University of Crete

Computer Science Department

Modelling and Dynamic Selection of Adaptation Rules for Multi-Cloud Applications

Eleni Politaki

Master's Thesis

Heraklion, April 2019

This work was partially supported by Institute of Computer Science, Foundation of Research and Technology-Hellas.

University of Crete

Computer Science Department

Modelling and Dynamic Selection of Adaptation Rules for Multi-Cloud Applications

Thesis submitted by Eleni Politaki

in partial fulfillment of the requirements for the Masters' of Science degree in Computer Science

Author:

Eleni Politaki, Computer Science Department

Committee approvals:

Dimitris Plexousakis, Thesis Supervisor

Evangelos Markatos, Committee Member

Kostas Magoutis, Committee Member

Departmental approval:

Antonios Argyros

Professor, Director of Graduate Studies

Heraklion, April 2019

Modelling and Dynamic Selection of Adaptation Rules for Multi-Cloud Applications

Eleni Politaki

Master's Thesis

Computer Science Department, University of Crete

Abstract

Nowadays, Cloud computing adoption has increased geometrically and many users prefer this type of technology to deploy and manage their applications. Today, there is a high number of Cloud providers, offering a great variety of Cloud services to meet users' demands. Furthermore, some enterprises prefer to deploy their applications in multiple Clouds in order to benefit from this plethora of offerings. Thus, one important challenge for the Multi-Cloud applications related to the dynamicity and uncertainty that even a single Cloud environment exhibits. As such the increasing complexity makes difficult the delivery of a suitable service level to the customers by the providers. Towards this direction, this thesis introduces two new extensions of the CAMEL modelling language, enabling applications to be adapted across multiple Clouds and different abstraction levels. In addition, an algorithm is proposed for the dynamic selection of the most appropriate adaptation rule for each problematic situation, based on its priority, according to the adaptation history of the application.

In the first part of this thesis, we focus on the proposed extensions of CAMEL, based on an existing cross-level and Multi-Cloud application adaptation architecture. Adaptation actions, rules and strategies are central adaptation-related notions that played a fundamental role in the extensions performed in the CAMEL meta-model. Adaptation rules match an event or event pattern, representing an occurrence of a critical situation, with adaptation workflows, which specify the concrete adaptation actions to be performed for addressing this situation, while adaptation strategies are necessary both for organizing the set of adaptation rules in the context of the same event or event pattern that triggers them, and for representing the application's adaptive behavior.

In the second part of this thesis, we elaborate on the dynamic selection algorithm of the most appropriate adaptation rule, within an adaptation strategy, which is based on its priority value for addressing a certain problematic situation represented by an event or event pattern. This priority value is calculated on the basis of a specific mathematical formula. In the adaptation history of each application recorded particular sensor measurements which are exploited for the computation of the quality attributes that participate in the priority formula to be calculated and come from previous executions of the adaptation rules selected.

Thus, the main contributions of this thesis aim to the better management of the applications that are executed in Multi-Cloud environments by the use of cross-layer adaptation workflows, and the dynamic selection of the most appropriate adaptation rule.

Supervisor: Dimitris Plexousakis

Professor

Μοντελοποίηση και Δυναμική Επιλογή Κανόνων Προσαρμογής για Εφαρμογές Πολλαπλών Υπολογιστικών Νεφών

Ελένη Πολιτάκη

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Περίληψη

Στις μέρες μας η χρήση περιβαλλόντων υπολογιστικού νέφους γίνεται όλο και συχνότερη. Ο αριθμός των χρηστών που προτιμούν αυτό το είδος της τεχνολογίας για να εγκαταστήσουν και να διαχειρίζονται το λογισμικό τους αυξάνονται. Σήμερα υπάρχουν πολλοί πάροχοι υπολογιστικού νέφους που προσφέρουν μια τεράστια ποικιλία υπηρεσιών. Επιπλέον, κάποιοι χρήστες προτιμούν τη χρήση πολλαπλών παρόχων υπολογιστικού νέφους έτσι ώστε να εκμεταλλεύονται στο μεγαλύτερο δυνατό βαθμό τα πλεονεκτήματα που τους προσφέρονται. Για αυτό το λόγο, μια σημαντική πρόκληση είναι η κατάλληλη αντιμετώπιση της αβεβαιότητας και της δυναμικής φύσης αυτού του είδους υπολογιστικών περιβαλλόντων. Για αυτόν τον λόγο δημιουργήθηκαν δύο επεκτάσεις της γλώσσας μοντελοποίησης CAMEL έτσι ώστε να υποστηρίζει την προσαρμογή των εφαρμογών σε περιβάλλοντα πολλαπλών υπολογιστικών νεφών και σε διάφορα αφαιρετικά επίπεδα. Επιπρόσθετα, στη συγκεκριμένη εργασία προτείνεται ένας αλγόριθμος για τη δυναμική επιλογή κατάλληλων κανόνων προσαρμογής, βασιζόμενοι στο ιστορικό ενεργειών προσαρμογής της εφαρμογής.

Στο πρώτο μέρος της εργασίας εστιάζουμε στις προτεινόμενες επεκτάσεις της CAMEL, με βάση μια υπάρχουσα αρχιτεκτονική αντιμετώπισης προβληματικών καταστάσεων, που εστιάζει σε όλα τα επίπεδα μια εφαρμογής που αναπτύσσεται σε πολλαπλά υπολογιστικά νέφη. Οι επεκτάσεις ορίζουν προχωρημένους κανόνες αντιμετώπισης καθώς επίσης και την καταγραφή του ιστορικού εκτέλεσης τους. Κυρίαρχες έννοιες σε αυτές τις επεκτάσεις είναι οι ενέργειες, οι κανόνες και οι στρατηγικές προσαρμογής. Οι κανόνες προσαρμογής αντιστοιχίζουν το γεγονός ή τα γεγονότα που περιγράφουν τις προβληματικές καταστάσεις που χρειάζονται αντιμετώπιση, με τις ροές εργασίας που περιγράφουν τις ενέργειες προσαρμογής. Οι στρατηγικές προσαρμογής οργανώνουν τους αντίστοιχους κανόνες με βάση το γεγονός ή τα γεγονότα που τους ενεργοποιούν, καθώς επίσης αναπαριστούν τη γενικότερη ικανότητα προσαρμογής της συγκεκριμένης εφαρμογής.

