University POLITEHNICA of Bucharest
Automatic Control and Computers Faculty
Automation and Industrial Informatics Department

**PhD Thesis Summary**

# Automatic Deployment of Modeling Environments Provided as Services in Cloud

Presented by

Drd.Ing. Marian LĂCĂTUȘU

Supervised by

Prof.Dr.Ing. Anca Daniela IONIȚĂ

**2023**

**Bucharest, Romania**

## Abstract

Moving to the cloud is a topic that tends to be present in all enterprises that have digitalized their activities. Advantages such as disaster recovery, collaboration, and cost-savings have attracted many organizations to adopt such a service-oriented approach, and some of them moved their entire activities to a public cloud. Also, in the context of the COVID-19 pandemic, the need for remote collaboration between teachers and groups of students has become an important educational aspect. The difficulties came from the fact that a vast majority of the existing e-learning platforms were not enough prepared for scaling up, or the teachers were not trained to use this kind of teaching exclusively. These limitations were also felt in the research activities, where the scarcity of collaboration between researchers has become an important problem under lockdown. My purpose was to solve some of the presented limitations and to help students, teachers, and researchers in research and educational activities regarding object-oriented modeling. As a result, I propose here a multicloud platform where students, teachers, and researchers who rely on collaboration can easily deploy their software environment to work remotely. There are few engineers specialized both in modeling and cloud computing at a high technical level. This approach wishes to create methods and tools to be used by engineers only specialized in one of the domains to be able to successfully perform such interdisciplinary tasks. A very important aspect is the complexity of the processes that needed to be followed for automatic and manual deployment. As a result, I model the manual and automatic deployment with a standard business process modeling language, I Investigate complexity metrics, I make a selection that fits our purpose and apply them for the two process models. Furthermore, I propose a new complexity metric relevant to our scope and I evaluate the processes and prove that the complexity may be up to 3 times higher for manual deployment. This result demonstrates that automation is very important for the service to be more accessible to users regardless of their technical knowledge. Therefore, I conceive a decision tree to support the multicloud automatic deployment and develop it to select the modeling environments and the following cloud providers: Amazon Web Services, Microsoft Azure and IBM Cloud. I validate the solution for the deployment in cloud of a modeling and metamodeling environment, i.e., WebGME, with an example of a modeling language for sensor networks and a specific model interpreter. Also, I evaluate the solution from the perspective of cloud monitoring and the performance of the service is tested using two cloud monitoring solutions. The service performs well on a minimal Kubernetes implementation and the Grafana-Prometheus bundle represents the right choice for cloud monitoring. Furthermore, I evaluate the decision algorithm from the perspective of its most valuable outcome of it, I describe the scenarios regarding the way this solution can be classified (PaaS (Platform as a Service) or SaaS (Software as a Service)), I compare and discuss the similarities and differences for multicloud.

## Acknowledgments

First of all, I want to thank my advisor, Professor Anca Daniela Ioniță for her guidance mentoring, support, and patience during the PhD studies. She is the best mentor and I couldn’t have these achievements without her guidance.

I want to also thank Professor Mariana Mocanu, Professor Florin Anton, and Professor Adriana Olteanu. As members of the PhD committee, they provided much-needed feedback and helped me improve my work.

Also, I want to thank my brother Florin Lăcătușu and my good friend Ioan Damian for their collaboration on multiple published articles.

Last but not least, I want to thank my family for their support during these years of PhD studies.

## Table of Contents

**Abstract** ........ 2

**Acknowledgments** ........ 3

**List of Tables** ........ 6

**List of Figures** ........ 6

1. Introduction ........ 7

1.1 Context and Motivation ........ 7

1.2 Research Storyline ........ 8

1.3 Thesis Overview ........ 9

**2. Analysis of the State of the Art** ........ 10

2.1 Modeling Aspects ........ 10

2.2 Cloud Computing Aspects ........ 11

**3. Multicloud Deployment of Modeling Environments** ........ 11

3.1 Objectives ........ 11

3.2 Method ........ 12

**4. Contributions to the Automatic Deployment of Modeling Environments** ........ 13

4.1 Manual Deployment of a Modeling Environment. Example for IBM Cloud ........ 13

4.2 Automatic Deployment Using the Multicloud Platform ........ 14

4.3 Complexity Metrics for Manual and Automatic Deployment ........ 16

4.3.1 Choice of complexity metrics for the Deployment Process Analysis and Results ........ 17

4.4 Monitoring Tools for Performance Analysis and Cloud Metrics ........ 17

4.5 Decision Algorithm that Helps the User Establish Deployment Details ........ 18

4.5.1 Deployment based on the ID3 decision tree ........ 18

4.5.2 Decision algorithm to deploy GME / WebGME in a multicloud environment ........ 19

4.6 Platform Implementation - Technologies ........ 21

**5. Evaluation and Validation** ........ 23

5.1 WebGME Deployment in Cloud - Features Validation Example ........ 23

5.2 Cloud Deployment - Performance Test Results ........ 24

5.3 Evaluation Method of the Decision Algorithm for the Deployment Options ........ 26

5.4 Cloud Deployment – Evaluation ........ 28

5.5 Discussion ........ 29

5.5.1 Modeling environment ........ 29

5.5.2 Complexity analysis ........ 30

5.5.3. Performance analysis ........ 31

5.5.4 Automated deployment platform ........ 32

**6. Conclusion** ........ 33

6.1 Summary of Original Contributions .......... 34
6.2 List of Publications .......... 36
6.3 Future Perspectives .......... 37
**Selected References** .......... 39

## List of Tables

| | |
|---|---|
| Table 4.1 | Deployment decisions for WebGME |
| Table 5.1 | Prometheus monitoring results for containers |
| Table 5.2 | Prometheus cluster metrics results |
| Table 5.3 | Sysdig monitoring results for containers |
| Table 5.4 | Sysdig cluster metrics results |
| Table 5.5 | Outcome evaluation |
| Table 5.6 | Metrics for SaaS model |

## List of Figures

| | |
|---|---|
| Fig. 3.1 | Solution implementation method |
| Fig. 4.1 | WebGME manual deployment tasks (IBM Cloud) |
| Fig. 4.2 | Environment deployment from multicloud platform |
| Fig. 4.3 | Multiple docker image Helm deployments |
| Fig. 4.4 | Decision tree after ID3 algorithm finished building it |
| Fig. 4.5 | Decision tree root |
| Fig. 4.6 | The left section of the decision tree |
| Fig. 4.7 | The right section of the decision tree |
| Fig. 4.8 | Overview of the technologies employed for the new platform |
| Fig. 5.1 | Part of the metamodel from the on-site metamodeling environment (after [6]) |
| Fig. 5.2 | Modeling environment migrated to the Cloud |
| Fig. 5.3 | Tree evaluation – Left part |
| Fig. 5.4 | Tree evaluation – Right part |

# 1. Introduction

## 1.1 Context and Motivation

Moving to the cloud is a topic that tends to be present in all enterprises that have digitalized their activities. This includes the need to work with software environments specific to various business domains, accessed as services supported by various cloud providers. In many business domains, people are interested to have a cloud-based solution that is easier to maintain in comparison with on-premises implementations. Advantages such as disaster recovery, collaboration, and cost-savings have attracted many enterprises to adopt such a service-oriented approach, and some of them moved their entire activities to a public cloud [1]. Cloud deployments are known to absolve users from completing tedious maintenance and resource allocation tasks [2]. Beyond this, to support all that is necessary for an application, including infrastructure, data, and development tools, one can adopt an Environment as a Service (EaaS). Such a solution is potentially multicloud, needs automation for the deployment and configuration of the environment, and has to manage resource consumption. Also, in the context of the COVID-19 pandemic, the need for remote collaboration between teachers and groups of students has become an important educational aspect. A possible solution is to move the entire activity to the Cloud, thus offering other advantages besides the much-needed interaction, such as disaster recovery and resource management. As cost savings are now more important than ever, one also needs to make the right choice between the Cloud providers where to deploy the educational environments. A platform focused on multicloud deployment and collaboration can facilitate one to benefit from the advantages mentioned above, providing usage simplicity for teams of students and researchers that need to continue their activity during the lockdown period. Remote learning is a type of education that is considered future-oriented and advantages both the student and the teacher due to the elimination of constraints such as time and space [3]; in particular, it helps students who live in a remote location and cannot attend the courses in person. Nowadays, the topic of remote learning has become crucial due to the COVID-19 pandemic that forced the closure of institutions such as schools and universities. This unexpected situation has also exposed the vulnerabilities of remote collaboration for learning and research purposes, and many educational institutions around the globe were caught unprepared for this kind of crisis [4]. The difficulties came from the fact that a vast majority of the existing e-learning platforms were not enough prepared for scaling up, or the teachers were not trained to use this kind of teaching exclusively. These limitations were also felt in the research activities, where the scarcity of collaboration between researchers has become an important problem under lockdown.

My purpose was to solve some of the presented limitations, to help students and teachers in research and educational activities regarding object-oriented modeling. Remote learning deprives teachers, researchers, and students of the hardware that was present in school laboratories or research facilities; however, part of this hardware can be replaced by moving everything entirely to the Cloud [5]. This can also be applied to object-oriented modeling tools, which can benefit from advantages such as collaboration and disaster recovery. I propose here a platform where students, teachers, and researchers who rely on collaboration can easily deploy their environment to work remotely. Nonetheless, the need for collaboration is essential for users who contribute to the same project.

There are few engineers specialized both in modeling and cloud computing at a high technical level. This approach wishes to create methods and tools to be used by engineers only specialized in one of the domains to be able to successfully perform such interdisciplinary tasks. For example, the platform helps engineers specialized in cloud to deploy a modeling environment without having modeling knowledge and, of course, can help modeling experts to make deployments in cloud without having cloud skills. So, the main research objectives are the following:

- Apply the advantages of cloud deployment in the modeling domain by developing a platform that assists and automates the multicloud deployment of a software environment
- Evaluate the performance of the platform and the complexity of the deployments
- Integrate side features, such as model interpreters, to further simplify the tasks that needed to be done without requiring technical skills.

## 1.2 Research Storyline

A first study was conducted to involve master's students in a collaborative research experience for developing an object-oriented modeling language specific to the domain of hazard management systems [6]. In this phase, the collaboration was based on class presentations and discussions, followed by the work with a desktop application for creating modeling tools – Generic Modeling Environment (GME) [5]. The research method was clearly defined in advance, but the collaboration was mainly oriented towards a conceptual work, whereas the technical contributions were individual. In a second phase, GME was deployed in a Cloud system based on IBM Service Delivery Manager (ISDM), replicating a separate virtual machine for each student subscribed to the Model-Driven Engineering course [7]. As a consequence, students could work in the same environment in the class and at home, and the teachers had access to monitor the progress. This provision of access to using a modeling tool is an example of Modeling as a Service, an approach situated above Software as a Service in the Cloud Service Models organization chart [8]. Cost-saving is a big focus in this period, so moving to Cloud can support this because resources are maintained as long as the user needs them.

However, not all Cloud providers have the same costs and services that can complement the offering of a modeling platform and the preferences can differ from one user to another, so the idea of multicloud comes up in this case with several alternatives. As a result, this work leans on the phases described above and introduces two new elements. On the one hand, it is used WebGME (Web-based Generic Modeling Environment) [9] to address the limitations regarding collaboration facilities. On the other hand, the deployment is done in one of the 3 supported clouds: AWS (Amazon Web Services), Azure, IBM Cloud and OpenShift. Also, to benefit from the advantages such as high availability, flexibility, and scaling provided by the containerized approach, I used Kubernetes and Docker instead of virtual machines. The focus is on simplicity because in the current context the targeted users have to use a lot of online platforms, which makes their work very difficult. Even if a platform has a fairly high theoretical utility, it can be inaccessible to many users if it has a very high level of complexity. Moreover, not all users are at the same level concerning their technical skills, so this argument must be taken into account when choosing a platform for educational purposes.

When initiating the use of cloud resources there may be processes to be followed that have many manual tasks and require specialized knowledge to be executed. This stands in the way of a much larger adoption of this kind of service provisioning. For example, the complexity of high-performance computing (HPC) workflows is analyzed in [10]. Li et al. model and analyze the workflows for the auto-scaling scheduler, the job management, and the metering management, and propose a containerized cloud platform to reduce their complexity and assist their users and administrators. The deployment of such a platform remains a challenging task. Apart from complexity, it is also important to attentively monitor the performance of such services for virtual machines and containers, based on standard benchmarks or in regard to what is offered by the native hardware [11]. Bystrov et al. evaluate communication- and computation-intensive services on virtual cloud resources, whose performance depends on MPI (Message Passing Interface) communication, load mapping, and overhead [12].

As a result, I also focused on the performance validation of the automatic deployment in comparison with manual deployment, from the perspective of user interaction and the prerequisites required for the manual tasks. I use existing complexity metrics, and I also propose a specific way to evaluate the complexity of human tasks for cloud deployment. The question is how big the differences in complexity are between the manual and the automated deployments, such as to motivate the development of a platform to automate most of the tasks. This would allow a specialist in the application domain to adopt the EaaS approach even without specific knowledge and know-how in cloud computing. Nonetheless, the performance of the resulting service is also important to the adoption of such a solution, and CPU and memory consumption monitoring are also necessary.

## 1.3 Thesis Overview

Chapter 2 will present the analysis of the state of the art, including the analytic methods, and limitations in contrast with the thesis objective. It will reflect the thesis domains and will present the lessons learned from studied bibliography. The main research areas were modeling and cloud. They are also divided into subjects such as metrics, theory elements, implementations, and also systems such as learning systems – the main area where the multicloud platform can be applied to. Alongside what is presented prior, this chapter will present articles that refer to deployment methods and cloud performance analysis.

