FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A model-driven approach for DevOps

Hugo Afonso da Gião

Programa Doutoral em Engenharia Informática

Supervisor: Jácome Cunha

Co-supervisor: Rui Pereira

January 3, 2023

A model-driven approach for DevOps

Hugo Afonso da Gião

Programa Doutoral em Engenharia Informática

January 3, 2023

Abstract

DevOps is a combination of tools and methodologies that aims at improving the software development, build and deploy processes by shortening this lifecycle and improving software quality. Despite its many benefits it still presents many challenges when it comes to its ease of use and accessibility. One of the reasons is the tremendous proliferation of different tools, languages, and syntaxes which makes the field quite difficult to learn and keep up to date.

This document contains a survey of the relevant work in the intersection of both the fields of DevOps and Model-driven development. From this survey we found that some work has been done in joining these fields, however, most of those works focus on a particular domain, task, or specific tools but do not offer a general approach that abstracts from the concrete technologies used.

Our goal with this project is to study a model-driven approach that can abstract the particularities of the different kinds of tools and processes for the different parts of the DevOps process. Moreover, we will also use model-driven techniques to allow users to express their pipelines without the need to know all the implementation details (e.g. configuration files) of all the tools they need to use.

Keywords: DevOps, IT, Model-driven engineering, Model-driven software engineering, Model-driven development, Models

Resumo

DevOps é uma combinação de ferramentas e metodologias cujo objetivo consiste em melhor o processo de desenvolvimento, construção e integração de software diminuindo o tempo deste e melhorando a qualidade do software produzido. Apesar de trazer vários benefícios existem vários obstáculos a adoção deste processo especialmente devido a dificuldades de uso e acessibilidade. Uma das razões para tal consiste na proliferação de diferentes ferramentas, linguagens e sintaxes o que dificulta o que torna conhecimentos previamente adquiridos obsoletos ou insuficientes.

Este documento contém um estudo sobre o trabalho relevante sobre a interceção das áreas de Model-driven engineering e DevOps. A partir deste estudo concluímos que foi realizado algum trabalho na intercessão destas áreas. Apesar disso encontramos que os focos destes trabalhos é em domínios, tarefas ou ferramentas especificas e não oferecem uma metodologia que permite uma abstração das tecnologias utilizadas.

O nosso objetivo com este projeto consiste em criar uma metodologia model-driven que permita abstrair das diferentes ferramentas e processos de DevOps. Para mais também pretendemos utilizar técnicas model-driven para permitir a utilizador criarem pipelines sem necessitarem de ter conhecimento específico sobre as ferramentas utilizadas (por exemplo não terem de tratar de ficheiros de configuração).

Keywords: DevOps, IT, Model-driven engineering, Model-driven software engineering, Model-driven development, Modelos

Acknowledgements

This work is financed by the ERDF - European Regional Development Fund, through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme under the Portugal 2020 Partnership Agreement, and by National Funds through the FCT - Portuguese Foundation for Science and Technology, I.P. on the scope of the UT Austin Portugal Program within project BigHPC, with reference POCI-01-0247-FEDER-045924.

Contents

List of Figures

List of Tables

Abbreviations

MDE	Model-driven engineering
MDSE	Model-driven software engineering
IT	Information technology

Chapter 1

Introduction

In 2022 the Information technology (IT) market accounted for $9,325.69 billion dollars worldwide [13]. It has been shown that 50% of IT costs are related to the maintenance and upkeep of existing systems and from those 50% correspond to emergencies, unplanned work, and changes [23]. This happens partially because in most organizations IT and development teams' goals often conflict since instead of working for a common goal, most organizations' development teams are responsible for reacting to and answering rapid changes in markets and customers' needs, while the IT teams role is to provide the customer with a secure, reliable and stable experience. This dynamic leads to "technical debt", that is decisions made over time lead to increasingly harder problems and slower operations and thus impede the achievement of the organization's goals [23].

Inspired by the Lean practices used in the manufacturing industry which significantly increased the number of orders fulfilled on time and lead to companies adopting those practices dominating the market, DevOps comes as a solution to the problems of the IT industry, this methodology comes from the union of the worlds of development (Dev) and IT operations (Ops), it is a means for software development and delivery by using various tools and techniques that integrate this two worlds. The integration of these two fields is achieved through automated development, build, deployment, and monitoring. One of the goals is to achieve reliable, secure, fast, and continuous delivery of software [23].

In the IT industry similarly to other industries the bar for development speed and software quality is constantly increasing, first with the adoption of Agile methodologies and then with the use of DevOps and cloud computing. Similarly to the Lean methodologies used in other industries the use of DevOps brings to organizations a considerable competitive advantage, this includes faster shipping times, more resilient, productive, and dynamic teams, better reaction to fast changes, improved software security, and better well being and job satisfaction and well being among employes [23].

However, organizations and developers face many obstacles when adopting DevOps, including changes in the architectural organization, dealing with problems in older infrastructure, and the integration of different DevOps tools [17]. Other problems faced by developers are in learning and using the specific DevOps tools which can be overwhelming and can decrease the improvements

expected with the adoption of DevOps [21].

Model-driven engineering (MDE) has been used to cope with numerous problems in computer science, from complexity to the increase of compatibility between different systems, to aid in the design process, or in improving the communication between different teams and teams members [30]. Models are in many cases visual languages or can be represented as such, and we believe they are a good approach to removing the complexity associated with DevOps, making it easier to adopt and use. Thus, in this project, we intend to use MDE to cope with the complexity of DevOps and to make it accessible to more users and in particular users with a less strong background in this kind of technology.

