Systematic Literature Review on Model Transformation for Advanced Generation of Pipelines within DevOps Solutions

Uldis Karlovs-Karlovskis
Faculty of Computer Science, Information Technology and Energy
Riga Technical University
Riga, Latvia
https://orcid.org/0009-0008-7816-3770

Oksana Ņikiforova
Faculty of Computer Science, Information Technology and Energy
Riga Technical University
Riga, Latvia
https://orcid.org/0000-0001-7983-3088

***Abstract*—Since its inception in 2009, DevOps has driven significant advancements in software development in terms of methodologies and support tools, developed both by industry, as well as a result of academic research. In turn to further enhance software development speed without compromising quality, this paper argues that DevOps as such also requires a digital solution for its automation. While previous studies have extensively explored the evolution of DevOps and its associated methodologies, these efforts have been insufficient. Existing research lacks comprehensive analysis and fails to address the full scope of DevOps formalization challenges and opportunities, which is the key prerequisite for automation. Researchers and practitioners are actively seeking innovative approaches, such as model-driven engineering, to expedite automation of software development processes even further. This systematic literature review aims to fill these gaps by providing a more thorough examination of the literature and offering deeper insights into the potential of combining DevOps and model-driven approaches. The paper investigates literature of the last 15 years on how to streamline DevOps pipeline generation that are grounded in model transformation techniques.**

Keywords— Model-Driven Engineering, build, deployment, Continuous Software Engineering, DevOps, pipeline, UML

I. Introduction

The convergence of the Fourth Industrial Revolution [1] and the accelerated shift to remote work during the COVID-19 pandemic has precipitated a comprehensive digital transformation across industries [2]. This rapid digitalization has intensified the demand for accelerated application development, making DevOps a critical component of modern software engineering [3]. Since its inception in 2009, DevOps has spurred the development of numerous tools and solutions from private industry, open-source communities, and academia. To further enhance software development speed without compromising quality, this paper argues that DevOps as such also requires a digital solution and formalization.

Enhancing software development speed has been a persistent focus in research for several decades. While traditional DevOps practices have yielded improvements [4], there is growing interest in leveraging more advanced techniques to accelerate this process. A pivotal moment in this evolution was the first workshop on model-driven DevOps at the 22nd ACM/IEEE MODEL-C conference in 2019 [5]. Building upon these foundational concepts, this paper investigates how model transformation can be applied to streamline DevOps pipeline [6] creation.

As a preliminary step in developing an innovative solution is research on related work in the area in turn to determine whether effective solutions already exist or if there are existing approaches that can be enhanced. This initial investigation is crucial for identifying gaps in the current landscape and ensuring that new developments build upon or improve what has already been established. Without this foundational research, efforts may be redundant or miss opportunities for meaningful advancement. Therefore, the goal of this paper is to conduct a systematic literature review and analysis within the context of automatization solutions in the area of DevOps pipeline generation using model transformation, a well-established formalism in IT product development [7]. The research aims to identify tools employed for advanced pipeline generation from domain models at different level of abstraction, assess the integration of model transformation within DevOps solutions, and investigate the input and output data notations used to perform these transformations.

In-depth full-text analysis of 42 studies published since 2009 revealed that only nine explicitly demonstrated the application of model transformations, primarily converting UML or BPMN models in XML format to automation scripts or Jenkins pipeline code. Nonetheless, the evolving landscape of the field emphasizes the necessity for continued research.

The paper is structured as follows: Section 2 provides a discussion on related work, Section 3 outlines the applied research methodology, Section 4 discusses the research findings, and Section 5 offers concluding remarks.

II. Related Work

The origins of DevOps can be traced back to 2009 and the visionary Belgian, Patrick Debois, the founder of the DevOpsDays conference and movement. Prior to Debois's pivotal role, Andrew Schafer, another influential figure, had been advocating for the concept of "Agile Infrastructure" for a few years. Their discussions on this topic at the Agile conference in 2008 were significant in shaping the early stages of DevOps. At the O'Reilly Velocity Conference, John Allspaw and Paul Hammond presented their groundbreaking case study at Flickr, demonstrating their ability to deploy to production more than ten times daily. Patrick Debois was among the online audience closely following their presentation. Inspired by these stories and his own experiences, Debois founded the DevOpsDays conference in Ghent, Belgium [8] but since the name “DevOpsDays” seemed too long for a hashtag to use on Twitter, a shortened version #DevOps emerged [9]. Since then, the DevOps movement rapidly gained acceptance among engineers

seeking to accelerate software development. This research focuses on research conducted 2009 to mid-2024. Before starting the research, four existing literature reviews that are potentially similar to the research described in this paper have been identified and then analyzed.

