Survey of Automation Practices in Model-Driven Development and Operations

Christophe Ponsard and Valery Ramon

April 12, 2022

Survey of Automation Practices in Model-Driven Development and Operations

Christophe Ponsard and Valery Ramon

CETIC Research Centre, Gosselies, Belgium

Abstract

Model-driven methods are gaining momentum in the industry to develop software intensive systems. To be effective in quality and efficient in productivity, they require a strong toolchain with seamless automation. The DevOps approach can help reach this by unifying software development and operations with a strong focus on automation and monitoring. The aim of this short paper is to review automation tasks that are specific to a model-driven context and to classify them according to a typical DevOps lifecycle covering design, code, testing, deployment and runtime activities. Tasks are identified based on different industry use cases experienced in our research centre or reported in the literature. Some challenges are identified and discussed, especially related to the use of bots in a model-driven context.

Keywords— DevOps, Model-Based System Development, Verification, Testing, Generation, Certification, Continuous Integration, Toolchain

1 Introduction

Engineering has been relying on modelling for quite a long time in many disciplines such as construction, electronics or aeronautics. It has widely proven capabilities to abstract and reason on real world artefacts at design time. However, its adoption is still in progress in the areas of software development (MDD - Model-Driven Development) [22] and system engineering (i.e. MBSE - Model-Based System Engineering) [11]. A common cause is the difficulty to face the problem of switching from a rigid document-based culture to a more dynamic model-based culture [15]. The latter relies on standardised and widely adopted modelling languages like SysML at system level, UML for software and also increasingly on Domain Specific

Languages (DSLs). Their visual syntax ease design and communication activities while their semantics enable automation.

Automation is crucial for the success of such methods because the cost of extra modelling activities must be regained on later activities, especially testing and bug fixing which can amount to a large ratio of the project budget. In addition to model-level verification and validation activities, the automated generation of artefacts is advised to get the most out of the modelling effort and to avoid introducing new flaws due to manual coding tasks.

Over the past few year DevOps has emerged as a strong way to bridge to the gap between Development (Dev) and Operations (Ops) by emphasizing communication and collaboration, continuous integration, quality assurance, and delivery with automated deployment utilizing a set of development practices [12]. It provides a reference framework for organising a systematic pipeline of tool supported activities ensuring both continuous development (CD) and integration (CI). Different specialisations have emerged such as DevSecOps which focuses on security activities [16] and, of interest here, ModDevOps that strongly advocates abstraction, automation, and monitoring at all steps of system construction [10]. Figure 1 illustrates the latter on the standard infinite loop used to depict the continuous DevOps lifecycle.

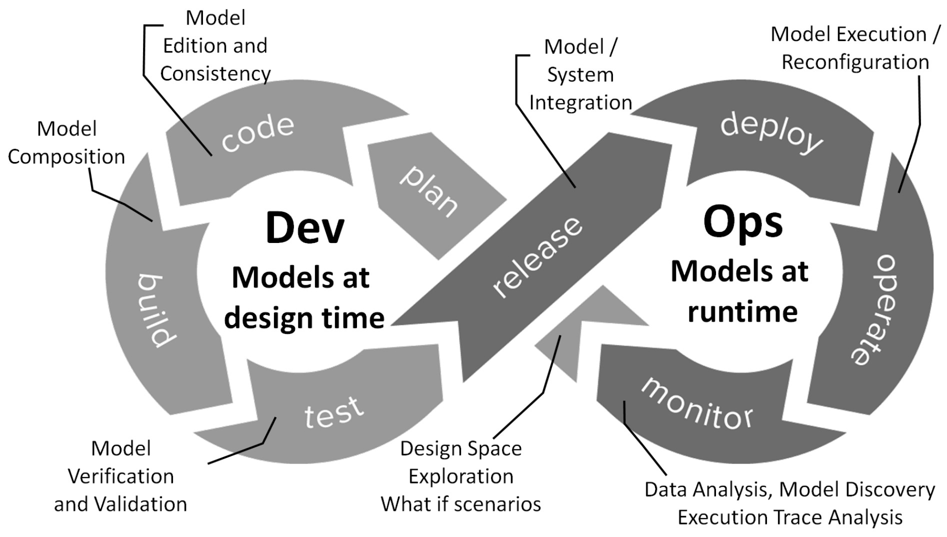
Model Edition and Consistency
Model Composition
code
plan
build
test
Dev
Models at design time
Model Verification and Validation
Model / System Integration
release
Design Space Exploration What if scenarios
Model Execution / Reconfiguration
deploy
operate
monitor
Ops
Models at runtime
Data Analysis, Model Discovery Execution Trace Analysis