1.1 Research Goals and Contributions

1.1.1 Research questions

Considering our motivation and goals our thesis aims to answer the following questions:

- **RQ1** - *Is it possible to design a modeling language that is capable of specifying the DevOps different phases?*
  Currently there is a lack of tools that allow users to do various DevOps steps in a way that abstracts from the technology underlying the process, our goal with this question is to study if there is a better process of modeling DevOps tasks that offers both the expressiveness of using the tools directly and is approachable and requires a low learning curve for new and inexperienced users.
- **RQ2** - *Is it possible to design a modeling language that is capable of specifying the DevOps for different domains?*
  Our next goal is to know if the language created can be used in different computing domains while being generic and maintaining the features that are attractive for novice users.
- **RQ3** - *Is it possible to define a methodology to integrate the different phases of DevOps and turn it into a unified process?*
  Our next goal is to evaluate if the different languages can be tied together into a unified DevOps process.

1.1.2 Venues and research groups

Here we present different venues for which we plan to submit our publications and research groups working in areas close to the area of this work. Table 1.1 contains some workshops relevant to the related fields of this work including *low code@MODELS* a workshop primarily focused on applying MDE to the low code field. The other workshop *HAPi DevOps@VL/HCC* is a new workshop whose focus is the intersection of the fields of Human-centered computing and DevOps.

The conferences for which we plan on submitting our work can be seen in Table 1.2, this table contains the name of the conferences, their field, and Core rank, these conferences are in both the

Venue	Field
low code@MODELS	Software engineering
HAPi DevOps@VL/HCC	Human-centered computing

Table 1.1: Workshops

fields of Software engineering and Human-centered computing and have ranks ranging from *B* to *A*∗.

Venue	Field	Core rank
ICMT	Software engineering	B
VL/HCC	Human-centered computing	B
CHI	Human-centered computing	A*
ICSE	Software engineering	A*
MODELS	Software engineering	A

Table 1.2: Conferences

In terms of journals, those can be seen in Table 1.3, this table contains the journals' name, their fields, and impact factor. Once again we plan on making our publications in the fields of Software engineering and Human-centered computing. These journals have impact factors from 2.351 for the lowest to 6.226 for the highest.

Venue	Field	Impact factor
TOCHI	Human-centered computing	2.351
TSE	Software engineering	6.226
JSS	Software engineering	2.829

Table 1.3: Journals

When it comes to the research groups we found research groups associated with researchers whose work focused on the fields of MDE and DevOps. Most of the people associated with these groups are doing work in one or both of those fields or have done significant work in the past. Table 1.4 contains those groups as well as their fields of research.

1.1.3 Contributions

With this work we make the following contributions:

- Modeling frameworks and tools for specific DevOps processes.
- Modeling frameworks and tools capable of defining whole DevOps pipelines without resorting to concrete technologies.

Group	Fields
Juan de Lara	Software Engineering MDE Domain-Specific Languages
Davide Di Ruscio	Software Engineering MDE Recommender Systems
Manuel Wimmer	Software Engineering MDE Web Engineering Industrial Engineering Refactoring
Gregor Engels	Software Engineering MDE
Nicole Forsgren	DevOps productivity IT use impacts knowledge management Knowledge Management Systems
Jez Humble	Software engineering Software systems

Table 1.4: Reserach groups

The already published publications associated with our work are the following:

- A model-driven approach for DevOps
  Hugo Gião
  IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'22) [18]
  Participation in VL/HCC'22 graduate consortium, this publication contains a preliminary version of the work plan for this thesis.

Future publications we plan on doing are as follows:

- A survey of the current work that has been done in the intersection of the fields of DevOps and MDE, this survey will be in part based on the contents in Chapter 3. This publication was submitted to the *HAPi DevOps@VL/HCC* conference. We plan on improving this version and submitting it to one of the previously shown conferences.
- Several publications retaining to the using MDE approaches to various phases of the DevOps process. Our goal here is to show the inner workings of these tools and frameworks and their validation with users.
- A publication retaining the development of whole DevOps pipelines using our MDE approach. Our goal with this publication is to expose the inner working of our framework or tool and showcase the results of an empirical evaluation.

1.2 How to Read this document

This document is organized in the following chapters:

- In Chapter 2 we give an overview of the fields and some key concepts of DevOps and MDE.
- Then in Chapter 3 we showcase the work that has been done in applying MDE techniques in the field of DevOps. This chapter includes the methodology used and detailed analysis and commentary on the results.
- We then proceed to Chapter 4 with our work plan. In this chapter, we expose the different tasks of our work plan as well as a detailed timeline.
- Finally in Chapter 5 we present some conclusions and future work.

Chapter 2

Background

2.1 DevOps

2.1.1 What is DevOps

In most organizations there is a conflict between development and IT teams, this results in slower times to markets and decreased efficiency, increased outages, and the amount of technical debt, worsening these problems over time. These problems happen often because development and IT teams have opposed goals since the goals of IT teams consist in providing the customer with a stable, secure, and reliable product while the development teams have to deal with the increasing changes in the market and deploying changes into production as quick as possible. These conflicts often lead to difficulties and often the lack of achievement of business goals for said organizations as well as leading to poor software, service quality, bad customer outcomes, and workarounds [23].

Most organizations are not able to deploy production changes fast or frequently, this happens because their process of deployment often relies on a slow and manual process. This process is often prone to errors and often involves outages, tiredness, and last minute fixes. We live in an age where competitive advantage requires fast time to market, high service levels, and relentless experimentation. These organizations thus find themselves at a significant competitive disadvantage that is due in large part to their inability to fix those problems at the core of their development process [23].