Στο δεύτερο μέρος της εργασίας ορίζεται η δυναμική επιλογή του καταλληλότερου κανόνα αντιμετώπισης με βάση την προτεραιότητά του, στα πλαίσια μια στρατηγικής αντιμετώπισης καταστάσεων. Η προτεραιότητα υπολογίζεται με συγκεκριμένο μαθηματικό τύπο, ο οποίος χρησιμοποιεί δεδομένα από προηγούμενες πυροδοτήσεις των κανόνων.

Επομένως, τα βασικά σημεία συνεισφοράς της συγκεκριμένης εργασίας είναι δύο: (i) η καλύτερη διαχείριση της ικανότητας προσαρμογής των εφαρμογών που αναπτύσσονται σε πολλαπλά υπολογιστικά νέφη μέσω κατάλληλων επεκτάσεων που προτείνονται στη γλώσσα μοντελοποίησης CAMEL και (ii) ένας προτεινόμενος αλγόριθμος για τη δυναμική επιλογή του κατάλληλου κανόνα προσαρμογής με βάση την προτεραιότητα του.

Επόπτης Καθηγητής: Δημήτρης Πλεξουσάκης

Καθηγητής

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον επόπτη καθηγητή μου κ. Δημήτρη Πλεξουσάκη για την καθοδήγηση του κατά την διάρκεια των μεταπτυχιακών σπουδών μου και την συμβολή του για την ολοκλήρωση της εργασίας αυτής.

Θα ήθελα επίσης να ευχαριστήσω τους καθηγητές κ. Ευάγγελο Μαρκάτο και κ. Κωνσταντίνο Μαγκούτη που δέχτηκαν με προθυμία να πάρουν μέρος στην τριμελή εξεταστική επιτροπή για την αξιολόγηση της εργασίας.

Πολλές ευχαριστίες θα ήθελα να εκφράσω στον κ. Κυριάκο Κρητικό και στον κ. Χρυσόστομο Ζεγκίνη για τον χρόνο που μου αφιέρωσαν, την πολύτιμη βοήθεια τους και την καθοδήγηση τους σε όλη την διάρκεια εκπόνησης της εργασίας.

Τέλος θα ήθελα να ευχαριστήσω όλους τους κοντινούς μου ανθρώπους που είναι πάντα δίπλα μου και με υποστηρίζουν. Μεγάλο ευχαριστώ στον φίλο μου, Παναγιώτη καθώς επίσης και στην μητέρα μου, Ευαγγελία, και στην αδερφή μου, Ιωάννα Μαρία, για την υποστήριξη και την αγάπη τους. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον πατέρα μου, Ευάγγελο, που φρόντισε να έχω την δυνατότητα να ολοκληρώσω τις σπουδές μου.

στους γονείς μου

Contents

List of Tables

List of Figures

Chapter 1

1. Introduction

In this chapter we analyze some basic concepts required in the context of this work. Firstly, Section 1.1 introduces some basic knowledge related to Cloud Computing. Section 1.2 elaborates on Multi-Cloud application management, while Section 1.3 provides details for a previous proposed adaptation framework, Finally, the Section 1.4 provides an outline of this thesis.

1.1 Cloud Computing

One of the most well known developing trends in recent years is Cloud computing. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. The use of Cloud computing implies a set of features and a number of issues and control worries [2].

Cloud computing chiefly offers lower implementation and maintenance costs by reducing user need for purchasing and supporting as much hardware. Another benefit of Cloud computing is flexibility because it implies both the high-performance of resources and increased reliability and availability of applications. There are some concerns mainly related to the security of data and the management of operations and services in order to support a self-service functionality in the Cloud. Nowadays, although there has been much technological progress in Cloud computing development, many research issues remain unsolved [3].

1.1.1 Service and Deployment Models

By following SOA (Service Oriented Architecture) every offering in the Cloud is characterized as a service. According to NIST (National Institute of Standards and Technology) [1], Cloud computing has three standard service and four deployment models (Figure 2.1).

The first service model is the Software as a Service (SaaS). This service model relates to the offering of existing software services like software applications and databases. SaaS is convenient, easy to access, and accessible from anywhere. It is scalable and secure for the users. The second service model is Platform as a Service (PaaS). This service model offers a highly developed environment, suitable for developers. They pay for what they need and they can develop their services quickly because they have the ability to use other pre-installed services. The last service model is Infrastructure as a Service (IaaS), which relates to the supply of computing resources, installed in data centers like servers, networking, storage etc, as a service. It can be useful for the companies who need to save the costs of buying and maintaining their own hardware. With the above service models, different Cloud environments could be created. A Cloud environment is an aggregate of the above service models. Different Cloud environments can be more fruitful for different cases, based on specific needs.

