

An Application of a DSML in Industry 4.0 Production Processes

Marko Vještica^{1[0000-0003-2368-5818]}, Vladimir Dimitrieski^{1[0000-0003-3234-6543]}, Milan Pisa-rić^{2[0000-0002-0764-4453]}, Slavica Kordić^{1[0000-0002-3992-0096]}, Sonja Ristić^{1[0000-0059-6062]} and Ivan Luković^{1[0000-0003-1319-488X]}

¹ University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia
{marko.vjestica, dimitrieski, slavica, sdristic, ivan}@uns.ac.rs

² Industrial Automation, KEBA AG, Linz, Austria
pisa@keba.com

Abstract. One of the goals of Industry 4.0 is to enable mass customization of products and to satisfy specific needs of customers. This goal is often hard to achieve in traditional manufacturing systems. To enable fast production changes, an automatic and flexible production is needed. In this context we propose a Model-Driven Software Development (MDSD) approach and a Domain-Specific Modeling Language (DSML) to model production processes. The language supports two levels of abstraction. A Master-Level (ML) model is used by a process designer to model process steps. A Detail-Level (DL) model is used by Orchestrator, a cluster of industrial computers that manages production, to fill existing ML models with a specification of production logistic and smart resources. A code generator is used to generate machine-readable or human-readable instructions from DL models. Generated code is used for automatic execution of production processes within a simulation or a shop floor. In this paper we provide an application of a DSML, which is capable of modeling production processes that are ready for automatic code generation.

Keywords: Production Process Modeling, Domain-Specific Modeling Languages, Model-Driven Software Engineering, Industry 4.0, Process Execution.

1 Introduction

Industry 4.0 aims to establish a flexible production in the traditional manufacturing [1], moving from mass production to mass customization [2]. Introducing variability to the production process in traditional manufacturing systems requires either stopping the production to reconfigure machines or having multiple production lines for every product variant. To achieve aimed production flexibility, production process variability need to be enabled during manufacturing, without a need to stop the production [3]. Thus, a customized product's price could be lower, as one of the key aspects of the mass customization is to provide products with prices that are near to mass production prices [4]. In order to establish production flexibility at the lower cost and to enable automatic production, we proposed a Model-Driven Software Development (MDSD) approach to

model and automate the execution of production processes [5]. The overview of the approach is given in Fig. 1 and described in short below.

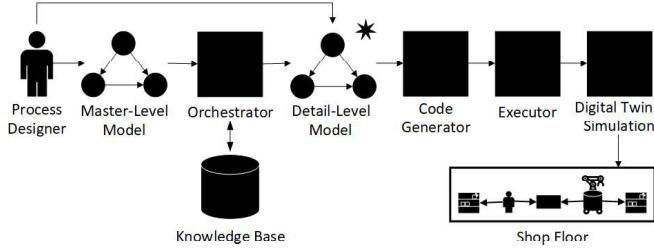


Fig. 1. The proposed MDSD approach

In the presented MDSD approach, a process designer uses a Domain-Specific Modeling Language (DSML) to specify production processes. This production process specification is a technical description of a production process, which we call a Master-Level (ML) model. It includes specification of: (i) process steps, (ii) required capabilities, i.e. skills to execute the step, (iii) input and output products, i.e. transformed resources like raw materials, components or finished goods, (iv) constraints and (v) capability parameters. In order to use such a production process model for automatic code generation and execution, additional information needs to be added. This information includes: (i) specific smart resources, e.g. storage, machines, robots and humans, which execute process steps, (ii) logistic information for product and resource movement and (iii) configuration tasks of smart resources, e.g. software setup, changing grippers, plugging into a charger or a workstation. This enrichment is automatically performed by Orchestrator and we denote the enriched process model as a Detail-Level (DL) model. It is up to process designers to either mark the process model for execution or perform manual interventions and optimization where they deem necessary.

Orchestrator is a software run on top of a cluster of industrial computers that is able to orchestrate smart resources and assign them to process steps for their execution [6]. It also needs to detect and configure new smart resources or reconfigure existing ones [7]. Each process step is specified with a required capability, a skill required to execute the process step. On the other hand, each resource offers a set of capabilities, i.e. skills it is able to perform. Using a knowledge base and the required and offered capabilities, Orchestrator is able to identify smart resources which are able to execute specific process steps. It is also able to identify storages that contain required products and process steps that need to be changed or added, e.g. transportation or machine configuration process steps, in order to execute the production process with the available resources. A DL model is used by the code generator to automatically generate instructions that are to be performed by humans or machines. The executor sends generated instructions to the digital twin that comprises both the simulation and the command proxies to the shop floor. In our case, the digital twin can be used as the simulation only or it can forward instruction to shop floor smart resources via the built-in proxies.