Over the 15 years since the emergence of DevOps, a multitude of continuous practices have gained prominence, each promising to accelerate specific stages of the software development lifecycle. A systematic review published in 2016 [10] was conducted to review the state of the art of continuous practices to classify approaches and tools, identify challenges and practices in this regard, and identify the gaps for future research. Of the 69 relevant papers analyzed, 25 delved into the integration of various tools to construct effective toolchains. At the time of this research, pipeline as code was still an emerging concept and consequently underrepresented in the analyzed studies. The authors identified several key challenges hindering the widespread adoption of continuous practices, including a lack of expertise, reliance on manual configuration, and the dynamic nature of customer environments. The authors further emphasized that overcoming challenges necessitates a carefully designed architecture and a deployment pipeline engineering as part of the development phase. A notable omission from the analyzed research is the concept of model transformation in continuous practices as a potential solution to the identified challenges.

In 2018, Dr. Nicole Forsgren, a prominent DevOps advocate and researcher, together with Jez Humble and Gene Kim, co-authored the influential book [11] on science of Lean Software and DevOps. This fundamental work was subsequently recognized with the prestigious Shingo Publication Award. Through four years of rigorous research, including data from the State of DevOps reports conducted in partnership with Puppet, later continued at Google [12], the authors developed a quantitative framework for measuring software delivery performance and identifying its key drivers, using rigorous statistical methods. While the authors propose a novel modeling approach for Continuous Delivery within organizations, this research does not delve into the specific methods and techniques associated with Model-Driven Engineering (MDE) [13].

A qualitative study conducted in 2020 [14], two decades after the widespread adoption of Continuous Integration, sought to identify bad practices. A comprehensive investigation encompassing 13 expert interviews from six companies and an analysis of 2322 Stack Overflow posts yielded a catalog of 79 CI anti-patterns (i.e. bad smells) organized in 7 categories. The authors conclude that while CI/CD adoption is on the rise, its implementation remains challenging, even in well-structured development environments. One prominent anti-pattern identified in the research was the inadequate versioning of pipeline code, often attributed to immature tooling and a lack of formalized practices. The study did not identify any anti-patterns specifically related to model transformation for pipelines, suggesting a limited adoption of this approach within the surveyed companies and online communities.

Conversely, in the same year of 2020 [15], a comprehensive literature review on deployment automation technologies introduced an essential deployment metamodel. This metamodel was grounded in a comprehensive analysis of 90 studies sourced from ACM and IEEE databases, complemented by the top 100 findings from a Google Search. Although primarily focused on cloud-native and infrastructure deployment technologies, the resulting model provides a valuable perspective on how a formalized approach can benefit broader software development automation. The authors further emphasize the necessity of such a model to facilitate the automated transformation of models into executable solutions. This research corroborates the notion that employing model transformation for DevOps pipeline development is an innovative concept with significant potential for exploration.

A 2022 [16] survey underscores the growing prominence of applying model-driven methods to software development performed in IT companies practically. This research focused on a five-year case study analysis to explore how bots and model-driven development can accelerate software development processes. Of the ten case studies analyzed, three incorporated the use of bots, each addressing a distinct challenge within the software development lifecycle. Authors concluded that there are interesting directions for further research. Despite being the most contemporary survey on model transformation applied to accelerating software development identified by the authors, this research is constrained by the following limitations:

- The five-year search scope from 2017 to 2021 may have limited the breadth of conclusions regarding the research object.
- The exclusive focus on artificial intelligence solutions, such as bots, is insufficient for establishing a comprehensive framework for the formalization of automated pipeline creation within DevOps solutions.