Figure 1: ModDevOps lifecycle [3]

A DevOps pipeline is highly automated so that the teams can focus on the value-added activities and on collaboration, especially across the Development and Operations borders. A number of tasks can be fully automated and operated by so called “bots” which are triggered by user commands or events, typical examples are nightly builds including non-regression testing or the scan of commits for detecting and reporting some possible flaws early. The exact definition of a bot is still emerging and empirical study has identified important characteristics such as autonomy, intelligent processing, and a tight integration [7].

In the scope of this paper, we are interested by identifying and classifying candidate tasks for automation in the context of MDD. We carry out our analysis by using the (Mod)DevOps as reference architecture for comparison and classification even though it might not be the actual implementation.

The methodology followed is based on the gathering and analysis of a number of case studies from our own experience and from the literature. At this stage of our research, we do not claim to be exhaustive and did not apply a systematic mapping survey approach. We rather confronted cases from a variety of domains with different focus (e.g. new system vs modernisation/enterprise architecture) and various kinds of requirements (including safety/security critical) in order to discuss specific DevOps support for MDD including the use of bots.

This paper is structured as follows. First, Section 2 reminds about standard DevOps automation before Section 3 surveys more specific automation related to MBSE. It is followed by a discussion in Section 4 which also highlight key challenges in connection with the use of bots. Finally, Section 5 draws some conclusions and identifies our future work.

2 Standard DevOps Automation

The focus of this paper is not to identify standard automation activities in DevOps. However, this section reminds about them in order to set the background so we can exclude them from the specific tasks targeting modelling or possibly state how they (inter)relate, e.g. as refinement or support.

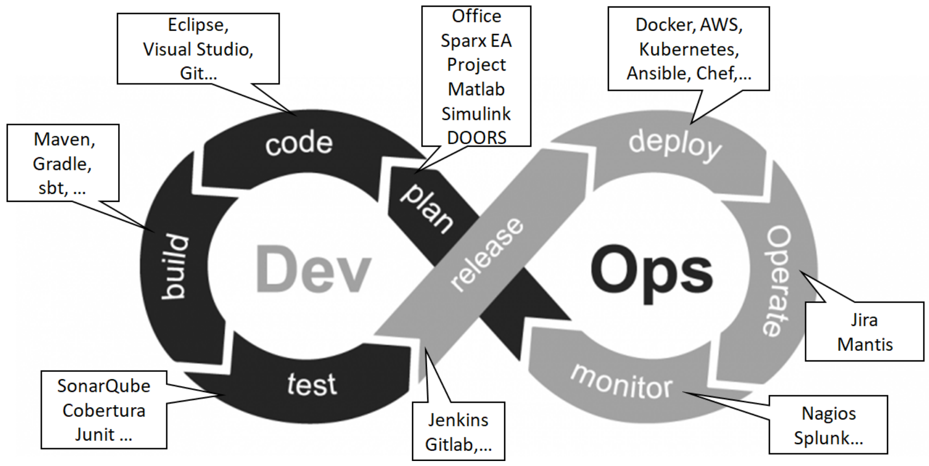
Eclipse, Visual Studio, Git...
Office Sparx EA Project Matlab Simulink DOORS
Docker, AWS, Kubernetes, Ansible, Chef,...
Maven, Gradle, sbt, ...
code
plan
deploy
build
Dev
release
Ops
Operate
Jira Mantis
test
monitor
SonarQube Cobertura Junit ...
Jenkins Gitlab,...
Nagios Splunk...