Chapter 3 presents the objectives of the thesis: the general and the specific ones. Those objectives refer to the research goals and present what was intended to be achieved during the years of doctoral study. Alongside the description of the objectives, chapter 3 also presents the research method that was used to achieve the objectives. As a result, every step that was done is described starting from the related work to the frameworks used for the development and architecture of the application. This represents the flow of the research.

Chapter 4 presents in detail the personal contributions achieved for this thesis. It describes what was done to achieve the objectives presented in Chapter 3. It is starting with the description of manual and automatic deployment, and the solution description for automatic deployment, the deployment methods are validated by using complexity metrics of the business process and by proposing a new and original complexity metric. After the complexity of the business processes is presented, the means of achieving the cloud metrics are explained, to obtain the complexity results for the cloud deployments. The chapter continues with the decision algorithm that was created to establish the deployment details and the five outcomes that result from it. The chapter ends with an overview of the technologies used for the implementation of the platform.

Chapter 5 represents the verification and validation of the contributions exposed in the previous chapters. It starts with the migration of a GME metamodel to a WebGME environment deployed on the Kubernetes cluster in IBM Cloud. This validated the primary role and functionality of the platform – it is working as expected and also identifies the difficulties of a metamodel migration from GME to WebGME. The chapter continues with the performance results of the cloud deployment represented by the monitoring results, including the evaluation of the decision algorithm presented in Chapter 4. The chapter persists with a discussion that targets the modeling environment in the context of a microservices architecture built on top of docker and Kubernetes. Nonetheless, the complexity metrics results are analyzed alongside the results of the performance of Cloud deployment. The chapter ends with a discussion of the automated deployment platform and the main capability that it offers.

Chapter 6 presents the conclusions alongside an overview of the original contributions and future perspectives. It also includes the list of published articles.

Appendix 1 chapter presents in detail the multicloud deployment platform from the backend and frontend perspectives. Part of the API provided by the backend is shown in detail alongside its Swagger specification. The Angular Frontend is presented alongside the features that the platform offers. It represents a visual overview of what was practically realized to sustain the proposed theory. Finally, it presents the view of monitoring deployable from a CLI connected to the Kubernetes cluster.

# 2. Analysis of the State of the Art

The two main domains of research for the thesis are Modeling and Cloud deployment. Those two main subjects are also split into sections that target the theoretical elements, performance, and validation. The research regarding learning systems is treated in a separate subchapter.

## 2.1 Modeling Aspects

I have studied subjects such as Model and Modeling as a Service, Model-Driven Architecture, and Model-Driven Engineering. From the articles that are in the Modeling and Model as a service category, I have achieved the required knowledge related to the implementation of a model and metamodeling environments in the cloud and the way of working with such tools. Furthermore, I have achieved theoretical knowledge related to this filed and the know-how implementation of a similar environment in the cloud. Many of the present articles had a brief description of the infrastructure that provided the model/modeling as a service capability. Furthermore, I have learned about the differences between modeling tools such as GME, WebGME, AToMPM, MDEForge, etc. Most of these tools had features in common such as collaboration, scalability, and model persistence. Those features were compared from the performance perspective. As a result, I made an idea related to these tools and their capabilities. Another studied topic is Model-Driven Architecture. I have learned about the history and theory of MDA alongside some analysis of approaches culminated with practical examples. In general, this approach encourages faster software development alongside flexibility in the technologies used. The next domain studied is related to multi-paradigm modeling in which, during the studied articles I have learned about some theoretical aspects related to the model, metamodel, semantics alongside some examples provided by the authors. Other studies were related to the approach applied in the cyber-physical systems field and the description of such implementation.

## 2.2 Cloud Computing Aspects

This is the second domain of the thesis besides the modeling environments. This section describes related work that refers to Kubernetes, cloud deployment, complexity metrics for cloud deployment, and performance analysis of the deployment.

# 3. Multicloud Deployment of Modeling Environments

## 3.1 Objectives

The main research objective is to apply the advantages of cloud deployment into the modeling domain, providing a bridge between the two skill sets: modeling and cloud computing. As a result, cloud experts will be able to deploy a modeling environment and model interpreters without having modeling skills, and modeling experts will be able to deploy modeling tools in cloud without the need for cloud skills. Therefore, a specific objective is to deliver a multicloud platform that can deploy modeling environments alongside model interpreters on one of the three clouds: AWS, Azure, IBM Cloud, and OpenShift Online. The platform capabilities can be expanded making it a hub where the users can deploy the modeling environment of their choosing and can control the deployment as they seem fit. The deployment of the environments is done in a few clicks by the registered user and can be utilized also by non-technical users. The platform can deploy the modeling environment and model interpreters in one of the three clouds and also OpenShift Online if advantages such as integrated monitoring are needed to be exploited by the users. Furthermore, the platform will have the ability to help the user in choosing the more suitable modeling environment for them by asking them a set of questions. Based on the answers, the algorithm will advise the users on which platform to deploy in Cloud. Due to this advice, the decision will be simpler, and it will also differentiate the use and utility of both of the available modeling environments.

In the context of the coronavirus pandemic, this platform could also help students and researchers with their tasks that are conducted under imposed lockdown. For example, this platform can supplement a laboratory environment and can provide the teachers with a more controllable manner for the deployed resources. Furthermore, the students will be able to collaborate on the projects by taking advantage of the features provided by the platform and cloud deployment benefits. This environment can be taken down after each use to save money. With pay as you go subscription that is provided by most cloud providers, costs can be reduced for environments that do not have the purpose to run continuously for a long time. As a result, the resources can be deleted, and the models/projects can be saved on cloud storage that will remain on the user’s subscription. This kind of storage is not as expensive as a Kubernetes cluster or even a medium-sized virtual machine. Also, in the same context, the teachers and students have some difficulties in adopting the new technologies, because due to the imposed lockdown, many platforms have arisen to offer more benefits to this type of user. Most of these platforms can be hard to use or difficulty can appear due to the multitude of new platforms so; in my opinion, simplicity is the key and the focus when the development of a new platform is started. As a result, the main objectives are:

1. Proposing a new multicloud platform that can automatically deploy a modeling environment in a cloud of the user's choice.
2. Evaluation of the automatic and manual deployment methods and their comparison based on criteria defined by complexity metrics.
3. Analysis of the performance and cloud deployment using cloud metrics.
4. Elaboration and implementation of a new decision algorithm for choosing the cloud deployment details.
5. Providing examples of model interpreters' integration with the multicloud platform and evaluation of the results.

## 3.2 Method

Based on the previously presented objectives, the method below determines the activities done to achieve these objectives.

The first and most important thing to understand is how what is proposed can be achieved, to determine how other researchers approached this issue and what tools are the most well-suited for such an implementation. Fig. 3.1 describes the steps that were done for the research, the realization of the solution and the achievement of the results. All of the presented steps are based on the previous ones.

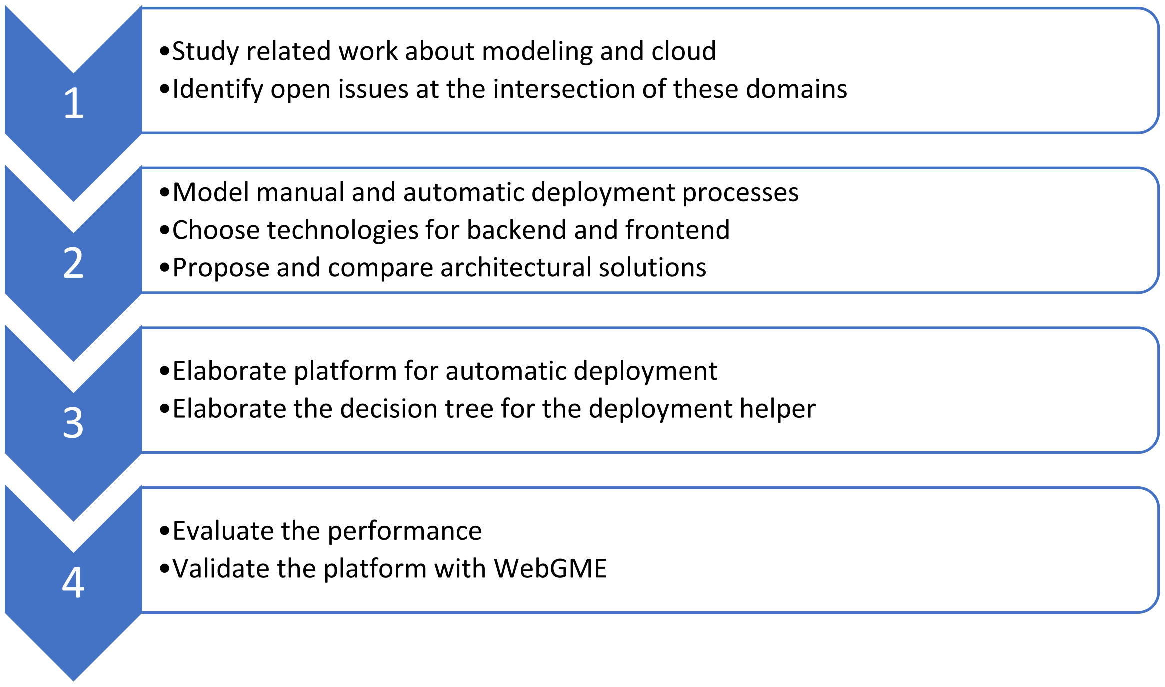

1
•Study related work about modeling and cloud
•Identify open issues at the intersection of these domains
2
•Model manual and automatic deployment processes
•Choose technologies for backend and frontend
•Propose and compare architectural solutions
3
•Elaborate platform for automatic deployment
•Elaborate the decision tree for the deployment helper
4
•Evaluate the performance
•Validate the platform with WebGME

**Fig. 3.1 Solution implementation method**

# 4. Contributions to the Automatic Deployment of Modeling Environments

## 4.1 Manual Deployment of a Modeling Environment. Example for IBM Cloud

The first test for the manual deployment was the implementation of the chosen modeling environment – WebGME in IBM Cloud. This is very important for users that want to contribute to a private modeling project and can be achieved using containerization (with technologies such as Docker and Kubernetes). Before the deployment, a Kubernetes service is needed to be deployed, using the IBM Cloud academic license. There is a pod that manages at least a container; for containerization purposes, it was used Docker, based on a custom WebGME image created with a Dockerfile. The Dockerfile for WebGME requires a configuration file that pinpoints to the Mongo Container. So, this is the first pre-requirement necessary before building the image.

The manual deployment and the requirements necessary for the deployment to be completed are presented in Fig. 4.1.

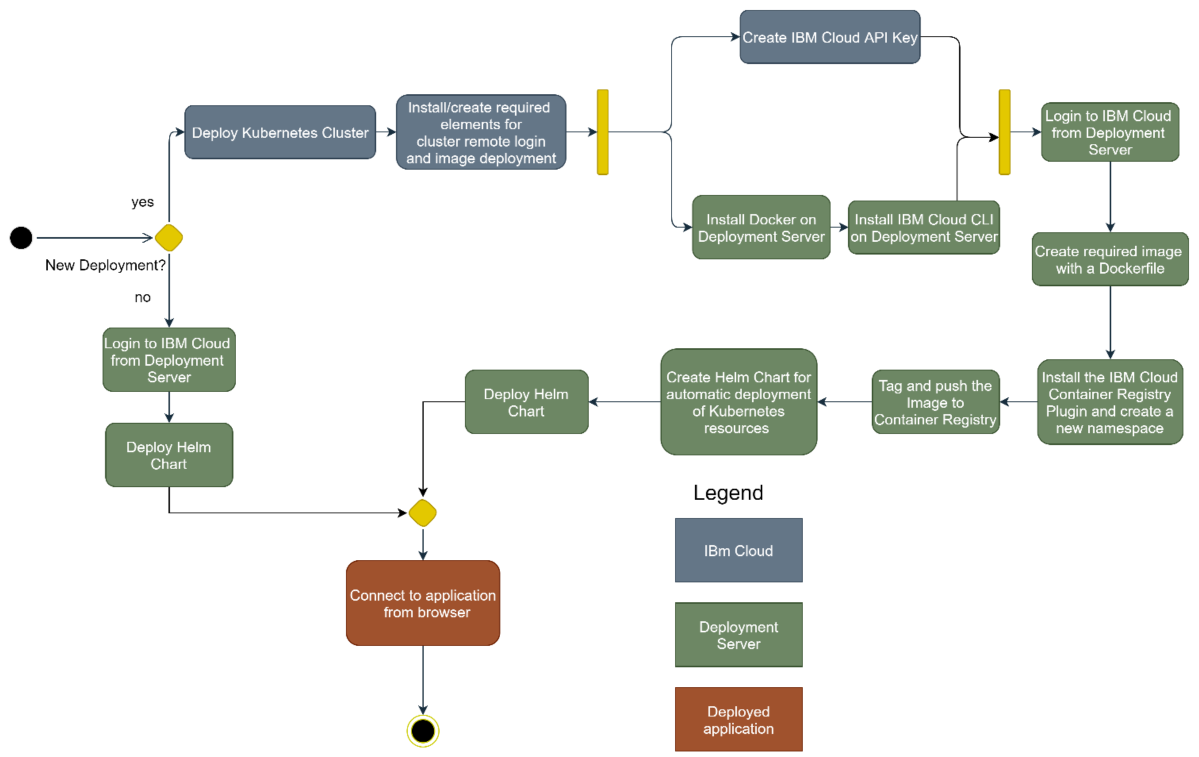

Create IBM Cloud API Key
Deploy Kubernetes Cluster
Install/create required elements for cluster remote login and image deployment
Login to IBM Cloud from Deployment Server
yes
New Deployment?
no
Install Docker on Deployment Server
Install IBM Cloud CLI on Deployment Server
Create required image with a Dockerfile
Login to IBM Cloud from Deployment Server
Deploy Helm Chart
Create Helm Chart for automatic deployment of Kubernetes resources
Tag and push the Image to Container Registry
Install the IBM Cloud Container Registry Plugin and create a new namespace
Deploy Helm Chart
Legend
IBm Cloud
Deployment Server
Deployed application
Connect to application from browser

**Fig. 4.1 WebGME manual deployment tasks (IBM Cloud)**

Fig. 4.1 describes the deployment process of the modeling environment in IBM Cloud. This method requires advanced technical knowledge from the user, and it is also very costly from a time perspective. The process starts with the deployment of the Kubernetes cluster if it is a new deployment and continues with the prerequisites for a deployment server. The prerequisites imply the creation of the API key (on the IBM Cloud platform), and the installation of Docker and IBM Cloud CLI on the deployment server. After that, the user needs to login into the IBM

Cloud workspace from the deployment server. Next, the user will create an image for the modeling environment. This image is based on a Dockerfile that needs to be configured. The configuration tasks continue with the installation of the IBM Cloud container registry plugin useful for connecting to this component. Also, a namespace needs to be created. The namespace has the role to group the deployed Kubernetes resources. After, that, the image is pushed to the IBM Cloud container registry. The next steps imply the creation of the deployment and services for WebGME and MongoDB. These services will be automatically deployed on the Kubernetes cluster rather than deploying the services one at a time. The process ends with the connection to the newly deployed environment. If the perquisites are already made and this is not a new deployment the matter will be simpler. The user only needs to login to the IBM Cloud account and deploy the Helm charts that are already created. With the help of the Helm charts, the user can also destroy the environment at will. This makes the deploy/destroy action very accessible. These actions need lots of technical knowledge and know-how for these technologies. In my opinion, a user shouldn't need this kind of skill just to deploy an environment more efficiently.

