

Towards a Model-Based DevOps for Cyber-Physical Systems

Benoit Combemale, Manuel Wimmer

▶ **To cite this version:**

Benoit Combemale, Manuel Wimmer. Towards a Model-Based DevOps for Cyber-Physical Systems. Software Engineering Aspects of Continuous Development, Springer-Verlag, pp.1-11, 2019. hal-02407886

HAL Id: hal-02407886
https://inria.hal.science/hal-02407886

Submitted on 12 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Towards a Model-Based DevOps for Cyber-Physical Systems

Benoit Combemale[1] and Manuel Wimmer[2]

[1] Univ. Toulouse & Inria, France
benoit.combemale@irisa.fr
[2] Johannes Kepler University Linz & CDL-MINT, Austria
manuel.wimmer@jku.at

Abstract. The emerging field of Cyber-Physical Systems (CPS) calls for new scenarios of the use of models. In particular, CPS require to support both the integration of physical and cyber parts in innovative complex systems or production chains, together with the management of the data gathered from the environment to drive dynamic reconfiguration at runtime or finding improved designs. In such a context, the engineering of CPS must rely on models to uniformly reason about various heterogeneous concerns all along the system life cycle. In the last decades, the use of models has been intensively investigated both at design time for driving the development of complex systems, and at runtime as a reasoning layer to support deployment, monitoring and runtime adaptations. However, the approaches remain mostly independent. With the advent of DevOps principles, the engineering of CPS would benefit from supporting a smooth continuum of models from design to runtime, and vice versa. In this vision paper, we introduce a vision for supporting model-based DevOps practices, and we infer the corresponding research roadmap for the modeling community to address this vision by discussing a CPS demonstrator.

1 Introduction

We are currently facing a dramatically increasing complexity in the development and operation of systems with the emergence of *Cyber-Physical Systems (CPS)* [9]. This demands for more comprehensive and systematic views on all aspects of systems (e.g., mechanics, electronics, software, and network) not only in the engineering process, but in the operation process as well [2]. Moreover, flexible approaches are needed to adapt the systems' behavior to ever-changing requirements and tasks, unexpected conditions, as well as structural transformations [6].

To engineer interdisciplinary systems such as CPS, *modeling* is considered as the universal technique to understand and simplify reality through abstraction, and thus, *models* are used throughout interdisciplinary activities within engineering processes which is often referred to Model-Based Systems Engineering (MBSE) [3]. However, in order to deal with current requirements such as the

flexible adaption of CPS to changing requirements, the operation processes of CPS as well as their interplay with the engineering processes and vice versa has to be taken into consideration. This raises the question how model-based DevOps practices for CPS can be achieved. Such practices are currently highly needed to reduce the time between identifying the necessity for a change and putting the appropriate change into production.

This paper discusses a vision for model-based DevOps practices for CPS (Section 2) as well as the challenges which have to be tackled in order to realize this vision by the help of a CPS demonstrator (Section 3). Finally, we conclude with an outlook on several research lines which may build on the discussed model-based DevOps practices, in the form of a short term research roadmap (Section 4) and long term perspectives (Section 5).

2 Overall Vision

While current DevOps principles apply to code integration, deployment, delivery and maintenance in the software industry, we envision the application of the very same principles at the model level for the development of CPS. In such a vision, the various domain-specific development models are seamlessly integrated with operations, either via models at runtime (e.g., model-based MAPE-K loop or digital twins) or via a combination of software and hardware components within a given environment.

Initially introduced for the design phases in software development, model-driven engineering (MDE) approaches cover nowadays the entire life cycle. First extended to support the elicitation of the requirements and the expected use cases, models have been then intensively used for automating the development and analysis of complex systems, and more recently to support dynamic reconfigurations of dynamically adaptable systems. As illustrated in Fig. 1, various tool-supported approaches have been explored and developed to cover all these phases. Most of these approaches are nowadays largely used in industry and help engineers to increase the quality and productivity of modern developments [11].

MDE approaches appear particularly useful in the context of systems engineering (a.k.a. MBSE), for the development of complex software-intensive systems, also referred as cyber-physical systems. Such systems involve heterogeneous concerns (including hardware and software). Models provide a uniform level of abstraction to reason over the entire system and support automation for analysis, development, monitoring and evolution.