Figure 2: Standard DevOps pipeline with typical tools

A DevOps infrastructure can involve a variety of tools at each phase as depicted in Figure 2. Those tools can be deployed on premises and/or in the Cloud. The outer communication layer is also a key for the successful operation by enabling communication through teams across the lifecycle, for example through collaborative chat platforms such as Slack or Mattermost. DevOps automation can be bounded to specific static rules such as nightly builds but is also supported by more flexible and dynamic agent called "bots" which can act as a bridge between the collaboration and the DevOps tools. The interaction with such bots can occur through chat commands posted in a channel of the collaboration tool and later acts to perform specific tasks [14]. A bot act autonomously with a level of human control and supervision to check the outcome either directly (e.g. direct answer for a real-time query) or through some dashboards (e.g. successful build process).

Scenarios can cover the whole lifecycle, for example: [21]

- *Proactive Planning*: create new user stories in sprint planning tools, assign them, update sprints and product backlogs.
- *Build and Continuous Integration*: execute on-demand or daily/nigthly build jobs and report back statistics.
- *Continuous Deployment*: perform deployments across environments, roll back deployments in case of failure, perform status checks or post-deployment actions such as VM restarts.
- *Infrastructure Provisioning and Configuration Management*: launch jobs to provision infrastructure or application environments. Enable monitoring and restarting services and servers.
- *Continuous Monitoring/Feedback*: gather application statistics and health status, perform analytics to feed a dashboard.

3 DevOps Automation for MDD

In this section, we first review automation needs. Then, we survey and classify existing practices in the scope of model-driven development. This includes some MBSE connectivity at the requirements and design phases (i.e in the DevOps plan step).

3.1 MDD Automation Needs

Dev phase - Figure 3 depicts a classical V-shaped model-driven development. The figure shows that the model becomes a central artefact. Efficient support to all model related activities in this phase is critical to ensure the expected productivity gain. The idea is to invest more in analysis to reduce the testing effort and the need to correct flaws and bugs discovered later in development or even in operations. Model related activities can be classified in model analysis and model transformation.

Standard model analysis needs are to:

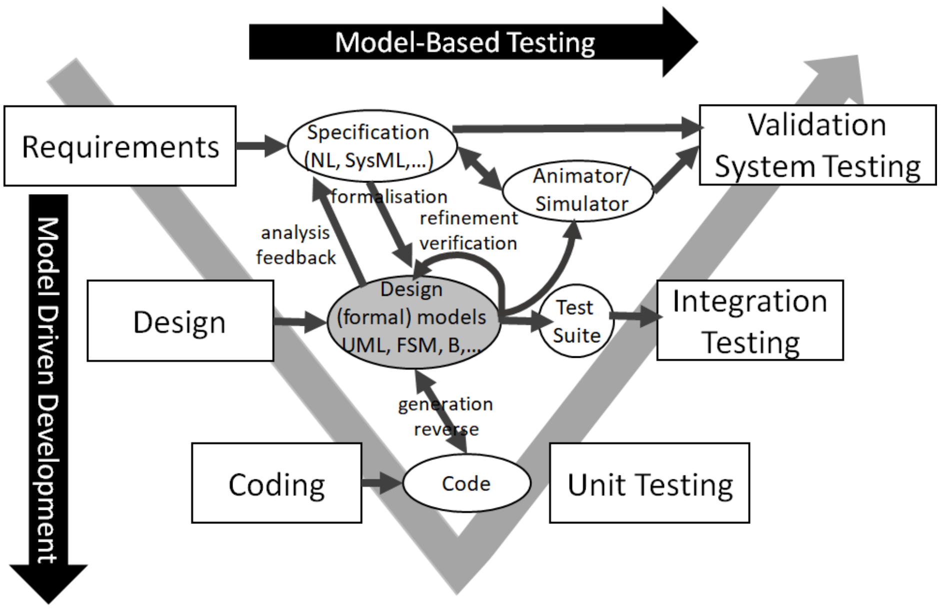

Figure 3: Summary of major activities in Model-Base Software Development

- assess model quality issues related to its relation with requirements (e.g. uncovered/missing/flawed requirements). This triggers iterative clarification between requirements and design models.
- performing traceability analysis and impact assessment across a model for different purposes, e.g. change request or refactoring.
- early model verification and validation: depending on the degree of formalisation, it can range from estimating some timing behaviours in a functional block model to formal verification of specific safety invariant (e.g. in B).
- possibly performing model animation if the model captures the dynamic behaviour in order to ease the model validation prior to more detailed design and development steps.

Model transformation is typically used to automatically generate the following artefacts from the model and thus achieve substantial productivity gains:

- standard lifecycle documents, previously produced manually in a document-oriented approach, e.g. design document, test plans, documents to support a certification process.
- test suites using a model-based testing approach.
- more refined models, closer to the code, e.g. in the B method.
- code, either stubs (from a structural model) or executable code (from a behavioural model).

Ops phase - Although less specific, MDD can extend into the operations phase, especially to achieve the following tasks:

- runtime monitoring of specific properties used as assumption for the correct operation of the system related to the environment including user behaviour or execution platform performance/responsiveness. This does not apply to critical requirements but helps to provision adequate and more dynamic response, especially in a DevOps cycle. Code monitors can be generated for this purpose.
- seamless runtime update and reconfiguration of the system relying on a deployment model shared between the DevOps platform and the application itself.

3.2 Survey of Automation Practices in MDD

To conduct our survey, we identified a number of case studies related to our MDD/MBSE activities with industrial partners or through reported use of such activities although we did not implement a systematic mapping study. Our goal was to achieve a decent coverage of a variety of application domains (automotive, space, logistics, railways...) and DevOps practices. Our search was limited to the past 5 past years and used keywords such as MBSE, automation, continuous integration, DevOps, bots and case study.

The following raw observations can be formulated:

- SysML and UML are widely reported as modeling framework. SysML is even more cited than UML probably because the whole software-based system is usually modelled.
- Most of the reported automation activities relate to the Dev part while the few activities reported in Ops are quite standard (configuration management and monitoring).
- Most model-related activities are reported with a bigger focus on model analysis and traceability rather than model transformation (not many references to test and code generation).
- The pipeline seems rather static with manual triggers and few reported use of bot technology except in two cases. The first (case 7) is in railways and stands out against other safety-critical applications: bots explicitly used to synchronise requirements templates, generate documents and automated builds [13]. The second (case 8) is related to security management, a domain with a specific variant called DevSecOps [16]. Note also a few cases cover security requirements.
- Our survey involves mainly large organisations known to be less agile and more subject to work in silos.