As a DSML is an integral part of the proposed MDSD approach which aims to improve flexibility and automation of a factory, and as no appropriate DSML has been

found for modeling all the aspects of a production process, we decided to create a new one which fits this need. In this paper we present basic concepts of the proposed DSML that is used to model production processes. In this paper an application of this DSML in an assembly use case is presented. The DSML is used by domain experts to model production processes using familiar, domain concepts. This should result in faster, easier and comprehensive modeling of production processes which are suitable for automatic execution that yields less faults. Using the DSML within our MDSD approach should also improve the manufacturing flexibility and the level of overall automation of production.

Apart from Introduction, this paper is structured as follows. An overview of the related work is presented in Section 2. Basic concepts of the proposed DSML for production process modeling and an assembly production use case are described in Section 3. Conclusions and future work are given in Section 4.

2 Related Work

Modeling production processes in Industry 4.0 is an important industrial informatics research topic [8], as production processes are digitally supported, and they need to be integrated with Cyber-Physical Production Systems [9].

There are many process modeling languages, but most of them are neither tailored for production processes, nor ready to model execution-ready production processes. There is the manufacturing process chart standard named KS A 3002 [10], however it lacks a tooling support for modeling and automatic execution [11]. Companies often utilize process charts or Bill of Materials (BOM) specifications, but none of them can fully describe production processes that could be automatically executed. BOM specifications are not sufficient to understand a production flow [11]. Bill of Materials and Operations (BOMO) specifications include the production flow, but cannot specify the selection and iteration patterns or smart resources.

Conceptual process modeling languages like Business Process Modeling and Notation (BPMN), Unified Modeling Language (UML) Activity Diagram, Petri Nets and Event-Driven Process Chains usually cannot support the material flow concept, which makes modeling production processes very difficult. To model production processes, BPMN extensions have been created [12], but a depiction of material flow is still hard to achieve [13] and there was an absence of uniformity [11]. Also, BPMN extensions were created for production process similarity measurements [11], but it is not possible to model selection or iteration patterns or smart resources. Using Systems Modeling Language (SysML) or Petri Nets to model production processes usually lack to model complex production processes or the material flow. To overcome usual lack of the material flow concept, a new material flow-oriented process modeling language – GRAMOSA has been created using UML profiles, but the material flow-oriented approach was complex [13]. Some languages lack production logistic specifications like DELMIA Process Engineer (DPE), while others are limited to simple linear process sequences like Value Stream Design (VSD) [13]. We have identified the lack of a modeling language capable to specify all relevant aspects of a production process in the

context of its formal definition suitable for automatic execution. That is the main reason why we have decided to create a new DSML instead of applying an existing one in our MDSD approach. The main goal of introducing such a DSML is to improve the production process flexibility and to enable formal specification of production processes that will allow automatic generation of program code aimed at automation of process execution.

3 Application of the DSML in an Assembly Use Case

In our previous work [5], we discussed a DSML usage for modeling production processes in Industry 4.0. We also identified concepts that must be supported by a DSML for production processes modeling that will enable automatic code generation and execution from such models. In this section, we present a use case to demonstrate the application of the developed DSML. For each DSML concept we provide an example of its use in the presented use case. The language is created by using Ecore meta-metamodel, while the graphical syntax and the modeling tool are created by using Eclipse Sirius. Concepts of the graphical syntax are presented on the left side of Fig. 2.

The presented use case describes an assembly process of a custom LEGO flag. The process is modeled by our DSML and the model is then passed for code generation which is executed both in the simulation and on the shop floor. Bricks of various colors are stored in a smart shelf while a human worker and an industrial mobile robot pick different bricks in parallel and assemble the flag on a brick pedestal at an assembly table. A human worker is using a tablet or a smart watch in order to receive instructions and to send feedback when an activity is finished.

