M. Riebisch, M. Tropmann-Frick (Hrsg.): Modellierung 2022,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2022 181

A Cross-Disciplinary Process Modelling Language for Validating Reconfigured Production Processes[1]

Sandro Koch[2], Tim Wunderlich[3], Jonas Hansert[4], Thomas Schlegel[4], Steffen Ihlenfeldt[3], Robert Heinrich[2]

Abstract: Modelling and reconfiguration of production processes require knowledge across different domains. This in-depth knowledge is necessary to avoid possible side effects that could threaten the production plant, the workpiece or the worker. Therefore, process modelling approaches allow adding additional data to the steps of a process. Such additions can be constraints, which need to be fulfilled before a step can be executed. Upon reconfiguration of production processes, these constraints need to be validated to ensure that the objective of the process is still met. However, this task demands expertise in the field of process modelling as well as in the domain of the production process and the production plant. To the best of our knowledge, state-of-the-art production process modelling approaches are unable to determine the semantic validity of a reconfigured production process. In this paper, we introduce a domain-specific modelling language dedicated to model and validate constraints between production steps. With this approach, we aim to assist the operator in reconfiguring production processes. We evaluate this approach in three case studies and show that our approach can detect violated constraints in production processes.

Keywords: domain-specific modelling language; process modelling; process validation; resilience; cyber-physical production systems

1 Introduction

The increasing prevalence and availability of Internet of Things (IoT) technology like sensors, actuators, nodes, and tags create a much higher and more dynamic integration into the physical world and new challenges for future software systems. The interconnection of IoT components increases the complexity of such systems and the potential for errors and failures grows, as well. Hence, also the risk of errors and failures increases. Due to the complexity of IoT systems and the human interaction with such systems, not all errors can be predicted. IoT are frequently confronted with unknown input and output values due to the constantly changing and uncontrollable execution context of the system. As a result, processes have to deal with highly volatile resources and states at runtime, which can fail at any time and result in the need for reconfiguration of process. Reconfiguring a failed process is complex, because the partial state of the system until the failure needs to be considered.

[1] This work was partially funded by the German Federal Ministry of Education and Research under grant 01IS18067A/B/D (RESPOND), and the KASTEL institutional funding.

[2] Karlsruhe Institute of Technology, Am Fasanengarten 5, 76131 Karlsruhe, GER, firstname.lastname@kit.edu
[3] Fraunhofer IWU, Reichenhainer Str. 88, 09126 Chemnitz, GER, firstname.lastname@iwu.fraunhofer.de
[4] Karlsruhe University of Applied Sciences, Moltkestr. 30, 76133 Karlsruhe, GER, firstname.lastname@h-ka.de

doi:10.18420/modellierung2022-004

In this paper, we present a *Domain-Specific Modelling Language* (DSML) that enables people who are not experts on the production system but are experts on the production process to model and modify production processes in an IoT system. In the development of the DSML, besides computer scientists, experts of mechanical and electrical engineering and experts in the field of modelling production processes were involved. The DSML allows the domain expert to model sub-processes with conditions that must be fulfilled. These conditions can be pre-conditions that must be fulfilled before a sub-process can be executed as well as post-conditions that are fulfilled after a sub-process has been executed. In case the conditions are not met, the automated verification detects such errors. As a result, non-domain experts can use the DSML given in this paper to model production processes in an IoT system.

Our contributions in this paper are: 1. A DSML that allows to model production processes in an IoT system. In contrast to general-purpose process modelling approaches, our approach focuses on the conditions and constraints of production processes. 2. A verification process that allows to validate reconfigured production processes. More details regarding the state-of-the-art is provided in Sect. 2.

The remainder of the paper is structured as follows: State-of-the-art of process verification is presented in Sect. 2. The requirements for the DSML are discussed in Sect. 3. The DSML is presented in Sect. 4. In Sect. 5, we evaluate the DSML by modelling three production processes based on two real world production lines and on one laboratory case study. The paper concludes in Sect. 6.