Table 1: Survey of Automation Practices in MDD

#	Year	Domain	Companies	Scope	Languages	Dev automation	Ops automation	Tooling	Ref.
1	2017	Space	fictional	cubesat	SysML, CONOPS	stereotype (semi-automation), dependency matrix, simulation	N/A	MagicDraw	[8]
2	2017	Logistics	LWN	sea crane	SysML UML	reuse, optimisation	N/A	Architect.	[23]
3	2017	Telecom	US gov	radar (modernisation)	UML, SysML, dataflow, CONOPS	traceability, consistency checks, system risk analysis, model publication	N/A	DOORS, DSM, Radiant web UI	[4]
4	2017	Space	NASA, ESA, DARPA	small satellite systems	SysML	viewpoints, rules	N/A	N/A	[1]
5	2019	Railways	infrabel	business system	EA/BPMN	Model quality Document generation	KPI monitoring	Sparx EA	[17]
6	2019	Automotive	General Motors	auton. automotive	dynamic modeLs	Test generation	N/A	CARLA, VIRES	[6]
7	2020	Manufacturing	Christian Doppler Lab	CPS, robotics	SysML, 3D CAD	code generation scripts, test case generators	deployment scripts	N/A	[3]
8	2021	Railways	D-Bahn Siemens	transp. system	SysML	reqbot, docgen, simulation	N/A	Capella, gitlab CI, jupyter	[13]
9	2021	Inform. security	SPARTA	firewall	data/threat/ infrastructure/certification models	traceability	config management,	frama-C, Open-SCAP	[5]
10	2021	Manufact. & logistics	ABB, FAGOR	control system	Finite State Machines	automated test gen for attacks, code checking	config of trace monitor	Uppaal	[19]

4 Discussion

4.1 Bots for DevOps Collaboration

The reported use of MDD is mainly in large organisations which tend to have dedicated departments dealing with the functional, safety and/or security dimensions, often in a wider engineering context. Consequently, DevOps collaboration means may be hindered by the mandatory use of corporate channels which may still be partly document-based. In early adoption phase of modelling, its extend may be limited to the functional dimension and rely on document flows for connecting with other departments, e.g. safety.

This explains that the use of bots is limited to focused activities such as model analysis at design stage (detecting quality issues, synchronisation with requirements management, etc.) or at collaboration boundaries using document generation. A prerequisite to a wider adoption is to make sure the model is truly accessible across departments, so each department can start developing its own viewpoints as mentioned in [1] and move to co-engineering practices [18]. Many modelling environment incorporate communication inside modelling tools for reviewing purposes (e.g. TEAMS for Capella, PostMania for Visual Paradigm). Such channels can also be used to interact with bots or by bots to report detected issues. In such a settings, bots can play the role of virtual modelling assistants or real-time model reviewers, resulting in a continuous feedback on the model quality [2].

4.2 Bot support for Dev to/from Ops Transitions

Our study highlights a focus on model design and transitioning to code/test and less on operations. This is confirmed in [3] which stresses the need for better support especially for cyber-physical systems. For Dev-to-Ops, virtual environments should be available for system level testing, standard deployments procedures should also be available and work in a similar way. For Ops-to-Dev, the monitoring should be supported by adequate descriptive runtime models easing the link back to the design model for enabling analysis and triggering possible evolution. This requires adequate languages for connecting runtime models back to design models, for the specification of indicators and identifying the information collection needs. A prerequisite is to make sure the pipeline is model-aware [9]. Complex monitors might also evolve from dedicated components to model generated or configured components based on a domain specific analysis enriched with runtime data, like threat models and vulnerability databases in cybersecurity. At this point, bots can be efficient to perform semantic reconciliation and impact analysis of design-time models with runtime data [2]. Conversely digital twins can also be analysed for deviation or kept synchronised using runtime bots.

4.3 Bots for more Incremental Processes

Many investigated domains are subject to certification constraints, mostly related to safety and increasingly cybersecurity. DevOps automation and specialised bots can support evidence gathering and document generation required by certification. The incremental approach is especially useful for supporting the need of fast update deployment, especially in response to cyberthreats [5].

5 Conclusion and Perspectives

This paper performed an overall survey of automation needs and practices for MDD using a DevOps approach. Although not exhaustive, our current analysis is consistent with other published works. It highlighted interesting directions to deepen our survey and further research in this area. We plan to extend our survey to better characterise the bot support for MBSE, especially from the operations phase.

We will also confront it with another mapping study in a wider scope [20]. There are many opportunities to introduce more bots in MDD inside modelling environments or in connection with runtime monitoring and data processing. The use of machine learning techniques can also help to improve and ease the performance of various diagnostics.