DevOps comes as a solution to those problems, the DevOps word comes from the combination of Development (Dev) with IT Operations (Ops). Indeed, this term is used to describe a culture where these two realms are no longer separated as in the past but are now just one. The integration of these two fields is achieved through automated development, build, deployment, and monitoring. One of the goals is to achieve reliable, secure, fast, and continuous delivery of software, to improve productivity in businesses and workers' well-being. Moreover, software engineers that use DevOps are more resilient and equipped for fast changes [23].

2.1.2 DevOps lifecycle

The DevOps process is made up of different phases comprising its lifecycle, these faces are necessary to guarantee the full range of benefits associated with the use of DevOps such as improved speed, software quality and more dynamic teams, the phases comprising this lifecycle are as follows [39]:

1. **Continuous development**

   The phase of continuous development involves the tasks of planning and developing software. This process is an application of the agile methodology to the world of software development, that is the whole software gets broken down into smaller pieces, and software is updated in smaller batches after being tested [39]. Tools used for this phase include AWS CodeDeploy, Jenkins, and GitLab [1].

2. **Continuous integration**

   This phase is at the core of the DevOps lifecycle and affects the process of committing new changes in the source code when doing continuous integration developers are forced to do more frequent software releases and these more frequent releases in conjunction with the use of various testing techniques such as unit testing and integration testing and other quality assurance practices such as code reviews allow them to be more secure and stable [39]. Continuous integration tools include Jenkins, TeamCity and GitLab [2].

3. **Continuous testing**

   Continuos testing involves the use of automatic testing tools such as Selenium, JUnit, Jenkins, and Docker [3]. This process saves organizations time and increases productivity [39].

4. **Continuous deployment**

   Continuous deployment affects the process of deploying new code to the production environment. This process is made automatically and reduces the need of having manual and planned releases since new changes to the source code are put into production after the process of integration and testing. This phase makes extensive use of container technology such as Docker to make use of standardized environments [39]. Other tools commonly used in this process include Jenkins and AWS CodeDeploy [4].

5. **Continuous monitoring**

   This phase entails performance monitoring and recording of the application, including potential errors such as high memory usage and networking problems. This monitoring allows development teams to find and correct errors earlier and substantially reduce IT costs. Some

[1] https://www.softwaretestinghelp.com/continuous-deployment-tools/
[2] https://smartbear.com/blog/top-continuous-integration-tools-for-devops/
[3] https://www.softwaretestinghelp.com/continuous-testing-tools/
[4] https://www.softwaretestinghelp.com/continuous-deployment-tools/

tools for this phase include Akamai mPulse, AppDynamics, and BMC Helix Operations Management [5].

6. **Continuous feedback**

   This phase comprises the gathering, analysis, and use of feedback coming from the client's end, this information includes performance and errors as well as customer feedback [39].

7. **Continuous operations**

   This phase includes the automation of all operations processes that help developers automate releases, detect issues faster and improve software products. This practice allows for faster delivery times and higher quality software [39].

2.2 Model-driven software engineering

Humans have a limited capacity to hold and process information, thus to understand the world around them they use various cognitive processes that change their perception of the world, one of those processes is the process of abstraction, which is the capability of finding similarities between various events and then generating a mental model that can represent reality using processes such as generalization, classification, and aggregation. Those processes are used instinctively since birth and are used in people's everyday lives. Abstractions are used in science and technology where this process is often referred to as modeling. A model can be defined as a simplified or partial reality intended to accomplish a certain task or reach an agreement. A model by its definition will always be an incomplete representation of the world [7].

Models have applications in different scientific fields such as physics or chemistry such as modeling atoms and the uniform motion model. These applications despite being far from perfect representations of the world still have many uses such as allowing people to understand the core concepts of the field, for education purposes, and as the basis of more complex theories. Mathematics also makes use of modeling, more notably models are often used to make various predictions about the world. Reduction and mapping are the two main features of models, reduction states that a model only reflects a (relevant) selection of the original's properties, and mapping states that models are based on an original individual, which is abstracted and generalized to a model. Models can also serve other purposes such as descriptive purposes, that is describing the reality of a system, and prescriptive purposes, this being determining the scope and details at which to study a problem [7].

Models have also become prevalent and increasingly useful for software development, this is due to many reasons such as the growing complexity of software systems and artifacts and the fact that those often need to be discussed with people with various levels of involvement in the system, the pervasiveness of software in peoples everyday lives and the frequent need to add features or create new pieces of software to satisfy customer demand. Another factor for the use of models in

[5] https://www.chaossearch.io/blog/continuous-monitoring-tools

this industry is the interaction with non-developers for which the abstraction from technical details helps understand the system [7].

The software industry uses models in many different ways including using them as sketches for communication purposes, as blueprints as a way to detail the system before its implementation, and as programs to develop the system. Furthermore, Model-driven software engineering (MDSE) has many use cases, the primary use case being software development automation also known as Model-driven development, software development automation consists of starting with a high-level representation of the desired system with the desired behavior and through a set of intermediate steps arriving at a piece of software conforming to the desired specification, some of its use cases include code generation. Another use of MDSE is system interoperability. this is guaranteeing the compatibility of the information exchanged between two or more systems. Reverse engineering is a task that also benefits from MDSE since older software systems often use obsolete technology and using MDSE developers can use these techniques to represent the older system and its requirements and then use the created model to generate a new version of the system [7].