## 4.2 Automatic Deployment Using the Multicloud Platform

Using the multicloud platform simplifies the user's work that needs to be done. In this case, the user will still have to deploy the Kubernetes cluster. This can be done very easily from the IBM Cloud web console. Also, the user needs to create and retrieve an API key. This task differs from one Cloud provider to another because the connection to the Kubernetes cluster requires other means that are specific to a cloud provider. In the case of IBM Cloud, just an API key is needed to be provided by the user at the specific step. First of all, the user needs to register to the platform after the Kubernetes cluster is deployed and the API key is retrieved. Next, the user should create to have the ability to add users and connections. This user can have many projects, so the current project that will be in use must be set as active. After that, the user will select to which cloud he/she wants to connect. The current options are IBM Cloud, AWS, Azure, and OpenShift Online. After these selections, the user will be prompted to add the connection requirements. For this example, the user will be required to provide the API key retrieved at the prior step and the Kubernetes cluster name. After that, the user will set the connection active because as in the project case, a user can have multiple connections to multiple Kubernetes clusters that reside in the three supported Clouds and OpenShift Online. The final step is represented by the deployment of the application and by the connection of the user to it.

In comparison with the steps that a user undertakes with the previously presented method, in this case, the advanced technical knowledge that is required for the configuration part is not needed and is done automatically from the cluster. Practically the user will only interact with the user-friendly interface of the Cloud provider and the multicloud platform. The deployment server is practically inexistent for the user and all of the tasks are handled by the backend of the application. The described tasks are represented in Fig. 4.2 below.

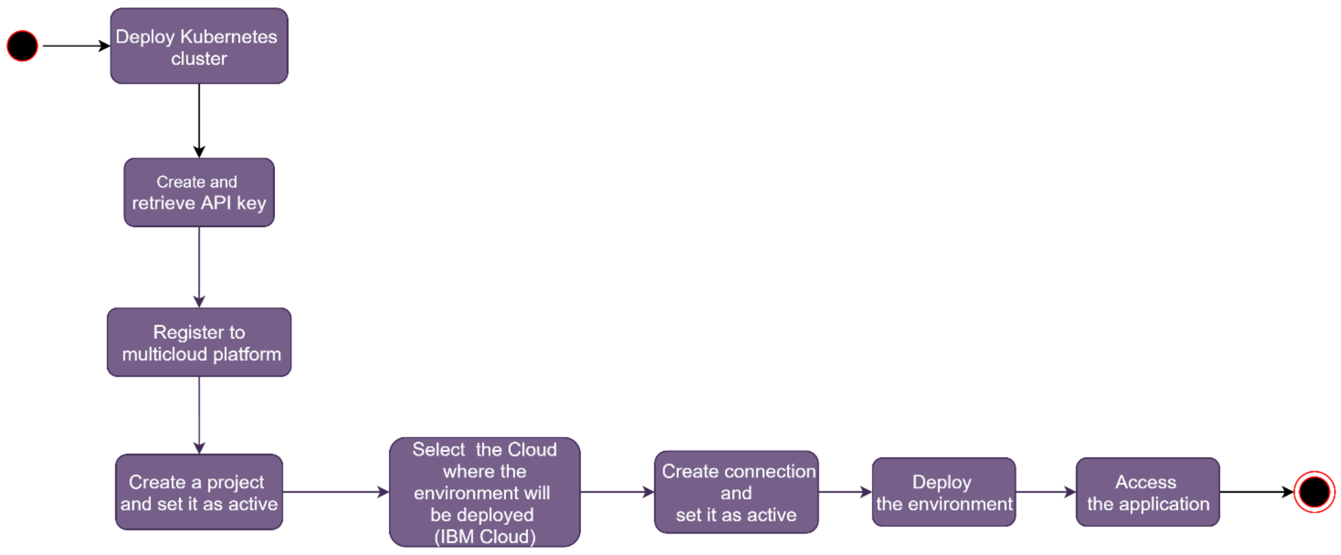

Deploy Kubernetes cluster
Create and retrieve API key
Register to multicloud platform
Create a project and set it as active
Select the Cloud where the environment will be deployed (IBM Cloud)
Create connection and set it as active
Deploy the environment
Access the application

**Fig. 4.2 Environment deployment from multicloud platform**

### *4.2.1 Model interpreter automatic deployment*

Two methods were proposed to provide the model interpreter as a service. The first solution involved the use of cloud storage, this is an issue if we take into account the overall solution, as the user is forced to invest in expensive methods to be able to access this feature. As a result, I have chosen the second method that implies the use of encapsulated interpreters in the docker image. The disadvantage of this method is the use of multiple images. The disadvantage is that it is hard to manage and update multiple images. For example, if a new version of WebGME is released, all of the images need to be updated accordingly. In comparison with the first method, this issue is acceptable because it is on the developer's side, not on the user's side. The developer needs to update its images and provide them to the user. The first method implies that the user needed to purchase a service in the cloud (storage). This is unacceptable in contrast to this solution.

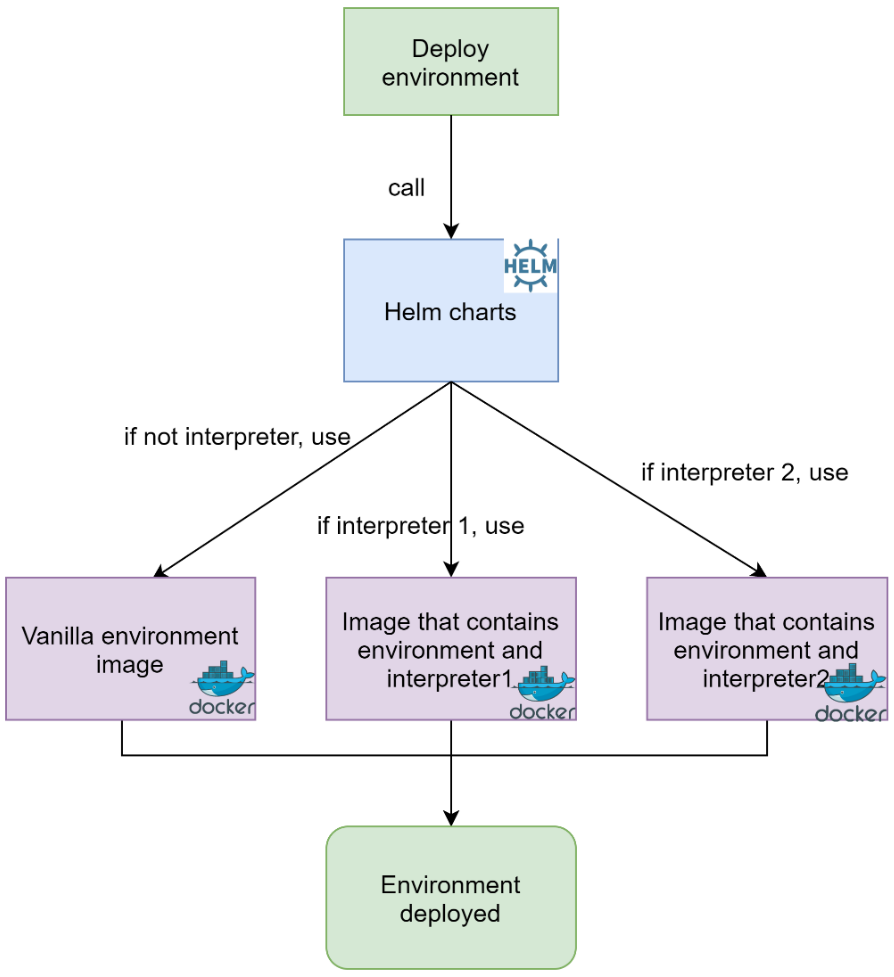


**Fig. 4.3 Multiple docker image Helm deployments**

Fig. 4.3 illustrates the multiple docker image method from the helm chart deployment perspective. At first look, this will have the same effect on the user: the environment will be deployed in the same way. Depending on the selected interpreter, the environment will be deployed with that plugin included and it will be shown in the WebGME web console, under the plugin selection menu.

## 4.3 Complexity Metrics for Manual and Automatic Deployment

After the presentation of the solution in the previous subchapters, I will present the complexity metrics applied to determine what is the difference in complexity between the manual deployment presented in Fig. 4.1 and the automatic deployment presented in Fig. 4.2. The manual and automated deployment processes are analyzed in Chapter 4.3.1, first based on complexity metrics presented in the scientific literature and then with the complexity of human tasks, a specific metric defined for the scope of our work.

### 4.3.1 Choice of complexity metrics for the Deployment Process Analysis and Results

First, I use a classic metric, cyclomatic complexity [13], for the complexity computation of the deployment processes. Cyclomatic complexity, introduced by Thomas McCabe in 1976, determines the complexity of the program.

Therefore, the cyclomatic complexity is 3 for the manual deployment process and 1 for the automated one. A lower complexity value for the latter indicates that the automated deployment requires less work from the user side and confirms the need for a platform developed for this purpose.

Second, I apply the Yaqin complexity formula, based on a series of metrics that are calculated for each component of the process seen as a graph [14].

After calculating all the equations, the Yaqin complexity (*YC*) is 78,5 for the manual deployment and 35 for the automated deployment.

Third, for the scope of my analysis, I define a specific metric based on the technical knowledge required to complete a manual task, which is one of our concerns. For this purpose, a weight is assigned to each of the process tasks, according to the following criteria:

- 1 - an easy task – minimum tutorial following needed
- 2 - medium task – specific technical knowledge required
- 3 - hard task – specific technical knowledge and experience necessary.

The complexity of the process from the point of view of the user who is responsible to execute the manual tasks is calculated as:

$$CH = \sum_{t=1}^{n} Wt \tag{1}$$

where *CH* is the complexity of human tasks involved in the deployment process, *Wt* is the weight for a given task and *n* represents the number of the final task.

Based on the criteria, the resulting complexity of human tasks (CH) — calculated with the specific metric proposed is 20 for the manual deployment and 8 for the automated one.

After computing the complexities (for manual and automatic deployment) following the explained criteria, the results are as expected. Manual deployment is 2 to 3 times more complex and requires more technical knowledge from the user in comparison with automatic deployment. Taking into account this measurement, the automatic deployment saves time and can be utilized by users without technical requirements. So, after presenting these cases I can fairly say that the automatic deployment brings a lot of value to the entire experience.

## 4.4 Monitoring Tools for Performance Analysis and Cloud Metrics

Alongside the process metrics studied in the previous subchapter, a topic as important is related to the cloud metrics and the performance of the deployment on the Kubernetes service. In this regard, I have chosen a more standard monitoring approach: Prometheus. According to [15], Prometheus is a monitoring, alerting tool, and time series database that provides a query language named PromQL. This language gave the user the ability to make expressions (queries) that can display a graph or a table in real-time. These days, Prometheus is very popular in the containerization world due to its default integration with the enterprise Kubernetes platform

OpenShift. As visualization software for the Prometheus tool, I have used Grafana which properly integrates with Prometheus. Prometheus offers four types of metrics used in different situations [16]:

- Counter - used to count the number of requests, increasing until it resets to zero
- Gauge – a counter that can go up and down; used mostly for memory and CPU metrics
- Histogram – used to compute request duration and also the sum of duration for all requests
- Summary – similar to histogram; additionally, it provides a sum of all of the values.

## 4.5 Decision Algorithm that Helps the User Establish Deployment Details

This chapter describes the decisive factors for the deployment of modeling environments from the multicloud platform. These decisions are made by leveraging the capabilities of the decision trees algorithm and also by providing a decisional algorithm that helps a user decide the deployment details.

The decision tree algorithms are applied to a given dataset that is based on the user's options. Chapter 4.5.1 describes the application of the ID3 algorithm to determine the split condition. At this step, we do not take into account the cloud where the modeling environment will be deployed.

### 4.5.1 Deployment based on the ID3 decision tree

I apply the ID3 algorithm to a dataset that represents the combination of deployment decisions and their outcomes. The first decision trees should predict in what conditions WebGME should be deployed. The dataset was created by taking into account the deployment decisions, as presented in Table 4.1.

