts location the activities cannot be done in parallel.

An activity, a smart resource and a smart material together constitute a production process step. Every process step needs information about the smart resource that needs to execute the activity on the smart material. Activities of process steps could be operations, transportations, configurations or inspections and they can contain different parameters about activity completion criteria (*e.g.* cut a metal plate at 10cm), and acceptance criteria (*e.g.* it is acceptable if a metal plate is cut between 9.9cm and 10.1cm). Completion and acceptance criteria are related to quality assurance, which is an important part of production processes. Many tests could be performed by human workers or robots in order to check if the quality of a product is acceptable and this should also be modeled with a production process language.

Also, different kinds of faults could occur during the production. A material could be damaged (*e.g.* a metal plate is scratched), a robot could be broken or stopped (*e.g.* a battery has been depleted), or a human worker executed an operation in an incorrect manner, or the operation is not done at all. Some errors could be handled in the general way (*e.g.* move a material to the recycle area), and others could be specific to handle (*e.g.* reassemble parts in specific order). There is a need to manage these situations with an error handler and thus errors, together with error handling activities, need to be described by a production process modeling language.

Besides the presented workflow, there are material, message and energy flows that need to be modeled. Ev-

ery time a smart resource has to execute an operation on a smart material, information about the material location needs to be known. In every production process step, there needs to be information about whether the material has to be acquired from a storage or the material is a result of a previous production process step. If the material has to be acquired from the storage, it will require transportation activities. Otherwise, the material is already present and the operation can be executed. In order to automatically execute code generated from models, all activities must be generated precisely, including each transport required.

A message flow represents a communication and a collaboration between smart resources. Both communication and collaboration between human workers, between a human worker and a machine, and between machines need to be specified with a production process modeling language. Thus, an integration of humans and machines in the smart factory could be specified.

An energy flow should also be considered while modeling production processes. For example, some robots must be plugged in before performing operations; some of them have a battery whose status needs to be tracked in order to charge it in time. Alternatively, other sources of energy could be used.

Except for the presented four types of flow, a production process modeling language should also support specification of process variations. Besides product variations in which similar products are different in only a few details, there are also process variations in which the same product could be created using different process steps.

All presented production process modeling concepts are needed for an abstract syntax of a DSML in order to prepare models for code generation and automatic execution. Also, a concrete DSML syntax is needed. There are two types of concrete syntaxes: a textual and a graphical syntax. However, there is no general answer which one is better [3]. The production process modeling language will be used by process designers and its models will be used by human supervisors and workers during the production. A graphical syntax is the better option in this particular case, because: (i) it enables visualized monitoring for human supervisors; (ii) it visualizes detected errors during the production; (iii) it enables graphical simulations of production process models; (iv) it creates manuals for human workers; and (v) it makes modeling easier for production process designers.

Monitoring of the production process has to be supported by the language to help human supervisors to track and control production. Production errors also need to be visualized in order to easily detect in which process steps errors have occurred. The visualization of production process simulations will help to find out which process steps have some defects and need to be optimized, or production costs could be lower by performing different changes on processes. Visualized production process models could also be used as a manual for human workers during production. Human workers will receive a production process model on their mobile devices with a manual of how to perform required activities. The graphical syntax will help process designers to ease production process modeling, but the language should also contain mechanisms to deal with the production process complexity.

Sub-processes will help deal with large production process models by hiding all of the details. However, large amount of transportation process steps and smart resources assigned to process steps can also increase complexity of models. To help process designers avoid complexities arising from transportation activities and smart resources, we propose a usage of Orchestrator and two abstraction levels of the production process modeling language.

Process designers will use the language at a higher level of abstraction to specify production process steps that include an activity and a material on which the activity will be performed. After the process steps are specified, Orchestrator will add smart resources that will execute process steps and transportation steps to support production logistics activities. These two levels of abstraction will ease modeling tasks performed by process designers. In the next section, we describe a system for automatic execution of production process models that includes the DSML with two abstraction levels and Orchestrator.

3.2 A System for Automatic Execution of Production Process Models

To enable flexible manufacturing of highly customized products at lower cost, we propose the implementation of a system for automatic execution of production process models. There are many possibilities and methodologies to choose from. The envisioned system will follow MDSD principles, centered on a DSML that is specific to a certain domain of application. In this particular case a domain of production processes modeling and execution.

A Model-Driven (MD) paradigm assumes orientation on models at all stages of system development. A complex system consists of several interrelated models organized through different levels of abstraction and platform specificity. MD as a prefix is an umbrella term to indicate, amongst others: MD Engineering (MDE), MD Software Engineering (MDSE), MD Development (MDD), MDSD and MD Architecture (MDA) [40].

The main goals of MDSD include the increase of development speed through automation and single point of system definition; the increase in software quality through formalization; the increase in component reuse and improved manageability of complexity through abstraction; greater domain expert inclusion in the development process; and better communication between different stakeholders in the software development process [4].

MDSD is usually centered on the DSML that can be seen as a specialization of a wider notion of Domain-Specific Languages (DSL) [24, 46]. The advantage of DSMLs in comparison to General Purpose Modeling Languages (GPML) is the closeness to the domain under observation and appropriateness of modeling concepts that are used for the given modeling task. By using such a language, a domain expert or a user familiar with the domain is able to specify the solution faster, with fewer errors, using more familiar concepts than is the case with GPMLs. An application of MDSD and DSMLs in industry systems' integration is given in [4].

Using the described concepts of the production process modeling language and its graphical syntax, process designers could create different models that are suitable for automatic execution. The envisioned system based on the MDSD approach, in which the language is used, is presented in Figure 2.

Process designers will use the production process modeling language at the higher level of abstraction. These models could also be obtained from product descriptions, *e.g.* Computer-Aided Design (CAD) models, using a process extractor. This process extractor will use knowledge extraction to conclude production process steps from the product description to generate the production process model. An example of process extraction from the digital product description to perform virtual assembly of the product can be found in the work of Sierla *et al.* [39].

Production process models at the higher level of abstraction will be used by Orchestrator that communicates with a knowledge base in order to add the following: (i) transportation steps in a model, (ii) smart resources to any production process step and (iii) energy flow elements, which represents energy consumption of the smart resources. The knowledge base will be used to match required skills for an activity specified in any production process step with skills of smart resources that are able to execute the specified activity.

Orchestrator will generate models at a lower level of abstraction that will be ready to be used by a code generator. Generated code needs to be human readable if instructions will be sent to mobile devices of human workers or machine readable if instructions will be sent to robots.