Modeling languages are one of the central components of MDSE. Modeling languages are the tools that allow for concrete representations of the mental model, modeling languages often consist of graphical representations and textual representations and feature a formal design that forces compliance with their syntax. These can be divided into two classes Domain-specific languages, languages that are built for a specific domain or function, such languages include HTML and SQL. The other group is General-Purpose Modeling Languages, these languages can be applied to any domain for modeling purposes, one example of those is UML [7].

Since models are at the core of MDSE, the next step of abstraction is to create models capable of representing those models. Since models are representations of the real world we can define models (called meta-models) to highlight the properties of the given models, meta-models can be used to define languages since they are capable of modeling the class of models that can be represented by the language [7].

Chapter 3

Related work

3.1 Research Method

Our research method consisted in querying several digital libraries, namely *Springer*, *ScienceDirect*, *ACM* and *IEEE Xplore*, for papers containing both topics of interests. Thus, we searched these libraries using the query *DevOps AND "model-driven"* in the entire text of the papers. The number of papers found can be seen in Table 3.1. After gathering the papers containing the search terms we manually filtered, by screening the content, the papers whose work involved the application of MDE techniques in various areas of DevOps. After filtering we ended up with 30 publications. Finally, we read the 30 papers and organized them according to their computing field and DevOps phase. The domain was extracted directly from each paper and for the phases, we used an existing classification [14]: lifecycle, these being development, integration, testing, monitoring, feedback, deployment, and operations. In the next section, we detail our findings.

Table 3.1: Number of papers found in each library

Venue	Development
Springer	406
ScienceDirect	300
ACM digital library	102
IEEE Xplore	31

3.2 Findings

The distribution of the publications according to the phases and domains is presented in Table 3.2. In this table, each publication is represented by its first author's initials and the year of publishing. In the following sub-sections, we discuss the publications in the context of the domain where they have been proposed.

Domain/Phase	Development	Integration	Testing	Monitoring	Feedback	Deployment	Operations
Cloud	J.S, 2018 [28]	J.S, 2018 [28]	R.K, 2020 [24]		M.G, 2015 [20]	H.A, 2016 [1] M.G, 2015 [19] D.W, 2016 [35] J.S, 2019 [29] D.W, 2016 [36] N.F, 2018 [16] H.A, 2018 [2] S.C, 2020 [10] M.A, 2016 [3] H.B, 2019 [6] G.C, 2020 [8] J.W, 2016 [37] J.S, 2019 [27]	
IoT	B.E, 2020 [15]	B.E, 2020 [15]					
Safety-critical systems			B.M, 2019 [25]	B.M, 2019 [25]	B.M, 2019 [25]		
Cyber-physical systems	B.C, 2020 [12] J.H, 2020 [22]	B.C, 2020 [12] J.H, 2020 [22]	B.C, 2020 [12] J.H, 2020 [22]	B.C, 2020 [12] J.H, 2020 [22]	B.C, 2020 [12] J.H, 2020 [22]	B.C, 2020 [12] J.H, 2020 [22]	B.C, 2020 [12] J.H, 2020 [22]
Blockchain	W.J, 2021 [34]	W.J, 2021 [34]	W.J, 2021 [34]	W.J, 2021 [34]	W.J, 2021 [34]	W.J, 2021 [34]	W.J, 2021 [34]
Machine-Learning	W.J, 2020 [33] N.B, 2021 [4]						
Big Data	G.S, 2015 [9]	G.S, 2015 [9]		G.S, 2015 [9]	G.S, 2015 [9]	G.S, 2015 [9]	
Generic	F.B, 2020 [5] A.C, 2020 [11]	F.B, 2020 [5] A.C, 2020 [11]	F.B, 2020 [5] A.C, 2020 [11]	F.B, 2020 [5] A.C, 2020 [11]	F.B, 2020 [5] A.C, 2020 [11]	F.B, 2020 [5] A.C, 2020 [11]	F.B, 2020 [5] A.C, 2020 [11] K.T, 2022 [32]

Table 3.2: Publications organized according to their DevOps lifecycle phases

3.2.1 Cloud

As far as the publications relative to cloud computing we have found that the majority of them focus on the deployment phase of the DevOps lifecycle. More precisely, most of those articles aim at creating a platform-agnostic way of deploying code and doing different operations in different cloud services (e.g. Amazon AWS[1], Google Cloud[2]) and improving reusability between those services as well as hiding technical details from the users [1, 19, 35, 29, 36, 16, 2, 10, 3, 6, 8, 37, 27].

These include by-passing vendor lock-in issuea with auto-scaling configurations [1], optimizing runtime capacity allocation across different vendors [19], managing resources between different providers by creating abstraction of cloud resources representations and orchestrations [35], providing an alternative to manual scripting and a way to do infrastructure provisioning using a domain-specific language that converts to code for different providers [29], providing a model-driven notation to visually represent, monitor and control cloud resource configurations [36], offering a notation for specifying both the provisioning and deployment of multi-cloud applications, and run-time environment for their continuous provisioning, deployment, and adaptation [16], connect a cloud platform independent model of services with cloud specific operations [2], graphically design cloud applications and deploy and manage them at runtime [10], allow quick deployment

[1] https://aws.amazon.com/
[2] https://cloud.google.com/

and redeployment of cloud applications for the purpose of continuous improvement [3], translating designed artifacts into DevOps specific artifacts [6], creating and managing applications based on serverless computing [8], enabling the seamless and interoperable integration of different approaches to model and deploy application topologies [37], and support multi-cloud infrastructure provisioning [27].