Based on a comprehensive analysis of existing literature reviews and exploring the integration of DevOps and model-driven development principles, the authors arrived at the following conclusions:

1) Given the ongoing relevance and value of DevOps practices, further research in this domain is warranted: scientific and industrial research continues to explore the impact of DevOps on software development acceleration. Scientific research methods will be instrumental in quantifying these effects and identifying better practices.

2) Existing literature reviews show that DevOps pipelines engineering lacks formalism: formalization would guarantee automated pipeline development, encompassing the entire process from application architecture to the orchestrated chain of tasks involved in building and deploying source code. A prevailing trend identified within the literature is the exploration of model transformations as a potential formalism, which may offer a significant advancement in the field of software development.

3) The majority of related work concentrates on infrastructure: existing literature primarily explores the application of model transformations for DevOps solutions within infrastructure management, encompassing cloud, container and hardware domains. This limited scope is insufficient to address the complexities involved in orchestrating software build and deployment.

4) To recap: a comprehensive literature review encompassing scientific publications from year 2009 to the present, when it has achieved mainstream adoption [17], is still required to establish a broader context and should

provide an in-depth insight into both theoretical and practical aspects. The research underscores the necessity of conducting a comprehensive literature review to identify foundational studies on formalization applicable to automated build and deployment within pipeline development.

Consequently, this systematic literature review focuses on DevOps solutions that generate build and deployment pipelines through the application of model transformation techniques.

III. Research Methodology

The research is conducted using the systematic literature review methodology used in [18]. The primary objective of this research is to identify existing research on model transformation techniques for advanced generation of pipelines within DevOps solutions. To provide a focused direction for the research, the three following research questions (RQs) were formulated:

- RQ1: Which tools are used for developing build and deployment pipelines within the model transformation domain?
- RQ2: What input data and notation is utilized for modeling build and deployment pipelines?
- RQ3: What is the essence of model transformation and what input or output data is used for advanced pipeline generation?

To address the RQs comprehensively, an initial literature pool is constructed by examining five databases of scientific papers. Firstly, the pool is filtered by reviewing study titles and abstracts. Secondly, a full-text assessment is performed for each remaining study to identify its relevance to the research scope. Subsequently, a snowballing technique is applied to identify additional relevant studies that may have been missed due to not being found with the search query. The following criteria are applied to select the initial pool of studies:

- Database: Scopus, ACM Digital Library, IEEExplore, Science Direct and Springer Link as the source databases.
- Range: 2009–2024 as the year of publication.
- Language: English.
- Subject area: Computer science.
- Title or abstract or keywords contains word pair “model driven” or “model-driven”.
- Title or abstract or keywords contains at least one of the words “architecture”, “software” or “engineering”.
- Title or abstract or keywords contains at least one of the words “devops”, “build” or “deployment”.
- Title or abstract or keywords contains at least one of the methods related to pipelines.

To identify potentially relevant studies, a search query is developed. The following example demonstrates a search query employed for Scopus, excluding keywords for brevity:

TITLE-ABS-KEY("model driven") AND (LIMIT-TO (LANGUAGE, "English")) AND (EXCLUDE (DOCTYPE , "cr")) AND PUBYEAR > 2008 AND TITLE-ABS-KEY(architecture OR software OR engineering) AND TITLE-ABS-KEY(devops OR build OR deployment) AND TITLE-ABS-KEY(pipeline OR script OR Toolchain OR automation OR "configuration management" OR "Continuous Integration" OR "Continuous Deployment" OR "Continuous Delivery" OR "Release Management" OR "Platform Engineering") AND NOT TITLE-ABS-KEY(...)

The initial search yields 80 studies on Scopus, with one duplicate identified. A subsequent search on ACM Digital Library produces 49 studies, including two duplicates from the previous search. A continued search in IEEExplore results in 111 studies, of which 23 are duplicates of previously identified papers. Due to ScienceDirect's limitations in complex query construction, three broad search queries are used, resulting in a combined total of 52 records. Finally, 386 additional studies are retrieved from SpringerLink using its full-text search function. A manual screening process of 643 studies identifies 25 relevant articles for in-depth full-text analysis. By applying a snowballing technique to the initial 25 studies and four previously identified literature reviews, additional 17 relevant papers are discovered, resulting in a final pool of 42 studies for in-depth full-text analysis.