2 State of the Art

Business Process Model and Notation 2 (BPMN2) is a historically grown, monolithic DSML for modelling and simulating business processes. Due to the strong coupling of the entities, Business Process Model and Notation (BPMN) is hard to maintain [Ro17] and to extend [HSR19]. However, the BPMN2 specification and other process modelling approaches like DECLARE [PSA07] do not support the validation of pre- and post-conditions of process models and extending these approaches would increase their complexity. With increasing complexity of process models, the need for automatic and reliable testing of these models increases, as well. This arises from the necessity to follow regulations in certain fields, e. g., patient care [Ly12] or pharmaceutical quality assurance [Al03], but also from the need for validating rescheduled manufacturing processes as a result of risk mitigation in Cyper-physical Production Systems (CPPS) [Ih21].

Validating process models during the modelling process can help to find failures and improve the quality [Kü10]. Previous research focuses on annotating business processes with semantic information regarding the constraints between certain process steps [WGH08; WHM08] and considers general business processes such as sales orders. Furthermore, declaring constraints between process steps has been examined as a vehicle to synthesise emergency

response workflows on the basis of predefined templates [ML17] and to dynamically generate and adapt process models describing software engineering workflows [GOR10a; GOR10b; GOR11]. Processes can also be represented as a combination of Unified Modeling Language (UML) and Object Constraint Language (OCL). This can be converted into logical expressions, which in turn can be used for validation [EST15]. Furthermore, Petri nets were used to allow verification of Event-driven Process Chain (EPC) models [Va99], but EPC does not provide a concise syntax for modelling constraints between process steps. Instead, linking multiple events to a process step requires the combination with logical connectors. It is also possible to model production processes using imperative languages such as Prolog, RuleML [BPS10] or Stanford Research Institute Problem Solver (STRIPS) [FN71]. The process structure is modelled as a set of facts and rules and the Prolog or RuleML engine is then used to validate the modelled process. In STRIPS, a set of states and actions with pre- and post-conditions can be defined. Due to the imperative nature of these languages, they are harder to comprehend than our approach.

While the risk of modelling processes that are not valid are reduced by these approaches, domain knowledge is still required to model these processes. In addition, the applicability of these approaches has not been empirically examined in the context of CPPS, yet. Finally, UML and OCL are General-purpose programming languages (GPPLs) and therefore significantly more difficult than a dedicated DSML for modelling and evaluating production processes.

3 Requirement Elicitation

We identified the following requirements that the language needs to comply with: Certain processes cannot be executed when certain conditions are not fulfilled at the beginning of the process. Thus, the first requirement **R1** for the language is to specify the preconditions of a process. After a processes is executed, the state of the plant or the workpiece can be changed, thus we derive **R2** that the language must be able to also model the postconditions of a process. It is also possible that after a process is executed, previously specified conditions are no longer relevant, thus we derive **R3** that the language must be able to remove conditions. A process can also explicitly exclude a condition, thus we derive **R4** that the language must be able to exclude conditions. Besides the syntactic requirements, we also require the language to verify whether a modelled process meets the specified condition, thus we derive **R5** that the language must be able to verify whether a process is valid.

4 A Language for Validating Reconfigured Production Processes

4.1 DSML Implementation

In this section, we introduce the language engineering framework used in our approach. Furthermore, we will explain key parts of the grammar of our language.

Xtext[5] is a framework for language engineering that allows to design and implement editing tools for textual *Domain-Specific Language*s (DSLs). The DSL is based on a grammar, like the Extended Backus–Naur Form (EBNF). The Xtext framework generates plug-ins for the Eclipse[6] *Integrated Development Environment* (IDE) without any additional effort. We chose Xtext, as it is integrated with the *Eclipse Modelling Framework* (EMF)[7], which is a modelling framework for code generation.