During the production, feedback data will be sent back to Orchestrator in order to monitor the production and detect if an error occurs to reconfigure production. Feedback data will also be gathered by an ERP and stored in a data storage in order to analyze data and find out about possible optimization, energy or time leaking.

Besides generating instructions for smart resources, the code generator also needs to be connected with a simulation. This way generated code could be tested in the simulation before the production, so the production process model could be optimized or fixed if needed.

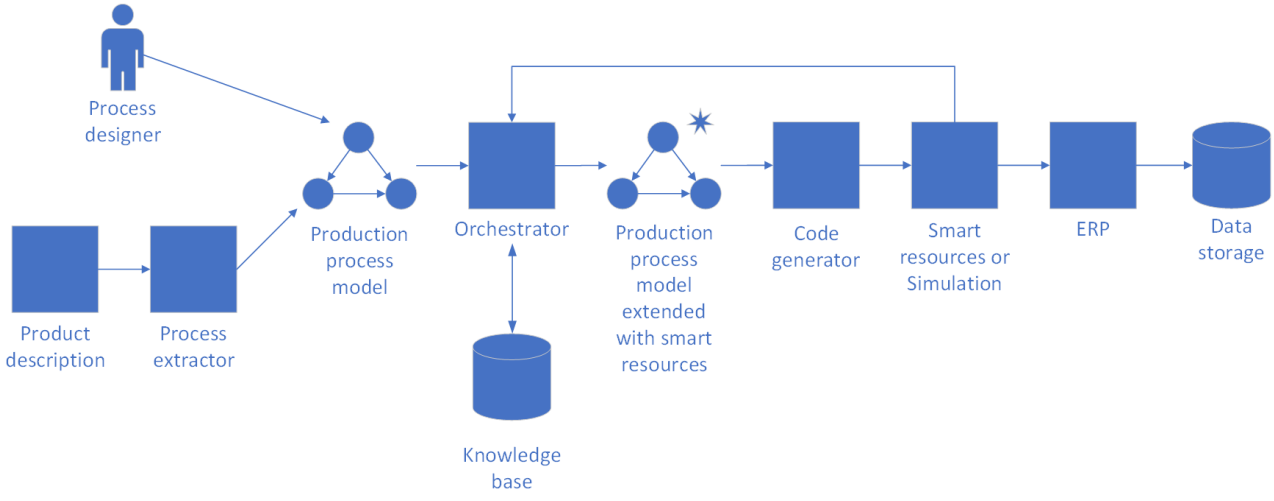

Figure 2: A system for automatic execution of production process models.

4 Modeling Production Processes with Existing Languages

In the previous section, requirements for modeling production processes in Industry 4.0 and preparing models for automatic production using a system based on an MDSD approach are identified and described. The question is whether a language exists that could fit in the envisioned system and that could describe production processes suitable for automatic execution, or whether we need yet another process modeling language as it is proposed in Section 3.1.

There are different process modeling languages that are used to specify production processes. Companies usually have process charts and Bill of Materials (BOM) specifications, but none of them can fully describe production processes that could be automatically executed. Conceptual process modeling languages like BPMN, UML and Petri nets are able to represent complex production process semantics and are extensively used in the practice. However, if production process models need to be ready for automatic code generation, aforementioned process modeling languages could be even more inadequate, as we describe in Section 5.

As one of the goals of this paper is to investigate whether it is possible to specify human-machine collaboration, and to automatically generate code from models in order to support production of highly customized products and their variations, we have checked these assumptions using three modeling languages: (i) BPMN, (ii) UML activity diagrams and (iii) Petri nets. We have used these languages to model production processes with all identified concepts needed for the automatic code generation and execution. We have analyzed whether these languages are able to support production of highly customized products or a novel DSML should be designed for this purpose.

In order to support the decision to design a novel DSML or to use one of the existing languages, in the following text a single example of a production process named *PlankSaw* is modeled by means of three aforementioned modeling languages. The example is constructed in a way that includes basic production process concepts described in the previous section. There are two planks that need to be sawed and packed as they will be later used in another process to assemble a shelf. In this particular example we only consider sawing and packaging of these two planks.

Two planks need to be sawed in parallel. We assume that, due to their dimensions, both of these planks cannot be simultaneously sawed by the same smart resource. Moreover, both planks must be inspected after they are sawed. If there is a defect in the planks, they need to be discarded, which is presented by a sub-process. Otherwise, they need to be placed in a box. A smart resource needs to hold the planks while another smart resource gets a box from a storage and places it beneath the planks. Afterwards, planks must be placed in the box and the box needs to be closed. After the box is closed, the production process is finished. In order to model production processes ready for automatic code generation, smart resources also need to be specified within the model.

4.1 A BPMN Example

In Figure 3, the production process *PlankSaw* is modeled by means of BPMN. There are three lanes that each represents a smart resource – two machines and a human worker. However, it is not possible to specify any additional property of the smart resource. Parallel process steps to pick and saw planks that are assigned to machines are specified. These two machines are chosen by Orchestrator as they were the closest to a working table.

Annotations can be used to specify input and output products and capabilities for each process step. In Figure 3 annotations are assigned only to parallel process steps to take less space and to keep the model simple as possible. In order to automatically generate code, there is a need to recognize and parse products and capabilities, especially if there are additional constraints or parameters. Using text in annotations would require the creation of a parser as the text is not part of the modeling language. Also, storage must be specified within annotations whether a product must be gathered from storage or placed in storage. Annotations have to be connected depending on whether an input product of one process step is an output product of the previous process step. Modeling a material flow is problematic using BPMN.

For example, the *Pick plank 1* process step has the input product *Plank 1* that needs to be gathered from a smart shelf, and has the output product *Picked plank 1*, which is the same as the input product as it is only picked from the smart shelf and is not changed. The *Saw plank 1* process step has the input product *Picked plank 1*, which is the same plank from the previous process step, and has the output product *Sawed plank 1*, which is sawed at a smart table.