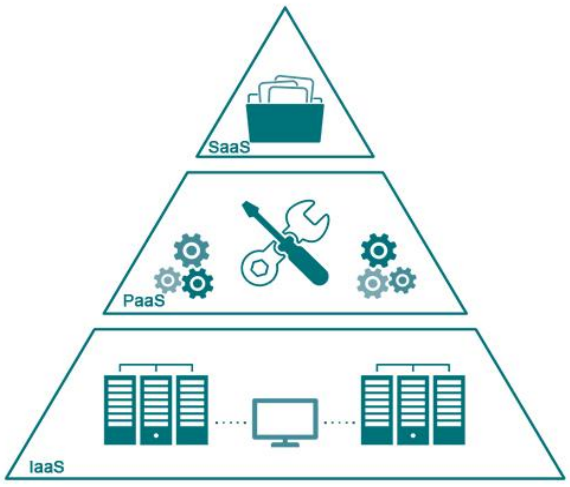


Figure 1. 1 : Cloud Service Models according NIST

As far as NIST is concerned there are also four deployment models. Every Cloud has some specific characteristics concerning the way in which services can be utilized. The first one is the public Cloud, which is owned by the Cloud provider and multiple customers. It is suitable for minimizing the operational and maintenance costs. The Cloud provider is responsible for the management and the maintenance of the Cloud. The disadvantage is that companies are not able to have their own security and management. A private Cloud is provided only for one company and can be owned by either the company, a third party vendor or a combination of the two. This deployment model is suitable for organizations which have their own infrastructure with highly sensitive information. A community Cloud is used by a specific community of users and can be owned either by a third party provider or by the set of organizations which use it. This type of Cloud is suitable for organizations with common needs. A Hybrid Cloud consists of two or more Cloud environments; the most common of which are either public or private. If there is sensitive data, the most suitable Cloud deployment model is the private. However, a company could use a private Cloud in this case and a public Cloud for all the other needs of the company. This type of Cloud is becoming very popular and the reason is that an organization can use a Hybrid Cloud in order to have control over the security of their private Cloud, in conjunction with the benefits of a public Cloud. Hybrid Clouds are thus the most economical of cloud deployment models; their only disadvantage being that it is difficult to manage their resources.

Apart from these four categories of Cloud Deployment Model which follow the NIST definition, there are other more specific deployment models such as the cross Cloud and Multi -Cloud. In the former we have a deployment across different Clouds, so that services from these Clouds can be utilized within the same application deployment, while in the latter, that is the Multi-Cloud Deployment Model, the use of services from multiple Clouds is maintained and can be performed at a single time.

1.2 Multi-Cloud Applications Management Challenges

One of the most significant parts from the customer side is the ability to choose the most suitable service, independent of the Cloud provider. The advantage that this choice gives, is the needed flexibility and the right combination of the different Cloud providers that could be the key towards greater efficiency and reduced costs. However this is a way for avoiding vendor lock-in issues. With Multi -Clouds customers can organize the data exported by the Providers and by processing and analyzing, create Cloud Computing environments containing manageable Clouds.

Due to the benefits of Multi-Clouds there is a move towards adopting their use. One of the most interesting challenges for the Multi-Cloud applications related to the dynamicity and uncertainty that even a Single Cloud environment exhibits. As such, the increasing complexity makes difficult the delivery of a suitable service level to the customers by the providers. The efficient recognition of user needs and an automated prediction can be made as to the optimal provision of services most suited to these needs.

Many challenges like this could be overcome by the use of adaptation workflows with actions on the running services. These actions are needed for the proper organization of all the required tasks to be done and to lead to the optimum result for the last users. In the case of Single Cloud the management of these actions is simpler than in the case of Multi-Cloud. Nevertheless, the Multi-Cloud environment is a superset of Single-Cloud environments. Adaptation workflows in Multi-Clouds have great importance and the reason is that it is difficult to synchronize the set of adaptation actions of the nested environments. Therefore, the same adaptation workflows could be implemented in Single-Cloud and Multi-Clouds but in the last case there is an increasing complexity. The goal of the adaptation actions is to predict the most appropriate task for the customer taking into account the historical records of the services used in the past and the current needs of the users.

So, in order to respond to this challenge, the Cloud deployment model should collect the metrics of all running applications, and through the analysis of these metrics should react to cases where adaptations to these applications are necessary. To facilitate this analysis, the execution histories should be recorded. SLAs also play an instrumental role in this monitoring and adaptation process. An SLA sets the expectations between the parties. The SLA has a certain service life cycle [4] and contains analysis of SLO conditions over the QoS metrics (Figure 2.2). The monitoring of the QoS play a fundamental role in the adaptation process [5]. The set of SLOs comprises a certain service level.

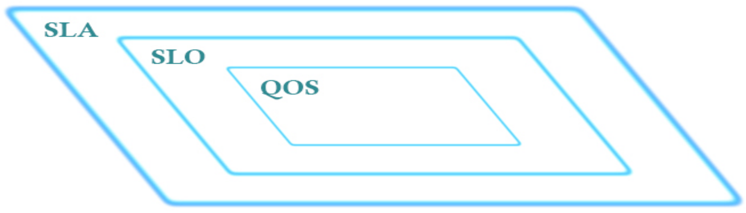


Figure 1. 2 : SLA structure

1.3 Cross-layer adaptation framework in Multi- Cloud

For the cross-layer adaptation of a service based application within the Multi-Cloud architecture a cross layer adaptation framework was proposed based on earlier work. As cross-layer we distinguish between 5 levels: IaaS, PaaS, SaaS, WfaaS (workflow as a service) and BPaaS (business process as a service) (Figure 1. 3) . The two last levels are an extension of the basic levels that NIST has already suggested. The WfaaS is represented with workflow tasks and at this layer the control flow of the corresponding adaptation tasks is organized. The BPaaS is introduced in the BPaaS Adaptation Framework [6] that has been made in order to organize the business processes (BPs) related with the provisioning of the needed adaptation actions on different abstraction levels of the Cloud. This framework can support the dynamic generation of adaptation workflows as well as the recording of the adaptation history. The BPaaS is a cross-layer adaptation framework which can be used in multi - layer Clouds. This framework correlates with the CAMEL meta-modelling language. In this work, two new extensions of the chief language are performed in order to specify all the needed operations and to suggest the most appropriate adaptation plan for each (critical) event. In the proposed extensions, the adaptation rules are responsible for the mapping between events and adaptation plans/ workflows. An adaptation plan is an adaptation workflow comprising the set of adaptation actions [7]. Event or event patterns [8] trigger an adaptation plan, i.e the execution of its adaptation workflow. Finally, the adaptation strategies organize the adaptation rules in the context of the same triggering event or event patterns. Finally, the system by itself selects the adaptation rule with the highest priority. This made by a dynamic selection algorithm of adaptation rules which use the records of the adaptation histories in order to compute the priority.