**Table 4.1** Deployment decisions for WebGME

| Subscription | Programming language | Collaboration | WebGME |
|---|---|---|---|
| **free** | C++/Java | yes | yes |
| **paid** | C++/Java | yes | yes |
| **fee** | javascript | no | yes |
| **paid** | javascript | no | yes |
| **free** | C++/Java | no | no |
| **paid** | C++/Java | no | no |

Fig. 4.4 depicts the resulting decision tree for the datasets presented in table 4.1. It also depicts the choices that a user needs to make to deploy WebGME.

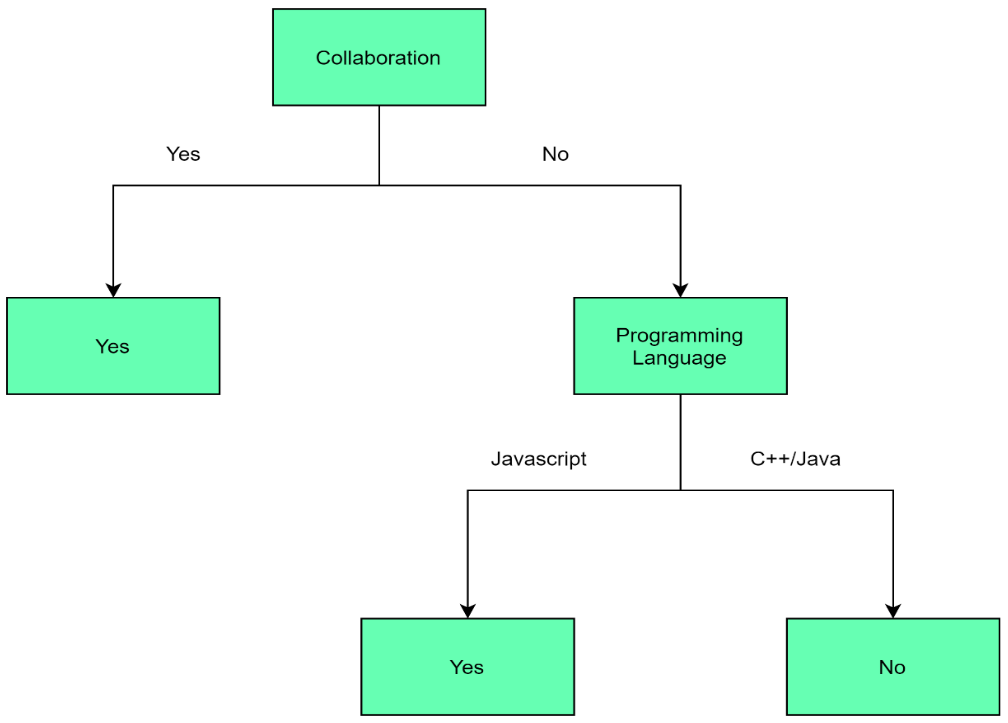


**Fig. 4.4 Decision tree after ID3 algorithm finished building it**

### 4.5.2 Decision algorithm to deploy GME / WebGME in a multicloud environment

After predicting a user's choice based on his or her preferences, such as collaboration or programming language, we propose an algorithm that advises the user which package is better to deploy, and in which cloud. In comparison with the datasets used earlier, I introduce supplementary features to differentiate the deployment choice. The following criteria are taken into account:

- Subscription: Paid, Reserved/on-demand
- Programming Language: C++/Java/Javascript
- Collaboration: Yes/No
- Time of use: uninterrupted for a month, environment destroyed after use.

The representation of the algorithm from Fig. 4.5, 4.6 and 4.7 is meant to help the user decide what and where to deploy based on question, choice, and outcome. For the paid subscription part of the tree, I am not making advice for the Cloud provider as the user is required to have a paid active subscription. For the free part, I am advising the choice of the Cloud provider because a free subscription is accessible to all users.

The first criteria of the split are based on the type of subscription that a user can own in the cloud (Fig. 4.5).

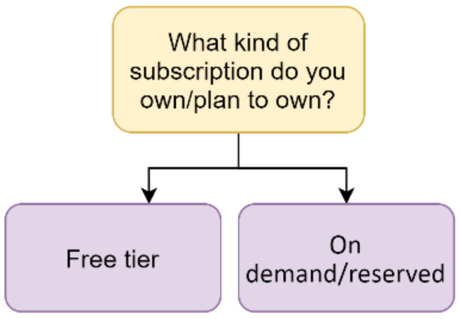


Fig. 4.5 Decision tree root

This criterion splits the decision trees into two halves that are generating different outcomes. The free tier part generates three outcomes and the On-demand/reserved part provides two outcomes.

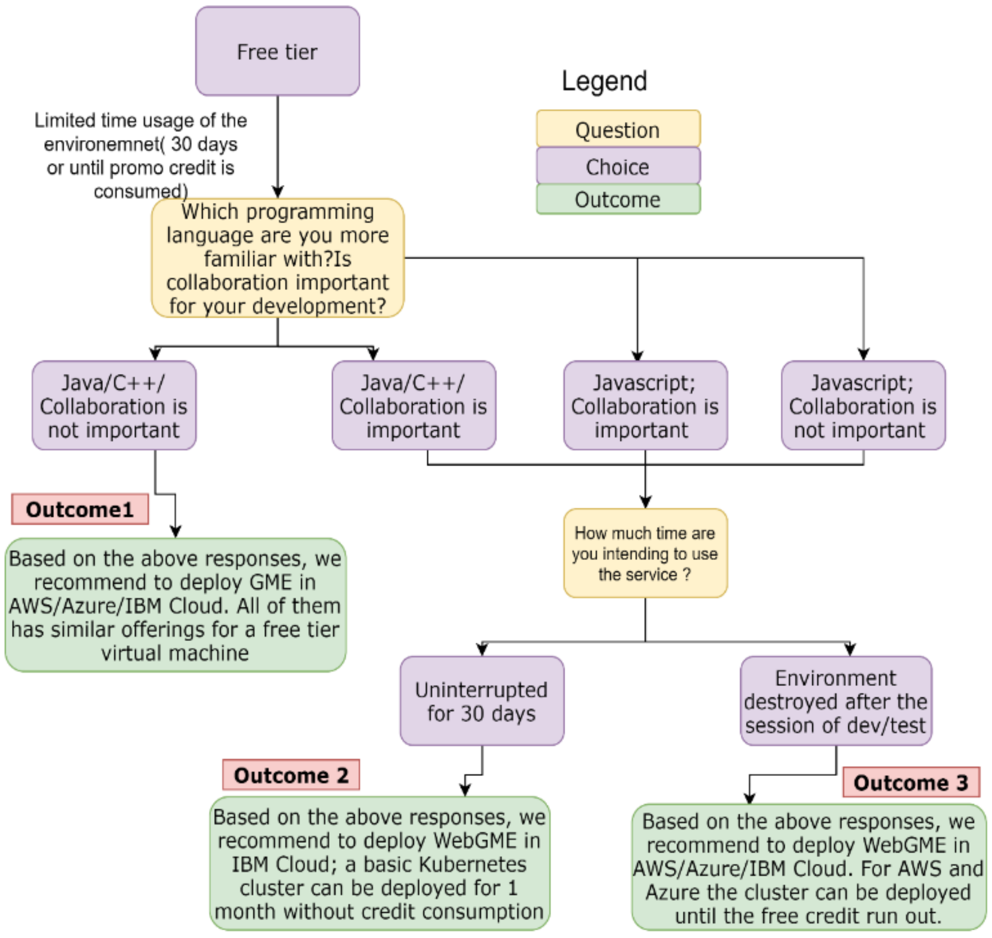

Free tier
Legend
Question
Choice
Outcome
Limited time usage of the environemnet( 30 days or until promo credit is consumed)
Which programming language are you more familiar with?Is collaboration important for your development?
Java/C++/ Collaboration is not important
Java/C++/ Collaboration is important
Javascript; Collaboration is important
Javascript; Collaboration is not important
Outcome1
Based on the above responses, we recommend to deploy GME in AWS/Azure/IBM Cloud. All of them has similar offerings for a free tier virtual machine
How much time are you intending to use the service ?
Uninterrupted for 30 days
Environment destroyed after the session of dev/test
Outcome 2
Based on the above responses, we recommend to deploy WebGME in IBM Cloud; a basic Kubernetes cluster can be deployed for 1 month without credit consumption
Outcome 3
Based on the above responses, we recommend to deploy WebGME in AWS/Azure/IBM Cloud. For AWS and Azure the cluster can be deployed until the free credit run out.

Fig. 4.6 The left section of the decision tree

The right section (Fig. 4.7) of the decision tree refers to the paid subscriptions and generates two Outcomes.

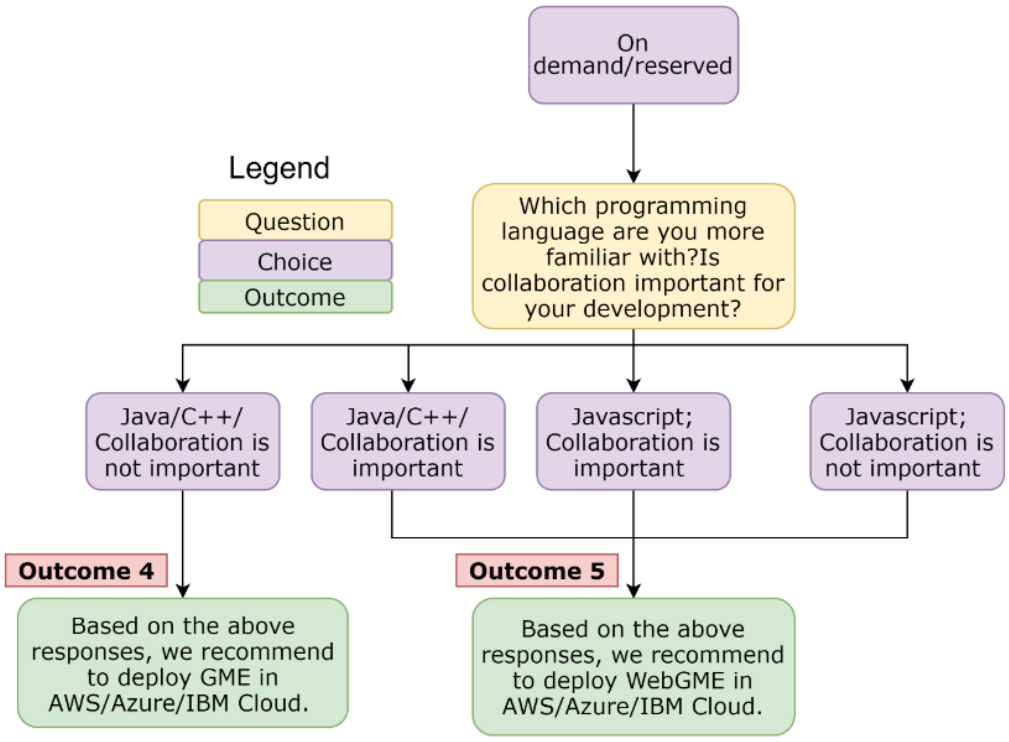

On demand/reserved
Legend
Question
Choice
Outcome
Which programming language are you more familiar with?Is collaboration important for your development?
Java/C++/ Collaboration is not important
Java/C++/ Collaboration is important
Javascript; Collaboration is important
Javascript; Collaboration is not important
Outcome 4
Outcome 5
Based on the above responses, we recommend to deploy GME in AWS/Azure/IBM Cloud.
Based on the above responses, we recommend to deploy WebGME in AWS/Azure/IBM Cloud.

**Fig. 4.7 The right section of the decision tree**

## 4.6 Platform Implementation - Technologies

As stated previously, the technologies used are the ones that are usually characteristics to Cloud deployment. The application uses the latest containerization technologies that are available on the chosen Cloud providers: AWS, IBM Cloud, Microsoft Azure, and OpenShift Online. The infrastructure contains a server from which the deployment is made. This server must be configured to run Docker, Terraform, and Helm. Also, the CLI tools for the cloud providers need to be installed on the deployment server. The deployment scripts use these technologies to deploy the modeling environment in one of the three cloud platforms and OpenShift Online. Alongside the deployment technologies, on the same server, the application and web server are installed to run the web portal. The multicloud platform was created using Node.js for the backend and Angular 10 for the frontend. These technologies were chosen due to the efficiency that can provide to the application based on criteria such as faster loading times and also the use of asynchronous programming provided by node.js and observables provided by the Angular framework that is also used for asynchronous programming and messages that are sent between components of the application. The persistence of data is ensured by MySQL database that runs on a docker container on the deployment server that has a persistent volume attached.