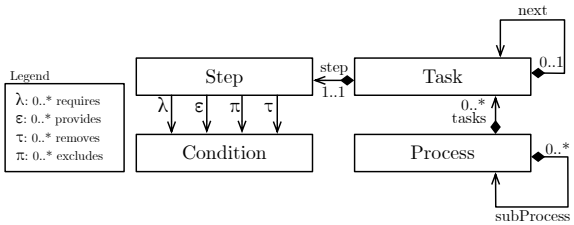

Fig. 1: Reduced illustration of the Process Model Validation Language (PMVL) syntax.

Fig. 1 shows a reduced model of the PMVL. A process consists of a sequence of further (sub)processes or tasks. We decided to introduce subprocesses to enable modularisation of processes and to allow reducing the size of a process. Thus, it is possible to reuse already existing processes and to reduce code duplication. The tasks in a process represent a sequence of steps that are executed in a certain order. One task of a process consists of a step. A step can have conditions that are either required (λ) and must be fulfilled, so that the step can be executed, or a step can provide (ϵ) conditions that are fulfilled after a step is executed. A step can also remove (τ) conditions that were fulfilled until the step was executed. Finally, a step can also exclude (π) conditions that would prevent the execution of the task.

The PMVL is divided in the declaration of processes, tasks, steps, and states. Conditions are part of steps and cannot be declared separately. Due to space limitations, the textual syntax of a process it described in more detail in our supplementary material [Ko22].

4.2 Production Process Validation

Besides the language definition, the implementation of validation steps is also supported by the Xtext framework. We distinguish between the following kinds of validation: 1) Inconsistent preconditions: Due to the backward references, the algorithm traverses through all steps leading to the current step. By traversing through these steps, contradicting preconditions can be identified and the corresponding steps are highlighted. 2) Inconsistent preconditions: Due to the forward references, the algorithm traverses through all steps following the current

[5] https://www.eclipse.org/Xtext/
[6] https://www.eclipse.org/eclipse/
[7] https://www.eclipse.org/modeling/emf

step. By traversing through these steps, contradicting postconditions can be identified and the corresponding steps are highlighted. 3) Branch inconsistency: If processes are determined to run in parallel, the algorithm checks contradicting conditions of the steps in a branch. If errors are found, the corresponding steps are highlighted. 4) Conditions already met: If a step does provide the same set or a sub-set of postconditions as the preceding step, the algorithm identifies the previous step via the backward reference. If the conditions are already met by a preceding step, the redundant step is highlighted. 5) Cycle detection: Via a depth-first search utilising the forward and backward references, the algorithm checks whether the process reaches the last step. If the depth-first search does not reach the last step in the processes, the corresponding processes that form a cycle are highlighted.

5 Evaluation

This section presents the evaluation of the PMVL based on three case studies. The evaluation design is explained in Sect. 5.1. The case studies are presented in Sect. 5.2. The evaluation results are presented in Sect. 5.3 and discussed in Sect. 5.4.

5.1 Evaluation Design

The evaluation of the PMVL follows the Goal Question Metric (GQM) approach [Ba94]. The goal is to analyse the PMVL to evaluate the accuracy of the PMVL instances with respect to precision and completeness of the identified errors from the point of view of the production plant domain experts in the context of reconfiguring and validating production processes.

To evaluate the accuracy of the PMVL instances, we compare a list of the identified warnings and errors to a reference list. The list of errors has been created using PMVL. The reference list has been created manually by the best available experts in developing and maintaining the respective case study production plant and its production processes. For the filling line case study, we asked the co-authors from the Karlsruhe University of Applied Sciences. For the assembly line case study, we asked the co-authors from the Fraunhofer Institute for Machine Tools and Forming Technology IWU. For the production line case study, we asked the plant experts from SITEC Industrietechnologie GmbH. Where the co-authors fulfill also the role of the domain experts who provide a reference list, we consider the list as correct and complete.