1.4 Thesis Outline

In the first part of this thesis we present a cross-layer workflow adaptation approach in a Multi-Cloud application system. To organize the corresponding adaptation actions we introduce adaptation rules activated by triggering events. One event can be mapped to multiple adaptation rules, each associated with a different adaptation workflow. We organized the adaptation rules with the same triggering event in adaptation strategies. Through a mathematical formula we give a priority value for each adaptation rule in the context of the triggering event. Towards the practical context of the thesis, we extended two parts of the ecore model of the language CAMEL, the metric and the adaptation/scalability; and we subsequently upgraded a third part - deployment one encompasses an introduction to the basic components needed in order to introduce the subsequent thesis components. In the second chapter, a use case is presented whose aim is to provide an application of this work in order to validate it. In the third chapter the analysis of the CAMEL extensions are presented. The fourth chapter contains the analysis of the mathematical formula for the priority of the adaptation rules in the context of the triggering event. In the fifth chapter we analyse the related work, and in the final chapter we refer to the conclusions and to future work in order to provide future work directions.

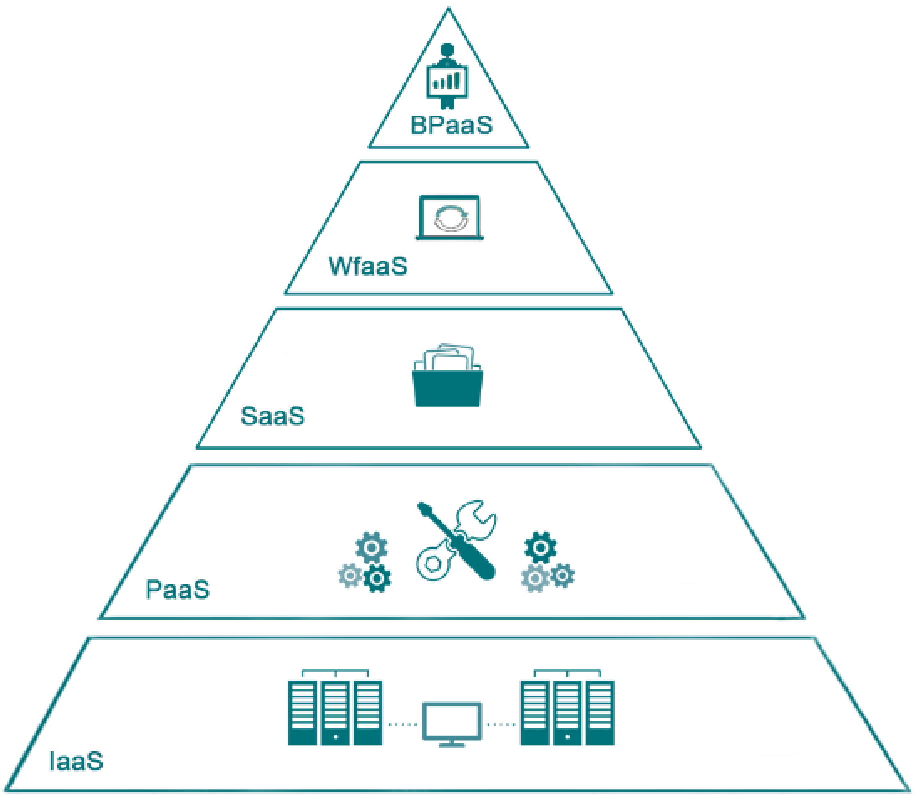


Figure 1. 3 : Extended Cloud Service Models

Chapter 2

2. Traffic Management Use Case

In this Chapter we introduce a use case paradigm in order to demonstrate the main thesis contributions. Firstly, we report the application specifications (Section 2.1) and then analyze the workflow structure (Section 2.2), the application requirements (Section 2.3) and its component parts (Section 2.4). Then there is the definition of all instances for our example (Section 2.5); an exemplary example of the application's operation (Section 2.6), and finally we have an example with indicative tables with SLO violations responsible for the triggering of adaptation rules.

2.1 Application Specification

Our example describes a Cloud application related to the management of various traffic-related events in the city of Heraklion. This application maps to a workflow which attempts to regulate the traffic under both normal and emergency situations in the city of Heraklion. This application follows a service-oriented architecture while it involves three main stakeholders:

Traffic Manager: controls an area in the Heraklion city and adjusts the traffic according to the evaluation of traffic and environmental conditions.

Rescue Forces: rescue forces, i.e., the traffic police and the fire brigade for the immediate response to critical situations.

Medical Forces: they are responsible for carrying out manual activities like First-aid. In fact, they follow a certain plan which is derived by the application.

We assume that the aforementioned stakeholders are represented by services which are mapped with Application tasks. The Traffic Manager is represented as a more generic and complex workflow of tasks as its workflow is consisted of more than one tasks. The services called by Traffic Manager tasks are the Monitoring, Assessment and Device Configuration Service. In the respected workflow the Traffic Manager has sequential control flow. The mapping between the services and the workflow tasks follows (Table 2.1):

Service	Task
Rescue Forces	Trf
Medical Forces	Tmf
Traffic Manager	(Tm , Ta , Td)

Table 2. 1 : Tasks and stakeholders correlations

Each of these services needs to take particular actions during the application operation. The Traffic Manager workflow is comprised of three main tasks (Tm,Ta,Td) which are realized by the following three services, respectively:

- **Monitoring Service** (called from Task Tm)

This service aims at collecting traffic information for the area of Heraklion. The data collected is forwarded to the Assessment Service.

- **Assessment Service** (called from Task Ta)

This service first structures and aggregates the data appropriately and then performs the respective analysis over them in order to finally produce the appropriate traffic management plan. This plan is forwarded to the Device Configuration Service.

- **Device Configuration Service** (called from Task Td)

This service can automatically adjust traffic lights based on the plan as well as demand the execution of certain activities by corresponding stakeholders that were mentioned above.