To facilitate subsequent analysis, the initial spreadsheet includes columns for title, author(s), and publication year for each study. A classification system with values "no" "partly" and "yes" is applied to categorize studies based on their alignment with the research objectives. Fig. 1 illustrates the number of studies per year that align with the research objectives. Additionally, columns for the addressed problem and author comments are included in the analysis.

To effectively address RQ1, it is essential to identify the specific tools employed for build, deployment, and pipeline management. The choice of tools often varies based on the specific programming platform employed. Consequently, it is essential to document these variations. To facilitate in-depth full-text analysis, the employed Integrated Development Environment (IDE) were recorded.

To address RQ2 effectively, this research focuses on identifying the input data formats, models and modeling notations utilized by other researchers to eventually generate build, deployment or pipeline automation. Consequently, all the models, notations and their types are registered in additional columns of the spreadsheet.

The motivation of RQ3 is to assess the maturity of model transformation techniques within the field, identify the primary tools and types of data employed for their implementation, and evaluate the level of community adoption for these approaches. An additional objective is to identify the most widely used tool for the model transformation or generation process. Additional columns are added to the spreadsheet to facilitate subsequent analysis.

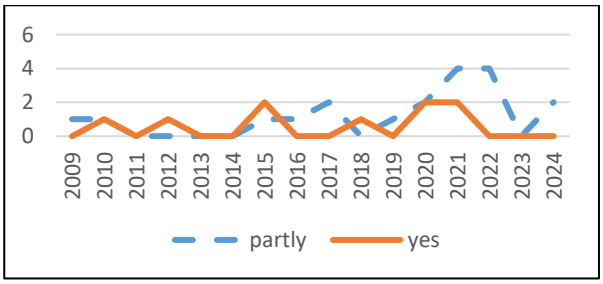

Fig. 1. Relevant studies per year.

Following in-depth full-text analysis of selected studies (a.k.a. marked as "partly") which are excluded from the further research, in addition to many excluded studies via the search query or analysis of the title or abstract, two primary trends emerged:

1) Infrastructure-focused: studies have proposed the use of MDE approaches for pipeline generation, often resulting in the orchestration of infrastructure components, such as cloud resources or Docker containers. The objective of this research is to investigate the orchestration of the component creation process using an MDE approach as an alternative to the current practice of manual scripting and configuration by engineers.

2) Organization-wide: several studies use a wide spectrum of input data to encompass various aspects of an organization, often extending beyond the scope of software development. Many of these studies are theoretical in nature and have not been empirically validated in a field experiment. While the vision of such research is compelling, achieving tangible results through this approach may be challenging, if not unattainable.

After in-depth full-text analysis and selection process illustrated in Fig. 2, nine studies are eligible to address the research questions.

IV. Results

The nine studies selected after analysis present fourteen alternative ways of model transformation applied to build, deployment, or pipeline generation (see Fig. 2). While studies [19], [20], [21] and [22] offer novel approaches to model transformation for pipeline generation, and a Continuous Integration system, they lack a practical validation. Future research should prioritize tangible outcomes, such as real-world implementations and case studies, to validate the effectiveness of these approaches. [23], written in 2010, predates the widespread adoption of pipelines and Continuous Delivery concepts. However, it offers a valuable early example of a model-driven approach and demonstrates the application of model transformation techniques to generate build steps for CruiseControl.net using UML Class diagrams [24] from the software model. Studies [25] and [26] explore the application of Platform-Independent Models (PIMs) to address configuration and deployment challenges within software development. Studies [27] and [28] present innovative research on generating pipeline code for Jenkins and GitLab systems.

A comprehensive mapping of source elements (transformation input) into target elements (transformation output) is conducted for each transformation path presented in the examined studies to extract key insights and summarize the existing research since 2009. The transformation path is presented as a chain of identified elements (see Fig. 3 and Fig, 4):

- the predefined transformation source elements (input type, input domain abstraction, input model and input notation).
- the corresponding transformation elements (transformation input data, transformation tool, transformation output data).
- the obtained transformation target elements.