While most of the added value in CPS comes from the software services built on top of the physical architecture (e.g., IoT, smart systems, flexible production systems, etc.), they face the same evolution than any other software services, including the restricted time to market to meet the final user expectations. Integrating the various approaches and ensuring a model continuum is thus the next level for the adoption of model-based approaches, supporting continuous delivery, monitoring and improvements of the systems.

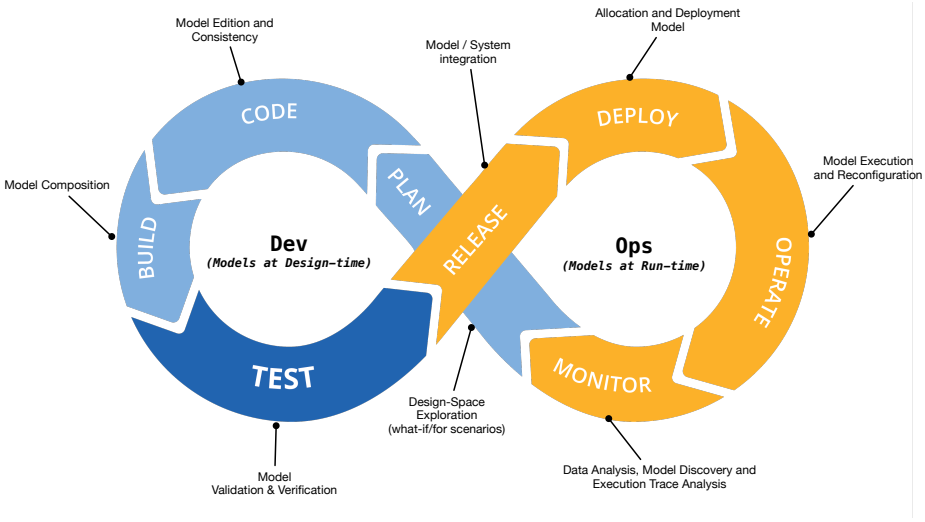


Fig. 1. A Model-Based DevOps Approach

Also, CPS are usually deployed in complex, ever changing, environments. Software services provide the intrinsic required adaptability, that must be validated with regards to heterogeneous hardware. DevOps principles bring monitoring at the first glance, and automate the continuous improvement while guaranteeing this is free of regression.

While it appears obvious that DevOps principles would be beneficial to the development of modern CPS, this requires to promote such principles at the model level. We review in the rest of this paper the challenges raised by such a vision, on the basis of a concrete CPS demonstrator introduced in the following section.

3 A CPS Demonstrator calling for Model-Based DevOps Practices

In this section, we discuss a CPS demonstrator developed at the Christian Doppler Laboratory for Model-Integrated Smart Production (CDL-MINT).[3] The CPS demonstrator is based on automating, operating, and maintaining a 6-axis robot with the notion of digital twins (cf. Fig. 2) providing different viewpoints such as logical and physical views as well as runtime data–called digital shadow [12]. In the following, we explain the main components of this demonstrator–being software and hardware parts.

[3] `https://cdl-mint.se.jku.at`

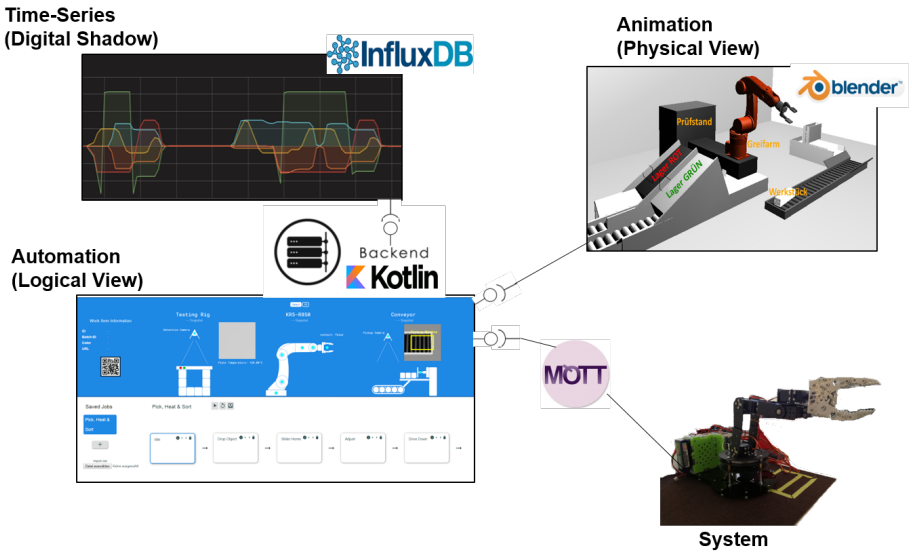


Fig. 2. A CPS Demonstrator Project.