Other projects offer contributions outside of the deployment phase. These include proposing a model-driven approach to abstract and automate a continuous delivery process of cloud resources in development, testing, and production environments using model-driven techniques and DevOps, which touches the development, integration, testing, feedback, deployment, and operations phases [28]. Another project that reached the same areas consists of a conceptual security model to facilitate the adoption of DevSecOps[3].) for the business processes over the cloud [24].

Regarding the modeling language used most of these projects adapted existing languages already used in the cloud computing domains such as TOSCA [28, 37, 27, 8, 6, 3, 10, 29]. We also found that the use of meta-models was prevalent in these publications [28, 37, 27, 6, 3, 10, 16, 29].

3.2.2 IoT

In the Internet of things (IoT) space, we found just one publication whose goal was to create a meta-model of the IoT capable of standardizing current and future IoT architectures. This project touches on the phases of development and integration [15].

3.2.3 Safety-critical systems

for safety-critical systems, we found one article whose work reached the phases of testing, feedback, and monitoring. In this work, the authors propose a modeling framework compatible with the DevOps principles of continuous testing and continuous integration for the design of safety-critical systems while obeying industry standards. In this work, the authors define their modeling language and make use of meta-models. The authors also expressed that the ability to reason and make proofs over the models was one factor for their use of model-driven engineering [25].

3.2.4 Cyber-physical systems

In terms of cyber-physical systems, we found two publications whose focus was to model the whole development lifecycle of these systems. Both publications created their languages and make use of meta-models. As for safety-critical systems, in both papers, the authors express that the ability to reason and make proofs over the models was one factor in their use of model-driven engineering [12, 22].

[3] DevSecOps is DevOps, but including security controls to provide continuous security assurance [26]

3.2.5 Blockchain

In the blockchain domain, we found a publication whose work consisted in creating a model-driven approach to the creation of smart contracts. These works affect all the phases of the DevOps lifecycle. The authors created their language and made use of meta-models. Once again, the ability to reason and make proofs over the models was one factor for their use of model-driven engineering [34].

3.2.6 Machine learning

In this space, we found two publications whose goals are to create methodologies and frameworks to build machine learning and autonomous systems using model-driven development. The authors of both articles created their language [33, 4], but only one publication made use of meta-models [33].

3.2.7 Big data

Regarding big data, we found a publication whose goal was to use model-driven engineering to create a framework that allowed for the creation of data-intensive software that supports reliability, efficiency, and safety requirements. The language created by the authors is of their own and they make use of meta-models [9].

3.2.8 Generic

We found three publications whose work is not dedicated to any particular domain. Two of these publications span the whole DevOps lifecycle [5, 11], while the other is an industrial case study with the goal of deriving requirements for a model-based approach for DevOps by studying the DevOps process used by this company [5]. The goal of the other article [11] is to build upon those requirements to create a conceptual framework for modeling and combining DevOps processes and platforms. In their approach developers can use their framework to create processes and platforms, processes being an application or service and platform a sequence of processes. Their approach however is dependent on the tools and platform used and does not offer a generic approach to doing each DevOps phase and the lifecycle. Both these projects have their languages and use meta-models [11].

We also found another project whose goal was to aid the management side of software projects by applying model-driven techniques. Their work touches on the operations phase of the DevOps lifecycle [32].

3.3 Summary

From Table 3.2 we can see that the majority of papers actuate in the domain of cloud computing and more specifically in the deployment phase. Nevertheless, we also found that some papers

tackled the other phases of the same domain, but in less quantity. We also found that of the papers relating to the deployment phase of cloud computing most of them focus on interoperability between cloud services. For the other domains, we found two papers for IoT, one for blockchain and another for safety-critical systems.

We also learned that the use of meta-models was prevalent in these projects and found that the ability to reason and create proofs over models was a factor in using model-driven approaches in the spaces of safety-critical systems, cyber-physical systems, and blockchain.

In terms of generic approaches, we found that some authors acknowledge the need for a model-driven approach to the DevOps process and that existing frameworks are still reliant on specific technologies.

Overall, we found a trend of growing interest over the past two years in publications in domains outside of cloud computing, since we can see that, while the vast majority of publications in this domain happened before 2020, the opposite is true for the other domains.

Chapter 4

Work plan

4.1 Tasks

Considering the work previously done on the subject, our goals, and motivations we devised the following tasks for our work, this list contains both tasks and the subtasks related to our work, the tasks are represented by numbers and subtasks by letters, the subtasks containing bold text represent milestones:

1. Analysis and study of the DevOps process

   Before initializing any work we need to gain a thorough understanding of the DevOps process as well as the work that has been done at the intersection of the fields of DevOps and MDE. To do so beyond reading about the subjects of DevOps and MDE we plan on surveying the intersection of these two fields in order of identifying what has been done and possible shortcomings which we can improve upon, we plan to write and submit an article with our findings to one of the conferences in Chapter 1. After this initial work, our goal is to approach industrial partners and find the different techniques used to implement DevOps as well as the most notable problems companies have in their daily lives. Rui Pereira, the co-supervisor of this project, is a team leader at OutSystems, the company responsible for one of the biggest low-code platforms, which grants us special access when it comes to possible studies and even access to internally gathered data. Moreover, we have been working with Magycal for some time now, a company that develops software for sports fans, and also uses DevOps in their work.

   We will meet with their DevOps teams using semi-structured interviews [38] to gather initial industrial evidence of the current trends in DevOps. Moreover, we will also try to understand previous processes they had and what they think the future might be so we can understand the evolution of DevOps in practice and prepare our methodologies for future changes. Although we will start with these two companies, we will also contact others to collect significant information about the state-of-the-art of DevOps in an industrial context. This work will provide us with sufficient material to write a scientific article that we will submit

to the industrial track of a conference showcased in Chapter 1, since this paper would be mainly focused on software engineering, a conference in those fields would prefered. This work will also give plenty of information so we can make decisions based on empirical evidence.