The deployment to cloud can be made without Helm but, to have a fully automatic deployment, these technologies help a lot in achieving this. The technical specification is presented in Fig. 4.8

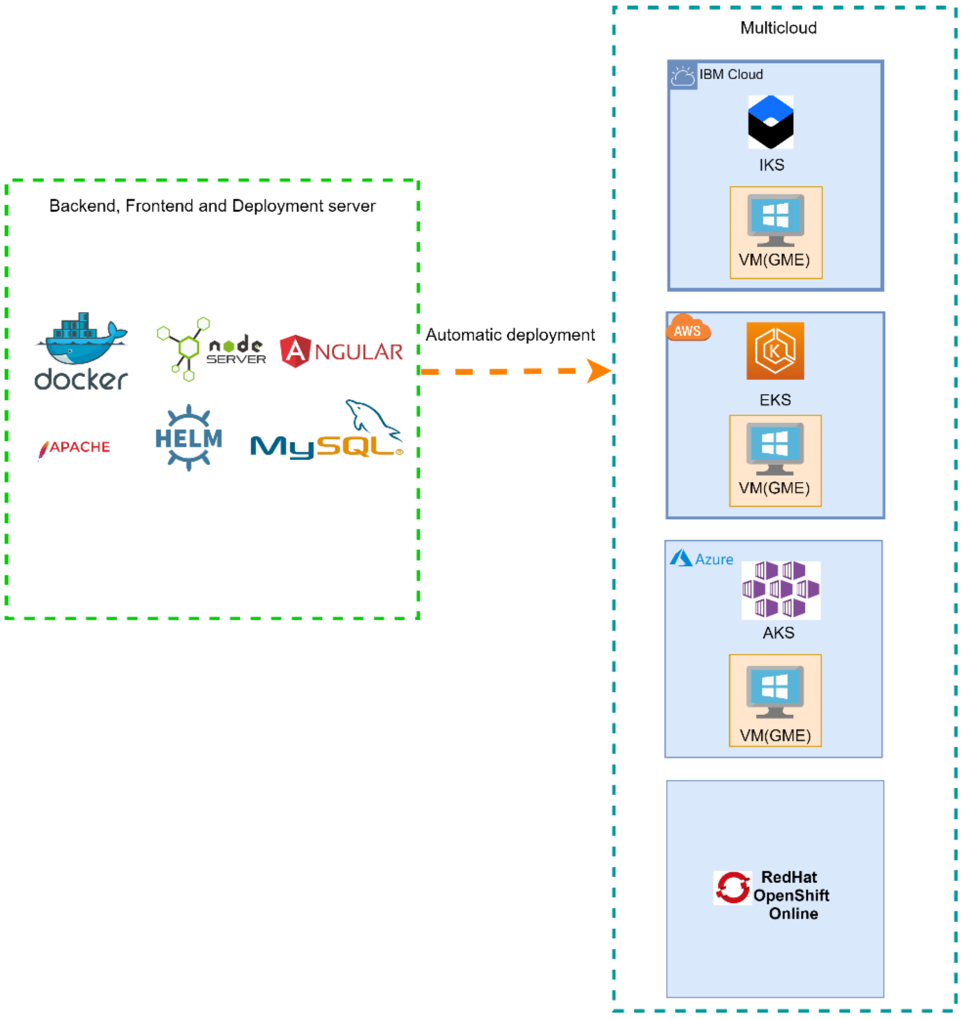


**Fig. 4.8 Overview of the technologies employed for the new platform**

Regarding the technologies used for deployment, they are:

- Helm for the deployment of Kubernetes resources
- Docker for image creation and management
- Apache – web server
- MySQL – data persistence

# 5. Evaluation and Validation

## 5.1 WebGME Deployment in Cloud - Features Validation Example

The WebGME Cloud implementation was tested for a metamodel that was previously presented in [6] and [17]. Fig. 5.1 shows a part of the abstract syntax of this metamodel, conceived for creating a graphical modeling language specific for sensor networks, which have become important to many systems for data collection and fusion capabilities [18]. The metamodel was represented in GME - the on-site metamodeling environment studied here. A *SensorNetwork* is composed of several *SensorNetworkUnit* elements, which can be units for communication, processing or sensing – the last ones potentially composed of multiple sensors. There may also be connections between these units, as well as dependencies between the software units that are part of the processing units. In addition to the elements represented in Fig. 5.1, the metamodel contains further details regarding the communication, memory, power, and sensor aspects. Based on this metamodel, we created models for testing purposes, to be able to find the information necessary for a side-by-side comparison.

The paradigm created with GME was imported in WebGME deployed in IBM Cloud, using the MetaGME importer plugin, via the .xmp file. Such an import is characterized, however, by several limitations [19], like the impossibility to import roles, constraints, cardinality, aspects and visual properties, as well as the usage of kinds instead of roles. The result of this export for the sensor network metamodel is illustrated in Fig. 5.2.

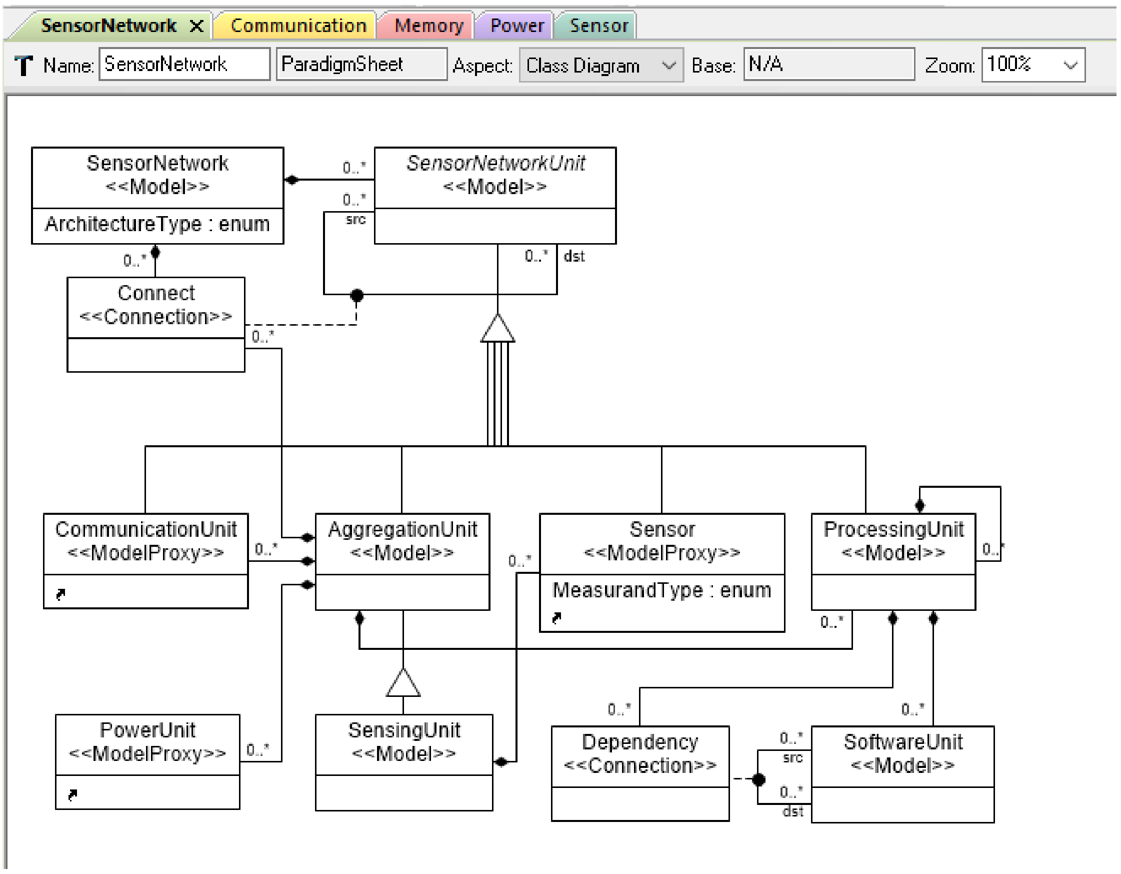


Fig 5.1. Part of the metamodel from the on-site metamodeling environment (after [6])

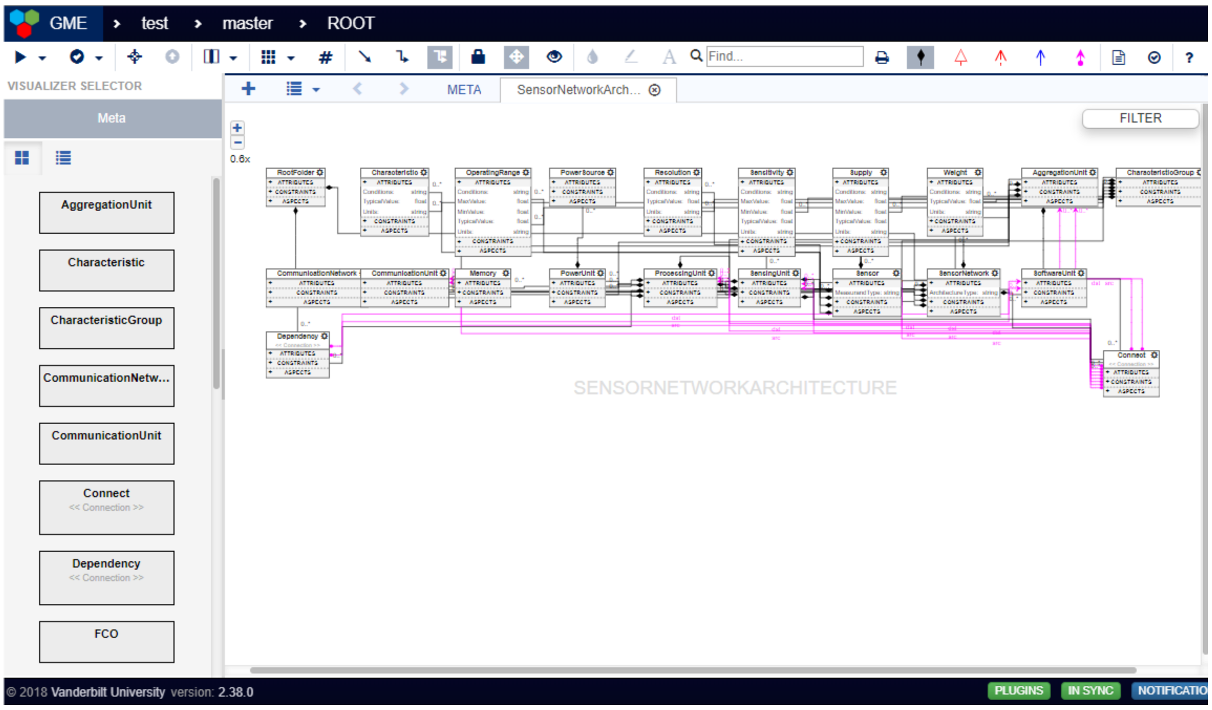


Fig. 5.2 Modeling environment migrated to the Cloud

## 5.2 Cloud Deployment - Performance Test Results

Apart from the results obtained at the statical evaluation, this section presents the performance obtained at executing the tests with Prometheus & Grafana and Sysdig, for the deployment of WebGME.

The results for the monitoring tool metrics are given in Table 5.1. These metrics are used for performance measurement, and they also indicate the usage of the application. They were obtained by running the Prometheus PromQL queries. The queries provide real-time data that help the user visualize the state of the deployed resources. These test results are obtained after the normal usage of the application by two users; the CPU resources are measured in thousands of cores - millicores (m). The Mongo and WebGME pods CPU consumptions are given in Table 5.1. Part of those metrics was presented in [20] and also in [21], a monitoring solution is presented for cloud-edge devices.

**Table 5.1** Prometheus monitoring results for containers

| Metric | Result (1 hour time span) |
|---|---|
| Container CPU Mongo | 2 m ÷ 8 m |
| Container Memory Mongo | 70 MB |
| Container CPU Webgme | 0.5 m ÷ 2 m |
| Container Memory Webgme | 90 MB |

The results of the cluster monitoring are presented in Table 5.2. They were obtained using the PromQL queries. The purpose of these queries is to monitor the state of the cluster resources during the application execution The results represent the average consumption during a time span of one hour.

**Table 5.2** Prometheus cluster metrics results

| Metric | Result |
|---|---|
| Cluster memory usage | 47.5 % |
| Total memory | 3.84 GB |
| Used memory | 1.82 GB |
| Cluster CPU usage | 21.34 % |
| Total CPU | 2 cores |
| Used CPU | 0.43 cores |

For comparison, Table 5.3 also showcases the results of the metrics presented on the Sysdig console. They are obtained in the same conditions as the ones encountered during the Prometheus PromQL querying and represent the CPU and memory consumptions for the Mongo and WebGME containers; the CPU resources are measured in cores. The Sysdig monitoring dashboard in IBM Cloud provides real-time data regarding the CPU and memory for the *mongo* and *webgme* containers, for 1 hour. The average CPU and memory consumptions monitored with this dashboard correspond to the values from Table 5.3.

**Table 5.3** Sysdig monitoring results for containers

| Metric | Result (1 hour time span) |
|---|---|
| Container CPU Mongo | 0.01 cores |
| Container Memory Mongo | 63.21 MB |
| Container CPU WebGME | 0.01 cores |
| Container Memory WebGME | 65.75 MB |

Regarding the memory and CPU usage of the cluster, they are monitored in a default dashboard on the Sysdig console. As this is not a custom dashboard, I obtained the percentage of consumed CPU and memory presented in Table 5.4. The results represent the average consumption for one hour.

**Table 5.4** Sysdig cluster metrics results

| Metric | Result |
|---|---|
| Cluster memory usage (%) | 30.74 % |
| Cluster CPU usage (%) | 23.63 % |

I have also studied the implementation of other agent-based monitoring solutions such as Dynatrace [22], AppDynamics [23], and NewRelic [24]. All of these solutions offer multiple capabilities and in some regards are more capable than Prometheus and also offer a SaaS (Software as a service) approach for the monitoring dashboard but, all of them require a pricey monthly subscription because they contain access to many features that are not needed in the current setup. Also, other platforms [25],[26],[27] that have also an educational and informal scope could benefit of such monitoring solutions and the implementation can be based on what is presented here.

## 5.3 Evaluation Method of the Decision Algorithm for the Deployment Options

The algorithm presented in the evaluation from chapter 4.5 is designed by determining which of the presented outcomes is the best for the user. The decision process was also presented in article [28]. First of all, I estimate a value created for each choice and a probability for each outcome based on the number of choices that lead to a similar outcome. As determined by applying the ID3 algorithm, the features with the higher information gain are collaboration and programming language. Equation 2 describes how the final value is obtained and Fig. 5.3 and Fig. 5.4 represent the decision tree (the left and right part of the tree) with weights attached to each outcome alongside the probability of each choice.