For realizing, operating, and maintaining the gripper robot, we follow a model-based methodology which is in line with Fig. 1. The gripper is modeled by using the Systems Modeling Language (SysML). In particular, we employ the block definition diagram and the state machine diagram.

The block definition diagram is used to define the structure of the gripper including its properties. For instance, we model the BasePosition (BP), MainArmPosition (MAP), and GripperPosition (GP) (cf. upper part of Fig. 3) to name just a few properties. These mentioned properties describe the angle positions of the axis of the gripper. These angle positions are set for different realizing different actions (e.g., for driving down, moving left/right, or for picking-up).

The intended behavior of the gripper is described by a state machine, i.e., by detailing the various states and state transitions (cf. middle part of Fig. 3). These states are for instance *DriveDown* and *PickUp*. The states set the variable values specifying the respective angle position to realize in these states.

During operation, the gripper acts as a continuous system and thus, moves in its environment on the basis of a workflow described by the state machine. The particular movements are recorded by axis sensors and returned as continuous sensor value streams. In our excerpt of the system, we show three sensor value streams for the three properties defined in the block definition diagram (cf. lower part of Fig. 3).

The building blocks for the development process as well as for the operation and maintenance processes of the discussed CPS demonstrator are as follows:

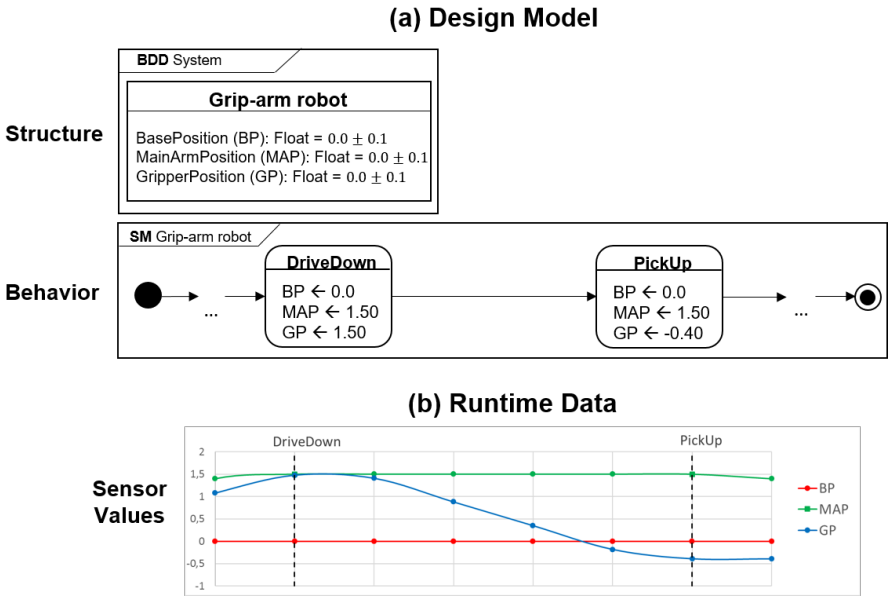


Fig. 3. Design View and Runtime View of the CPS Demonstrator (Excerpt).

- **Logical modeling languages** to define the intended logical structure and behaviour of the system (provided by the SysML language).
- **3D CAD modeling language** for representing the geometry and kinematics of the physical components of the gripper.
- **Code generator** to produce the necessary control code from the logical models for the particular controller platform.
- **Controller**, an extended version of a Raspberry Pi, with connections to the sensors and actuators of the physical device. For the gripper, every axis has a dedicated actuator and sensor which are connected to the controller via GPIO (general-purpose input/output) pins.
- **Physical device**, i.e., the gripper, with sensors and actuators.

In order to move from a classical model-based engineering to a model-based DevOps process as outlined in Fig. 1, dedicated extensions to the previously described setting are required in order to realize an efficient and effective usage of models. In particular, design models have been employed for development in many different settings and domains and allowed for automating the code generation process. In addition, several work also proposed to use runtime models during the operation of a system. However, the transition from development to operation and vice versa has been mostly overlooked. In the following, we shortly summarize the need to take these transitions into account.

