# A Low-Code Development Environment to Orchestrate Model Management Services

Arsene Indamutsa, Davide Di Ruscio, Alfonso Pierantonio

# A Low-Code Development Environment to Orchestrate Model Management Services*

Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio

Università degli Studi dell'Aquila, Italy
{firstname,lastname}@univaq.it

**Abstract.** The current digital transformation in production systems has positioned model-driven engineering (MDE) as a promising development solution to leverage models as first-class entities and support complex systems' development through dedicated abstractions. Models are specified through domain-specific languages and consumed by dedicated model management services, which implement automation and analysis services. Achieving complex model-driven tasks that involve several model management services and multiple model repositories can be a difficult and error-prone task. For instance, modelers have to identify the proper atomic operations among available services, connect to remote model repositories, and figure out their composition to satisfy the final goal. Different composition proposals have been introduced in MDE even though a satisfactory solution is still missing. In this paper, we propose a *low-code development environment* to support citizen developers to plan, organize, specify and execute model-management workflows underpinning the development of complex systems. Thus, developers are relieved from managing low-level details, e.g., related to the discovery, orchestration, and integration of the needed model management services.

**Keywords:** Production System Development · Low-code Development Platform · Domain Specific Language · Cloud-based Model Repository · Workflow Engine

## 1 Introduction

Production systems are highly interwoven systems that span through different engineering fields such as mechanical, electrical, network, software engineering, and control systems [21]. Such a synergistic integration introduce additional complexities due to the management of cross-disciplinary methods and the integration of heterogeneous artifacts together with their supporting tools. Model-Driven Engineering (MDE) is a software discipline that leverages the adoption of models to support the understanding and the engineering of large, complex, and interdisciplinary systems, such as production systems while minimizing their

* This work is funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie – ITN grant agreement No 813884

complexity through the systematic adoption of abstractions [3]. Models are specified through graphical or textual domain-specific languages, and dedicated model management services consume them, e.g., to perform early analysis or automatically generate target software artifacts [2].

To foster reusability of modeling artifacts, and to support collaboration among modelers and developers [8], over the last decade, several model repositories have been proposed by the MDE community [2]. Thus, specific system models, and developed model management tools are made available in some of the existing repositories to promote their reuse during the development of new systems. However, in such contexts, model-based development of complex software systems requires the definition of different processes, consisting of the coordinated composition of different tools and the usage of various languages. Thus, developers must discover from available repositories the needed modeling tools and related model management services and work on their composition to develop the wanted system. The current main challenges that make the composition of model management operations a strenuous activity are the following:

- Current composition tools mainly deal with locally available resources;
- Composition mechanisms like ANT tasks are specific for the particular ecosystem at hand (e.g., Epsilon[1]);
- The development of complex engineering processes require technical expertise that citizen developers (i.e., domain experts with limited programming skills) might not necessarily have, though deemed to be aware of involved services.

In this paper, we propose the adoption of a low-code development platform (LCDP) to develop complex model management processes. LCDPs provide intuitive visual environments to citizen developers to build fully operational applications, which do not require a strong programming background [14]. The considered context is characterized by atomic model management operations provided as services by (potentially) different providers. The envisioned LCDP supports the discovery and the orchestration of the services needed to develop the wanted composed process. The objective is to develop an event-driven approach based on trigger-action programming as done by LCDPs like IFTTT and Zapier among popular services [20]. In such platforms, users can connect various independent services, organize and customize them in a specific flow to achieve their goal [13]. Similarly to such services, the proposed platform will support high-level abstraction and automation to compose model management services provided by different repositories. The current work is under development, and its code repository is publicly available online.[2]

The paper is organized as follows: Section 2 presents the background and motivation of this work. Section 3 presents the proposed approach. The related work is discussed in Section 4, whereas Section 5 concludes the paper and describe prospective work.

[1] https://www.eclipse.org/epsilon/doc/workflow/
[2] https://github.com/Indamutsa/model-management-services.git

## 2 Background and motivation

The digital transformation undergoing traditional production systems heavily impacts product types' variability, and customization possibilities during their life cycles [22, 19]. Significant efforts and investments are required to implement and maintain complex production systems, limiting their acquisition by small and medium-sized enterprises (SMEs) [21]. Such systems' complexity is amplified when relying on the use of code-centric approaches that have proven daunting due to the arduous effort involved in programming, customization, and integration of complex heterogeneous systems coming from different engineering domains and processes [5].