After an inspection of planks, a selection is modeled, which is supported by BPMN, as well as a sub-process of discarding the planks. A message flow and a collaboration of smart resources can be implemented using parallelism gates and message events. These are modeled in a case when planks need to be placed in a box. Also, a notation of

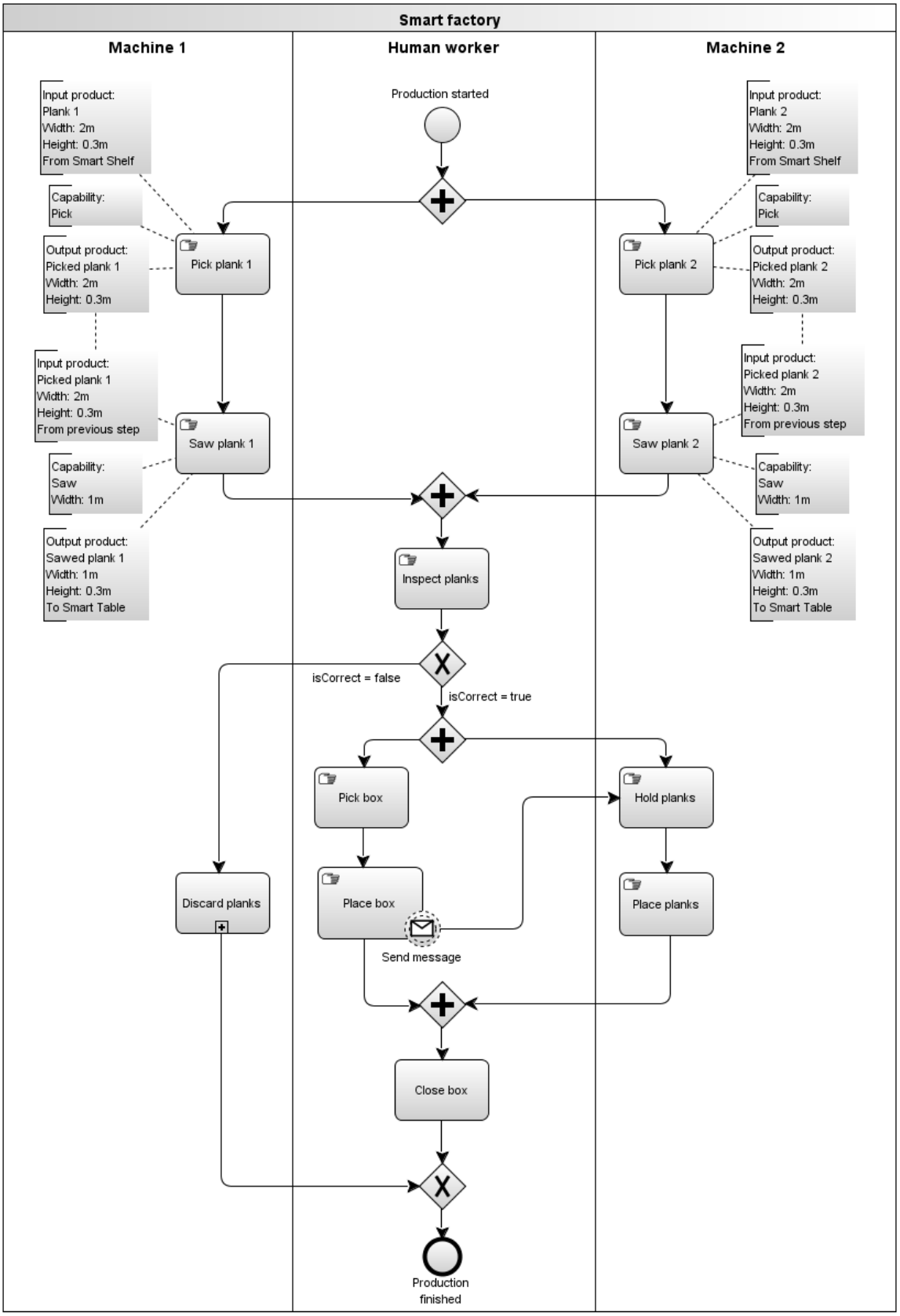

Figure 3: A BPMN production process model.

process steps, *i.e.* operation, transportation and inspection, cannot be modeled.

BPMN can be used to specify some of the basic concepts, but it lacks the semantics of production processes as it is tailored to model business processes. It is not possible to model production processes that are ready for automatic code generation and execution of the code, especially when modeling products, capability constraints and parameters and the material flow.

4.2 A UML Activity Diagram Example

The UML activity diagram of the *PlankSaw* production process is presented in Figure 4. Three lanes are again used to represent smart resources, but with this language it is not possible to model resource properties.

It is possible to specify parallel process steps in order to pick and saw two planks. A capability is presented within an activity, *i.e.* a process step, and the capability parameters