- **Moving from Dev-to-Ops**: While most MDE tooling allow for moving from the model level to the code level, further activities in the direction of operations are often not explicitly supported. However, for settings such as described for the CPS demonstrator, we need further support to test the controller combined with the system to automatize before moving to the actual system level as well as to automatically deploy the control code on particular platforms.
- **Moving from Ops-to-Dev**: In addition to the use of runtime models to perform self-configuration and optimization within a particular design, monitoring is important to understand the actual operation of a system to explore new designs. This means, in addition to prescriptive and predictive runtime models, we also need descriptive runtime models which can be linked back to the design to reason about possible re-designs, model improvements such as providing a higher precision or energy minimisation. In summary, we need a way to link the data streams from the systems to our design models to close the loop.

In the following section, we detail these two research dimensions by the concrete challenges we are facing and outline some directions to take.

4 Research Roadmap

In this section, we present our research roadmap (cf. Fig. 4) by discussing a set of important challenges which have to be tackled to realize the aforementioned vision (cf. Fig. 1). We categorize the challenges in two kinds: (*i*) we present the challenges to continuously move from model-based development to operations, and (*ii*) the challenges to continuously move from operations to model-based development. We ground the challenges by giving concrete links to the CPS demonstrator introduced in the previous section.

4.1 From Dev to Ops

C1: Integration of MDE techniques: In the past decade, a plethora of different techniques for validation, verification, evolution, transformation of models have been proposed. However, how these techniques may be bundled into a pipeline for continuously integrating, building, testing, and deploying models into production environments is less explored. The only exception is the work by García and Cabot [4] who married continuous deployment technologies and model-driven technologies.

For the CPS demonstrator, model changes have to trigger code generation scripts, test case generators, deployment scripts for running the code on a virtual representation of the physical gripper, i.e., its digital twin, in the simulation platform. As soon as this virtual level is certified, the code has to be deployed in the production environment to test it on the real physical device. For this process, we require a pipeline which can connect modeling tools, simulation tools,

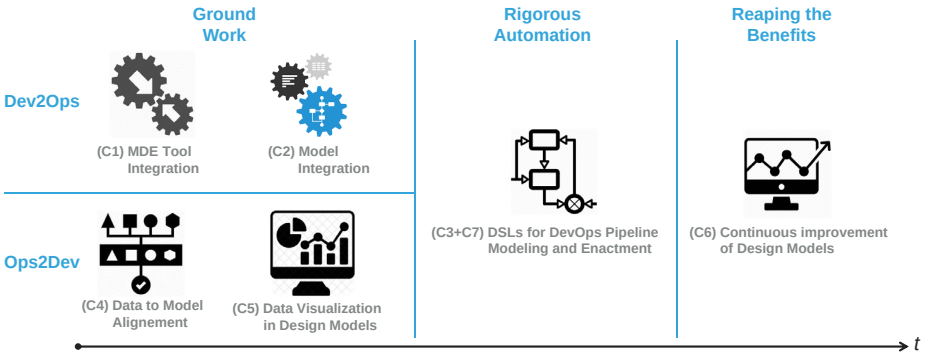


Fig. 4. Summary of the Research Challenges to Support a Model-Based DevOps for Cyber-Physical Systems.

code generators, testing tools, as well as continuous deployment tools. In the best case, these pipeline should allow for incremental techniques to save computation costs and to guarantee an instant re-deployment.

C2: Integration of heterogeneous artefacts: While current model-based technologies provide common services for model-based artefacts by following certain meta-modeling standards or other conventions, other artefact kinds such as technical drawings, software components or hardware descriptions cannot be directly integrated with models. However, this would be highly needed in order to allow for a progressive integration starting in the engineering process and going until the deployment process.

For the CPS demonstrator, integration between the logical controller model and the physical models, i.e., the 3D CAD models, is required in order to run virtual simulations before moving to the production site. Currently, these type of tools are often realized on different technologies with different languages (even legacy meta-languages used to define these languages) and simulators.

C3: Languages for Dev-to-Ops pipelines: Previously a lot of research has been spent on languages for megamodeling, i.e., how different models are connected, and for model transformation chains, i.e., how models are pushed through a network of transformations. However, more specific languages may be needed to describe the pipelines from Dev-to-Ops. Such languages would allow to explicitly model the process instead of scripting these processes in different technologies. We proposed one approach going in this direction in the past by extending Gradle with explicit megamodeling and transformation chain DSL [10].