For evaluating the accuracy of the PMVL, we ask the research question: **How is the accuracy of our approach?** To answer this question, we use the metric $\boldsymbol{F_1}$ **score**. F_1 is a harmonic mean of precision and recall, by aggregating the amount of true positives, false positives, and false negatives. The amount of true positives, false positives, and false negatives is calculated by comparing the list of identified errors with the reference list. An error derived

by PMVL is considered as a true positive, if the error is also marked in the reference list (i. e., there is an error in the reference list for the same line as in the derived list). An error derived by PMVL is considered as a false positive, if the error is not marked in the reference list (i. e., there is no error in the reference list for an error found by PMVL). An error not derived by PMVL is considered a false negative, if the error is marked in the reference list (i. e., there is an error in the reference list with no corresponding error in the derived list).

After calculating the numbers for true positives (t_p), false positives (f_p), and false negatives (f_n), precision and recall are calculated as follows: $precision = \frac{t_p}{t_p + f_p}$ and $recall = \frac{t_p}{t_p + f_n}$. F_1 score is calculated as the harmonic mean of precision and recall: $F_1 = 2\,\frac{precision \times recall}{precision + recall}$.

5.2 Case Studies

This section gives an overview of the three case studies used to evaluate PMVL.

Filling Line: For evaluation and demonstration purposes, the filling line was set up in the Karlsruhe University of Applied Sciences laboratory. It has nine pump stations and a carriage that transports a cup to them. The carriage and the pump stations are considered separate systems. There is a pump and a scale to weigh the bottle from which the liquid is pumped. A stepper motor moves the carriage horizontally and a scale holds the cup. The Workflow Management System (WfMS) sends Message Queuing Telemetry Transport (MQTT) messages to control the filling line. The pumps' positions must be measured in motor steps before the carriage can be utilised. The carriage is moved to the start point, then two liquids are filled at distinct stations, and finally the carriage is moved to the end place.

Assembly Line: The second case study comprises a six-station assembly line connected by a conveyor belt. The assembly line handles an aluminium piece that will be cut by a mill and dot peened by a dot peen marking machine. The dot peen marking machine engraves an Data Matrix code (DMC), allowing traceability of the manufacturing process. The workpiece also features a hole for a steel ball to be pressed into. The hole is measured either mechanically by inserting a cone or visually using a camera. A successful manufacturing process requires four criteria: 1. The engraving is cut into the workpiece. 2. The DMC is engraved. 3. A steel ball is forced into the workpiece's hole. 4. The finished object is put into the tray. While the engravings are independent of the other procedures, the steel ball may only be pressed in once the hole diameter has been confirmed. Moreover, the steps can be performed in any order as long as the skid on the conveyor belt, the mill, and the dot peen marking machine can fit a workpiece that has already been joined with the steel ball. As a result, certain production steps can be rescheduled to make the process more resilient while still meeting the criteria mentioned above.

Electrochemical Machining Line: The Electrochemical Machining (ECM) case study is based on a production line of our industrial partner SITEC Industrietechnologie GmbH. In

terms of non-disclosure agreements, details about the exact functionality of the stations are changed, but the overall process stays the same. The ECM line handles one workpiece at a time in three stations. The workpiece is sealed at the start of the process. At the line's entry point, there must be at least one workpiece. A conveyor belt transports the workpiece to the first station to remove the sealing. The workpiece is then conveyed to the second station. Station 2 takes workpieces from the conveyor belts and performs ECM. The workpiece is then conveyed to the third station to re-seal the workpiece and transports it out. To finish the production process, the workpiece must be machined and resealed.

5.3 Evaluation Results

In Tab. 1 and Tab. 2, we present the evaluation results for each process model. The evaluation data can be found in our supplementary material [Ko22]. In Tab. 1, the numbers of true positives (t_p), false negatives (f_n), and false positives (f_p) are shown for each plant. The table includes precision, recall, and the F_1 metric. The table separates the results for inconsistent preconditions, inconsistent postconditions, conditions already met, branch inconsistency, and cycle detection. The results of using PMVL are compared to the experts' estimation.

	Filling Station					**Assembly Line**					**Sealing Line**				
	Pr.	**Po.**	**Met.**	**Br.**	**Cy.**	**Pr.**	**Po.**	**Met.**	**Br.**	**Cy.**	**Pr.**	**Po.**	**Met.**	**Br.**	**Cy.**
t_p	4	5	3	1	2	4	2	3	1	1	4	4	3	2	1
f_n	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
f_p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tab. 1: Results for the individual case studies (Pr.: preconditions, Post.: postconditions, Met.: conditions already met, Br.: branch, Cy.: Cycle).