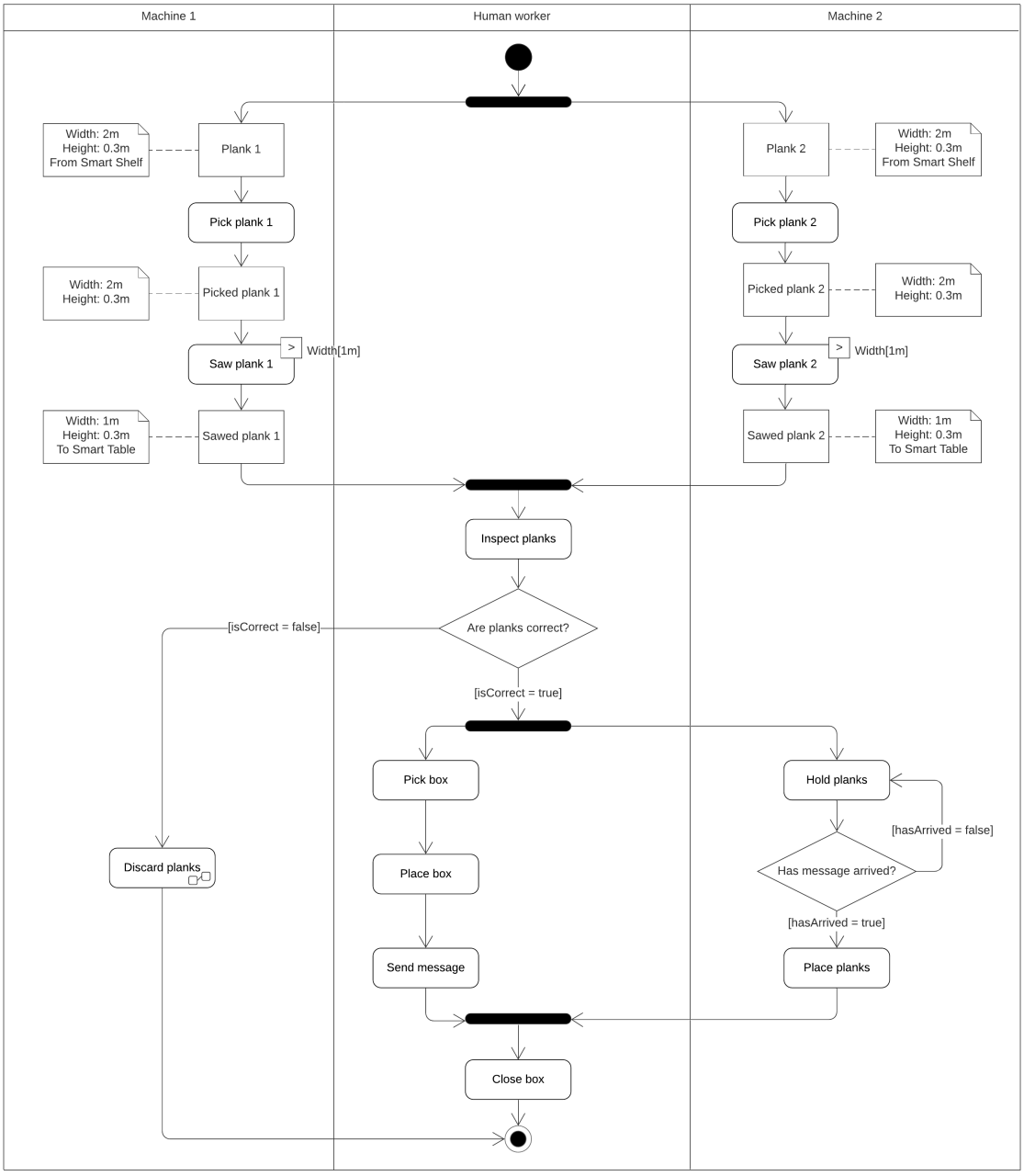

Figure 4: A UML activity diagram production process model.

could be specified within the activity as input parameters. Capability constraints must be added as annotations.

Input and output products are modeled as objects and their constraints and storages as text annotations attached to objects. Storage properties are hard to specify within annotations. Like BPMN models, it would be hard to recognize and parse text from UML activity diagram annotations in order to automatically generate code and execute it. To specify whether a product is a result of a previous process step, different objects and activities can be connected.

For example, the *Plank 1* object represents an input product of the *Pick plank 1* activity, and an output product of the same activity is the *Picked plank 1* object, which is the same as the input product. The material flow must be carefully modeled. It is important to know whether an input product is gathered from storage or it represents an output product from some of the previous process steps. The name of these two objects is not the same since the state of the plank changed – from the state in which the plank is stored in a smart shelf, to another state in which the plank is already picked. The *Picked plank 1* object is then used in the *Saw plank 1* activity as an input product, and after it is sawed, an output product of the same activity is the *Sawed plank 1* object with a shorter width than it was before the activity.

Modeling of material flows is complex, which is more obvious when using UML activity diagrams comparing to BPMN. Similar to the previous BPMN example, input and output products are only presented for parallel process steps, while others are hidden from the diagram to make the model as simple as possible.

A selection is supported by UML activity diagrams as it is presented after the inspection of planks. A sub-process of discarding planks can also be modeled as a decomposed activity. In comparison with BPMN, it is harder to model a message flow and a collaboration of different smart resources. The message flow and the collaboration can be modeled by means of a synchronization. For example, on the human worker side there is the *Send message* activity, and on the second machine side it has to be checked whether the message arrived using the selection pattern. This solution would be problematic for an automatic code generation because there is a need to continuously check whether the message has arrived. The message flow could also be modeled using signals, but their purpose is to communicate with an external participant. As it is with BPMN, process step notations cannot be modeled.

Like BPMN, UML activity diagrams can be used to specify some of the basic concepts, but they lack the semantics of production processes. Because of objects and capability parameters, UML activity diagrams provide better concepts in order to model process steps comparing with BPMN. However, by using BPMN it is possible to specify the message flow and the collaboration with greater detail as the message attributes could be specified and the message flow could be presented more clearly using relationships. If UML activity diagrams are used to model production processes, it is not possible to automatically generate executable code, especially when modeling capabilities and constraints, the material flow and the message flow, *i.e.* the collaboration.

4.3 A Petri Net Example

The model of the *PlankSaw* production process expressed by the concepts of Petri nets is presented in Figure 5. Capabilities of process steps can be specified by adding transitions in a Petri net model. Products, smart resources, storages, constraints and parameters cannot be added within the model. They can only be specified as a text that is attached to each transition. It would be very hard to parse a text like that and this text would also be very hard to read by humans. Thus, process designers would have difficulties modeling production processes.

Transitions with multiple relationships are used to specify parallelism. In the example, there are parallel process steps – to pick and saw different planks, that are modeled between transitions with multiple relationships. After an inspection of planks is specified, a selection pattern is modeled using places with multiple relationships.

As opposed to BPMN and UML activity diagrams, subprocesses cannot be modeled using Petri nets, and consequently they are modeled as simple transitions, *e.g.* the *Discards planks* process step. A message flow and a collaboration of smart resources, *e.g.* putting planks in a box, is modeled in a similar way as it was done by means of UML in Section 4.2. A transition is used to represent a process step of sending a message after the box is placed, and the selection pattern is used to check whether the message arrived to place planks in the box.

As has been previously described in the BPMN and UML examples, it can be concluded that it would be very hard to generate code and execute it based on Petri net production process models. Likewise, process step notations cannot be modeled by means of Petri nets.

Petri nets can be used to model only a few basic concepts of production processes. They are too generic to model any specific concepts of production processes. In the presented example, we can see that it is not possible to specify process steps, the material flow, the message flow and subprocesses of production process models that are ready for automatic code generation.