To enhance readability of the identified transformation paths, the obtained transformation target elements are divided into two schematic presentations:

1) Fig. 3 presents automation of build or deployment as the transformation output of source, resulting in the generation of build or deployment scripts within DevOps solutions.

2) Fig. 4 presents automation of pipeline as the transformation output, resulting in advanced pipeline generation within DevOps solutions.

To address RQ1 regarding the tools employed for build and deployment pipelines, it was necessary to differentiate between tools specifically designed for build and deployment processes and those that encompass the broader scope of pipeline orchestration. Some studies discuss build and deployment process automation, while others focus on the generation of pipelines specifically. While build and deployment processes have been foundational aspects of software delivery for some time, the concept of pipelines as a unified construct emerged more prominently around 2016, gaining significant traction with tools like Jenkins 2.0 [29]. As illustrated in Fig. 3, the mapping identifies six transformation paths focused on build processes and four transformation paths centered on deployment activities. The analysis revealed the dominance of the tool Jenkins, while others like Pluto, Maven, or CruiseControl.net demonstrated a more limited presence. The comparative analysis of Fig. 4 for pipeline generation shows a different result – a specific tool named Urano combines two related transformation paths, when Jenkins and GitLab each has one. When comparing Jenkins to GitLab, it is important to take into consideration that GitLab features pipelines only staring from 2015 [30]. Despite its

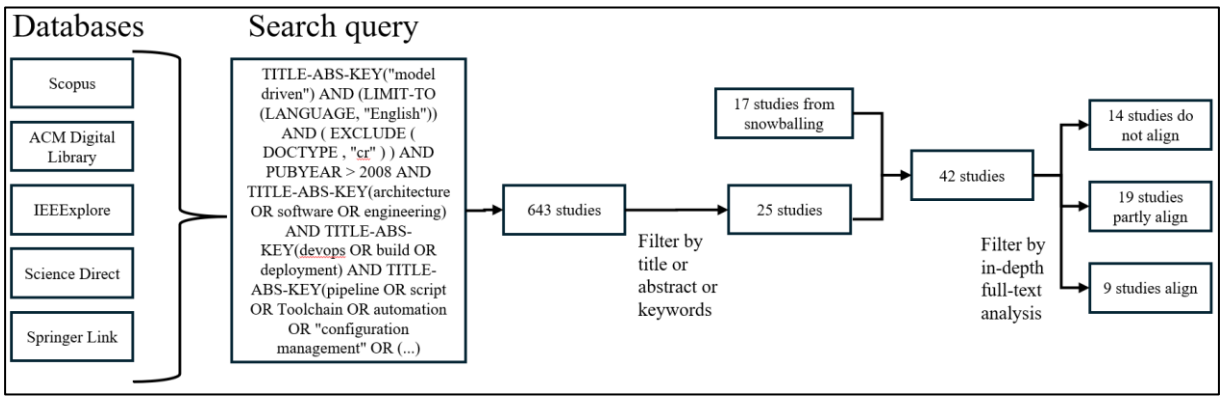

Fig. 2. The funnel of study selection process.