Our DSML supports modeling at two levels of abstraction in order to make the modeling easier for process designers. They are creating higher-level abstraction models, i.e. ML models. An ML model can be extended with additional specifications of execution details. In that way a DL model, at a lower level of abstraction, is created. Execution details can be added to an ML model manually or automatically by means of Orchestrator, as it is illustrated in Fig. 1. Based on these two abstraction levels, we have classified the language elements into two groups: (i) elements that are needed to model ML production processes and (ii) elements that are needed to be added into existing ML models to create DL models. The ML model of our LEGO use case is presented in the central part of Fig. 2, while the DL model is presented on the right side of Fig. 2.

At the higher level of abstraction, process designers model process steps that denote either an operation or an inspection activity. A process step can include (i) an input product, that can be e.g. raw material or component, and that can be picked from a storage or is a result of the previous process step, (ii) a capability, i.e. skill required to execute the step and (iii) an output product, which is a result of executed capability on the input product and is either placed in a storage or transferred to the next process step. Every product and capability can have constraints, e.g. width, height, color, which are considered by Orchestrator when it decides which resources are able to perform a process step. Some capabilities require parameters to be specified e.g. position must be specified in order to place a brick on a brick pedestal. Also, a material flow can be

specified by defining if a product needs to be picked from a storage, placed to a storage, or it is a result of the previous process step. As specified process steps need to be connected using relationships – links between process steps, the language has a concept to model a workflow. Process steps can be connected so as to form a sequence or a set of parallel workflow branches. Additionally, selection and iteration patterns can be also added to the workflow. For this purpose, we are using a gate concept – a modeling concept that is used to connect multiple workflow branches. The language also supports a message flow, i.e. collaboration of resources. This concept is used if two or more process steps need to be executed in parallel, but one process step must not finish its activity or start the next one before it gets a message that another process step finished its activity. Using described concepts, a process designer can be focused only on process steps that must be executed and need not to worry about production logistic and resources that will execute the process steps.

