Design-Operation Continuum Methods for Traffic Master

Aitor Arrieta[1], Aitor Agirre[2], Goiuria Sagardui[1] and Maite Arratibel[3]

[1]Mondragon University, Spain
[2]Ikerlan, Spain
[3]Orona, Spain

Keywords: DevOps, Traffic Master, Dispatching Algorithms.

Abstract. The lifecycle of lifts could last up to 30 years. As with any other electrical or mechanical component, software of lifts also requires a maintenance process. Maintenance copes with (1) hardware obsolescence and/or degradation, (2) bug fixing, (3) new functionalities, (4) requirements changes, etc. This evolution requires reliable and automatic engineering methods for developing and operating lifts. Advances in the last few years have resulted in a more efficient development process, improving modelling and simulation techniques to validate complex systems from the early phases of development. However, once the system is deployed, methods used during operation and maintenance do not have synergies with methods used during the design. The steps from the development to operation, i.e. testing, delivery and deployment, often require certain manual work to guarantee reliability. Current software development approaches are not applicable or require extension for lifts, where evolution is constant. Furthermore, learning from operational data to enhance the design is becoming a necessity in this sector.

The ADEPTNESS project seeks to investigate and implement a streamlined and automatic workflow that makes methods and tools for the software development and maintenance of lifts to be seamlessly used during design phases as well as in operation. The ADEPTNESS framework uses a novel embedded microservices-based architecture for the context of lifts. The generation and reuse of test cases and oracles from initial phases of the development to the system in operation and back to the laboratory for further analysis will be investigated. This will guarantee a faster and more reliable detection of faults before a new software release is deployed into the lift installations. This deployment will be automatic and synchronised to improve the agility of the whole workflow that covers design-operation continuum. Additionally, test oracles will run both at design-time as well as at operation, permitting the continuous validation of a software release.

1 INTRODUCTION

The traffic master and its software are one of the core components of a system of lifts. As other components, its software requires maintenance over years to deal with different aspects like bugs or problems that arise in operation (e.g., unforeseen situations). While in other types of systems (e.g., web systems) design operation continuum methods like DevOps have emerged, in complex systems like lifts, where the software is embedded and needs to interact with physical components, such methods pose several challenges.

The Adeptness H2020 project is a project that aims at adapting DevOps methods to the context of Cyber-Physical Systems. Specifically, to validate such methods, one of the use-case encompasses the traffic master of Orona's lifts. In this paper, we summarize the methods we developed and instantiated in the context of lifts. Specifically, we instantiate the reference architecture based on embedded microservices in the use-case to allow for design-operation methods. In addition, we carry out an analysis of the benefits this would pose.

2 ARCHITECTURE OF EMBEDDED MICROSERVICES FOR DEVOPS OF TRAFFIC MASTER

2.1 Architecture definition methodology

The first step when designing the architecture based on embedded microservices was to define a methodology for the architecture definition. This methodology consisted of six main steps [1]:

1. Use-case definition: Use-case scenarios were defined by domain experts from the lift industry. These can be found in [2].
2. Stakeholder requirements: By considering these scenarios, we elicited a set of requirements for the architecture, which can be found in [3].
3. DevOps toolchain requirements elicitation: System requirements and subsystems requirements were elicited by having as input the stakeholder's requirements, accessible in [3].
4. Requirements analysis and microservice identification: A system architect performed the first analysis with the elicited requirements and a set of microservices was identified.
5. Interface identification: The different interfaces were defined and integrated with one microservice template developed during the Adeptnes H2020 project. This template is available both in C and Python.
6. Instantiation: Lastly, the different templates were intanted and integrated

2.2 Microservices-based architecture for DevOps of the traffic master

Figure 1 shows an overview of the proposed microservice-based architecture. The architecture contains microservices at two computational levels: (1) the cloud and (2) the fog, i.e., a local network close to the lift. The architecture can be divided into four main subsystems: (1) the automation server, (2) the deployment subsystem, (3) the monitoring subsystem and (4) the validation subsystem.