The results for the filling line case study show that all pre- and postcondition errors, already-met-condition-errors, branching-errors, and cycling-errors identified by the domain experts were also correctly identified by PMVL (t_p). No additional errors were identified by PMVL (f_p). None of the errors from the experts' list were missing (f_n).

The results for the assembly line case study show that all pre- and postcondition errors, already-met-condition-errors, branching-errors, and cycling-errors identified by the domain experts were also correctly identified by PMVL (t_p). No additional errors were identified by PMVL (f_p). None of the errors from the experts' list were missing (f_n).

The results for the ECM line case study show that all postcondition errors, already-met-condition-errors, and cycling-errors identified by the domain experts were also correctly identified by PMVL (t_p). No additional errors were identified by PMVL (f_p). Two of the errors the experts identified were missing (f_n). First, one precondition error required a time constraint, which the PMVL is currently not able to model, and one branching error was masked by a second branching error. When the second branching error was fixed, the second

error was identified by PMVL, as well. Although the first error could be identified after a fix, we still consider it as a f_n.

	Prec.	**Postc.**	**Met con.**	**Branch**	**Cycle**	**Total**
t_p	12	11	9	4	4	**40**
f_n	1	0	0	1	0	**2**
f_p	0	0	0	0	0	**0**
Precision	1.0	1.0	1.0	1.0	1.0	**1.0**
Recall	0.92	1.0	1.0	0.8	1.0	**0.95**
F_1	0.96	1.0	1.0	0.89	1.0	**0.98**

Tab. 2: Evaluation results for the case studies combined (Prec.: preconditions, Post.: postconditions, Met con.: conditions already met).

In Tab. 2, the precision, recall, and F-measure are depicted. The results show that PMVL does only identify errors that are also present in the reference list (Precision). When modelling and validating production processes, we do not highlight problems that are not true errors, so we do not mislead the user. However, we are unable to identify all of the errors in the reference list (Recall).

5.4 Discussion and Limitations

The results of the evaluation show that all postcondition errors, already-met-condition-errors, and cycle errors have been identified correctly by PMVL. The missing precondition error could not be modelled with PMVL. We have to extend the language in order to be able to add runtime constraints and thereby model such errors. The missing branching error could be identified after fixing the error, masking the missing error. The user of PMVL must be aware that after fixing the identified errors, more errors could occur. Overall, PMVL shows promising results regarding validating reconfigured production processes. Regarding the requirement **R5**, we come to the conclusion that in general, PMVL is able to model and validate reconfigured production processes, but needs to be extended to model and validate time-constrained production processes.

6 Conclusion and Future Work

In this paper, we introduced PMVL, a language for the modelling and validation of production processes in the context of IoT and CPPS. It allows domain experts to specify sub-processes with conditions that can be used by non-domain experts. The design process of PMVL is driven by the input of industry partners with the need to model and validate production processes. The evaluation of PMVL demonstrates that the DSML can be used to specify real-word production processes and identify errors such as inconsistent pre- and postconditions, branching errors or cycles. Future work includes the application of PMVL to further disciplines and augmenting PMVL with additional features like modelling production processes with time constraints and to include more complex production processes.

References

[Al03] Aleem, H.; Zhao, Y.; Lord, S.; McCarthy, T.; Sharratt, P.: Pharmaceutical process validation: an overview. Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering 217/2, pp. 141–151, 2003.

[Ba94] Basili, V. R. et al.: The Goal Question Metric Approach. In: Encyclopedia of Software Engineering. Wiley, 2:528–532, 1994.

[BPS10] Boley, H.; Paschke, A.; Shafiq, O.: RuleML 1.0: the overarching specification of web rules. In: International Workshop on Rules and Rule Markup Languages for the Semantic Web. Springer, pp. 162–178, 2010.