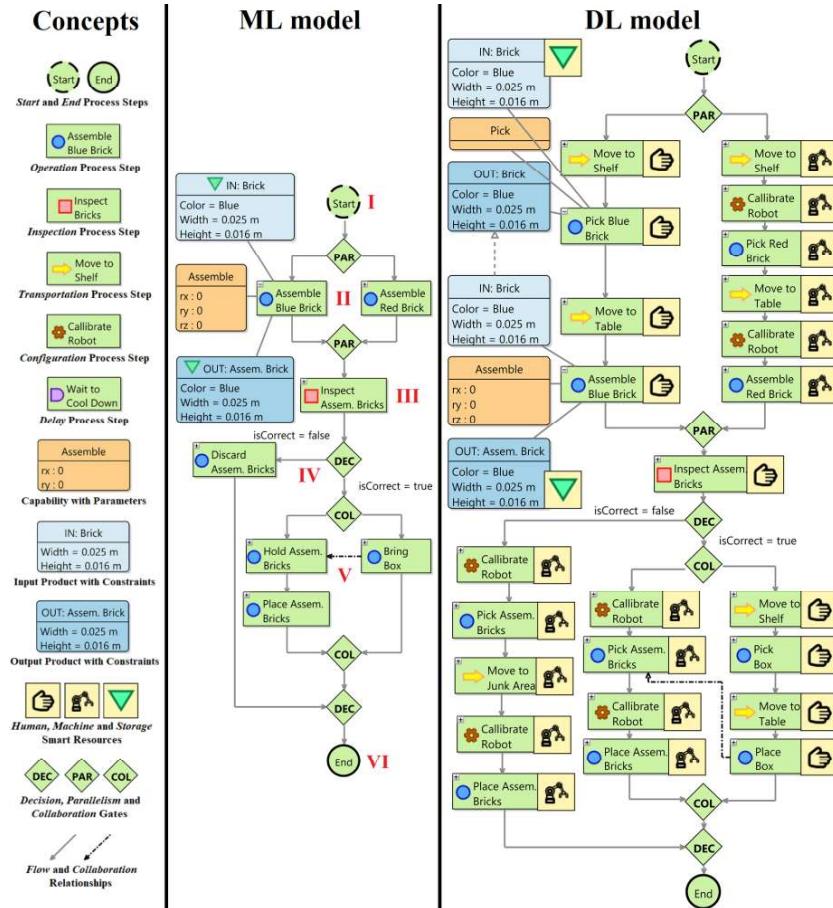


Fig. 2. A production process model example

In our LEGO use case, the ML model has: (i) the start process step, (ii) parallelism gates (*PAR*) and between them parallel assembly process steps, (iii) the inspection process step, (iv) decision gates (*DEC*), which have two branches that leads to product discard or packaging, (v) collaboration gates (*COL*) with packaging process steps and (vi) the end process step. An assembly process step is an operation process step, which is denoted by a circle on the left side of a process step name. Flow Process Chart (FPC) is broadly used to specify the production process flow and process designers are familiar with its graphical and symbolic representation. That is why we have decided to take over some common symbols from FPC for corresponding concepts in the DSML, like circle symbol for an operation process step. An operation process step needs to have input and output products and a capability. Products and a capability are graphically represented by rectangles of different color connected to a process step. They can be hidden from a diagram using a +/- button at the top left corner of a process step, so a process model could be more or less complex depending on the designer needs. Due to length limitations, these detailed specifications are depicted just for one process step in the central part of Fig. 2, while for the rest they are specified, but not shown. In the depicted ML model, the input product can be seen – a blue brick that is gathered from a storage. A storage is presented by a triangle icon, same as in FPC, on the left side of a product name, representing that a product must be picked from a storage. The assembly capability needs to be executed on the input product, and the output product is the assembled blue brick on the brick pedestal, basically the partially assembled flag with this brick in it. After assembly process steps, the inspection process step is needed, which notation is presented by a rectangle icon, same as in FPC. If assembled bricks have not passed the inspection, they are discarded. Otherwise, packaging of assembled bricks is required. This should be done by doing two activities in parallel. One activity is to hold assembled bricks and another one is to bring a box beneath them. The first activity should not be finished before the message arrives that the box is brought under assembled bricks. This message flow is presented by a dashed arrow between these two process steps. After the message arrives, assembled bricks need to be placed in the box.

At the lower level of abstraction, all concepts from the higher level of abstraction exist, but the language also has additional concepts like resources, specific storages and new process step representations. Orchestrator uses these concepts to fill existing ML models with production logistics and smart resources. At this level, process steps can also represent activities like transportation, configuration, i.e. calibration of machines, or delay, i.e. necessary waiting. A specific storage must be defined for every input product that must be gathered from a storage and for every output product that must be placed in a storage. Process steps additionally have a resource that will execute a capability on input products. A resource can be a human worker or a machine. This is important information especially for code generation. For every process step human-readable or machine-readable instructions will be generated, depending on the provided information. Using all presented concepts, production process models are ready for code generation and consequently for an execution.

In the LEGO use case, we present in detail only one “pick” and one “assemble” process step of the DL model. Other process steps are expanded in a similar way. One

assembly process step is assigned to a human worker, and another is assigned to a mobile robot. Both human worker and mobile robot must execute additional transportation process steps in order to pick a brick from the smart shelf and place it on the assembly table. In contrast to the human worker, the mobile robot in this use case needs to execute additional configuration process steps after transportation. The mobile robot is not equipped with the machine vision modules and therefore it must be calibrated after each movement to determine its position. Transportation and configuration process step notations are presented by an arrow, same as in FPC, and a gear wheel icon respectively. These process steps only include a capability, without products. The input product of the “pick” process step has the specific storage, i.e. the smart shelf, added by Orchestrator. Its output product is the same picked product. The input product of the “assemble” process step is the output product from the “pick” process step, which means it does not have to be gathered from a storage. Its output product is the brick, which will be assembled at the specific storage, i.e. the assembly table.

Using the DL model, it is possible to generate instructions to both mobile robot and human worker. The executor sends instructions to the mobile robot or the human worker and waits for their response. Once a resource completes an activity, the next one is sent. The process is finished after the execution reaches the end process step.

4 Conclusions and Future Work

In this paper we presented an application of the developed DSML for modeling production processes in an assembly use case and described its basic concepts. This language is used as one of the main elements of the MDSD approach that aims to improve the flexibility and the automation level of production. The DSML can be used to (i) make faster and more precise process designs, (ii) make less faults during process design, (iii) enable faster changes of production process models in the era of Industry 4.0 and (iv) model a human-machine interaction. Also, the DSML models are used by Orchestrator to manage the production, as it is important to plan process activities and integrate processes within the industrial system [14].