For the CPS demonstrator, we first require open APIs on all levels: software modeling tool, 3D CAD modeling tool, programming IDEs, simulation platforms, etc. In addition to open tools, a particular language to describe the complex

process of moving from the modeling activities to the finally deployed system with necessary interaction points, e.g., a human has to validate the graphical simulation of the virtual gripper models, is required.

4.2 From Ops to Dev

C4: Tracing Operational Data to Design Models: The first challenge in this category is to map back the runtime data (e.g., measures about performance, energy consumption, masses, costs, etc.) into the documentation provided on top of development design models. Existing modeling languages often lack a viewpoint for operations or provide do not provide dedicated guidelines how such information may be represented, e.g., see UML, SysML, and many DSLs. Dedicated extensions to these languages are required to link to operational data or to store summaries of operational data in models.

For the CPS demonstrator, we have to record and represent the realized positions of the gripper to reason about tolerance ranges on the model level and to validate the precision of the final system. In SysML such information is currently not representable. However, there are some dedicated efforts for SysML in the standardization process of SysML v2 [4] to provide state histories for components.

C5: Embedded Visualization of Operational Data in Design Models: Operational data is naturally becoming huge in size for complex systems. Even if operational data is already traced to design models, current modeling languages and modeling editors most often fail short in visualization aspects. Additional requirements for visualization of design models occur such as how to visualize the underlying quality of the data such as uncertainties. Integrating sophisticated visualization techniques from the information visualization community [1] seems beneficial in order to provide an understanding of operational data embedded in design models at a glance.

For the CPS demonstrator, dedicated diagrams have to be supported to visualize the runtime data (we refer again to the bottom of Fig. 3). Just showing large runtime data in property windows in current modeling editors is not helpful for modelers to reason on runtime events and values. New diagram types are needed for our current modeling languages and tools to visualize time series information (for instance as different kind of charts [5]) or time series visualization tools have to be integrated in the engineering tools.

C6: Utilizing Operational Data for Continuous Checks and Improvements of Design Models: Runtime models have gained considerable attention in model-driven engineering, mostly in the context of self-* systems. Exploiting runtime models for continuously checking, profiling, enriching and improving

[4] `http://www.omgsysml.org/SysML-2.htm`

[5] For an example, see: `https://sparxsystems.com/enterprise_architect_user_guide/14.0/model_publishing/define_a_time_series_chart.html`

design models (possibly through additional predictive models) would allow to reason about the next versions of a system's design [7]. Runtime models are indeed already very helpful here, but currently not all runtime models are in line with the design models. For instance, assume the transform of the runtime models back into traces which can be replayed by simulators for animation, exploration, etc., on the design models.

For the CPS demonstrator, we need the possibility to play in the runtime traces from the physical system, e.g., to reproduce errors which occurred during operation, in the virtual representation (both, physical and logical view)). This may require dedicated transformations of runtime logs of the system to the design model level. These data transformations may be systematically engineered as coupled transformations with respect to the design time model transformations employed to reach the code level.

C7: Languages for Ops-to-Dev pipelines: Dedicated languages are needed to support the modeling of Ops-to-Dev pipelines. For instance, such languages are required to provide provenance for the extracted runtime models and linked design models, for the specification of indicators, e.g., metrics, KPIs, of interest, as well as the required data exchange between different monitoring, analysis, and design tools.

For the CPS demonstrator, we need dedicated languages to describe properties such as state realizations, request/response times, precision of certain actions, etc. This further requires to have a hybrid query language which is on the one hand powerful on very large data, e.g., time series recorded on the system level and on the other hand is able to produce at the same time model structures to populate runtime models and to compute derived properties which may be attached to the design models, e.g., for a given command for the gripper to move to a particular position, the average realized position may be annotated to the action in the state machine.

4.3 Synopsis

To sum up, the road ahead summarized in Fig. 4 we see as follows. In particular, in stage 1, the challenges C1, C2, C4, and C5 can be considered as ground work which is required to lift MDE to the next level for both phases, design and operation. As soon as these challenges are tackled, a rigorous automation support is required to build and enact DevOps pipeline efficiently. Thus, challenges C3 and C7 have to be tackled in stage 2. Finally, as soon as the foundations are achieved and an appropriate level of automation support is reached, the benefits of realizing a continuous engineering process by continuously improving the design models based on runtime data (cf. challenge C6) may be realized in the final stage.

5 Looking Ahead

Looking ahead the vision presented in this paper, we present in this section different perspectives that would leverage the implementation of the proposed research roadmap.