2.2 Workflow Structure

In general, services have input and output parameters while they can also interact with other services. The result of running a task depends on the service that implements the respective functionality. The main input parameter of the Traffic Management Application is the related area where traffic should be regulated. We have a new instance of the traffic management workflow; and thus of the respective application for each different area in Heraklion. The main input parameter of the workflow is passed to the first service that needs to be executed, i.e., the Monitoring Service. The output of the Monitoring Service is monitoring data which are forwarded to the Assessment Service. The Assessment Service receives this data and produces as output the traffic management plan to be enacted by the Device Configuration Service. The latter obtains this plan and produces as output the corresponding actions to be performed by the respective stakeholders. The following figure (Figure 2.1) depicts the workflow of tasks and how it is mapped to respective services.

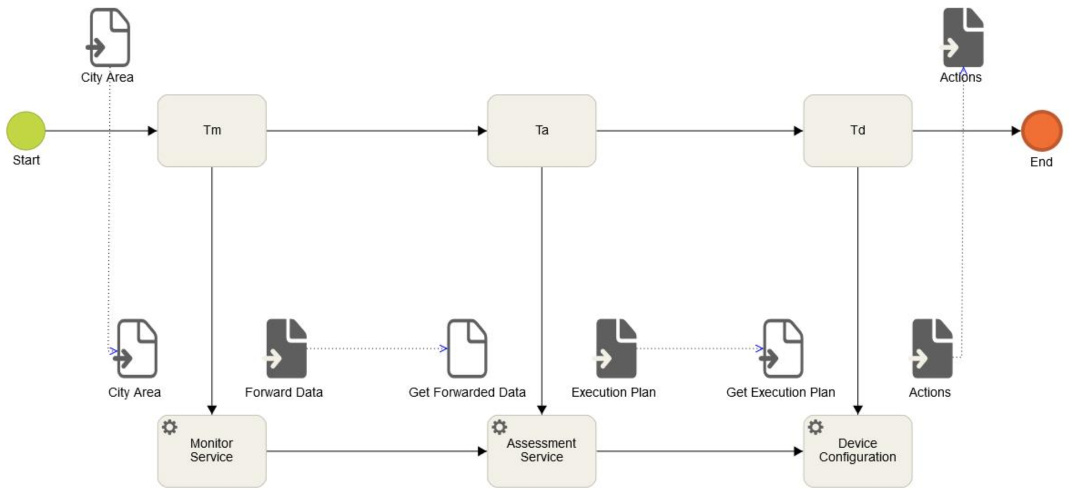


Figure 2. 1 : Tasks and Services Correlations

As this application runs, a problem may arise such as the destruction of a sensor. Then an Adaptation Workflow must act to ensure that the application works properly.

2.3 Service Requirements

The respective requirements need to be fulfilled by the corresponding services that realize mapping tasks. The Assessment Service has high requirements on both computational power and service availability for the following reasons: it is responsible for filtering data according to the most valuable information, focuses on analyzing and creating a response plan for a critical situation. The Assessment service requires a certain storage capacity as it needs to store and maintain a great set of data. The Monitoring Service also needs high storage capacity to cover the aggregation of the sensors. Finally, the Monitoring and Device Configuration Services must be stored in close geographic areas with respect to the sensor infrastructure as these two services interact with the sensors (e.g information - gathering sensors) and actuator (e.g sensors to inform drivers). An indicative table with the requirements of the three services of the Traffic Management workflow is supplied below (Table 2.2):

Involved Tasks	**Requirement**
Ta	High computational power
Ta	High availability
Ta, Tm	High storage capacity
Tm and Td	Closeness with sensor infrastructure

Table 2. 2 : Indicative requirements of the services

As far as the hosting of the components of the corresponding services is concerned, we assume that the Monitoring and Device Configuration Services are deployed on a private/municipal Cloud located at the city of Heraklion. The Monitoring and Device Configuration Services are hosted on a “medium” public VM (4GB RAM, 4-core CPU and 40GB disk). The Assessment Service is deployed on a "high" VM (8GB RAM, 8-core CPU and 80GB disk), through a PaaS provider, due to its higher computational and storage requirements.

2.4 Application Components

The following figure (Figure 2.2) depicts the overall system, including the Cloud layers (IaaS, PaaS, SaaS, WfaaS) involving the respective components situated on these layers and their dependencies. At the WfaaS layer, we can see the set of the application workflow tasks and at the SaaS layer we have the respective services. The PaaS layer involves the PaaS services of our example. Finally, at the IaaS layer we have the required infrastructure for the deployment of the application. A PaaS provider could have its own infrastructure but could also rely on the infrastructure of another provider. In our use case we have a PaaS provider with its own infrastructure. This is more clear at the figure (Figure 2.3).