This complexity sparked the need for flexible approaches that adapt to systems behaviour regarding the ever-changing requirements, structural transformations, and unexpected conditions [19]. To develop production systems and cyber-physical systems in general, MDE promotes the adoption of models as machine-readable and processable abstractions specified employing dedicated languages such as System Modeling Language (SysML).[3] Dedicated tools are employed to support development and analysis tasks, to integrate engineering processes and stakeholders' perspectives, and they also foster information exchange during different engineering process [3].

To simplify the development of complex systems, trigger-action programming paradigms can be employed to facilitate automation and abstraction. Such a paradigm is employed, for instance, in the Internet of Things (IoT) domain to develop applications in smart home management, agriculture, e-health, industrial automation, and robotics [15]. Systems like IFTTT, and Zapier are examples of LCDPs that facilitate business automation processes by giving users the means to specify processes [13]. In particular, they permit the creation of new services known as recipes out of custom chaining of services based on conditional statements [20]. For instance, a user can like a particular post on Facebook and automatically archive it on a corresponding storage in the cloud [11].

In [4] authors propose the Modeling as a Service (MaaS) initiative as an approach to deploy and execute model-driven services over the Internet. In such a direction, over the last years, several model repositories have been proposed to promote the reuse of modeling artifacts and to enable their remote execution on demand as services. For instance, in [2] authors propose MDEForge as a platform to enable the remote application of different services like model validation, transformation, and analytics. The composition of model management services is under the complete responsibility of developers that, e.g., have to define the programmatic orchestration of the services of interest and the ways data have to be produced and consumed.

By relying on the concepts and tools previously mentioned in a model-driven engineering setting, we aim at developing complex model management composed operations by specifying custom workflows. We leverage trigger-action paradigm

[3] https://sysml.org/

so that based on particular events (e.g., when you upload your model on a given repository), corresponding actions will be executed. For instance, some analytics are performed on the uploaded model depending on some condition. Then, if the performed analytics produce higher values than a certain threshold, some model transformations are executed, and the produced models are saved to enable the application of additional manipulations.

## 3 Proposed low-code development environment

In this section we describe the proposed low-code development environment. The front-end of the proposed platform is presented in Section 3.1. The core services of the platform are presented in Section 3.2 and its related limitations are mentioned in Section 3.3

### 3.1 Environment front-end

Figure 1 shows a mock-up of the proposed environment providing users with the ability to create and automate workflows on cloud-based repositories using graphical environments with drag and drop capabilities, and custom scripting by the use of a domain specific language as referred to in figure 3. The custom scripting is enabled by an editor where the user can programmatically express complex expressions of the workflow. The services and extensions on the repositories are organized in decoupled and distributed microservices to emphasize the separation of concerns and foster individual service maintainability, scalability, and extensibility [16].

According to the explanatory workflow shown in Fig. 1, the citizen developer might want to upload a Performance Model Interchange Format (PIMF) model [10] and generate a corresponding SySML model out of it. Then, she can validate the model, calculate dedicated metrics, extract some metadata, and once done, merge the obtained information into another SySML model. The obtained model can be stored in the repository, and the user can be notified together with the complete execution logs. The services used in the above scenario are remotely accessed as services through APIs, and the storage systems are distributed services consisting of several network nodes.

Developing and execute model management workflows like the one shown in Fig. 1 without proper support can be time and resource consuming, laborious, hard to maintain and error-prone. The proposed approach aims at enabling citizen developers to create and automate workflows based on selected model management services using a graphical environment with drag and drop capabilities. The proposed environment is based on the metamodel shown in Fig. 2. According to the shown metamodel fragment, workflows consists of nodes, which are an abstract representation of activities referred to as actions and decisions. Several events can trigger activities, and the node can receive different types of inputs, such as modeling artefacts and variables. Events of interest and their