From this we created the following subtasks:

(a) Research relevant works and read the most important literature concerning the fields of DevOps and MDE.

(b) Create a survey about the work at the intersection of those two fields

(c) **Write a scientific article with contents of the survey**

(d) Define a semi-structured interview based on scientific articles describing DevOps (with a special focus on industrial reports)

(e) Contact organizations and collect information

(f) Analyze typical DevOps workflows

(g) Find biggest difficulties companies face

(h) **Write a scientific article reporting the main findings**

2. Design and implementation of a generic framework for the DevOps tasks

   DevOps is characterized by an enormous amount of tools available for each particular task. This implies DevOps teams need to learn quite some tools, languages, and notations to be able to define concrete DevOps processes. This is quite challenging given the number of tools, not stable as tools are created/changed frequently, and difficult to integrate new members as there are too many concepts to be learned before starting to work. Thus, our goal is to create a framework capable of specifying the different parts of the DevOps process independently of the concrete technological choices of each DevOps team.

   To this end, we will study model-driven methodologies and propose a meta-model that can be used to generate different models for the three parts of the process we mentioned before [14]. This meta-model will need to include concepts from development, build management, and deployment. Each of these models can then be instantiated in the different tools. For instance, given a model for build management, one can instantiate it in tools such as Gradle or Ant. The intent is that our framework can generate the necessary code or configuration files based on the model for existing tools, or even for tools that may appear in the future, being only necessary to create the necessary mapping between the model and the new tool's concrete syntax.

   Such a framework will allow DevOps teams to have a single specification of the different parts of the process and generate concrete instances of the tools they choose for each project or even in different periods.

Each of the models and the meta-model will be validated using case studies collected in the initial part of this project. This validation will be mostly an applicability validation, that is, we will evaluate if the models are sufficient to represent a significant amount of tools. After this validation, we can create a meta-model. These (meta-)models will then be implemented using a framework such as Epsilon, we plan on using this framework since it offers many capabilities such as automatically generating Eclipse IDE's for the specified language, interoperability with Java, and speed when compared with other tools, furthermore this framework is widely used in the industry by an organization such as NASA, BOSH, ORACLE, and IBM [31], after its implementation, we plan on empirically validate the language with DevOps teams, possibly from OutSystems.

From this we created the following subtasks:

(a) Design a model for different DevOps tasks (e.g development, build automation, and deployment management)

(b) Design a meta-model based on the models created

(c) Implementation of the (meta-)models in a framework

(d) Empirical evaluation of implementation's usability

(e) **Write a scientific article with the models and their evaluation, as well as a tool demo article**

3. DevOps pipelines without concrete technologies in mind

   In this part of the work, we will study how to integrate the different steps of the DevOps process into a unified process. Each team has its way to integrate the different parts of the process and thus we need to find a way to allow them to define arbitrary combinations of the DevOps steps.

   We will study how to extend the previously proposed models to include concepts from pipelines. We will also study other languages such as the Business Process Modeling Notation (BPMN) which is quite used in different industrial contexts and thus may be more appealing for industrial practitioners.

   Once more, the language we will propose will be validated for its completeness by applying it to several case studies. Nevertheless, we will evaluate it empirically with practitioners so we can assess its usability (efficiency, effectiveness, and satisfaction) by real users.

   From this we crteated the following subtasks:

   (a) Design a language to specify DevOps pipelines

   (b) Validate the language's applicability

   (c) Implement the language in the framework started in Part 2

   (d) Evaluate the usability

(e) **Write and scientific and a tool demo article**

4. Thesis writing

4.2 Timeline

The timeline for this project can be seen in Figure 4.1, in the *Task* column of this figure orange cells represent tasks, white cells represent the subtasks that aren't milestones and green cells represent the milestones. We plan to start with task number 1 in March 2022 and end it in December 2022, during this task we plan on spending six months on the various tasks related to reading literature about the fields relevant to our work, doing a survey concerning the work done at the intersection of those fields and writing a paper on the subject, during the final months of this period we plan on starting the process of contacting organizations and developing the survey, we then plan to do the actual studie and of course, finalizing the writing of an article on the subject.

After this we plan on starting task number 2 in January 2023 and ending on November 2023, during this task we plan on building models for different DevOps tasks, creating a meta-model that can describe those models and implement them in a framework, our goal is then to validate the models with the users and finally write an article relating to the subject.

Task number 3 will start when task 2 is still ongoing since the created models and meta-models might need to be changed so they can accommodate the concept of pipelines. We plan on beginning this task on June 2023 and ending it in February 2025, this task is by far the longest, and even though it will be partially done concurrently with task 2 this time is justified since this task contains the most challenging and influential portions of this project. During this task we plan on creating the language, then validating its applicability with existing workflows, and then implementing it in the framework used for the languages created in task number 2, we then evaluate the usability with users and finally plan on writing a paper about the subject.

Finally, during the whole process, we plan on iteratively writing the thesis document.