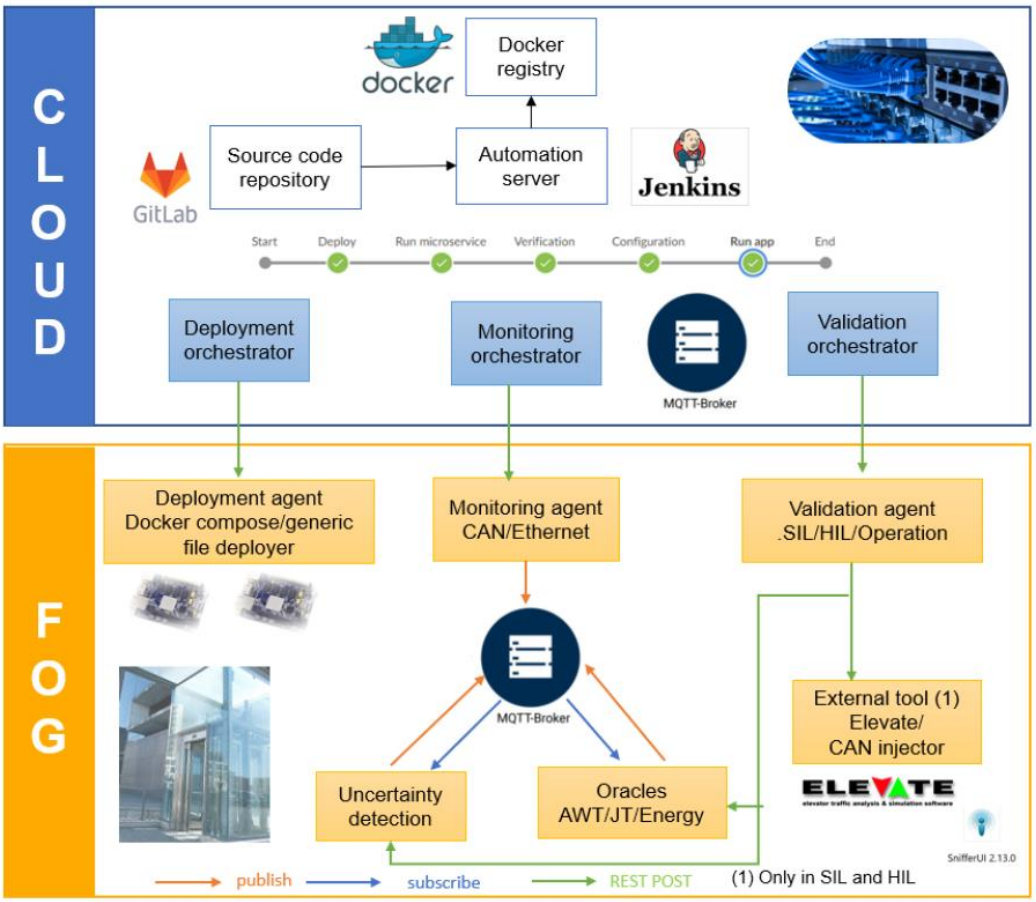

Figure 1 Overview of the architecture for DevOps of the Traffic Master

Common microservices interfaces

All the microservices have certain common interfaces to allow for communication among them. This communication can be either synchronous or asynchronous. For synchronous communication, we used REST communication whereas for asynchronous we used MQTT.

The following interfaces are common for Synchronous communication via REST:

- /adms/v1/ping [GET]: Ping service to check that the service is alive. Returns an empty 200 response if the microservice is working correctly.
- /adms/v1/info [GET]: Provides basic information about the microservice. It returns a JSON object containing the microservice ID and microservice role within the architecture.
- /adms/v1/performance [GET]: Provides CPU and memory usage metrics. It returns a JSON object containing the free and allocated memory and the CPU usage.
- /adms/v1/status [GET, PUT]: Permits getting or changing the execution status of the microservice. GET calls to this endpoint will return a JSON object containing the status of the microservice. Changes to the microservice status will be performed by sending a JSON object with the desired state. The possible states for the microservice are “Ready” and “Running”.

For asynchronous communication, the following interfaces are common to all microservices, which communicate through MQTT:

- /adms/v1/discovery [PUB]: On microservice launch, the microservice publishes a hello message in this topic including the identifier, microservice role and its MQTT and REST endpoints, defined as a JSON object.

The automation server:

When developers propose a change in the code, this is committed to a GitLab repository. In such a repository, an automation server based on Jenkins orchestrates different jobs, including a verification phase where different test executions are performed by following advanced techniques like metamorphic testing [4], uncertainty-wise testing [5] and machine-learning based techniques [6]. Such tests are carried out either at the Software-in-the-Loop (SiL) test level, or the Hardware-in-the-Loop (HiL) test level. The former is a simulation-based testing method where Elevate is used as the core simulator. The latter refers to a real-time emulation where most of the hardware of the system is real, while the mechanical and electrical parts are emulated with real-time infrastructure. When all verification activities are finished and all tests pass, the automation server triggers the deployment microservice to deploy the new software version in the required installations.

Deployment subsystem:

The deployment subsystem is in charge of downloading, decompressing and executing different microservice artifacts, including the new dispatching algorithm versions, on the edge. It is composed on two different microservices: (1) the deployment microservice and (2) the deployment agent. The former receives a plan for deployment triggered by the automation server. The core task is to send these deployment instructions to the different deployment agents installed in each node (i.e., lifts installations across the globe). The latter is a microservice that is installed in each edge node to perform the actual deployment. Two different types of components can be deployed: (1) a docker-compose based deployer that deploys docker containers and (2) a generic deployer to deploy any kind of files. This way, the deployment agent has a high flexibility.