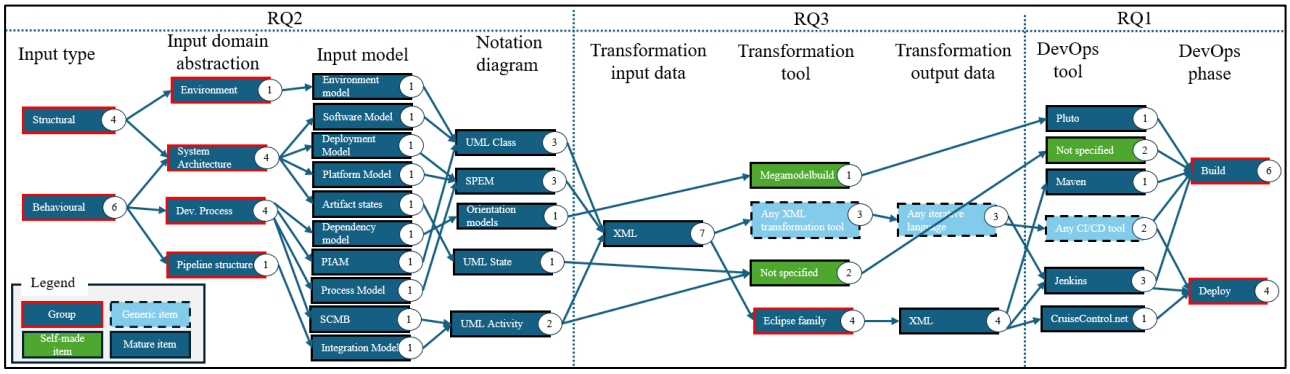

Fig. 3. Mapping of input model classifications, resulting in the generation of build or deployment scripts within DevOps solutions.

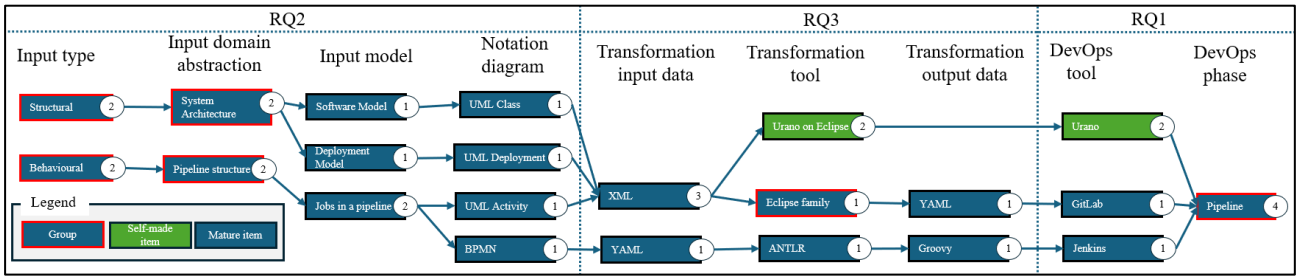

Fig. 4. Mapping of input model classifications, resulting in the advanced generation of pipelines within DevOps solutions.

prominence in the identified transformation paths, subsequent research has not documented the continued development or usage of the Urano tool beyond 2018.

To address the RQ2 regarding the input data and employed notation, a classification and abstraction of the identified transformation paths was necessary. Firstly, all input data had a structural or behavioral property. A comprehensive analysis of the input data led to its categorization into four distinct domain abstractions used as transformation input: Environment, System Architecture, Development Process, and Pipeline Structure. Following the classification of input data, various corresponding models were mapped to each class. To complete the analysis, the following notations used for visualizing the models are identified: UML Class diagram [24], SPEM [21], Orientation models [22], UML State diagram [24], UML Activity diagram [24], BPMN [31], and UML Deployment diagram [24]. Analysis of Fig. 3 and Fig. 4 shows that structural and behavioral input are used with similar frequency. Despite the diversity of models employed across different transformation paths, System Architecture and Development Process emerged as the predominant input domain abstractions, accounting for six and four transformation paths respectively. The Pipeline Structure input domain abstraction is exclusively utilized for modeling pipelines as output artifacts. Among the identified modeling notations, UML Class diagrams were used more frequently than others, while the UML Activity diagrams were often employed to define sequential process steps. The three transformation paths employing SPEM diagrams originated from a single study, limiting their generalizability and potential impact.

RQ3 aims to identify widely adopted tools that leverage model transformation techniques for advanced pipeline generation. While a handful of innovative research endeavors have emerged over the years, the Eclipse framework and its associated tools remain the predominant choice for the majority of transformation paths. XML emerged as the most common input data format in the analyzed transformation paths, with ten out of fourteen paths utilizing this standard. Three of the identified transformation paths were purely theoretical, lacking empirical validation. The practical implementations within the transformation paths demonstrated the generation of various pipeline artifacts, including Jenkins pipelines in both XML and Groovy formats, GitLab Runner pipelines in declarative YAML format, and Maven and CruiseControl.net pipelines in XML format. The singleton transformation paths involving Megamodelbuild and Urano were not accompanied by documented evidence of continued development.

Beyond the primary research questions, the analysis revealed a notable prevalence (eight studies of 28) of Java-based programming ecosystems across the majority of studies. While industry-specific trends are not readily apparent, the driving motivation behind many studies is the increasing complexity of pipeline automation and the shortage of skilled professionals within organizations.

