Towards the Automatic Identification of Optimal Configurations of CI/CD Pipelines of Software Development Projects: Symbolic, Meta-heuristic and Statistical Approaches

Guy Rostan DJOUNANG NANA[1,2]

LIRMM, Univ Montpellier, CNRS, Go2Scale, Montpellier, France,
guy.djounang-nana@lirmm.fr, guy@go2scale.com

Abstract. One major DevOps activity is the establishment of Continuous Integration and Continuous Delivery or Continuous Deployment (CI / CD) pipelines to automate many tasks of the software life cycle, such as build, functional testing, quality testing, security checks, packaging, or deployment. Implementation and maintenance of these pipelines are repetitive, tedious and time-consuming. In addition, some developer teams do not have the required skills. The objective of this PhD is to develop a CI/CD pipeline generator which will provide optimal CI/CD configurations regardless of the target execution platform. This will also facilitate the CI/CD configurations cross-platform migration. To achieve our goal we define a 4-step process methodology alongside with three strategies, i.e. (meta-)heuristic, symbolic and statistical for identifying optimal CI/CD pipelines. In this paper we also present our preliminary results on determining a common model for founding the pipeline variability.

Keywords: Software Engineering · Feature model · variability · DevOps · CI/CD · pipeline · workflow · Meta-heuristic Approach · Statistic Approach · Symbolic Approach

1 Introduction and Motivations

DevOps is an approach for bringing together two key functions of a company's IT department responsible for developing applications [3,6], namely the "Dev" and the "Ops" which respond to two different needs. One key aspect of DevOps is the establishment of Continuous Integration and Continuous Delivery or Continuous Deployment (CI / CD) pipelines to automate many tasks of the software life cycle [1,5]. A pipeline is defined as a configurable automated process that will run one or more jobs, such as build, functional testing, quality testing, security checks, packaging, or deployment jobs.

Nevertheless the implementation of this workflow is not without constraints [8,4]. The configuration of these CI/CD pipelines is currently carried out manually by the teams; it is a time-consuming and tedious job to set them up and

maintain them for each project [8]. Additionally, various platforms exist to run these pipelines, each with very different functionalities and syntax [1]. So, for a DevOps team that does not have required skills or wants to migrate from a platform to another, the task is even more painful: they have to learn syntax and concepts of the new target platform which is wasted time that they could have used for their core business activities.

In literature, we distinguish two principal categories of approaches that try to resolve the pipeline definition problem: manual-assisted and automatic-specific approaches. The manual-assisted approaches reduce the cost of fully manual implementation of pipeline but they do not guarantee the validity or the optimality (with regard to non functional requirements) of the resulting pipeline. For their part, the automatic-specific approaches, as their name indicates, focus either on a single stage of the software life cycle [7,11], or on a single element / attribute of the pipeline [9], or on an application domain [2] or even on contexts of particular applications [10].

In practice, a CI platform like GitHub Actions or a portable devkit for CI/CD tool like Dagger provides actions, safely shared, and reusable building blocks that encapsulate complex but frequently repeated tasks. Those actions facilitate the creation of jobs but the pipeline implementation still remains manual. Another CI platform like GitLab goes beyond by providing an Auto DevOps solution that detects the code language of a software project and then uses predefined CI/CD job templates to create and run a default pipeline. A solution such as r2devops provides an auto-pipeline allowing to generate a better customizable pipeline for a project. But these two last solutions achieve the selection of the different jobs of the pipeline by hard coded rules, and are platform specific. Dagger, the previously mentioned solution, deals with the CI lock-in, so developers and operators just automate actions with their favorite programming languages, all tie in a declarative language named CUE, and run them on any Docker-compatible CI platform. Dagger has a main drawback for DevOps teams which already have their CI/CD configurations for a specific platform and which wants migrate to another CI platform. They need not only to learn the new language CUE but also to rewrite all their configurations for the first time.