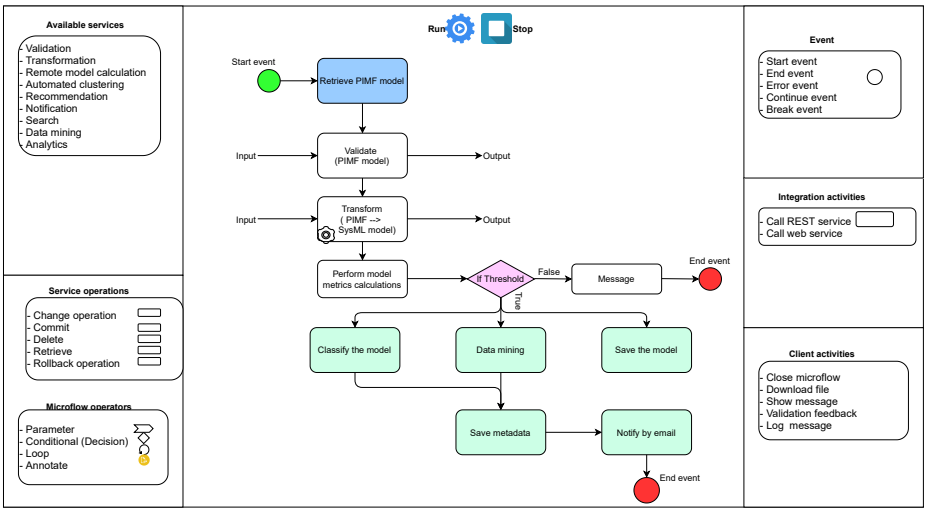


**Fig. 1.** Example mock-up of graphical task workflow environment

sources are defined as seen in Fig 1, and they result from different providers that trigger specific actions as instructed. Nodes represent decoupled, and independent microservices orchestrated when the specified workflow is executed.

### 3.2 Core services

Figure 3 shows a logical view of the system and the corresponding stakeholders, notably two prominent actors involved, i.e., *citizen developers* and *software engineers*. The former can specify task workflows through the provided environment, whereas the latter can extend the repository services by adding new functionalities. The typical user (citizen developer) can access the repository, select services to automate, configure triggers and actions, and authorize task workflows. Interestingly, advanced support is provided to recommend modeling elements while editing workflows, and analyse, test, and deploy models by means of a dedicated DevOps support. Such support is provided by the workflow definition and analysis component. The service integration component ensures seamless integration of external and internal services. Once the modeled workflow is ready, the incoming model (task workflow) encoded in format such as xml/json, is transformed and executed by the engine. Mining and analysis services are performed before the model is persisted with the help of MDEForge [2]. Hence a backup is performed to facilitate data and service recovery, and rollbacks in case program execution encounters an impediment.

The user has control over the use of her task workflow, configuration, and termination. The developer can do whatever the user can do, but also, she can register services, establish connectors, and define triggers and actions related to the added service. The task workflow expressed as models are managed by an

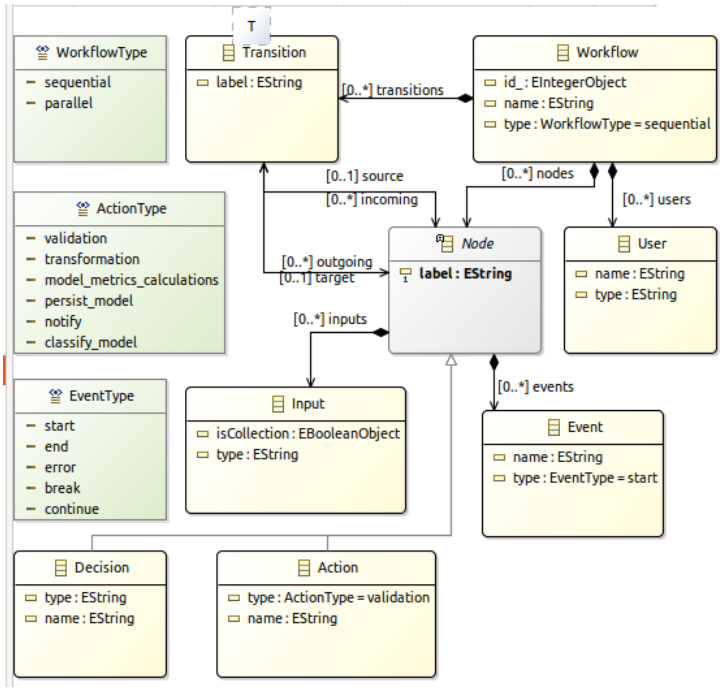


**Fig. 2.** Fragment of the proposed workflow metamodel

engine and persisted as models by MDEForge, a cloud-based model repository, [2] to ensure their management and reusability. The engine has dedicated components that ensure the security, privacy integrity of services and data from several data sources. The engine benefits from inherent services from MDEForge such as recommendation system, quality assurance, service integrator, data mining and analytics as seen in Figure 3.