V. Conclusion

Since researchers are trying to automate software development the formalization of DevOps solutions is crucial for enhancing the whole development process to be more efficient and scalable. One of the ways to achieve high enough level of any process automation is integration of Model-Driven Engineering principles. Application of model transformations seems promising also in providing a structural framework to formalize DevOps process. Therefore, it is vital to explore the current state of the art in the research area. The systematic literature review described in this paper aimed to investigate the application of model transformation to advanced generation of pipelines within DevOps solutions. By conducting a systematic literature review and analyzing relevant studies, key findings are the following:

- While steady progress has been made in the research of model-driven approaches for DevOps pipeline automation, further investigation is warranted to fully realize their potential.

- A variety of tools and frameworks are employed for model transformations, with Eclipse-based solutions being particularly popular.
- While the literature offers valuable insights into novel pipeline automation use cases, there is a need for more focused research in pipeline generation from System Architecture in combination with other models.
- To guide future research efforts and leverage existing knowledge, focusing on generation of pipelines within the Java ecosystem, with Jenkins or GitLab as foundational technologies for the DevOps solution, presents a pragmatic approach.

In conclusion, this research has provided valuable insights into the current state of application of model transformation to generation of pipelines within DevOps solutions in turn to summarize existing advancements in the area, as well as to identify potential directions for further development and starting innovative directions that can significantly improve automation process. While significant progress has been made, there is still room for further innovation and research to unlock the full potential of model transformation integration within DevOps solutions.

ACKNOWLEDGMENT

This research leading to these results has been supported by the EU Recovery and Resilience Facility within the Project Nr. 5.2.1.1.i.0/2/24/I/CFLA/003 "Implementation of consolidation and management changes at Riga Technical University, Liepaja University, Rezekne Academy of Technology, Latvian Maritime Academy and Liepaja Maritime College for the progress towards excellence in higher education, science and innovation" academic career PhD grant (ID 1017).

REFERENCES

[1] F. Kalota, D. Hua, and A. Behforouz, "The fourth industrial revolution: An overview of technologies and socio-economic implications," in *Industry 4.0 Key Technological Advances and Design Principles in Engineering, Education, Business, and Social Applications*, CRC Press, 2024, pp. 1–16.

[2] E. Salamah, A. Alzubi, and A. Yinal, "Unveiling the Impact of Digitalization on Supply Chain Performance in the Post-COVID-19 Era: The Mediating Role of Supply Chain Integration and Efficiency," *Sustainability (Switzerland)*, vol. 16, no. 1, p. 304, Jan. 2024.

[3] R. Jabbari, N. Bin Ali, K. Petersen, and B. Tanveer, "What is DevOps? A systematic mapping study on definitions and practices," *ACM International Conference Proceeding Series*, vol. 24-May-2016.

[4] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps" *IEEE Softw*, vol. 33, no. 3, pp. 94–100, May 2016, doi: 10.1109/MS.2016.68.

[5] F. Bordeleau *et al.*, "Preface to the 1st Workshop on DevOps@MODELS," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Sep. 2019, pp. 587–588.

[6] "What is a DevOps pipeline? A complete guide." Accessed: Aug. 27, 2024. Available: https://github.com/resources/articles/devops/pipeline

[7] J. G. Süß, S. Swift, and E. Escott, "Using DevOps toolchains in Agile model-driven engineering," *Softw Syst Model*, vol. 21, no. 4, pp. 1495–1510, 2022, doi: 10.1007/s10270-022-01003-2.

[8] "The Origins of DevOps: What's in a Name?" Accessed: Aug. 27, 2024. Available: https://devops.com/the-origins-of-devops-whats-in-a-name/

[9] "The Incredible True Story of How DevOps Got Its Name" Accessed: Aug. 27, 2024. Available: https://newrelic.com/blog/nerd-life/devops-name

[10] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017.