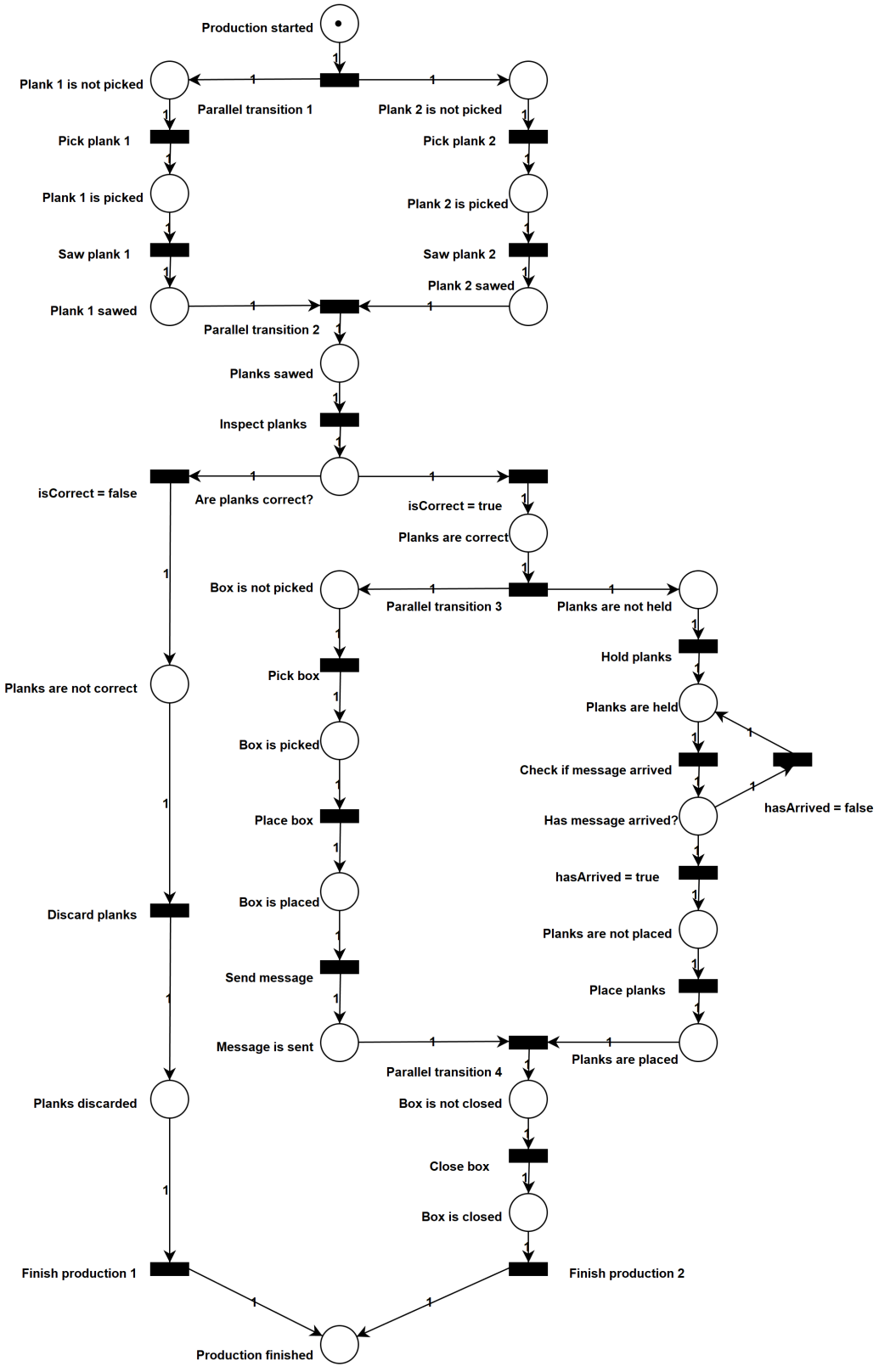

Figure 5: A Petri net production process model.

4.4 A Comparison of the Represented Languages

In the previous examples, some basic production process concepts have been modeled by means of BPMN, UML activity diagrams and Petri nets, respectively. In Table 1 we present these languages and their capabilities to present production process concepts that have been identified and presented in this paper. The numeric rating scale from 1 to 5 is used to represent the Language-Concept support level. A language L supports a concept C at level n as follows: 5 – fully, 4 – mostly, 3 – partly, 2 – weakly, and 1 – without support.

Table 1: Possibilities of languages to model production process concepts

	BPMN		UML activity diagram		Petri nets	
Process step	Partly possible	3	Mostly possible	4	Weakly possible	2
- Capability	Using activities	4	Using activities	4	Using transitions	4
- Product	Using annotations, but lack to describe properties	3	Using objects	4	-	1
- Resource	Using lanes, but lack to describe properties	3	Using lanes, but lack to describe properties	3	-	1
- Constraint	As text description in annotations	2	As text description in annotations	2	-	1
- Parameter	As text description in annotations	2	Using activity parameters	5	-	1
Workflow	+	5	+	4	+	4
- Sequence	+	5	+	5	+	5
- Selection	+	5	+	5	+	5
- Iteration	+	5	+	5	+	5
- Parallelism	+	5	+	5	+	5
- Unordered	Using ad-hoc sub-processes	5	-	1	-	1
Material flow	Weakly possible	2	Weakly possible	2	-	1
- Storage	As text description in annotations	2	As text description in annotations	2	-	1
- Product equivalent	Connecting annotations	2	Connecting objects with activities	2	-	1
Message flow / Collaboration	+	5	Using activities and selection, or using signals	3	Using transitions and selection	2
Energy flow	Using annotations, but lack to describe properties	3	Using annotations, but lack to describe properties	3	-	1
Sub-process	+	5	+	5	-	1
Process variations	Using selection	2	Using selection	2	Using selection	2
Error handling	+	4	Using exception handlers	4	Using transitions and selection	2
Quality assurance	Using activities and selection	2	Using activities and selection	2	Using transitions and selection	2
- Completion criteria	As text description in annotations	2	As text description in annotations	2	-	1
- Acceptance criteria	As text description in annotations	2	As text description in annotations	2	-	1

Using BPMN it is possible to model most of the production process concepts in comparison to UML and Petri nets. Still, it is hard to model process steps, a material flow, an energy flow, process variations and quality assurance. Also, an error handling can be modeled, but models would be hard to read. The error handling should be modeled through a different layer. To overcome the issues, BPMN extensions could be created. However, BPMN is still business oriented and it is not tailored for a production domain.

UML activity diagrams have the same deficiencies as BPMN when they are used to model production processes, but they are slightly better in specification of process steps as they can specify products and capability parameters. Additionally, UML activity diagrams lack concepts needed to model unordered process steps, *i.e.* an ad-hoc sub-process in BPMN, and a message flow. UML profiles could be created to enable modeling some missing concepts. However, even using UML profiles, it would be hard to model production processes ready for automatic code generation. We have chosen UML activity diagrams to model production processes as their purpose is to model flowcharts. UML is family of modeling languages and it may be possible that a few different languages together could describe production processes. This would additionally load process designers as they would need to know different modeling languages to fully describe production process. Thus, the complexity of production process models would grow.

Petri nets are too generic to model production processes, especially if models are needed to be specified in detail so they could be used for automatic code generation. Unlike BPMN, an advantage of Petri nets is that they have only a few basic elements to model processes. But because of this, production process models become too complex even with few process steps.

A common conclusion can be drawn for all three tested languages. These languages are not created to support modeling of production processes suitable for automatic execution. Even if human-machine collaboration could be modeled using message flows, it is hard to specify properties of both humans and machines as smart resources. Usual lack of support to model all the details of process steps and a material flow indicates that automatic code generation and execution would be hard to achieve using these languages. Thus, the production of highly customized products would be hardly supported.