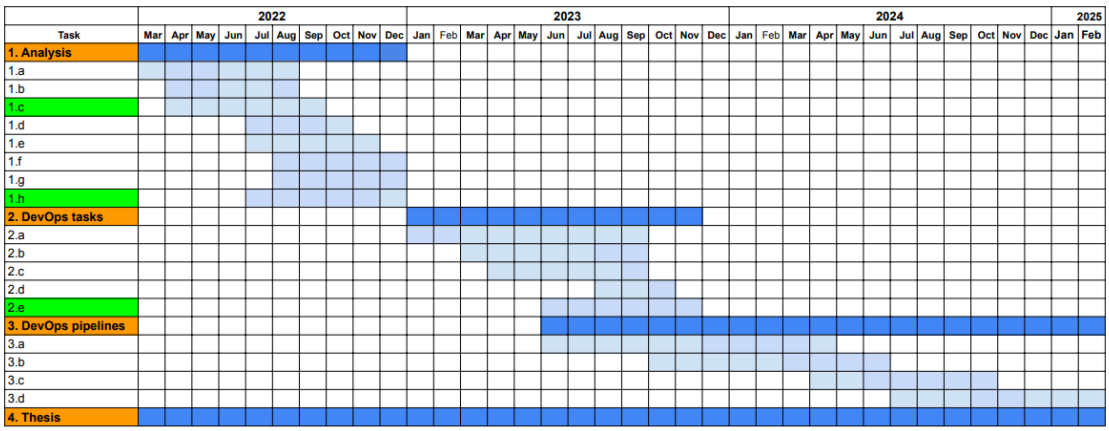

Figure 4.1: Detailed timeline for our work plan

Chapter 5

Conclusions and Future Work

DevOps is a challenging subject and despised its wide use, applications, and undeniable benefits, it is still not used by a considerable number of organizations, mostly due to the complexity associated with the use and pairing of the different DevOps tools. MDE has already been of use for various tasks of software engineering and despite being applied to the field of DevOps in the past most of the projects using it use it for a specific, purpose domain, or tool. Our goal is then to create a generic approach to DevOps using MDE in a way that is technology and domain agnostic.

Currently, we got an article submitted to *VL/HCC's* graduate consortium, relating to our work proposal, we are also finalizing a survey retaining the intersection of the fields of DevOps and MDE, this survey showcases the evolution and domains of application and approach to this problem. Our immediate goal is to finish and submit this survey, for which a preliminary version has been submitted to the workshop *HAPi DevOps@VL/HCC* and try to publish the resulting article to one of the conferences listed in Chapter 1. We also plan on contacting organizations and starting elaborating a survey to evaluate the use of DevOps in the enterprise.

References

[1] Hanieh Alipour and Yan Liu. A model driven method to deploy auto-scaling configuration for cloud services. In *Proceedings of the 4th International Workshop on Release Engineering*, RELENG 2016, page 23, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Hanieh Alipour and Yan Liu. Model driven deployment of auto-scaling services on multiple clouds. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 93–96, 2018.

[3] Matej Artač, Tadej Borovšak, Elisabetta Di Nitto, Michele Guerriero, and Damian A. Tamburri. Model-driven continuous deployment for quality devops. In *Proceedings of the 2nd International Workshop on Quality-Aware DevOps*, QUDOS 2016, page 40–41, New York, NY, USA, 2016. Association for Computing Machinery.

[4] Nelly Bencomo. Modeling autonomic systems in the time of ml, devops and microservices. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 731–731, 2021.

[5] Francis Bordeleau, Jordi Cabot, Juergen Dingel, Bassem S. Rabil, and Patrick Renaud. Towards modeling framework for devops: Requirements derived from industry use case. In Jean-Michel Bruel, Manuel Mazzara, and Bertrand Meyer, editors, *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 139–151, Cham, 2020. Springer International Publishing.

[6] Hayet Brabra, Achraf Mtibaa, Walid Gaaloul, Boualem Benatallah, and Faiez Gargouri. Model-driven orchestration for cloud resources. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 422–429, 2019.

[7] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice*, volume 1. morgan and claypool, 09 2012.

[8] G. Casale, M. Artač, W.-J. van den Heuvel, A. van Hoorn, P. Jakovits, F. Leymann, M. Long, V. Papanikolaou, D. Presenza, A. Russo, S. N. Srirama, D. A. Tamburri, M. Wurster, and L. Zhu. Radon: rational decomposition and orchestration for serverless computing. *SICS Software-Intensive Cyber-Physical Systems*, 35(1):77–87, Aug 2020.

[9] Giuliano Casale, Danilo Ardagna, Matej Artac, Franck Barbier, Elisabetta Di Nitto, Alexis Henry, Gabriel Iuhasz, Christophe Joubert, Jose Merseguer, Victor Ion Munteanu, Juan Fernando Perez, Dana Petcu, Matteo Rossi, Craig Sheridan, Ilias Spais, and Daniel Vladuic. Dice: Quality-driven development of data-intensive cloud applications. In *2015 IEEE/ACM 7th International Workshop on Modeling in Software Engineering*, pages 78–83, 2015.

[10] Stéphanie Challita, Fabian Korte, Johannes Erbel, Faiez Zalila, Jens Grabowski, and Philippe Merle. Model-based cloud resource management with tosca and occi, 2020.

[11] Alessandro Colantoni, Luca Berardinelli, and Manuel Wimmer. *DevOpsML: Towards Modeling DevOps Processes and Platforms*, page 1–10. Association for Computing Machinery, New York, NY, USA, 2020.

[12] Benoit Combemale and Manuel Wimmer. Towards a model-based devops for cyber-physical systems. In Jean-Michel Bruel, Manuel Mazzara, and Bertrand Meyer, editors, *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 84–94, Cham, 2020. Springer International Publishing.

[13] The Business Research Company. Information technology global market report 2022, by type, organization size, end user industry, Mar 2022.

[14] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *IEEE Software*, 33(3):94–100, 2016.