[11] "Nicole Forsgren, PhD, Accelerate: The Science of Lean Software." Accessed: Aug. 27, 2024. Available: https://nicolefv.com/writing

[12] "2023 State of DevOps Report." Accessed: Aug. 27, 2024. Available: https://cloud.google.com/devops/state-of-devops

[13] Marco. Brambilla, Jordi. Cabot, and Manuel. Wimmer, "Model-driven software engineering in practice," p. 166, 2012. Available: https://books.google.com/books/about/Model_driven_Software_Engineering_in_Pra.html?hl=lv&id=2tVu-wC4XkAC

[14] F. Zampetti *et al.*, "An empirical characterization of bad practices in continuous integration," *Empir Softw Eng*, vol. 25, no. 2, pp. 1095–1135, Mar. 2020.

[15] M. Wurster *et al.*, "The essential deployment metamodel: a systematic review of deployment automation technologies," *Software-Intensive Cyber-Physical Systems*, vol. 35, no. 1, pp. 63–75, Aug. 2020.

[16] C. Ponsard and V. Ramon, "Survey of Automation Practices in Model-Driven Development and Operations," in *Proceedings - 4th International Workshop on Bots in Software Engineering, BotSE 2022*, 2022, pp. 14–17.

[17] "Emerging Tech Impact Radar: DevOps." Accessed: Aug. 27, 2024. Available: https://www.gartner.com/en/documents/5577027

[18] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Inf Softw Technol*, vol. 55, no. 12, pp. 2049–2075, Dec. 2013.

[19] H. Steudel, R. Hebig, and H. Giese, "A build server for model-driven engineering," in *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, in MPM '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 67–72.

[20] A. Bartusevics and L. Novickis, "Model-based approach for implementation of software configuration management process," in *MODELSWARD 2015 - 3rd International Conference on Model-Driven Engineering and Software Development, Proceedings*, 2015.

[21] A. Colantoni, L. Berardinelli, and M. Wimmer, "DevOpsML: towards modeling DevOps processes and platforms," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, in MODELS '20. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3417990.3420203.

[22] P. Stevens, "Connecting software build with maintaining consistency between models: towards sound, optimal, and flexible building from megamodels," *Softw Syst Model*, vol. 19, no. 4, pp. 935–958, Jul. 2020, doi: 10.1007/S10270-020-00788-4/FULLTEXT.HTML.

[23] V. Garci'a-Di'az, J. P. Espada, E. R. Nunez-Valdez, B. C. P. G-Bustelo, and J. M. C. Lovelle, "Combining the continuous integration practice and the model-driven engineering approach," *Computing and Informatics*, vol. 35, no. 2, pp. 299–337, Jan. 2010.

[24] "Unified Modeling Language Specification Version 2.5.1." Accessed: Aug. 27, 2024. Available: https://www.omg.org/spec/UML/2.5.1

[25] A. Bartusevics and L. Novickis, "Models for implementation of software configuration management," in *Procedia Computer Science*, 2015, pp. 3–10.

[26] L. F. Rivera, N. M. Villegas, G. Tamura, M. Jiménez, and H. A. Müller, "UML-driven automated software deployment," in *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*, in CASCON '18. USA: IBM Corp., 2018, pp. 257–268.

[27] T. Tegeler, S. Teumert, J. Schürmann, A. Bainczyk, D. Busch, and B. Steffen, *An Introduction to Graphical Modeling of CI/CD Workflows with Rig*, vol. 13036 LNCS. 2021.

[28] T. F. Düllmann, O. Kabierschke, and A. van Hoorn, "StalkCD: A Model-Driven Framework for Interoperability and Analysis of CI/CD Pipelines," in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Sep. 2021, pp. 214–223.

[29] "What's New in Jenkins 2.0." Accessed: Aug. 27, 2024. Available: https://developers.redhat.com/blog/2016/08/24/whats-new-in-jenkins-2-0-2

[30] "Gitlab-runner Release History." Accessed: Aug. 27, 2024. Available: https://gitlab.com/gitlab-org/gitlab-runner/blob/main/

[31] Business Process Model and Notation Specification Version 2.0.2." Accessed: Aug. 27, 2024. Available: https://www.omg.org/spec/BPMN

https://doi.org/10.1109/ITMS64072.2024.10741933
Electronic ISBN: 979-8-3315-3383-0. Print on Demand (PoD) ISBN: 979-8-3315-3384-7.