In this paper, we focus on the problem of building a CI/CD pipeline auto-generator regardless of the target CI platform. In Section 2, we formulate the research questions to address during this PhD, and present our research methodology. Then, Section 3 presents the preliminary results of the first two months of our work while in Section 4 we detail our work plan.

2 Research Questions and Methodology

2.1 Research Questions

As present above, setting up CI/CD pipelines is tedious and time-consuming, and that a variety of CI platform or tool exist, each with its own syntax and concepts. This observation prompts us to ask ourselves the following research Questions (RQs):

RQ-1 : What are the core concepts and features of CI/CD pipeline domain ?
RQ-2 : What guides developers and operators when building pipelines for a software project ?
RQ-3 : How to automatically identify valid and optimal CI/CD pipeline configurations ?
RQ-4 : How to automatically get from a resulting optimal pipeline a CI/CD configuration for any target CI platform ?

Answer the above RQs will help us to achieve our PhD goal: build a tool for automatic identification of optimal configurations of CI/CD pipelines of software development projects.

2.2 Research Methodology

We define a four-step process for answering our RQs (Figure 1).

Workflow variability The first step of our PhD journey is to make the domain engineering of CI/CD pipeline on one side and software projects on the other side. For both CI/CD pipeline and software projects, it aims to identify all their features and the dependency constraints between those features. For CI/CD pipelines, the outcome is a common pipeline domain model and its feature model (pipeline variability), thus answering to RQ-1. For software projects, the result is a variability model that allows us to classify them.

Project analyzer It is about analyzing all artifacts (source code, packages structure, tests and deployment files, etc.) of a software project for identifying its features. Then, we can determine the class of the project and get a set of valid CI/CD pipeline configurations. This is an answer to RQ-2. Note that a valid CI/CD pipeline configuration is a configuration that can be executed without error.

Jobs selector From all possible valid pipeline configurations, we have to select those that satisfy the non functional requirements of the DevOps team. We refer to those configurations as optimal pipeline configurations. To realize the selection of optimal pipeline configurations we plan to explore three strategies.

- **Strategy 1 (Meta-heuristic approach)**: we are going to define an optimization model based on meta-heuristic algorithms which use one or more objectives functions.
- **Strategy 2 (Symbolic approach)**: we are going to explore some pattern mining approaches combined with similarity and recommendation-based approaches.
- **Strategy 3 (Statistical approach)**: we are going to use Machine Learning and Deep Learning models.

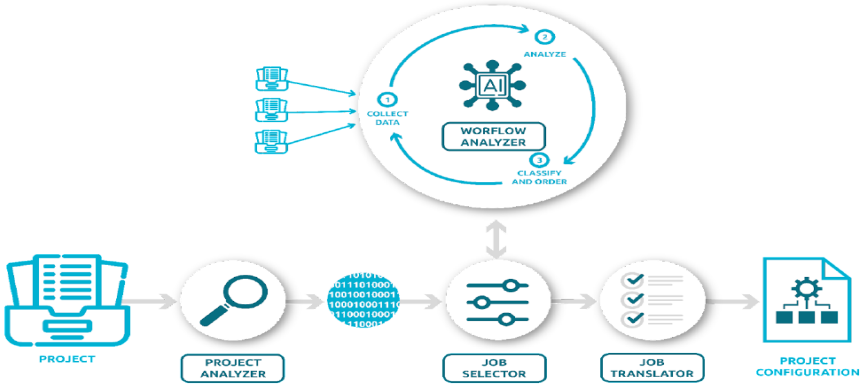

Fig. 1. Process of automatic identification of optimal CI/CD pipeline configurations

Job translator Once optimal CI/CD pipeline configurations are selected, we should automatically translate them into the configuration format of the target platform, making our solution a pipeline cross-platform generator. We can do that by using model to model transformation rules.