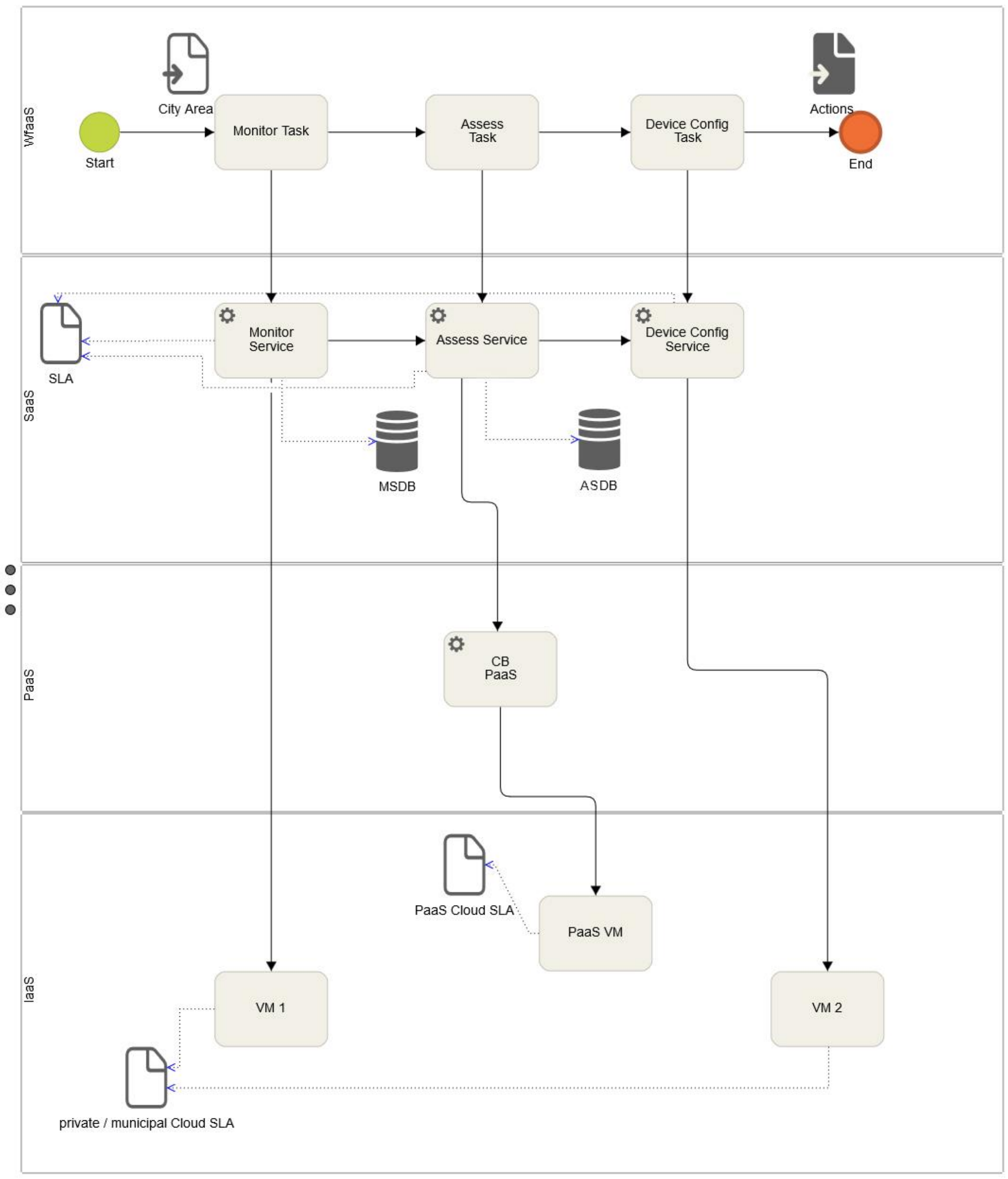


Figure 2. 2 : Application Components

Each infrastructure provides the required software for the deployed services, as well as a servlet container. A servlet container is a featured service that acts as a server to service components and can be offered by a PaaS provider. For the Traffic Management Application, the CB provider offers the servlet container service as an add-on. Only the CB provider has access to the Assessment service VM, so the adaptation system cannot manage it. Each VM uses an Apache Tomcat Application service for hosting the applications. The Drools Rule Engine is required by the Assessment Service and by the Device Configuration Service. This Rule Engine is used by the Assessment Service to decide on the level of tasks that need to be performed in the current situation, while the Device Configuration Service has to execute the plan given by the Assessment Service.

This can be a high-level plan that needs to be concretized by the Drools engine. Apache Tomcat Application and Drools Engine are nested components in the services. The ASDB database is used by Assessment Service for storing the monitored events and extracting aggregated values. Also the database MSDB is used by the Monitoring Service for the aggregation of the data. All the above are parts of the running applications.

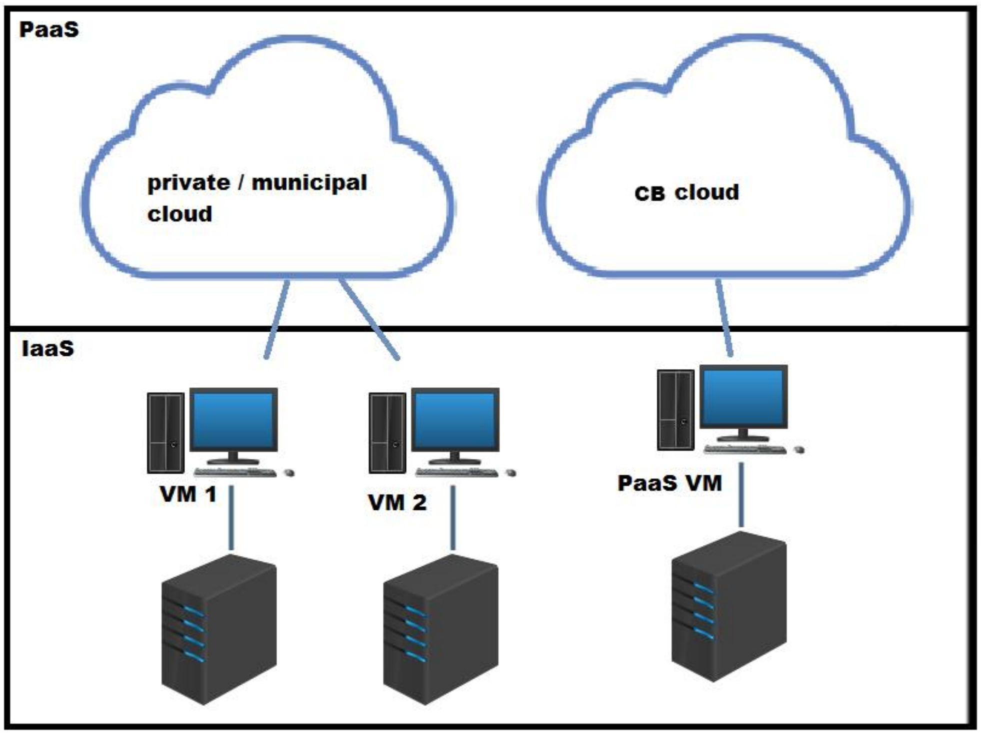


Figure 2. 3 : Clouds Infrastructure

2.5 Application Instances

In the Traffic Management use case, we could have multiple instances from the same type workflow. Thus, we assume that we have two different instances of a workflow which have different labels on their respective elements (Figure 2.4). The first concerns the zone_A, which includes the center of Heraklion, and the second the zone_B, includes its suburbs.