The language has been evaluated by process designers on the shop floor within a small-scale industrial production setup. We plan to further evaluate the language in additional industrial use cases and also by independent researches and process designers in order to improve the domain concept coverage and the tooling stability. Also, we plan to further investigate related research and provide a systematic literature review on production process modeling and execution.

Although a lot of concepts identified in [5] were already implemented in the presented DSML, there are additional concepts that can be added in order to improve the language domain coverage, like: (i) unordered sets of process steps, (ii) quality assurance with completion and acceptance criteria, (iii) error handling flows, (iv) process variations and (v) sub-processes. Also, we plan to implement a new modeling tool feature to monitor execution of every process step and thus enable detection of delays or badly modeled process steps. Finally, ML and DL models could be generated from existing product specification formats, like Computer Aided Design (CAD) models.

Acknowledgment

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

References

1. Lu, Y.: Industry 4.0: A survey on technologies, applications and open research issues. *J. Ind. Inf. Integr.* 6, 1–10 (2017). <https://doi.org/10.1016/j.jii.2017.04.005>.
2. Crnjac, M., Veža, I., Banduka, N.: From Concept to the Introduction of Industry 4.0. *Int. J. Ind. Eng. Manag.* 8, 21–30 (2017).
3. Dorojević, K., Profanter, S., Cabral, J., Ferreira, P., Zoitl, A.: Agile Operational Behavior for the Control-Level Devices in Plug&Produce Production Environments. In: Proceedings of 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). pp. 49–56. IEEE, Zaragoza, Spain (2019).
4. Zhao, H., McLoughlin, L., Adzhiev, V., Pasko, A.: “Why do we not buy mass customised products?” – An investigation of consumer purchase intention of mass customised products. *Int. J. Ind. Eng. Manag.* 10, 181–190 (2019). <https://doi.org/10.24867/IJIEM-2019-2-238>.
5. Vještica, M., Dimitrijević, V., Pisarić, M., Kordić, S., Ristić, S., Luković, I.: Towards a formal description and automatic execution of production processes. In: Proceedings of 2019 IEEE 15th International Scientific Conference on Informatics. pp. 463–468. IEEE, Poprad, Slovakia (2019). <https://doi.org/10.1109/Informatics47936.2019.9119314>.
6. Keddis, N.: Capability-Based System-Aware Planning and Scheduling of Workflows for Adaptable Manufacturing Systems, (2016).
7. Pisarić, M., Dimitrijević, V., Babić, M., Veselinović, S., Dušić, F.: Towards a Plug-and-Play Architecture in Industry 4.0. In: Proceedings of 17th International Scientific Conference on Industrial Systems (IS’17). pp. 136–141., Novi Sad, Serbia (2017).
8. Xu, L.D.: Enterprise Systems: State-of-the-Art and Future Trends. *IEEE Trans. Ind. Inform.* 7, 630–640 (2011). <https://doi.org/10.1109/TII.2011.2167156>.
9. Xu, L.D., Xu, E.L., Li, L.: Industry 4.0: state of the art and future trends. *Int. J. Prod. Res.* 56, 2941–2962 (2018). <https://doi.org/10.1080/00207543.2018.1444806>.
10. Korean Standards Service Network (KSSN): KS A 3002 Standard, <https://www.kssn.net/en/>, last accessed 2020/04/05.
11. Ahn, H., Chang, T.-W.: Measuring Similarity for Manufacturing Process Models. In: Moon, I., Lee, G.M., Park, J., Kiritis, D., and von Cieminski, G. (eds.) *Advances in Production Management Systems. Smart Manufacturing for Industry 4.0.* pp. 223–231. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-99707-0_28.
12. Zor, S., Schumm, D., Leymann, F.: A Proposal of BPMN Extensions for the Manufacturing Domain. In: Proceedings of the 44th CIRP International Conference on Manufacturing Systems. pp. 1–7., Madison, Wisconsin, USA (2011).
13. Lütjen, M., Rippel, D.: GRAMOSA framework for graphical modelling and simulation-based analysis of complex production processes. *Int. J. Adv. Manuf. Technol.* 81, 171–181 (2015). <https://doi.org/10.1007/s00170-015-7037-y>.
14. Stevanov, B., Gračanin, D., Kesić, I., Ristić, S.: An Application of Period Batch Control Principles and Computational Independent Models for Supporting the Overhaul Process of the Railway Braking Devices. *Int. J. Ind. Eng. Manag.* 4, 95–101 (2013).