3 Preliminary Results

We started our work on the one hand with the realization of the domain engineering of the CI/CD pipelines, which consisted of making pipelines models for several CI platforms: CircleCI (fig. 2), GitHub Actions (fig. 3), with a deeper focus on GitLab (fig. 4). These CI platforms have their own specific features and concepts, and also share common or similar features despite being implemented differently. We plan to study deeper the pipeline domain model of other CI platforms and build a common meta-model. Then, we are going to build a feature model of CI/CD pipeline which answers RQ-1, to get the ***Workflow variability***.

On the other hand, we started answering RQ-2 by designing a questionnaire. It consists in exploring software projects and figure out what features would push "Devs" or "Ops" to use certain jobs. The result of this step are going to allow us to build the ***Project Analyzer***.

4 Work Plan

This PhD started in March 2022 and should be defended in March 2025. It is based on a collaboration between the LIRMM (Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier) and the company Go2Scale, under the supervision of Djamel Seriai, Arnaud Castellort, Marianne Huchard

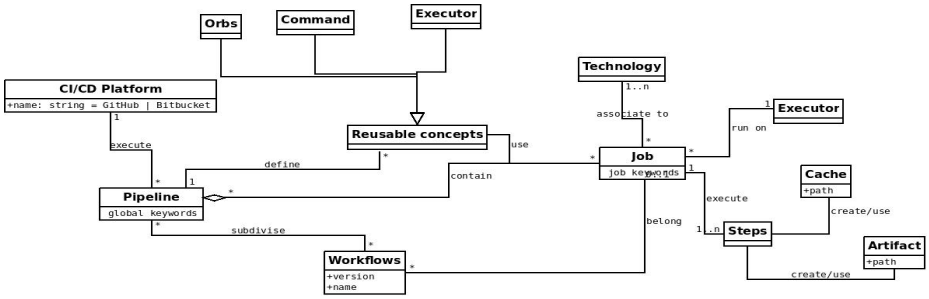

Fig. 2. CircleCI pipeline domain model

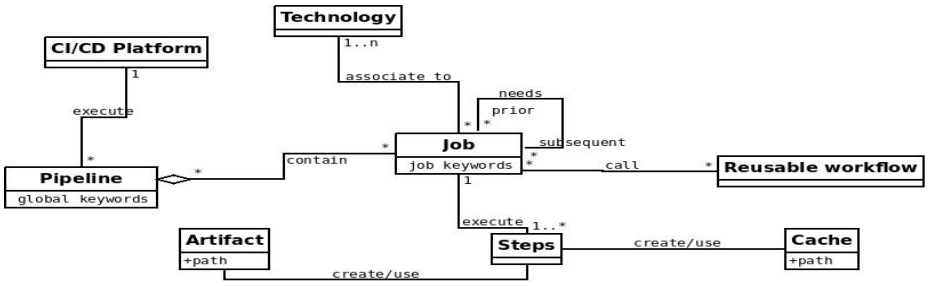

Fig. 3. GitHub pipeline domain model

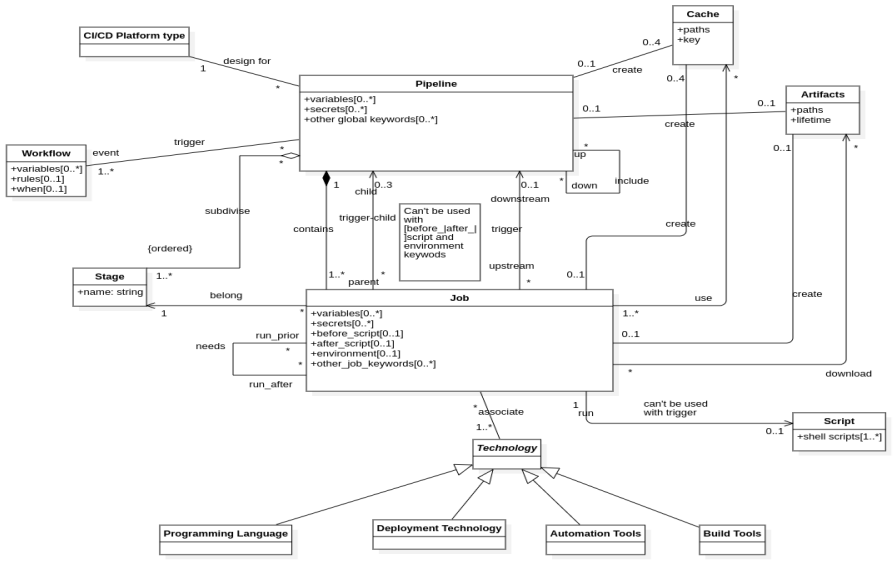

Fig. 4. GitLab pipeline domain model

and Anne Laurent for the academic part, and Thomas Boni for the industrial part. From March to April 2022, we began working on *Workflow variability* and *Project analyzer*. These tasks could still require a semester and a quarter respectively. After that we are going to move to *Job selector*, which is the main and challenging task of the PhD. It could last more than one year for exploring and implementing all the three strategies. Finally, we are going to end with the implementation of *Job translator* for the remaining time.

5 Conclusion

This paper aims to present our work on automatic identification of optimal configurations of CI/CD pipelines. We discuss about the preliminary results of workflow variability of CI/CD pipeline and identification of software development project features studies. We noticed that pipeline features are CI platform and project specific. For the next months, we will focus on building pipeline meta-model and features model that take into account the variety of CI platform, and the implementation of features extraction from software projects.