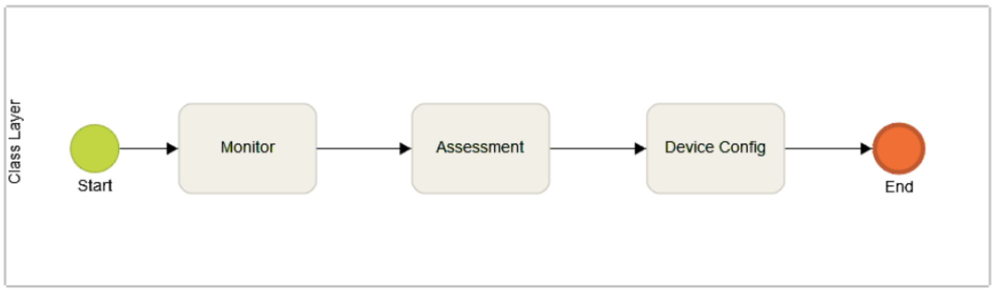


Figure 2. 4 : Class Layer representation

For an application we have instances of tasks and workflows. In fact, if we consider workflow engines, usually we talk about different deployments of the same workflow. For each deployment, one or more instances of the workflow can be generated and executed (Figure 2.5 **).**

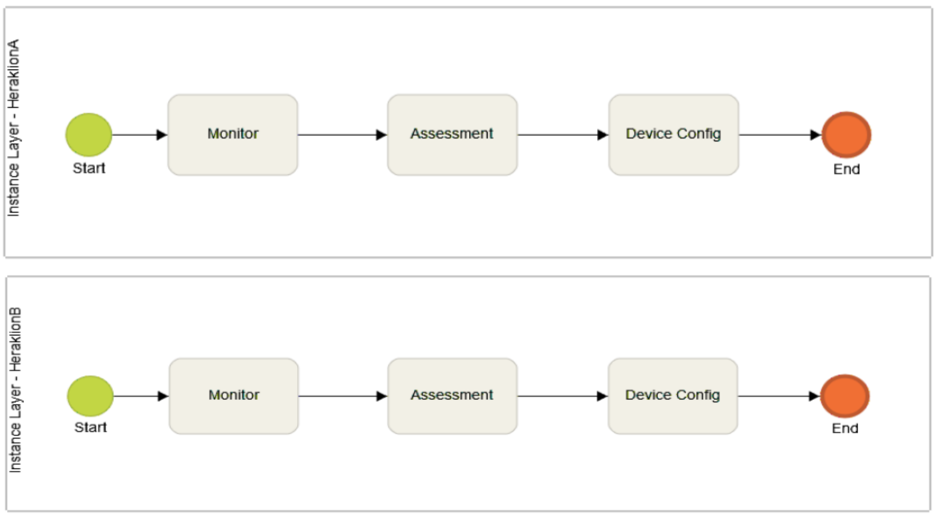


Figure 2 5 : Instance Layer Representation

Based on the logic of multi instances running in parallel, an example of a running application will be detailed below.

2.6 Running Example Application

In order to demonstrate the main functionalities of the running example application, in this section we elaborate more on two different cases (instances); (i) a normal case and (ii) an emergency case. We will then describe in detail the flow of operations in these two cases.

<u>1st instance - Emergency Case</u>

In the case of an accident, the Monitoring Service immediately informs the Assessment Service about the accident severity; the latter then assesses this incident and comes up with the actions to be performed. Then, the Assessment Service informs the Device Configuration Service about the adaptation plan with the actions to be performed and their order (Figure 2.6) depicts such an emergency case.

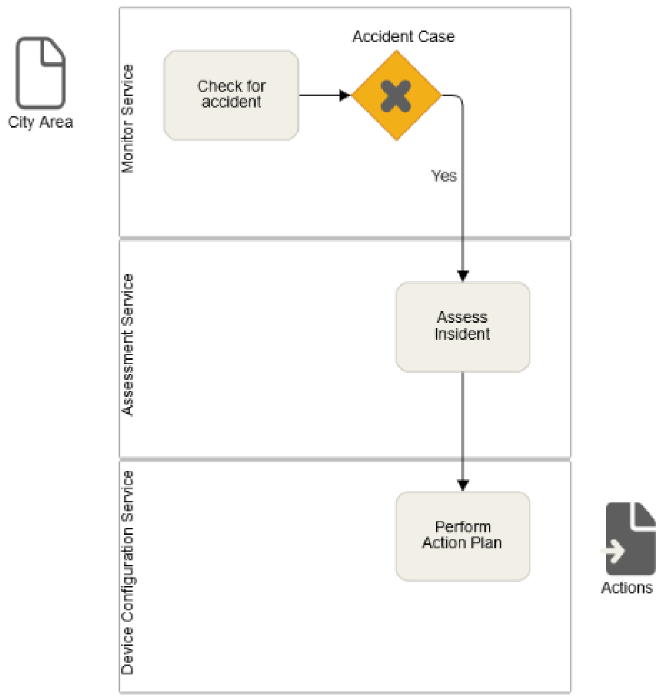


Figure 2. 6 :Emergence case operations of services

<u>2nd instance - Normal Case</u>

In a normal case, the Monitoring Service collects the environmental data, such as temperature, humidity and others; checks calendar data related to some special days within the year, such as National Holidays, and also measures the traffic on the roads of the respective area of the corresponding instance taking into consideration the traffic flow density (i.e., the number of cars passing from a specific point in a 24-hour base). These functionalities are provided by separate components of the application. This data is collected by specialized sensors that have been installed at the managed area. The Monitoring Service passes the data to the Assessment Service, which, in turn, processes and analyzes this big amount of data. The outcome of this analysis is a traffic management plan. After that, the Device Configuration Service is responsible for performing the traffic device reconfiguration in order to decongest the area’s traffic in the places where traffic congestion has been identified. We can see at the next figure (Figure 2.7) a part of a normal case execution of two separate operations of the traffic manager application components.