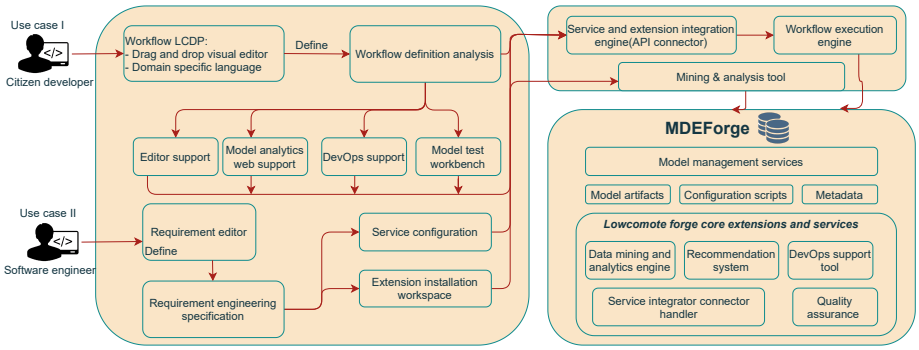


**Fig. 3.** Logical view describing the backend and frontend aspects of the system

Service orchestration is enabled by Kubernetes technology[4], which comes bundled with out-of-box benefits such as auto-scalability and extensibility to handle a flux of users, support continuous integration and load balance containerized workloads and services [7] [17]. By managing distributed and containerized services, we define the utilization and boundaries of resources, reduce hardware costs, deploy resilient, loose coupled, self-healing, elastic distributed services [7]. In addition, we benefit from service discovery, load-balancing, storage orchestration by mounting the volume of your choice on-the-fly from several providers and technologies, and importantly enable automatic bin packaging while ensuring automated rollout and rollbacks to a given state [17].

### 3.3 Limitations

The first and foremost challenge that trigger-action programming solutions have faced are security and privacy concerns as suggested by previous research [18], especially when independent services are involved [1]. Existing approaches have limited facilities supporting the reusability of already created task workflows to avoid replication and adequately manage the collections of modeled workflows [18]. The proposed approach aims at addressing such issues even though debugging facilities are not supported yet. Moreover, while developing task workflows, it is crucial to avoid ambiguity in terminologies that hamper efficiency creation and the use of task workflows [18]. The proposed approach does not provide yet any mechanism to check the terminology used for naming workflows tasks.

## 4 Related work

Berardinelli et al. [3] identified relevant challenges that hinder the adoption of model-driven approaches for cyber-physical production systems engineering and discussed issues related to integrating several modelling tools. An automated engineering toolchain has been proposed to perform early design and validation. Vogel-Heuser et al. [19] presented an approach to support the model-driven engineering of manufacturing systems. The SysML-AT language (SysML for automation) has been proposed to specify both functional and non-functional hardware components' requirements. Chen et al.[6] presented an approach for automatic translation of natural language descriptions into executable If-Then programs. The system helps users to synthesize If-Then programs by proactively predicting triggers and actions related to their descriptions using neural networks. Dzulqornain et al. [9] also developed a real-time monitoring and controlling smart aquaculture system based on IFTTT and cloud integration. The system facilitates interoperability and integration of sensors, system controllers, client data visualizations, and system monitors. Quirk et al. [12] presented an approach to map natural language descriptions with If-Then patterns to executable programs. They use semantic parser-learners that utilize already defined recipe

[4] https://kubernetes.io

descriptions to train semantic parsers that automatically map these descriptions to executable programs.

Differently from such related work, we presented a low-code environment to specify and execute workflows of model management tasks in this paper. The approach is agnostic from the languages used to specify the models that are transformed and later analyzed.

## 5 Conclusion

This paper proposed a novel approach to support the development of complex model management operations. In particular, a low-code development platform is presented by resembling the functionalities offered by currently available LCDPs like IFTTT and Zapier. Such platforms permit the development of complex processes by integrating and executing different services. The proposed approach proposes the adoption of a microservice-based architecture to integrate and execute model management services, which are orchestrated on the cloud according to specifications given by the user by means of a BPMN-like modeling language. The proposed approach aims to overcome current challenges faced by traditional modelling environments that heavily rely on locally downloaded resources. Such environments are limited in their scalability and extensibility and their services exhibit high coupling with the local environment. The complete implementation of the proposed low-code development environment is ongoing and its application on real scenarios, validation, and testing remain as future work.