[15] Badr El Khalyly, Abdessamad Belangour, Mouad Banane, and Allae Erraissi. A new metamodel approach of ci/cd applied to internet of things ecosystem. In *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, pages 1–6, 2020.

[16] Nicolas Ferry, Franck Chauvel, Hui Song, Alessandro Rossini, Maksym Lushpenko, and Arnor Solberg. Cloudmf: Model-driven management of multi-cloud applications. *ACM Trans. Internet Technol.*, 18(2), jan 2018.

[17] Georges Bou Ghantous and Asif Qumer Gill. Devops: Concepts, practices, tools, benefits and challenges. In *PACIS*, 2017.

[18] H. Da Giao. A model-driven approach for devops. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Los Alamitos, CA, USA, September 2022. IEEE Computer Society.

[19] Michele Guerriero, Michele Ciavotta, Giovanni Paolo Gibilisco, and Danilo Ardagna. A model-driven devops framework for qos-aware cloud applications. In *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 345–351, 2015.

[20] Michele Guerriero, Michele Ciavotta, Giovanni Paolo Gibilisco, and Danilo Ardagna. Space4cloud: A devops environment for multi-cloud applications. In *Proceedings of the 1st International Workshop on Quality-Aware DevOps*, QUDOS 2015, page 29–30, New York, NY, USA, 2015. Association for Computing Machinery.

[21] Mubin Ul Haque, Leonardo Horn Iwaya, and M. Ali Babar. Challenges in docker development: A large-scale study using stack overflow. In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ESEM '20, New York, NY, USA, 2020. Association for Computing Machinery.

[22] Jerome Hugues, Anton Hristosov, John J. Hudak, and Joe Yankel. Twinops - devops meets model-based engineering and digital twins for the engineering of cps. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, MODELS '20, New York, NY, USA, 2020. Association for Computing Machinery.

[23] G. Kim, J. Humble, P. Debois, and J. Willis. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. ITpro collection. IT Revolution Press, 2016.

[24] Rakesh Kumar and Rinkaj Goyal. Modeling continuous security: A conceptual model for automated devsecops using open-source software over cloud (adoc). *Computers & Security*, 97:101967, 2020.

[25] Bart Meyers, Klaas Gadeyne, Bentley Oakes, Matthias Bernaerts, Hans Vangheluwe, and Joachim Denil. A model-driven engineering framework to support the functional safety process. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 619–623, 2019.

[26] Mary Sánchez-Gordón and Ricardo Colomo-Palacios. Security as culture: A systematic literature review of devsecops. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, page 266–269, New York, NY, USA, 2020. Association for Computing Machinery.

[27] J. Sandobalin, E. Insfran, and S. Abrahao. Towards model-driven infrastructure provisioning for multiple clouds. In Bo Andersson, Björn Johansson, Chris Barry, Michael Lang, Henry Linger, and Christoph Schneider, editors, *Advances in Information Systems Development*, pages 207–225, Cham, 2019. Springer International Publishing.

[28] Julio Sandobalin. A model-driven approach to continuous delivery of cloud resources. In Lars Braubach, Juan M. Murillo, Nima Kaviani, Manuel Lama, Loli Burgueño, Naouel Moha, and Marc Oriol, editors, *Service-Oriented Computing – ICSOC 2017 Workshops*, pages 346–351, Cham, 2018. Springer International Publishing.

[29] Julio Sandobalin, Emilio Insfran, and Silvia Abrahão. Argon: A model-driven infrastructure provisioning tool. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 738–742, 2019.

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

[31] Laurence Tratt and Martin Gogolla. *Theory and Practice of Model Transformations*, volume 6142. Springer, 01 2010.

[32] Konstantinos Tsilionis and Yves Wautelet. A model-driven framework to support strategic agility: Value-added perspective. *Information and Software Technology*, 141:106734, 2022.

[33] Willem-Jan van den Heuvel and Damian A. Tamburri. Model-driven ml-ops for intelligent enterprise applications: Vision, approaches and challenges. In Boris Shishkov, editor, *Business Modeling and Software Design*, pages 169–181, Cham, 2020. Springer International Publishing.

[34] Willem-Jan van den Heuvel, Damian A. Tamburri, Damiano D'Amici, Fabiano Izzo, and S. Potten. Chainops for smart contract-based distributed applications. In Boris Shishkov, editor, *Business Modeling and Software Design*, pages 374–383, Cham, 2021. Springer International Publishing.

[35] Denis Weerasiri, Moshe Chai Barukh, Boualem Benatallah, and Jian Cao. A model-driven framework for interoperable cloud resources management. In Quan Z. Sheng, Eleni Stroulia, Samir Tata, and Sami Bhiri, editors, *Service-Oriented Computing*, pages 186–201, Cham, 2016. Springer International Publishing.

[36] Denis Weerasiri, Moshe Chai Barukh, Boualem Benatallah, and Cao Jian. Cloudmap: A visual notation for representing and managing cloud resources. In Selmin Nurcan, Pnina Soffer, Marko Bajec, and Johann Eder, editors, *Advanced Information Systems Engineering*, pages 427–443, Cham, 2016. Springer International Publishing.

[37] Johannes Wettinger, Uwe Breitenbücher, Oliver Kopp, and Frank Leymann. Streamlining devops automation for cloud applications using tosca as standardized metamodel. *Future Generation Computer Systems*, 56:317–332, 2016.

[38] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[39] L. Zhu, L. Bass, and G. Champlin-Scharff. Devops and its practices. *IEEE Software*, 33(03):32–34, may 2016.