Also, even if production process designers could be trained to model production process using these languages, none of these languages are conceptually familiar to them. They would need to invest a lot of time and effort to learn how to use any of these languages and their modeling tools. Difficulties to learn how to use a modeling tool that supports one of these languages is not caused by the complexity of the domain, but by the complexity of the modeling language and the tool.

All aforementioned conclusions lead us to believe that a solution to modeling production processes suitable for automatic execution would be the creation of a novel language that would be specific to the production domain.

5 Related Work

Production process modeling is an important industrial informatics research topic [53], but it is still not sufficiently covered [29]. The production processes modeling languages need to be highly used in Industry 4.0 as production processes are digitally supported, and one of the goals of Industry 4.0 is to create an integration of production processes and the cyber-physical factory [54]. It will be crucial to model production processes in order to understand, control and optimize process operations [53]. Consequently, the production cost will be lowered due to the lowering of production failures and energy costs, accompanied by the lowering of production time.

Another difficulty is automatic execution of process descriptions that would be nearly impossible in other circumstances. These are the reasons why production processes will be increasingly engineered with virtual representation that requires abstract thinking and modeling using a tool [7]. Many process modeling languages have been used to model production processes and in this section we discuss existing languages and their extensions created to support the production process modeling.

As stated in the previous section, process charts and BOM specifications are frequently used to model production processes. However, it is not possible to model all details needed for the automatic execution using process charts. Korean manufacturing process chart standard KS A 3002 [17] can be used to specify production processes, but the tooling support and the possibility to automatically execute models are missing [1]. BOM specifications are not enough to understand the production flow of production processes [1]. To model the production flow, Bill of Material and Operations (BOMO) [14] specifications have been created, but they still lack concepts to model selection and iteration patterns, as well as resource information that would be required for automatic production.

In the previous section, we also stated that conceptual process modeling languages like BPMN, UML and Petri nets have been used to model production processes. However, these languages are usually not enough to support

modeling of production processes with the created models suitable for the automatic execution.

Lütjen and Rippel [19] stated that BPMN lacks concepts to model material flow, which makes modeling production processes particularly difficult. To model production processes, BPMN extensions have been created [57], but a specification of material flow is still hard to achieve [19] and there exists an absence of uniformity [1]. Also, BPMN extensions have been created for production process similarity measurements [1], but it is not possible to model selection or iteration patterns or smart resources that would execute production process steps. Witsch and Vogel-Heuser [51] implemented a manufacturing execution system modeling language based on BPMN that included modeling of production processes. However, production tasks were not specified with all the details to enable automatic execution.

Meyer *et al.* [25, 26] proposed BPMN extensions to create IoT-aware process models. Both human workers and IoT devices are included within the language and are modeled to work on different tasks in the smart factory. Graja *et al.* [11] presented the BPMN4CPS modeling language that extends BPMN to enable modeling of CPS production processes. However, none of these languages support the execution of production process models [38]. This is the reason why Schönig *et al.* [38] extended BPMN to integrate IoT devices within production process models, but also to be able to use BPMN process models in existing business process management execution systems. Still, it is difficult to model the material flow, smart resources and products.

UML activity diagrams are used to model production processes, however models are not meant to be used for automatic execution, nor are they intuitive for process designers. Thus, they could be undesirably complex [51]. Also, as the modeling of the material flow concept is hard to achieve, a new material flow-oriented process modeling language – GRAMOSA has been created using UML profiles. Using GRAMOSA it is possible to transform factory data models into executable simulation models, but the material flow-oriented approach is complex [19].

Using the UML use case language and BPMN extensions, Petrasch and Hentschke [28] created an IoT-aware process modeling method to model IoT-aware production processes. They extended this language and created an Industry 4.0 process modeling language [29] to model all the technological details. However, none of these languages provide the details on how to execute the models [38].

Petri nets are used to model the behavior of manufacturing systems, but they do not consider technical requirements and automation [51]. They also lack the possibility to model the material flow. However, hierarchical timed-colored Petri nets have been used to model priority-based distributed production processes [22], which also included modeling of the material flow. Also, Petri nets have been used to model processes composed of assembly operations and to generate work plans [6]. Similarly, Petri nets have been used to model and analyze assembly workflows in order to improve the production scheduling [32].

6 Conclusions

An objective of this paper was to check whether it is possible to: (i) specify human-machine collaboration as it is very important in Industry 4.0 to enable flexible productions of different products and product variants, (ii) automatically generate instructions from production process models to improve production management, and (iii) automatically execute generated instructions to support production of highly customized products.

Thus, we identified and described requirements for modeling production processes in Industry 4.0 that are suitable for automatic execution. We also proposed an implementation of a system based on an MDSD approach that will support automatic code generation and execution of production process models. These models will be specified using a language that will support different abstraction levels so it could make modeling easier for process designers by hiding execution details from them.

However, to the best of our knowledge, none of existing languages are able to specify production process models with all required details for automatic execution and also to be simple enough for a human comprehension. Also, we could not find any research that presented basic concepts to model production processes suitable for automatic execution, and compared these concepts with possibilities of existing modeling languages to model them.

The described insufficiencies of existing languages to model production processes are reasons why we propose a creation of a novel DSML. The comparison presented in this paper can be seen as a pilot study for a detailed survey on a broader set of modeling languages that will be the proof of concept for the decision to create the novel DSML aimed at a formal specification of production processes that are suitable for automatic execution. Research is mostly directed towards a usage of specific modeling techniques rather than established ones [52]. The reason for that is because existing modeling languages are not tailored for a specific domain of production processes, especially if all the details are needed to be modeled.

In order to prepare process models for automatic production, the DSML needs to be specified. Based on our

claims from the previous sections, the DSML has to support: (i) modeling of all production process concepts described in Section 3.1, (ii) graphical syntax, (iii) generation of human-readable or machine-readable instructions, (iv) generation of manuals for human workers, (v) different levels of abstraction, (vi) production process monitoring, (vii) error handling, (viii) dynamic changes of models during the production, and (ix) production simulations.

The DSML will contribute in the following: (i) better and faster production process implementation, (ii) faster production process changes, (iii) better integration of human workers, human supervisors, machines and production processes, (iv) fewer production faults, and (v) lower production time and costs. A modeling tool will be implemented to support process designers and an appropriate code generator will also be implemented to generate instructions and make automatic production possible. Created production process models can be used in different simulation scenarios to test or predict the behavior of production processes. The DSML will be tested within the envisioned MDSD-based system using the simulation and on the shop floor with smart resources and smart materials.

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

References

[1] Ahn H., Chang T.-W., Measuring Similarity for Manufacturing Process Models, Moon I., Lee G. M., Park J., Kiritsis D., von Cieminski G., editors, Advances in Production Management Systems. Smart Manufacturing for Industry 4.0, Cham, 2018, Springer International Publishing, 536, 223–231, ISBN 978-3-319-99706-3 978-3-319-99707-0.