[EST15] Estañol, M.; Sancho, M.-R.; Teniente, E.: Verification and Validation of UML Artifact-Centric Business Process Models. In (Zdravkovic, J.; Kirikova, M.; Johannesson, P., eds.): Advanced Information Systems Engineering. Springer International Publishing, Cham, pp. 434–449, 2015, ISBN: 978-3-319-19069-3.

[FN71] Fikes, R. E.; Nilsson, N. J.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial intelligence 2/3-4, pp. 189–208, 1971.

[GOR10a] Grambow, G.; Oberhauser, R.; Reichert, M.: Employing semantically driven adaptation for amalgamating software quality assurance with process management. In: Second International Conference on Adaptive and Self-Adaptive Systems and Applications 2010 (ADAPTIVE 2010). Pp. 58–67, 2010.

[GOR10b] Grambow, G.; Oberhauser, R.; Reichert, M.: Semantic workflow adaption in support of workflow diversity. In: The Fourth International Conference on Advances in Semantic Processing (SEMAPRO 2010). Pp. 158–165, 2010.

[GOR11] Grambow, G.; Oberhauser, R.; Reichert, M.: Semantically-driven workflow generation using declarative modeling for processes in software engineering. In: 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops. IEEE, pp. 164–173, 2011.

[HSR19] Heinrich, R.; Strittmatter, M.; Reussner, R. H.: A Layered Reference Architecture for Metamodels to Tailor Quality Modeling and Analysis. IEEE Transactions on Software Engineering/, 2019, ISSN: 0098-5589.

[Ih21] Ihlenfeldt, S.; Wunderlich, T.; Süße, M.; Hellmich, A.; Schenke, C.-C.; Wenzel, K.; Mater, S.: Increasing Resilience of Production Systems by Integrated Design. Applied Sciences 11/18, p. 8457, 2021.

[Ko22] Koch, S.: A Cross-Disciplinary Process Modelling Language for Validating Reconfigured Production Processes - Supplementary Material, version v1.0, Mar. 2022, URL: https://doi.org/10.5281/zenodo.6334307.

[Kü10] Kühne, S.; Kern, H.; Gruhn, V.; Laue, R.: Business process modeling with continuous validation. Journal of Software Maintenance and Evolution: Research and Practice 22/6-7, pp. 547–566, 2010.

[Ly12] Ly, L. T.; Rinderle-Ma, S.; Göser, K.; Dadam, P.: On enabling integrated process compliance with semantic constraints in process management systems. Information Systems Frontiers 14/2, pp. 195–219, 2012.

[ML17] Marrella, A.; Lespérance, Y.: A planning approach to the automated synthesis of template-based process models. Service Oriented Computing and Applications 11/4, pp. 367–392, 2017.

[PSA07] Pesic, M.; Schonenberg, H.; van der Aalst, W. M.: DECLARE: Full Support for Loosely-Structured Processes. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007). Pp. 287–287, 2007.

[Ro17] Rostami, K.; Heinrich, R.; Busch, A.; Reussner, R.: Architecture-based Change Impact Analysis in Information Systems and Business Processes. In: 2017 IEEE International Conference on Software Architecture (ICSA2017). IEEE, pp. 179–188, 2017, ISBN: 978-1-5090-5729-0, URL: https://doi.org/10.1109/ICSA.2017.17.

[Va99] Van der Aalst, W. M.: Formalization and verification of event-driven process chains. Information and Software technology 41/10, pp. 639–650, 1999.

[WGH08] Weber, I.; Governatori, G.; Hoffmann, J.: Approximate compliance checking for annotated process models. In: 1st International Workshop on Governance, Risk and Compliance-Applications in Information Systems (GRCIS'08). 2008.

[WHM08] Weber, I.; Hoffmann, J.; Mendling, J.: Semantic business process validation. In: Proceedings of the 3rd International Workshop on Semantic Business Process Management (SBPM'08), CEUR-WS Proceedings. Vol. 472, Citeseer, 2008.