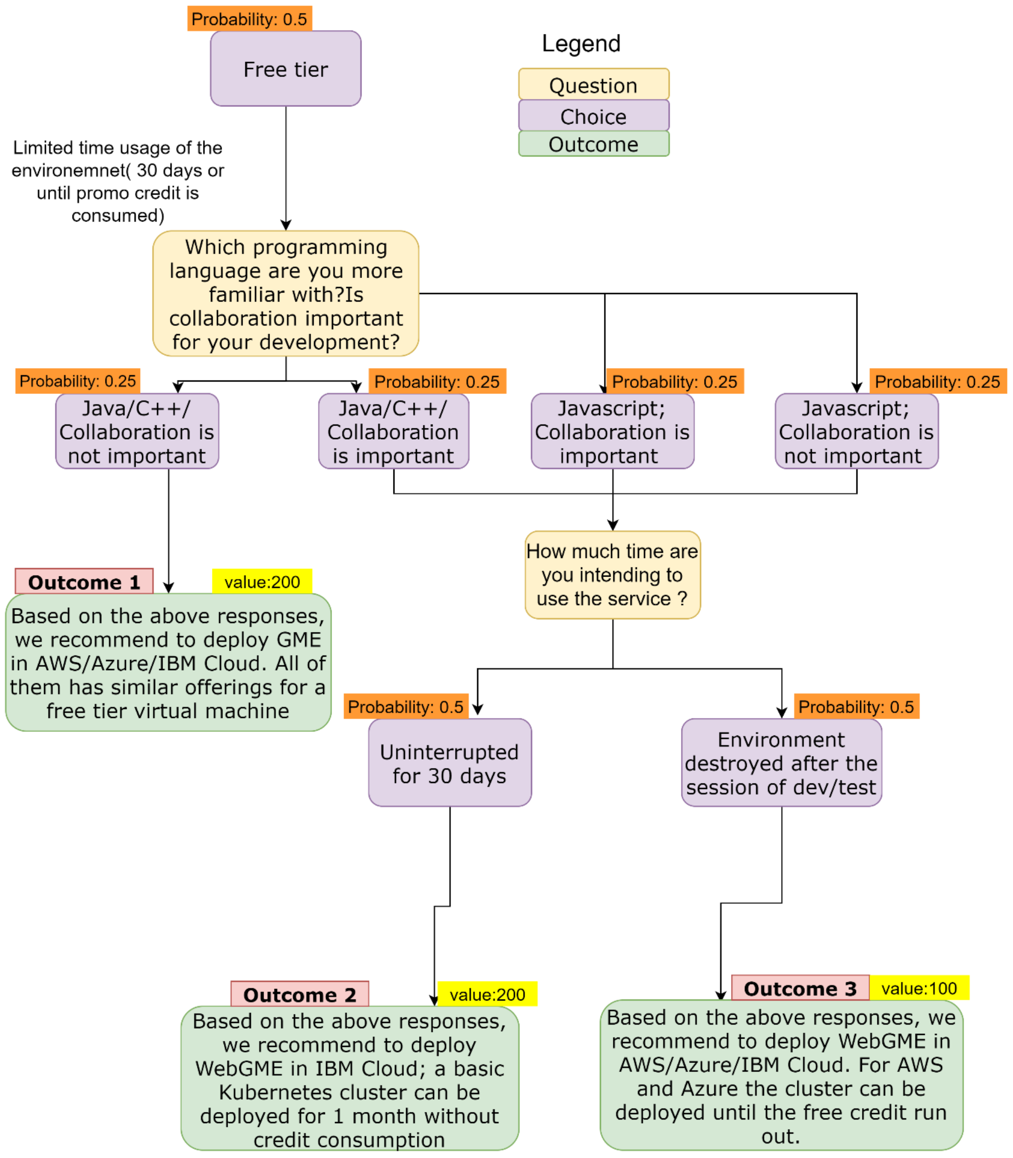
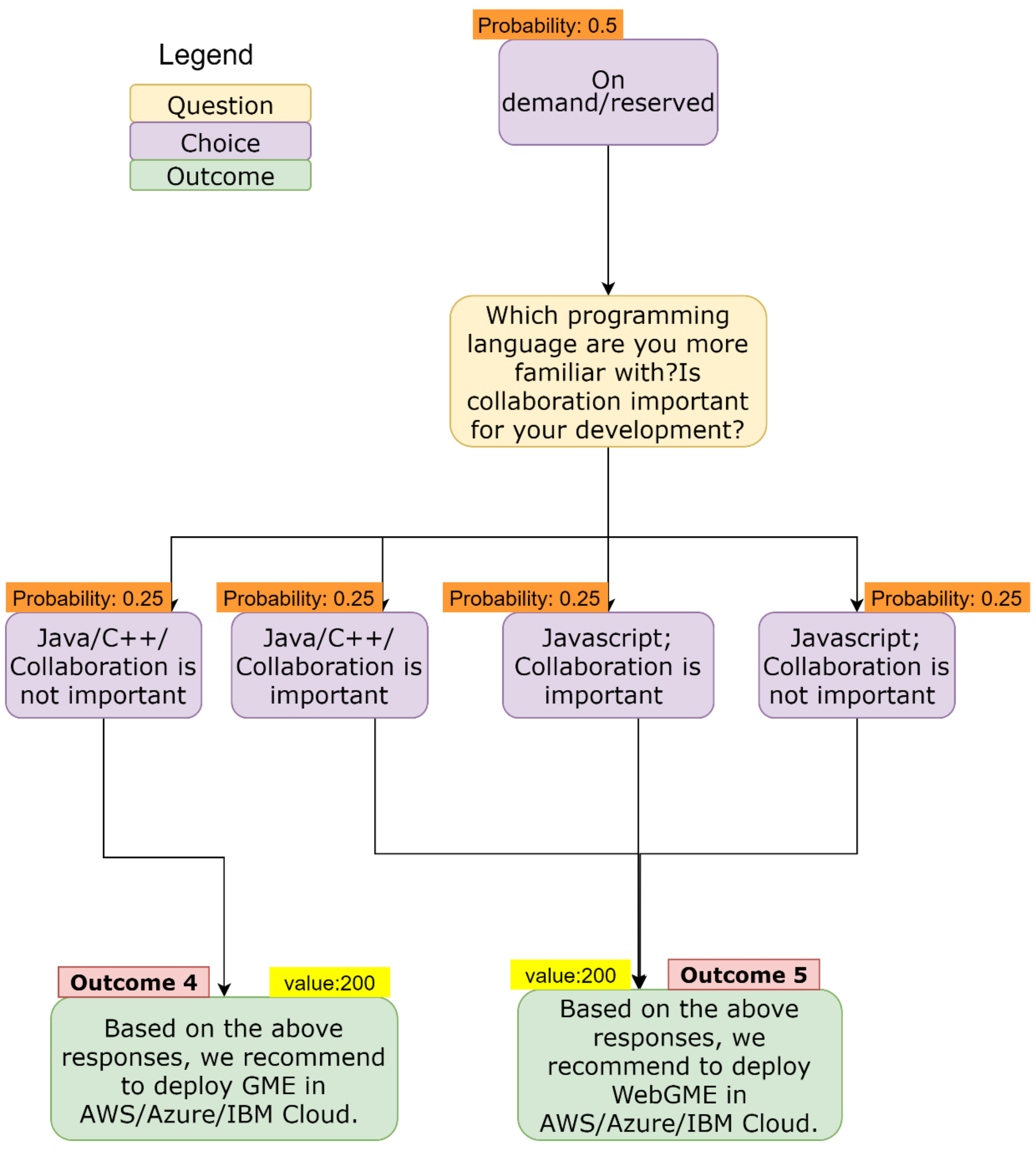

Probability: 0.5
Free tier
Legend
Question
Choice
Outcome
Limited time usage of the environemnet( 30 days or until promo credit is consumed)
Which programming language are you more familiar with?Is collaboration important for your development?
Probability: 0.25
Java/C++/ Collaboration is not important
Probability: 0.25
Java/C++/ Collaboration is important
Probability: 0.25
Javascript; Collaboration is important
Probability: 0.25
Javascript; Collaboration is not important
How much time are you intending to use the service ?
Outcome 1
value:200
Based on the above responses, we recommend to deploy GME in AWS/Azure/IBM Cloud. All of them has similar offerings for a free tier virtual machine
Probability: 0.5
Uninterrupted for 30 days
Probability: 0.5
Environment destroyed after the session of dev/test
Outcome 2
value:200
Based on the above responses, we recommend to deploy WebGME in IBM Cloud; a basic Kubernetes cluster can be deployed for 1 month without credit consumption
Outcome 3
value:100
Based on the above responses, we recommend to deploy WebGME in AWS/Azure/IBM Cloud. For AWS and Azure the cluster can be deployed until the free credit run out.

Fig. 5.3 Tree evaluation – Left part



Legend
Question
Choice
Outcome
Probability: 0.5
On demand/reserved
Which programming language are you more familiar with?Is collaboration important for your development?
Probability: 0.25
Java/C++/ Collaboration is not important
Probability: 0.25
Java/C++/ Collaboration is important
Probability: 0.25
Javascript; Collaboration is important
Probability: 0.25
Javascript; Collaboration is not important
Outcome 4
value:200
Based on the above responses, we recommend to deploy GME in AWS/Azure/IBM Cloud.
value:200
Outcome 5
Based on the above responses, we recommend to deploy WebGME in AWS/Azure/IBM Cloud.

**Fig. 5.4 Tree evaluation – Right part**

$$Final\ value\ = \left(\sum_{n \in N} Probability\ (n)\right)\ \times\ value \qquad (2)$$

where N represents the number of available choices that lead to an outcome on a path. For example, on Outcome 1 (Fig. 5.3), N=2 as two choices leads to the outcome.

For each outcome, Table 5.5 shows the sum of probability, the assigned value, and the final value, calculated with equation 2. The sum of probability is the sum of the probability of all of the choices that lead to that outcome. For example, if a question has four choices, every choice has a 0.25 probability to be chosen randomly. Based on equation 2, one evaluates the final value of each of the outcomes. The outcomes with the highest value are the most valuable and the most suitable for users. The value for each outcome is assigned based on the constraints or lack of constraints that each one has. In this case, only Outcome 3 was assigned a lower value, because the deployment is temporary and there is a penalty for the interruption of service. As a result, the algorithm evaluates which of the outcomes is more valuable for the user. Moreover, the choices that have a higher combined probability have the highest advantage, as the assigned value is the same for most outcomes.

**Table 5.5** Outcome evaluation

| Outcome | Value | ∑Probability | Final Value |
|---|---|---|---|
| **Outcome 1** | 200 | 0.75 | 150 |
| **Outcome 2** | 200 | 1.75 | 350 |
| **Outcome 3** | 100 | 1.75 | 175 |
| **Outcome 4** | 200 | 0.75 | 150 |
| **Outcome 5** | 200 | 1.25 | 250 |

Based on this calculation the algorithm determines that the most valuable outcomes are the ones that are a result of multiple choices, and this means that it suits the needs of multiple users. This is the reason why I consider them the fittest choices.

The algorithm has the role to help the user decide the deployment options. After the deployment options are received by the user, he/she runs the algorithm, and the actual deployment will be made based on the details obtained at the decision step. The decision part has a very important role because it is the first step of the deployment: to determine the options that a user has, based on their needs. Of course, there is room for improvement from the perspective of enhancing the intelligence of the decision part. For example, a user would be able to add her/his deployment options and an improved algorithm, based on machine learning [29], would analyze the data and provide the options to the user.

## 5.4 Cloud Deployment – Evaluation

This evaluation is be made by taking into account the quality model presented in [30]. The metrics presented here are derived from the service and software quality model. The metrics are presented in Table 5.6 alongside a score from 0 to 3 that reflects the level of the attribute. An attribute that is assigned 0 means that it was not implemented and is due for implementation, 1 as a score means that there is room for improvement in that area, 2 means that is acceptable but can still be improved, and 3 means that it is almost perfect.

**Table 5.6** Metrics for SaaS model [30]

| Portability | Efficiency | Reliability | Usability | Maintainability | Security |
|---|---|---|---|---|---|
| **2** | 2 | 1 | 3 | 2 | 1 |

According to [30], portability refers to adaptability and "instalability". I assigned the score of 2 because of the easiness of installation using the deployment platform and the way that can be adapted and deployed in multiple cloud environments. Of course, there is room for improvement here.

Efficiency refers to the utilization of resources and, in this case, I have demonstrated prior that resource consumption remains low during the utilization of the software. This is the same when interpreters are deployed. This can be tuned greatly by imposing memory and CPU limits that can control the resource consumption and limit it to the intended consumption range.

According to [30] reliability refers to the maturity of the software and the recoverability. The way of deployment is not mature enough, but the software itself that is deployed is already tested. Regarding the deployed interpreters, they are not mature enough until they are properly tested and developed for a longer period. Regarding recoverability, there is no backup method for the data implemented. So, the environment can be redeployed in case of failure, but the data is lost and is not recovered. So, these aspects are due for improvement.

Regarding usability, the platform is easy to use, and also the modeling environment. A tutorial section can be an improvement, but right now it is replaced by help sections that are present on the platform.

According to [30] maintainability refers to stability, changeability (refers to code modification), and testability.  The platform can be maintained very easily and regarding code modification, the frontend is modular based on angular components and service, so another feature can be easily implemented, or an older feature can be dropped. The same discussion is on the backend as another controller can be implemented or dropped. Regarding testability, testing tools can be implemented on the Kubernetes cluster to test the environment. This of course can be improved with an admin section with more features that can help the user complete tasks much easier.

Regarding security, a solution that scans vulnerabilities is needed on the Kubernetes cluster and to scan the Docker images. The docker images are already scanned when they are pushed in the IBM Cloud container registry and can be scanned also on Dockerhub if I upgrade to a pro subscription. Also, pod security policy in Kubernetes can be still configured (deprecated in Kubernetes version 1.21; it will be removed in version 1.25 and replaced by another solution) to restrict pod creation and update. Similarly, in article [31], a comparison of cloud edge monitoring solutions is presented and highlights the advantages of such cloud deployment and devices.