Monitoring subsystem:

This subsystem configures different monitors based on a monitoring plan, which provides telemetry data from different sources (e.g., dispatching algorithm or the main bus). Two microservices are considered: (1) monitoring orchestrator and (2) monitoring agent. The former parses the monitoring plan triggered from the automation server and configures all the monitoring agents. The latter is deployed in the edge and is responsible for reading the operational variables from different sources and publishing them asynchronously through MQTT. Two main monitors are used in the case study. On the one hand, the CAN monitor, which allows for monitoring operational data shared through the CAN bus. On the other hand, the instrumented code monitor, which allows for monitoring data from instrumented code. All this data is published using the SenML [7] format.

Validation subsystem:

The validation subsystem supports the verification of new traffic master versions at different test levels, including (1) SiL, through the use of Elevate, (2) HiL and Operation. It is composed of five different microservices. The validation orchestrator microservice is located in the cloud and manages the execution of a validation plan. The validation agents are located in the fog and launch validations at the SiL and HiL test levels. The oracle microservice receives inputs and outputs of the system under test and provide test verdicts (i.e., pass, fail or inconclusive). The uncertainty microservice works similarly to the oracle microservice but aim at detecting unforeseen situations (e.g., ghost passengers, wrong behavior of the user). Lastly, the external tool microservice provides a communication link with external tools like Elevate.

3 EXPECTED BENEFITS

We prototyped these DevOps methods with the traffic master of our lifts. A video of the full prototype can be viewed in the following link: https://youtu.be/uoq9n9k4kgc

Based on this prototype, we carried out an evaluation to assess which are the core benefits of using DevOps practices to develop new traffic master versions. The expected main benefits for each subsystem can be summarized as follows [1]:

- Deployment subsystem: The main expected benefit is the need for not requiring manual intervention when deploying the software releases, and having full control over the configuration of the release.
- Monitoring subsystem: It will reduce the effort of analyzing problems in lifts' installations and enable continuous remote monitoring.
- Validation subsystem: It provides systematic and advanced testing methods that will enable increasing the number of bugs detected before releasing a new traffic master version. Furthermore, it will enable continuously testing of new releases not only at design-time but also at operation, enabling the possibility of rolling back to a previous version when the new traffic master version is faulty.

4 CONCLUSION

In this paper, we propose the instantiation of a reference architecture based on microservices to support the design-operation continuum engineering of our lift traffic master. This reference architecture has been proposed in the Adeptness H2020 project. Future work includes a comprehensive evaluation of the methods and a thorough cost-benefit analysis of using these methods.

REFERENCES

[1] Gartziandia, A., Ayerdi, J., Arrieta, A., Ali, S., Yue, T., Agirre, A., ... & Arratibel, M. (2021, March). Microservices for Continuous Deployment, Monitoring and Validation in Cyber-Physical Systems: an Industrial Case Study for Elevators Systems. In 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C) (pp. 46-53). IEEE.

[2] Requirements-and-validation-tests – https://adeptness.eu/wp-content/uploads/2020/09/D1.1-ANNEX-A-Requirements-and-validation-tests.pdf

[3] Requirements-and-validation-tests – https://adeptness.eu/wp-content/uploads/2020/11/D1.1-REQUIREMENTS_v1.1.pdf

[4] Ayerdi, J., Segura, S., Arrieta, A., Sagardui, G., & Arratibel, M. (2020, October). Qos-aware metamorphic testing: An elevation case study. In 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE) (pp. 104-114). IEEE.

[5] Galarraga, J., Marcos, A. A., Ali, S., Sagardui, G., & Arratibel, M. (2021, October). Genetic Algorithm-based Testing of Industrial Elevators under Passenger Uncertainty. In 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 353-358). IEEE.

[6] Arrieta, A., Ayerdi, J., Illarramendi, M., Agirre, A., Sagardui, G., & Arratibel, M. (2021, May). Using machine learning to build test oracles: an industrial case study on elevators dispatching algorithms. In 2021 IEEE/ACM International Conference on Automation of Software Test (AST) (pp. 30-39). IEEE.

[7] SENML format – https://tools.ietf.org/html/rfc8428

ACKNOWLEDGMENTS

This publication is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871319.

BIOGRAPHICAL DETAILS

Dr. Aitor Arrieta: He is a lecturer and researcher at Mondragon University. He obtained a PhD degree in software engineering in 2017. Since then, he has been involved in both research projects and transfer projects, having strong collaborations with Orona.

Dr. Aitor Agirre: He is a senior researcher at Ikerlan research center. Currently he is involved in the Adeptness H2020 project and also in transfer projects with Orona.

Dr. Goiuria Sagardui: She is the coordinator of continuous education at Mondragon University, and she is also a researcher working in both research projects and transfer projects, especially with Orona.

Maite Arratibel: She is a senior engineer in Orona, who coordinates the research and development of the traffic master. Before, she worked as an engineer in Ikerlan.