[2] Almada-Lobo F., The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES), Journal of Innovation Management, 2016, 3(4), 16–21, ISSN 2183-0606.

[3] Dejanovic I., Tumbas M., Milosavljevic G., Perisic B., Comparison of Textual and Visual Notations of DOMMLite Domain-Specific Language, Local Proceedings of the Fourteenth East-European Conference on Advances in Databases and Information Systems, Novi Sad, Serbia, 2010, 131–136.

[4] Dimitrieski V., Model-Driven Technical Space Integration Based on a Mapping Approach, Ph.D., University of Novi Sad, Faculty of Technical Sciences, Serbia, 2017.

[5] Dorofeev K., Profanter S., Cabral J., Ferreira P., Zoitl A., Agile Operational Behavior for the Control-Level Devices in Plug&Produce Production Environments, Proceedings of 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019, 49–56.

[6] Duque D. A., Prieto F. A., Hoyos J. G., Trajectory generation for robotic assembly operations using learning by demonstration, Robotics and Computer-Integrated Manufacturing, 2019, 57, 292–302, ISSN 07365845.

[7] Erol S., Jäger A., Hold P., Ott K., Sihn W., Tangible Industry 4.0: A Scenario-Based Approach to Learning for the Future of Production, Procedia CIRP, 2016, 54, 13–18.

[8] Farooqui A., Bergagard P., Falkman P., Fabian M., Error handling within highly automated automotive industry: Current practice and research needs, 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, 2016, IEEE, 1–4, ISBN 978-1-5090-1314-2.

[9] Gao Z., Cecati C., Ding S. X., A Survey of Fault Diagnosis and Fault-Tolerant Techniques-Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches, IEEE Transactions on Industrial Electronics, 2015, 62(6), 3757–3767, ISSN 0278-0046, 1557-9948.

[10] Gorecky D., Schmitt M., Loskyll M., Zuhlke D., Human-machine-interaction in the industry 4.0 era, Proceedings of 2014 12th IEEE International Conference on Industrial Informatics (INDIN), Porto Alegre RS, Brazil, 2014, IEEE, 289–294, ISBN 978-1-4799-4905-2.

[11] Graja I., Kallel S., Guermouche N., Kacem A. H., BPMN4CPS: A BPMN Extension for Modeling Cyber-Physical Systems, Proceedings of 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Paris, France, 2016, IEEE, 152–157, ISBN 978-1-5090-1663-1.

[12] Object Management Group, Business Process Model and Notation (BPMN), Version 2.0.2, 2014, Technical report.

[13] Object Management Group, Unified Modeling Language, Version 2.5.1, 2017, Technical report.

[14] Jiao J., Tseng M. M., Ma Q., Zou Y., Generic Bill-of-Materials-and-Operations for High-Variety Production Management, Concurrent Engineering, 2000, 8(4), 297–321.

[15] Keddis N., Capability-Based System-Aware Planning and Scheduling of Workflows for Adaptable Manufacturing Systems, Ph.D., Technical University of Munich, Germany, 2016.

[16] Kolberg D., Zühlke D., Lean Automation enabled by Industry 4.0 Technologies, IFAC-PapersOnLine, Ottawa, Canada, 2015, 48 of 3, 1870–1875.

[17] Korean Standards Service Network (KSSN), KS A 3002 Standard, https://www.kssn.net/en/, accessed 2020/04/05

[18] Loskyll M., Schlick J., Hodek S., Ollinger L., Gerber T., Pirvu B., Semantic service discovery and orchestration for manufacturing processes, ETFA2011, Toulouse, France, 2011, IEEE, 1–8, ISBN 978-1-4577-0017-0.

[19] Lütjen M., Rippel D., GRAMOSA framework for graphical modelling and simulation-based analysis of complex production processes, The International Journal of Advanced Manufacturing Technology, 2015, 81(1-4), 171–181, ISSN 0268-3768, 1433-3015.

[20] Lu Y., Industry 4.0: A survey on technologies, applications and open research issues, Journal of Industrial Information Integration, 2017, 6, 1–10, ISSN 2452414X.

[21] Luo Y., Zhang L., Tao F., Ren L., Liu Y., Zhang Z., A modeling and description method of multidimensional information for manufacturing capability in cloud manufacturing system, The International Journal of Advanced Manufacturing Technology, 2013, 69 (5-8), 961–975, ISSN 0268-3768, 1433-3015.

[22] Lv Y. Q., Lee C. K. M., Wu Z., Chan H. K., Ip W. H., Priority-Based Distributed Manufacturing Process Modeling via Hierarchical Timed Color Petri Net, IEEE Transactions on Industrial Informatics, 2013, 9(4), 1836–1846, ISSN 1551-3203, 1941-0050.

[23] Makris S., Michalos G., Chryssolouris G., RFID driven robotic assembly for random mix manufacturing, Robotics and

Computer-Integrated Manufacturing, 2012, 28(3), 359–365, ISSN 07365845.

[24] Mernik M., Heering J., Sloane A. M., When and how to develop domain-specific languages, ACM Computing Surveys, 2005, 37 (4), 316–344, ISSN 03600300.

[25] Meyer S., Ruppen A., Magerkurth C., Internet of Things-Aware Process Modeling: Integrating IoT Devices as Business Process Resources, Advanced Information Systems Engineering. CAiSE 2013. Lecture Notes in Computer Science, Valencia, Spain, 2013, Springer International Publishing, 7908, 84–98, ISBN 978-3-319-98176-5 978-3-319-98177-2, Series Title: Notes on Numerical Fluid Mechanics and Multidisciplinary Design.

[26] Meyer S., Ruppen A., Hilty L., The Things of the Internet of Things in BPMN, Advanced Information Systems Engineering Workshops. CAiSE 2015. Lecture Notes in Business Information Processing, Stockholm, Sweden, 2015, Springer International Publishing, 215, 285–297, ISBN 978-3-319-19242-0 978-3-319-19243-7, Series Title: Lecture Notes in Business Information Processing.

[27] Neugebauer R., Hippmann S., Leis M., Landherr M., Industrie 4.0 - From the Perspective of Applied Research, Procedia CIRP, 2016, 57, 2–7.

[28] Petrasch R., Hentschke R., Towards an Internet-of-Things-aware Process Modeling Method - An Example for a House Suveillance System Process Model, Proceedings of 2nd Management and Innovation Technology International Conference (MITiCON2015), Bangkok, Thailand, 2015, Information Technology Management, Faculty of Engineering, Mahidol University, 168–172.