## 5.5 Discussion

### 5.5.1 Modeling environment

The microservices cloud infrastructure for WebGME was implemented using Kubernetes and Docker. As a result, WebGME can be privately accessed via a public IP of the Kubernetes worker and privately managed on a public Cloud. For these activities, two deployments and two new services were realized.

A new configuration for the WebGME docker image has been introduced, in order to suit the MongoDB connectivity. WebGME in Cloud is expected to work better than other desktop and web-based solutions (classic WebGME and AToMPM), according to the following criteria: high availability, disaster recovery, private collaboration, resource management, and private modeling environment. However, they are influenced by the IBM Cloud,AWS and Azure offerings and Worker performance. *High availability* means the ability of a system/application to run without failure for a large amount of time. For our work, *disaster recovery* was considered the capacity of the system to be still available if a natural/non-natural disaster occurs. *Private collaboration* and *private modeling environment* refer to the fact that the Git-like features that contain the modeling process are accessible and visible to certain users. *Resource management* is a criterion that is inherited from the Cloud implementation, where resources can be deployed/deleted as needed.

All of the presented criteria represent strengths that the private implementation of WebGME in Cloud has in comparison with similar tools such as classic WebGME and AToMPM. These improvements can help teams in achieving teamwork goals and private environment management alongside cost savings guaranteed by the cloud provider costs plans.

### 5.5.2 Complexity analysis

The results from Section 4.3.1 shows that the cyclomatic complexity for automated deployment is 1, whereas the one for manual deployment is 3. As expected, the automated method has a lower cyclomatic complexity value than the manual one. The implication of this finding is important to my goal, as it is a first step to proving that the development of an automated deployment platform would make a significant difference. A lower complexity value indicates fewer errors, a process that is easier to understand by the user, easier to test, and easier to maintain.

For the two deployment processes (represented in Fig. 4.1 and Fig. 4.2) the Yaqin complexity (*YC*) results presented in Section 4.3.1 also differ. First, we notice that the first process, for the manual deployment, is more complex than the automated deployment due to the difference in the number of branches. However, this is not the only reason; this is one of the strengths of the method proposed in [14], because the Yaqin metric involves many parameters and it is very sensitive in finding the differences between the branches' numbers and types (OR, AND, and XOR). The presented results led us to observe that the overall Yaqin complexity is more than double for the manual deployment process than for the automated one. This result confirms the advantage of a deployment platform and its simplicity of use in comparison with the manual method.

As a supplementary evaluation, let us also consider the complexity of the human tasks (*CH*) estimated in Section 4.3.1. This is more specific to the scope of our thesis and the assignment of the weights is motivated based on the required technical knowledge for every task. The assigned weights are between 1 and 3, with the value 3 for the most complex tasks. In the case of automated deployment, not a single task was given a weight of 3, in contrast with the manual deployment. According to this metric, CH is 20 for the manual deployment, and only 8 for the automated one i.e., almost 2 to 3 times lower. This is a significant difference, though a deployment platform allows an important reduction of manual work, also enlarging the range of people who can deploy the software environment they need to the cloud, without the need for highly specialized technical knowledge in cloud computing. Note that the CH metric defined in Section 4.3.1. is similar to the YC metric, because it also employs cognitive weight to evaluate the difficulty of understanding the software. With the specific metric, I customize this method to the tasks needed to prepare/deploy/use an environment. The chosen criteria may be also applied to other similar processes, which require various kinds of interaction with the user to perform manual tasks. This metric can be used also for other software environments [32] deployed in cloud.

#### *Multicloud deployment complexity*

For the deployment on other cloud platforms, the steps presented in Fig. 4.1 and Fig. 4.2 are similar, with several differences regarding the cloud provider-specific offerings. For example, for the manual deployment, task 3 in Fig. 4.1 should implement specific Azure or AWS connection methods, such as service principal ID and password, or AWS access key ID and secret access key. Task 5 would be replaced with the installation of the respective cloud CLI and task 6 would connect to the chosen cloud platform. For task 8, there are several alternatives up to the preferences of the user. For example, one can use AWS Elastic container registry, Azure container registry, or Docker Hub. Of course, task 12 would also change to require the connection to the targeted cloud platform. For the automated deployment process, task 2 in Fig. 4.2 would require the user to retrieve the specific AWS and Azure credentials; task 5 becomes the selection of the targeted cloud, and task 6 would create the connection to the selected cloud from the multicloud platform. Therefore, the steps are slightly different to suit the cloud provider-specific and not changed entirely. The complexity value remains the same for deployment in other cloud platforms.

### 5.5.3. Performance analysis

For performance testing, I used Prometheus and Sysdig for monitoring the Kubernetes cluster and the deployed containers. The container CPU and memory metrics indicate the usage of an application and its efficiency. If an application is inefficient and the memory increases at an alarming level for the cluster, an OOM (Out of Memory) signal may appear on the respective worker logs; in such a case, the pod that consumes too much memory is evicted and rescheduled. If the graph shows alarming memory and CPU consumption, the memory limits must become higher than the memory request but not as high as the entire worker memory.

After calculating these metrics for our testing environment (see Table 5.1) and based on the cluster memory and CPU usage from Table 5.2 and Table 5.4, I consider that the current sizing is capable to sustain the WebGME modeling environment alongside the monitoring solution. The memory consumption of the Kubernetes pods can be controlled by introducing memory requests and limits to the container definition inside a deployment. Those configuration parameters have the role of preventing the container to use more resources than it is intended. When the limit is reached, the container is restarted [33]. However, the strain may increase if many users actively utilize the same service i.e., access the modeling environment as a service at the same time. In comparison to these findings, obtained with Prometheus & Grafana, the Sysdig results from Table 5.4 are relatively similar. Both tests return results below 100 MB for the consumed memory, for both WebGME and Mongo. Due to the limitations of resources on the Kubernetes cluster used in our testing environment, this cannot be tested at the same time. The difference in the case of CPU is because the measuring unit used in both tests is different. In the Sysdig monitoring dashboard, cores are used as CPU monitoring units, whereas on Grafana the thousandth of a core (m) has been chosen. Hence, the approach used in Grafana is more sensitive to lower CPU consumption in comparison with the core approximation used on the Sysdig dashboard.

The results from Table 5.2 and Table 5.4 prove a lower memory consumption when Prometheus & Grafana are not installed on the cluster, although the CPU consumption is similar. This may be considered an advantage of the Sysdig IBM Cloud monitoring, alongside the fact that it requires no configuration from the user side. The disadvantage is that Sysdig is not present in this form in other cloud platforms, such as Azure and AWS.

### 5.5.4 Automated deployment platform

The research presented showed that an automated deployment would make an important difference in the accessibility of a software environment delivered as a service in the cloud. The resulting platform for automated cloud deployment is written in Node.js for the backend part, and in Angular for the frontend. The platform is designed to deploy multiple kinds of software environments to cloud environments having different providers, with the possibility to add new ones to the list.

The platform is capable to automate the deployment of the WebGME modeling and metamodeling environment in IBM Cloud, AWS, and Azure. Currently, the deployment is realized using technologies such as Docker, Kubernetes, and Helm; they were chosen because they also work well with other important cloud providers, not only with IBM Cloud. To start the deployment, a user is required to register to the platform, create a project, set it as active, and register the cloud connection credentials in the form required for the chosen cloud provider. As future work, the platform will also support the deployment of the Kubernetes service; depending on the chosen cloud provider, the user will have a tutorial on how to deploy a Kubernetes service on the cloud platform.

Thus, the platform for automated software environment deployment in the cloud provides non-technical users with the opportunity of an easier setup and a fast deployment of the needed software environment in the cloud. It is also possible to create multiple environments for testing, development, and production whenever needed. The deployed software environment can also be automatically deleted by the user through the platform. This kind of platform proved to be very useful in times of pandemic as depicted in articles [34] and [35].

# 6. Conclusion

First of all, the thesis presented the state of the art on topics such as modelling as a service, cloud deployment, and performance analysis, which are a natural fit for this kind of research. The goal was to realize a multicloud deployment platform that can help its users, represented by students and researchers, to deploy a software environment in multicloud, even without vast technical knowledge. As a result, the specific objectives were the following: the conception and realization of a new multicloud platform, the evaluation of the deployment methods, and the comparison between the automatic and manual deployment to understand the needs for such a platform from the complexity point of view. Also, the platform grew in the direction of helping the user decide the deployment details and deploying modeling interpreters alongside the modeling environment. As a result, the thesis presented the methods for each of the proposed objectives and the resulting multicloud deployment platform. Regarding complexity, my goal was to present a set of complexity metrics for the computation of the business processes that represent the manual deployment procedure of the modeling environment on a Kubernetes cluster, and the automatic deployment from the multicloud platform. This computation was made to identify an appropriate business case for the development of the multicloud platform. With this perspective in mind, I have studied articles that helped me to choose the metrics that are suited for my case. The articles were very explicit in terms of the formulas used for the metrics and the reason why a metric is used. As a result, I chose to use two complexity metrics: cyclomatic complexity, which is very robust, and Yaqin metric, which is more complex in comparison with the former, since it contains many equations that have the role to increase the precision of the result. Using both of these metrics on the business process of manual and automatic deployment with the platform, I have concluded that automatic deployment is almost three times less complex than manual deployment, and the need for a multicloud platform is indeed welcomed. Finally, I proposed a new metric that is based on the difficulty of the tasks. The weight assignment criteria are based on the amount of knowledge needed for a user/developer to complete a task. For each of the tasks, I have described the reason behind the weight with a real-world example of how to achieve it. The weight given was between 1 and 3, with the 3 being the most complex tasks. In the case of automatic deployment with the platform, not a single task was awarded a weight of 3, in contrast with the manual deployment where the complexity score was more than double.

Another aspect investigated is the performance of the service. I have compared and implemented two monitoring solutions for the environment: Prometheus with Grafana and IBM Cloud monitoring (Sysdig). After configuring the environment, I started by using the PromQL language that queries the Prometheus database for metrics. I first worked directly with the Prometheus web console and next, I created my dashboards in Grafana. I have described in detail the queries used for monitoring the following parameters: CPU and memory usage for Mongo and WebGME containers, namespace status, pod, and deployment status, and finally the resource consumption at the Kubernetes cluster level. Next, I have implemented the IBM Cloud monitoring solution that is based on Sysdig. I have compared both of the solutions side by side and, although the IBM Cloud monitoring solution is easier to set up, it has the disadvantage that it is only deployable on IBM Cloud (for a multicloud solution, I cannot rely on different services on different cloud platforms to have a similar performance) and also it has an expensive monthly subscription that in this case cannot justify the cost, as Prometheus and Grafana can deliver similar results free of charge.

The thesis also proposed a new decision algorithm for choosing the deployment options that the platform offers. This algorithm has the role to help a user decide which is the best-suited modeling environment and cloud to deploy. First of all, I presented a dataset with deployment choices, and I applied the ID3 algorithm to the dataset to create a decision tree. The ID3 algorithm was applied to predict in which conditions to deploy GME and WebGME. For this part, the Cloud provider was omitted. After this experiment, I created a decision algorithm that takes into account all of the deployment details. As a result of the ID3 algorithm, the paths were also guided by the same decision factors: collaboration and programming language alongside new additions such as the cluster availability period. After elaborating on the algorithm, it was also integrated within the multicloud platform.

The thesis also described the scenarios regarding the way this solution can be classified. It was presented why the solution can be considered at the same time as PaaS or SaaS depending on the point of view of the user.  It was also presented the options that were considered for the development and deployment of the model interpreter. The second option that implies the encapsulation of the interpreter in the docker image was chosen due to the advantages that it has. It was also described the new addition to the Dockerfile for this implementation to work. Also, it was described some evaluation methods for the objectives in order to demonstrate their utilities or if the method is eligible to be used in this scenario.

## 6.1 Summary of Original Contributions

A list of the original contributions of this thesis is presented below.

1. State-of-the-art study and lessons learned. A first objective of this study was to identify the required references that will help in achieving the required knowledge in the field and the implementation of the final project. I selected articles that describe subjects such as modeling and cloud, decisional algorithms, cloud metrics and complexity metrics. They were chosen based on the description of theoretical elements and practical examples. The main contribution in this regard is represented by the lessons learned from the presented topics.
2. Deployment of the WebGME modeling environment in IBM Cloud, accompanied by an example of metamodel migration from a source metamodel for sensor networks, initially developed in GME - the on-site variant of this environment. WebGME in cloud is accessible to all the members of a development team at the same time, and the modifications on the same project can be made by them on any device.
3. Comparison of WebGME deployed in cloud with the traditional solutions and other modeling tools. Presentation of the advantages of such deployment and the need for it.
4. Proposed a detailed cloud architecture for the multicloud deployment platform.
5. Elaborating a detailed user story that presents the situation in which a platform like this is needed.

6. Presenting the idea behind the deployment, and envisioning the use case for the platform, from the user perspective.
7. Proposed a decision algorithm that can help the user decide which modeling environment to deploy (WebGME or GME) and in which cloud.
8. Represented detailed process models for automatic cloud deployment and manual cloud deployment.
9. The choice of technologies used for the infrastructure. The choosing of the technologies represents one of the contributions of the Infrastructure/Cloud/Application architect in a commercial project.
10. Multicloud application design from the font-end perspective.
11. Elaborate features for the application that has the role to suit the user's needs.
12. Presented the implementation methods for the backend and frontend.
13. Identify educational settings changes caused by the COVID-19 pandemic.
14. Propose a technical solution to problems like collaboration difficulties, the need for cost savings, and more efficient resource management. The thesis presented a platform that can help to overpass the presented limitations and has the role to deploy modeling tools, such as Web-based Generic Modeling Environment, in the cloud.
15. Developed a platform that promotes the idea of multicloud as the replacement for laboratory environment in the modeling field. The ease of use for the platform was also a focus and a contribution; the platform can be used without technical knowledge as a prerequisite.
16. Provided a robust comparison between a traditional deployment method that requires vast technical knowledge and the use of the newly developed platform.
17. Applied the studied complexity metrics on the deployment business process and compared the results.
18. Proposed a new complexity metric centered on the difficulty of the tasks. The weight assignment criteria are based on the amount of knowledge needed for a user/developer to complete a task and a complete description of all the reasons behind weight assignments.
19. Provided two monitoring solutions for the deployed environment.
20. Comparison of the monitoring tools based on the performance results, cluster load, features, and costs.
21. Detailed analysis of the results for the business process complexity metrics.
22. Detailed analysis of the performance of the monitoring tools and cluster metrics.
23. Provided a detailed discussion of the results in the context of manual and automatic deployment and their performance.