References

[1] Awele I Anyanhun and William W. Edmonson. Inter-satellite communication MBSE design framework for small satellites. In *Annual IEEE International Systems Conference, SysCon, Montreal, QC, Canada, April 24-27, 2017*, pages 1–7. IEEE, 2017.

[2] Jordi Cabot et al. Cognifying model-driven software engineering. In *Software Technologies: Applications and Foundations - STAF Collocated Workshops, Marburg, Germany, July 17-21, Revised Selected Papers*. Springer, 2017.

[3] Benoit Combemale and Manuel Wimmer. Towards a model-based devops for cyber-physical systems. In *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, 2020.

[4] Jeremiah Crane et al. Mbse for sustainment: A case study of the air force launch and test range system (ltrs). In *AIAA SPACE and Astronautics Forum and Exposition*, page 5302, 2017.

[5] Sébastien Dupont et al. Incremental common criteria certification processes using devsecops practices. In *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Sept.*, 2021.

[6] Joseph D'Ambrosio et al. An mbse approach for development of resilient automated automotive systems. *Systems*, 7:1, 01 2019.

[7] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. An empirical study of bots in software development: Characteristics and challenges from a practitioner's perspective. In *Proc. of the 28th ACM Joint Meeting ESEC/FSE*, 2020.

[8] Sanford Friedenthal and Christopher Oster. *Architecting Spacecraft with SysML: A Model-based Systems Engineering Approach.* Createspace Independent, 2017.

[9] Jokin García and Jordi Cabot. Stepwise adoption of continuous delivery in model-driven engineering. In *DEVOPS*, 2018.

[10] Jerome Hugues and Joe Yankel. From Model-Based Systems and Software Engineering to ModDevOps. CMU Resarch Review, 2021.

[11] INCOSE. Systems engineering vision 2020. Seattle, WA: International Council on Systems Engineering, 2007.

[12] Ramtin Jabbari, Nauman Ali, Kai Petersen, and Binish Tanveer. What is devops?: A systematic mapping study on definitions and practices. pages 1–11, 05 2016.

[13] Viktor Kravchenko. An example of model-centric engineering environment with Capella and CI/CD. Capella Days Conference, 2021.

[14] Carlene Lebeuf, Margaret-Anne D. Storey, and Alexey Zagalsky. Software bots. *IEEE Softw.*, 35(1):18–23, 2018.

[15] Azad M. Madni and Shatad Purohit. Economic analysis of model-based systems engineering. *Systems*, 7(1), 2019.

[16] Håvard Myrbakken and Ricardo Colomo-Palacios. Devsecops: a multivocal literature review. In *International Conference on Software Process Improvement and Capability Determination*, pages 17–29. Springer, 2017.

[17] Christophe Ponsard. Assessing IT Architecture Evolution using Enriched Enterprise Architecture Models. BENEVOL19 (talk) `http://soft.vub.ac.be/benevol2019/papers/BENEVOL\_2019\_paper\_17.pdf`, 2019.

[18] Christophe Ponsard, Jeremy Grandclaudon, and Philippe Massonet. A goal-driven approach for the joint deployment of safety and security standards for operators of essential services. *J. Softw. Evol. Process.*, 33(9), 2021.

[19] Andrey Sadovykh et al. Veridevops: Automated protection and prevention to meet security requirements in devops. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5*. IEEE, 2021.

[20] Sivasurya Santhanam et al. Bots in software engineering: a systematic mapping study. *PeerJ Computer Science*, 8(e866), February 2022.

[21] Shriniwas Sathe. The Role of Bots in DevOps. `https://devops.com/the-role-of-bots-in-devops`, 2019.

[22] D. C. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31, Feb 2006.

[23] Thomas Vosgien et al. A Federated Enterprise Architecture and MBSE Modeling Framework for Integrating Design Automation into a Global PLM Approach. In *14th IFIP Int. Conf. on Product Lifecycle Management (PLM)*, July 2017.