## References

1. Baruah, B., Dhal, S.: A two-factor authentication scheme against FDM attack in IFTTT based Smart Home System. Computers and Security **77**, 21–35 (2018)
2. Basciani, F., Di Rocco, J., Di Ruscio, D., Di Salle, A., Iovino, L., Pierantonio, A.: MDEForge: An extensible Web-based modeling platform. CEUR Workshop Proceedings **1242**(September), 66–75 (2014)
3. Berardinelli, L., Mazak, A., Alt, O., Wimmer, M.: Model-Driven Systems Engineering: Principles and Application in the CPPS Domain, pp. 261–299 (05 2017)
4. Brunelière, H., Cabot, J., Jouault, F.: Combining Model-Driven Engineering and Cloud Computing. In: MDA4ServiceCloud'10 Workshop co-located with ECMFA (Jun 2010)
5. Chen, X., Nophut, C., Voigt, T.: Manufacturing execution systems for the food and beverage industry: A model-driven approach. Electronics **9**(12) (2020)
6. Chen, X., Liu, C., Shin, R., Song, D., Chen, M.: Latent attention for if-then program synthesis. Advances in Neural Information Processing Systems (2016)
7. David, O., Lloyd, W., Rojas, K., Arabi, M., Geter, F., Ascough, J., Green, T., Leavesley, G., Carlson, J.: Model-as-a-service (MaaS) using the Cloud Services Innovation Platform (CSIP). Proceedings - 7th International Congress on Environmental Modelling and Software, iEMSs 2014
8. Di Ruscio, D., Franzago, M., Malavolta, I., Muccini, H.: Envisioning the future of collaborative model-driven software engineering. Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017

9. Dzulqornain, M.I., Harun Al Rasyid, M.U., Sukaridhoto, S.: Design and Development of Smart Aquaculture System Based on IFTTT Model and Cloud Integration. MATEC Web of Conferences **164** (2018)
10. Llad, C.M., Smith, C.U.: Performance model interchange format (pmif 2.0): Xml definition and implementation. In: Quantitative Evaluation of Systems, International Conference on. IEEE Computer Society, Los Alamitos, CA, USA (sep 2004)
11. Ovadia, S.: Automate the Internet With "If This Then That" (IFTTT). Behavioral and Social Sciences Librarian **33**(4), 208–211 (2014)
12. Quirk, C., Mooney, R., Galley, M.: Language to code: Learning semantic parsers for if-This-Then-That recipes. ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference **1**, 878–888 (2015)
13. Rahmati, A., Fernandes, E., Jung, J., Prakash, A.: IFTTT vs. Zapier: A Comparative Study of Trigger-Action Programming Frameworks (2017)
14. Sahay, A., Indamutsa, A., Ruscio, D.D., Pierantonio, A.: Supporting the understanding and comparison of low-code development platforms. 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) pp. 171–178 (2020)
15. Surbatovich, M., Aljuraidan, J., Bauer, L., Das, A., Jia, L.: Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of ifttt recipes. In: Proceedings of the 26th International Conference on World Wide Web. pp. 1501–1510 (2017)
16. Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. IEEE Cloud Computing **4**(5), 22–32 (2017)
17. Tomarchio, O., Calcaterra, D., Di Modica, G.: Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks. Journal of Cloud Computing **9**(1), 1–24 (2020)
18. Ury, B., Ho, M.P.Y., Brawner, S., Lee, J., Mennickenz, S., Picard, N., Schulze, D., Littman, M.L.: Trigger-action programming in the wild: An analysis of 200,000 IFTTT recipes. Conference on Human Factors in Computing Systems - Proceedings pp. 3227–3231 (2016)
19. Vogel-Heuser, B., Schütz, D., Frank, T., Legat, C.: Model-driven engineering of manufacturing automation software projects – a sysml-based approach. Mechatronics **24**(7), 883–897 (2014), 1. Model-Based Mechatronic System Design 2. Model Based Engineering
20. Vorapojpisut, S.: A Lightweight Framework of Home Automation Systems Based on the IFTTT Model. Journal of Software **10**(12), 1343–1350 (2015)
21. Weißenberger, B., Flad, S., Chen, X., Rösch, S., Voigt, T., Vogel-Heuser, B.: Model driven engineering of manufacturing execution systems using a formal specification. In: 2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA). pp. 1–8 (2015)
22. Zacharewicz, G., Daclin, N., Doumeingts, G., Haidar, H.: Model driven interoperability for system engineering. Modelling **1**(2), 94–121 (2020)