24. Presentation and implementation of a feature for the deployment platform in the form of a decisional algorithm that has the goal to help the user choose the details for the cloud deployment. Creation of a dataset based on the identified user needs. Applied the ID3 algorithm on the dataset that contains a combination of deployment decisions to predict the paths. Considering the resulting trees (after applying the ID3 algorithm), I have created my decision algorithm where I also added the other deployment options. The decision algorithm leads to 5 outcomes.
25. I have evaluated the algorithm by determining which of the outcomes is the best for the user. This evaluation was done based on the choice probability and an outcome value (based on the constraints that each outcome has).
26. Implementation of the decision algorithm in the multicloud platform.
27. Analyzed the solution from the PaaS and SaaS points of view and concluded that can be both at the same time.
28. Developed two solutions and presented the architecture for both of them regarding the strategy of the interpreter deployment alongside the modeling environment.
29. Implemented the solution in the multicloud platform (AWS, Azure, IBM Cloud and OpenShift Online).
30. Evaluated the cloud deployment method based on a quality model for SaaS.

## 6.2 List of Publications

For this thesis, I have used content form articles 1,4,6 and 9 from the below list.

1. **Marian Lacatusu**, A. D. Ionita, Florin Lacatusu and I. Damian, "Decision support for multicloud deployment of a modeling environment," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 247-251, doi: 10.1109/ICCP56966.2022.10053951

2. Florin Lacatusu, A. D. Ionita, **Marian Lacatusu**, I. Damian and D. Saru, "A Comparison of Cloud Edge Monitoring Solutions for a University Building," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 253-257, doi: 10.1109/ICCP56966.2022.10053978

3. Ioan Damian, **Marian Lacatusu**, Florin Lacatusu, Anca Daniela Ionita, Web Services for Guiding Persons with Locomotor Impairments in Public Spaces, *2022 26th International Conference on System Theory, Control and Computing* (ICSTCC), 2022, pp.639-644, IEEE, INSPEC, doi:10.1109/ICSTCC55426.2022.9931861, **WOS:000889980600107**

4. **Lăcătuşu, Marian**; Ionita, A.D.; Anton, F.D.; Lăcătuşu, Florin Analysis of Complexity and Performance for Automated Deployment of a Software Environment into the Cloud. *Appl. Sci.* **2022**, *12*, 4183. https://doi.org/10.3390/app12094183, *Impact factor:* 2.84 - **Q2**, **WOS:000794735600001**

5. Lăcătușu, Florin; Ionita, A.D.; **Lăcătușu, Marian**; Olteanu, A. Performance Evaluation of Information Gathering from Edge Devices in a Complex of Smart Buildings. *Sensors* **2022**, *22*, 1002. https://doi.org/10.3390/s22031002, *Impac factor:* 3.84 – **Q2**, **WOS:000760176500001**

6. **Marian Lacatusu**, Florin Lacatusu, Ioan Damian, Anca Daniela Ionita, Multicloud Deployment to Support Remote Learning, *15th International Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4601-4606, IATED Digital Library, doi: 10.21125/inted.2021.0936

7. Ioan Damian, **Marian Lacatusu**, Anca Daniela Ionita, Florin Lacatusu, Software Services to Support Faculty Management in Times of Pandemic, *15th International Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4634-4639, IATED Digital Library, doi: 10.21125/inted.2021.0943

8. Florin Lacatusu, I. Damian, A. D. Ionita and **Marian Lacatusu**, Smart Building Manager Software in Cloud, *University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 83, no. 4,* 2021., *Impact factor:* 0.37, **WOS:000741473700003**

9. **Marian Lacatusu** and A. D. Ionita, "Metamodeling environment in Cloud," *University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 82, no. 3,* 2020, *Journal:* U.P.B. Sci. Bull Series C, *Impact factor:* 0.37, **WOS:000557847800003**

10. Olteanu, Adriana**; Lacatusu, Marian**; Ionita, Anca Daniela; Lacatusu, Florin, ”Platform for Informal Education and Social Networking to Increase Awareness Regarding Nuclear Vulnerabilities” - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 333-340. **WOS:000467466800045**

11. Adriana Olteanu, Florin Lacatusu, **Marian Lacatusu**, Iulian Craciun, Anca Daniela Ionita, ”Mobile Application for Crisis Situations in a University Campus” - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 280-287. **WOS:000467466800038**

## 6.3 Future Perspectives

From a future perspective, the adoption of the multicloud platform to be used in a university environment is the most natural continuation. The teachers, students and modeling experts can use the platform as it is (from the features perspective).

Regarding technical improvements, the deployment of the Kubernetes service can be added as a feature alongside the deployment of the modeling environments and their interpreters. To obtain infrastructure deployment directly from the platform, the pricing component must be explained to the users, as well as the options that he/she has for deployment. This requires a Kubernetes cluster with one master and one worker, and the deployment times should differ between the three supported cloud providers (IBM Cloud, AWS, Azure) and OpenShift online. With such a feature the user only needs an active cloud subscription as a prerequisite. This can be a good addition and a way to simplify the remaining manual tasks that a user needs to be done.

In the future, other deployment options can be added to the current catalog for the platform to be more versatile. For example, other software environments can be added to the deployment catalog. Thus, the platform for automated software environment deployment in the cloud provides non-technical users with the opportunity of an easier setup and a fast deployment of the needed software environment in the cloud. It is also possible to create multiple environments for testing, development, and production whenever needed. The deployed software environment can also be automatically deleted by the user through the platform. So, the supported cloud providers can be also improved with the addition of support for new providers such as Google Cloud, Alibaba Cloud, and many other examples.

Regarding the decision part, a user would be able to add her/his deployment options and an improved algorithm, based on machine learning [29], would analyze the data and provide the options to the user.

## Selected References

[1]. Tomarchio, O.; Calcaterra, D.; Modica, G.D. Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks, *J Cloud Comp,* **2020,** vol 9, issue 49. https://doi.org/10.1186/s13677-020-00194-7

[2]. Giannakopoulos, I.; Konstantinou, I.; Tsoumakos, D. . Cloud application deployment with transient failure recovery. *J Cloud Comp,* **2018,** vol. 7, issue 11. https://doi.org/10.1186/s13677-018-0112-9

[3]. S. Jung and J.-H. Huh, "An Efficient LMS Platform and Its Test Bed," *Electronics,* pp. 8, 154., 2019.

[4]. W. Ali, "Online and Remote Learning in Higher Education Institutes: A Necessity in light of COVID-19 Pandemic.," *Higher Education Studies*, pp. 10 ,16, 2020.

[5]. GME, Generic Modeling Environemnt, Accessed 25 November, 2020., Retrieved from: http://www.isis.vanderbilt.edu/projects/GME

[6]. A. D. Ionita, F. Anton and A. Olteanu, "A. Sensor Network Modeling as a Service," *In Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018),* pp. 346-353, 2018.

[7]. A. D. Ionita, "Research Experience of Master's Students for Modeling Hazard Management Systems," *Proceedings of INTED2017 Conference 6th-8th March 2017, Valencia, Spain,* pp. 8865-8870, 2017.

[8]. S. Popoola, J. Carver and J. Gray, "Modeling as a Service: A Survey of Existing Tools," *in Proceedings of the Model-Driven Engineering Languages and Systems Conference*, 2017

[9]. WebGME, WebGME Web page, Accessed 30 November, 2020., Retrieved from: https://webgme.org/

[10]. Li, G.; Woo, J.; Lim, S.B. HPC Cloud Architecture to Reduce HPC Workflow Complexity in Containerized Environments. *Appl. Sci.* **2021**, *11*, 923. https://doi.org/10.3390/app11030923

[11]. Shah, S.A.R.; Waqas, A.; Kim, M.-H.; Kim, T.-H.; Yoon, H.; Noh, S.-Y. Benchmarking and Performance Evaluations on Various Configurations of Virtual Machine and Containers for Cloud-Based Scientific Workloads. *Appl. Sci.* **2021**, *11*, 993. https://doi.org/10.3390/app11030993

[12]. Bystrov, O.; Pacevič, R.; Kačeniauskas, A. Performance of Communication- and Computation-Intensive SaaS on the OpenStack Cloud. *Appl. Sci.* **2021**, *11*, 7379. https://doi.org/10.3390/app11167379

[13]. Shepperd, M. A Critique of Cyclomatic Complexity as a Software Metric, Software Engineering Journal, 1988, pp. 30-36

[14]. Yaqin, M. A.; Sarno, R.; Rochimah, S. Measuring Scalable Business Process Model Complexity Based on Basic Control, *International Journal of Intelligent Engineering and Systems 13* (2020): 52-65

[15]. Prometheus Overview. Available online: https://prometheus.io/docs/introduction/overview/ (accessed on 5 April 2021).

[16]. Prometheus configuration Kubernetes. Available online: https://devopscube.com/setup-prometheus-monitoring-on-kubernetes/ (accessed on 1 May 2021).

[17]. M. Lacatusu and A. D. Ionita, "Metamodeling environment in Cloud," University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 82, no. 3, 2020, Journal: U.P.B. Sci. Bull Series C

[18]. M. Kenyeres, and J Kenyeres: "Multi-Sensor data fusion by average consensus algorithm with fully-distributed stopping criterion: comparative study of weight designs.", UPB Scientific Bulletin, Series C: Electrical Engineering and Computer Science, vol. 81, iss. 2, 2019, pp. 27-42

[19]. MetaGMEParadigmImporter: https://github.com/webgme/webgme-engine/tree/master/src/plugin/coreplugins/MetaGMEParadigmImporter, Accessed March 26, 2020

[20]. Lăcătușu, M.; Ionita, A.D.; Anton, F.D.; Lăcătușu, F. Analysis of Complexity and Performance for Automated Deployment of a Software Environment into the Cloud. *Appl. Sci.* **2022**, *12*, 4183. https://doi.org/10.3390/app12094183.

[21]. Lăcătușu, F.; Ionita, A.D.; Lăcătușu, M.; Olteanu, A. Performance Evaluation of Information Gathering from Edge Devices in a Complex of Smart Buildings. *Sensors* **2022**, *22*, 1002. https://doi.org/10.3390/s22031002

[22]. Dynatrace documentation. Available online: https://www.dynatrace.com/platform/ (accessed September 2021)

[23]. AppDynamics documentation. Available online: https://www.appdynamics.com/product/infrastructure-monitoring/cloud-monitoring (accessed September 2021)

[24]. NewRelic documentation. Available Online: https://docs.newrelic.com/docs/new-relic-solutions/get-started/intro-new-relic (accessed September 2021)

[25]. Olteanu, Adriana**;** Lacatusu, Marian; Ionita, Anca Daniela; Lacatusu, Florin, " Platform for Informal Education and Social Networking to Increase Awareness Regarding Nuclear Vulnerabilities" - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 333-340

[26]. Ioan Damian, Marian Lacatusu, Florin Lacatusu, Anca Daniela Ionita, Web Services for Guiding Persons with Locomotor Impairments in Public Spaces, *2022 26th International Conference on System Theory, Control and Computing* (ICSTCC), 2022, pp.639-644.

[27]. Adriana Olteanu, Florin Lacatusu, Marian Lacatusu, Iulian Craciun, Anca Daniela Ionita, "Mobile Application for Crisis Situations in a University Campus" - The

International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 280-287

[28]. M. Lacatusu, A. D. Ionita, F. Lacatusu and I. Damian, "Decision support for multicloud deployment of a modeling environment," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 247-251, doi: 10.1109/ICCP56966.2022.10053951.

[29]. Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN COMPUT. SCI.* **2**, 160 (2021). https://doi.org/10.1007/s42979-021-00592-x

[30]. H. Yang and Y. Kim, " Design and Implementation of Fast Fault Detection in Cloud Infrastructure for Containerized IoT Services," *Sensors*, pp. 20, 4592, 2020.

[31]. F. Lacatusu, A. D. Ionita, M. Lacatusu, I. Damian and D. Saru, "A Comparison of Cloud Edge Monitoring Solutions for a University Building," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 253-257, doi: 10.1109/ICCP56966.2022.10053978.

[32]. F. Lacatusu, I. Damian, A. D. Ionita and M. Lacatusu, Smart Building Manager Software in Cloud, *University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 83, no. 4*, 2021.

[33]. Kubernetes requests and limits. Available online:https://kubernetes.io/docs/concepts/configuration/manage-resources containers/ (accessed on August 16, 2021).

[34]. Marian Lacatusu, Florin Lacatusu, Ioan Damian, Anca Daniela Ionita, Multicloud Deployment to Support Remote Learning, 15th International Technology, Education and Development Conference (INTED 2021) Proceedings (2021), pp. 4601-4606.

[35]. Ioan Damian, Marian Lacatusu, Anca Daniela Ionita, Florin Lacatusu, Software Services to Support Faculty Management in Times of Pandemic, 15th International Technology, Education and Development Conference (INTED 2021) Proceedings (2021), pp. 4634-4639, IATED Digital Library, doi: 10.21125/inted.2021.0943.