[29] Petrasch R., Hentschke R., Process modeling for industry 4.0 applications: Towards an industry 4.0 process modeling language and method, Proceedings of 2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, Thailand, 2016, IEEE, 1–5, ISBN 978-1-5090-2033-1.

[30] Petri C. A., Kommunikationen mit Automaten, Ph.D., University of Bonn, Germany, 1962.

[31] Pisarić M., Dimitrieski V., Babić M., Veselinović S., Dušić F., Towards a Plug-and-Play Architecture in Industry 4.0, Proceedings of 17th International Scientific Conference on Industrial Systems (IS'17), Novi Sad, Serbia, 2017, 136–141.

[32] Qian C., Zhang Y., Jiang C., Pan S., Rong Y., A real-time data-driven collaborative mechanism in fixed-position assembly systems for smart manufacturing, Robotics and Computer-Integrated Manufacturing, 2020, 61, 101841, ISSN 07365845.

[33] Qu T., Lei S. P., Wang Z. Z., Nie D. X., Chen X., Huang G. Q., IoT-based real-time production logistics synchronization system under smart cloud manufacturing, The International Journal of Advanced Manufacturing Technology, 2016, 84(1-4), 147–164, ISSN 0268-3768, 1433-3015.

[34] Raptis T. P., Passarella A., Conti M., Data Management in Industry 4.0: State of the Art and Open Challenges, IEEE Access, 2019, 7, 97052–97093.

[35] Roblek V., Meško M., Krapež A., A Complex View of Industry 4.0, SAGE Open, 2016, 6(2), 1–11, ISSN 2158-2440, 2158-2440.

[36] Rodič B., Industry 4.0 and the New Simulation Modelling Paradigm, Organizacija, 2017, 50(3), 193–207, ISSN 1581-1832.

[37] Sanchez B. B., Alcarria R., Sanchez-de Rivera D., Sanchez-Picot A., Enhancing Process Control in Industry 4.0 Scenarios using Cyber-Physical Systems, JoWUA, 2016, 7, 41–64.

[38] Schönig S., Ackermann L., Jablonski S., Ermer A., IoT meets BPM: a bidirectional communication architecture for IoT-aware process execution, Software and Systems Modeling, 2020, 19, 1443–1459, ISSN 1619-1366, 1619-1374.

[39] Sierla S., Kyrki V., Aarnio P., Vyatkin V., Automatic assembly planning based on digital product descriptions, Computers in Industry, 2018, 97, 34–46, ISSN 01663615.

[40] Stahl T., Voelter M., Model-Driven Software Development: Technology, Engineering, Management, 2006, John Wiley and Sons, Ltd., Chichester, England, 1st edition, ISBN 0-470-02570-0.

[41] Stock T., Seliger G., Opportunities of Sustainable Manufacturing in Industry 4.0, Procedia CIRP, Ho Chi Minh City / Binh Duong, Vietnam, 2016, 40, 536–541.

[42] Svingerova M., Melichar M., Evaluation of Process Risks in Industry 4.0 Environment, Katalinic B., editor, DAAAM Proceedings, DAAAM International Vienna, 2017, 1, 1021–1029, ISBN 978-3-902734-11-2.

[43] Thoben K.-D., Wiesner S., Wuest T., "Industrie 4.0" and Smart Manufacturing – A Review of Research Issues and Application Examples, International Journal of Automation Technology, 2017, 11(1), 4–16, ISSN 1883-8022, 1881-7629.

[44] Trstenjak M., Cosic P., Process Planning in Industry 4.0 Environment, Procedia Manufacturing, Modena, Italy, 2017, 11, 1744–1750.

[45] Vaidya S., Ambad P., Bhosle S., Industry 4.0 – A Glimpse, Procedia Manufacturing, Maharashtra, India, 2018, 20, 233–238.

[46] van Deursen A., Klint P., Visser J., Domain-specific languages: an annotated bibliography, ACM SIGPLAN Notices, 2000, 35(6), 26–36, ISSN 03621340.

[47] Vještica M., Dimitrieski V., Pisarić M., Kordić S., Ristić S., Luković I., Towards a formal description and automatic execution of production processes, Proceedings of 2019 IEEE 15th International Scientific Conference on Informatics, Poprad, Slovakia, 2019, IEEE, 463–468, ISBN 978-1-7281-3180-1.

[48] Wagner T., Herrmann C., Thiede S., Industry 4.0 Impacts on Lean Production Systems, Procedia CIRP, Taichung City, Taiwan, 2017, 63, 125–131.

[49] Wan J., Cai H., Zhou K., Industrie 4.0: Enabling Technologies, Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things, Harbin, 2015, 135–140.

[50] Wang S., Wan J., Li D., Zhang C., Implementing Smart Factory of Industrie 4.0: An Outlook, International Journal of Distributed Sensor Networks, 2016, 12(1), 1–10, ISSN 1550-1477, 1550-1477.

[51] Witsch M., Vogel-Heuser B., Towards a Formal Specification Framework for Manufacturing Execution Systems, IEEE Transactions on Industrial Informatics, 2012, 8(2), 311–320, ISSN 1551-3203, 1941-0050.

[52] Wortmann A., Barais O., Combemale B., Wimmer M., Modeling Languages in Industry 4.0: An Extended Systematic Mapping Study, Software and Systems Modeling, 2020, 19, 67–94, ISSN 1619-1366, 1619-1374.

[53] Xu L. D., Enterprise Systems: State-of-the-Art and Future Trends, IEEE Transactions on Industrial Informatics, 2011, 7(4), 630–640, ISSN 1551-3203, 1941-0050.

[54] Xu L. D., Xu E. L., Li L., Industry 4.0: state of the art and future trends, International Journal of Production Research, 2018, 56 (8), 2941–2962, ISSN 0020-7543, 1366-588X.

[55] Zhang Y., Qian C., Lv J., Liu Y., Agent and Cyber-Physical System Based Self-Organizing and Self-Adaptive Intelligent Shopfloor, IEEE Transactions on Industrial Informatics, 2017, 13(2), 737–747, ISSN 1551-3203, 1941-0050.

[56] Zhong R. Y., Xu X., Klotz E., Newman S. T., Intelligent Manufacturing in the Context of Industry 4.0: A Review, Engineering, 2017, 3(5), 616–630, ISSN 20958099.

[57] Zor S., Schumm D., Leymann F., A Proposal of BPMN Extensions for the Manufacturing Domain, Proceedings of the 44th CIRP International Conference on Manufacturing Systems, Madison, Wisconsin, USA, 2011, 1–7.