References

1. Berg, J.E.C.: DevOps. Building CI/CD Pipelines with Jenkins, Docker Container, AWS ECS, JDK 11, Git and Maven 3 (2019)
2. El Khalyly, B., Belangour, A., Banane, M., Erraissi, A.: 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). pp. 1–6. IEEE, Kenitra, Morocco (Dec 2020)
3. Forsgren, N., Humble, J., Kim, G.: Accelerate: the science behind DevOps: building and scaling high performing technology organizations. IT Revolution (2018)
4. Hemon-Hildgen, A., Rowe, F., Monnier-Senicourt, L.: Orchestrating automation and sharing in DevOps teams: a revelatory case of job satisfaction factors, risk and work conditions. European Journal of Information Systems **29**, 474–499 (Sep 2020)
5. Hüttermann, M.: DevOps for developers. The expert's voice in Web development, Apress, New York (2012)
6. Jones, C.: Using code generation to enforce uniformity in software delivery pipelines. In: Bruel, J.M., Mazzara, M., Meyer, B. (eds.) Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment. pp. 155–168. Springer International Publishing, Cham (2019)
7. Lai, S.T., Susanto, H., Leu, F.Y.: Combining Pipeline Quality with Automation to CI/CD Process for Improving Quality and Efficiency of Software Maintenance. In: Barolli, L., Yim, K., Chen, H.C. (eds.) Innovative Mobile and Internet Services in Ubiquitous Computing. pp. 362–372. Springer International Publishing, Cham (2022)
8. Leite, L., Rocha, C., Kon, F., Milojicic, D., Meirelles, P.: A Survey of DevOps Concepts and Challenges. ACM Computing Surveys **52**, 127:1–127:35 (Nov 2019)

9. Mahboob, J., Coffman, J.: A Kubernetes CI/CD Pipeline with Asylo as a Trusted Execution Environment Abstraction Framework. In: 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC). pp. 0529–0535. IEEE, NV, USA (Jan 2021)
10. Nogueira, A.F., C.B. Ribeiro, J., Zenha-Rela, M.A., Craske, A.: Improving La Redoute's CI/CD Pipeline and DevOps Processes by Applying Machine Learning Techniques. In: 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC). pp. 282–286. IEEE, Coimbra (Sep 2018). https://doi.org/10.1109/QUATIC.2018.00050, https://ieeexplore.ieee.org/document/8590203/
11. Rangnau, T., Buijtenen, R.v., Fransen, F., Turkmen, F.: Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines. In: 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC). pp. 145–154 (Oct 2020)