Business concerns, as presented in the BizDevOps approach [5], require to reason over the global system. Such an approach would benefit from the application of the DevOps principles at the model level as models are closer to the application domain and provide a comprehensive representation of the system, including its environment and possible extra functional properties related to business concerns. For this, an additional integration dimension opens up. In particular, there is the need for aligning enterprise models and design models which is provided by reference enterprise architecture frameworks. Finally, for reporting back the performance of the system on the business level, runtime monitoring of requirements as well as enterprise models seems beneficial.[6]

The smooth combination of the Dev-to-Ops and Ops-to-Dev continuums would provide advanced feature to support live modeling [8]. Live modeling environments would provide continuous and immediate feedback to modelers about the impact of their changes on the execution behavior of a model eliminating any delay between modelers' actions and feedback on their effects. Therewith, they should offer flexibly to explore the design space easing the development of complex software-intensive systems, facilitating learning, and improve quality and efficiency in the development process. In addition to applying operations at the level of the digital twin or the system itself, this would enable the simulation of the operations themselves to explore what if scenarios.

Finally, promoting DevOps principles at the model level enables to push backward its use early in the development process. Hence, DevOps principles would not only apply to the integration, deployment and delivery of the global system, but can also apply at a finer grain for the different concerns addressed during the development process, and across the various abstraction levels. These two dimensions (separation of concerns and levels of abstraction) complement the Dev-Ops dimension, and would possibly lead to powerful development process where automation and continuous feedback is not only available at the level of the global system, but also at the level of the different concerns and across the various levels of abstraction.

Acknowledgments

This work has been partially supported and funded by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, by the FWF under the grant numbers P28519-N31 and P30525-N31, and the Inria / Safran collaboration GLOSE.

[6] As an example from industry see: `https://www.softwareag.com/info/innovation/enterprise_digital_twin/default.html`

References

1. Aigner, W., Miksch, S., Schumann, H., Tominski, C.: Visualization of Time-Oriented Data. Human-Computer Interaction Series, Springer (2011)
2. Broy, M., Schmidt, A.: Challenges in engineering cyber-physical systems. Computer **47**(2), 70–72 (2014)
3. Estefan, J.: Survey of model-based systems engineering (mbse) methodologies. Incose MBSE Focus Group pp. 1–47 (2007)
4. García, J., Cabot, J.: Stepwise adoption of continuous delivery in model-driven engineering. In: Proceedings of DEVOPS (2018)
5. Gruhn, V., Schäfer, C.: BizDevOps: because DevOps is not the end of the story. In: Proceedings of the International Conference on Intelligent Software Methodologies, Tools, and Techniques. pp. 388–398. Springer (2015)
6. Lee, E.A.: Cyber Physical Systems: Design Challenges. In: Proceedings of the 11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC). pp. 363–369 (2008)
7. Mazak, A., Wimmer, M.: Towards liquid models: An evolutionary modeling approach. In: Proceedings of the 18th IEEE Conference on Business Informatics (CBI). pp. 104–112 (2016). https://doi.org/10.1109/CBI.2016.20, https://doi.org/10.1109/CBI.2016.20
8. Tendeloo, Y.V., Mierlo, S.V., Vangheluwe, H.: A multi-paradigm modelling approach to live modelling. Software and Systems Modeling **18**(5), 2821–2842 (2019). https://doi.org/10.1007/s10270-018-0700-7, https://doi.org/10.1007/s10270-018-0700-7
9. Vangheluwe, H., Amaral, V., Giese, H., Broenink, J., Schätz, B., Norta, A., Carreira, P., Lukovic, I., Mayerhofer, T., Wimmer, M., Vallecillo, A.: MPM4CPS: Multi-Paradigm Modelling for Cyber-Physical Systems. In: Proceedings of the Project Showcase @ STAF 2015. pp. 1–10 (2016)
10. Weghofer, S.: Moola - a Groovy-based model operation orchestration language. Master's thesis, TU Wien (2017)
11. Whittle, J., Hutchinson, J., Rouncefield, M.: The state of practice in model-driven engineering. Software, IEEE **31**(3), 79–85 (2014)
12. Wolny, S., Mazak, A., Wimmer, M., Konlechner, R., Kappel, G.: Model-driven time-series analytics. Enterprise Modelling and Information Systems Architectures **13**(Special), 252–261 (2018). https://doi.org/10.18417/emisa.si.hcm.19, https://doi.org/10.18417/emisa.si.hcm.19