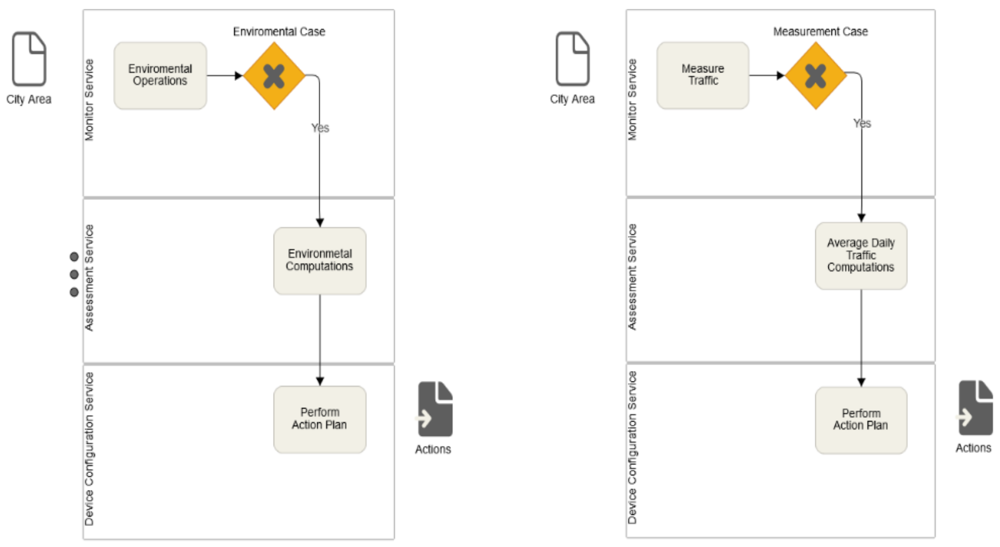


Figure 2. 7 : Normal cases operations of services

In the figure (Figure 2.8**)** we have a representation of the Traffic Manager Application Workflow. We discern the Monitoring, Assessment and Device Configuration Services. Each of these services exposes a set of methods which could map to the tasks of a workflow. So, each service does not correspond to just a single method. In the respective workflows we have multiple tasks. Two or more tasks could be realized through a service. Thus in the workflow of the figure we performs the orchestration of some operations which runs separately and in parallel for every instance of the application. The operations based on the referred operations of the current use case.

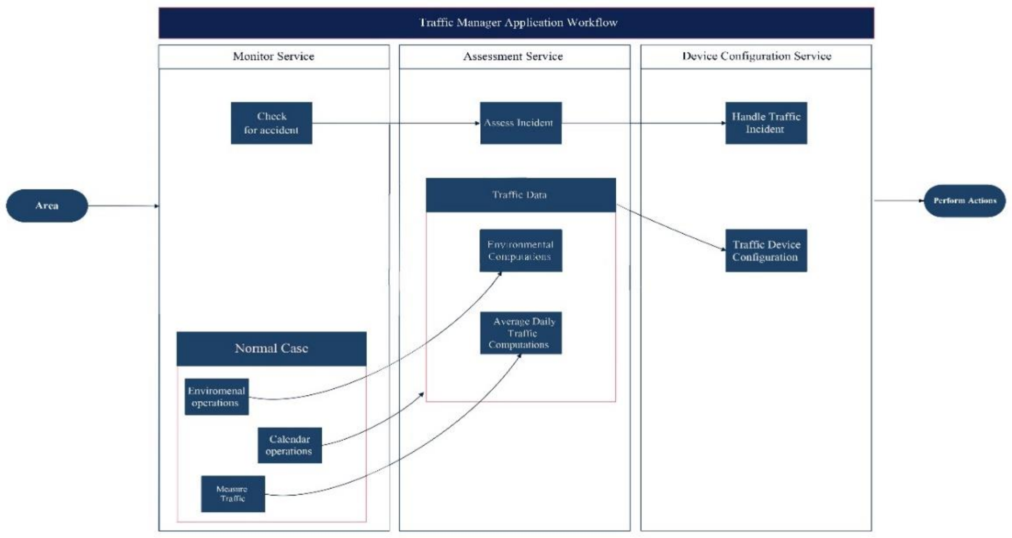


Figure 2. 8 : General Cases Plan

2.7 SLO requirements

As far as it concerns the Traffic Manager Application, we supply below an indicative SLO table (Table 2.3, Table 2.4, Table 2.5) for each of the utilized services according to the emergency and normal cases of execution of our use case which is detailed in Section 2.6. The SLOs are mapped within SLAs to penalties in case that they are violated. The assessment of SLOs relies on the evaluation of metric conditions. The SLO violations trigger events which can cause the execution of adaptation rules for every separate instance of the traffic management application. In the following example the metrics conditions based on the execution time and the availability of the application.

Monitoring Service

Execution time SLO	The service execution time should not exceed 11 seconds in the emergency case, and 20 seconds in the normal case.
Availability SLO	The availability of the Monitoring Service should be greater than 99,99% in emergency case and 99% in normal case.

Table 2. 3 : SLO violations of Monitoring Service

Assessment Service

Execution time SLO	The assessment of the emergency case should be completed within 20 seconds and within 10 seconds in normal cases.
Availability SLO	The availability of the Assessment Service should be greater than 99,99% in critical cases and 99% in normal cases.

Table 2. 4 : SLO violations of Assessment Service

Device Configuration Service

Execution time SLO	The handling of the emergency cases should be completed within 30 minutes, as it requires manual activities and within 10 seconds in normal cases.
Availability SLO	The availability of the Device Configuration service should be greater than 99,999% in critical cases and 99,9% in normal cases.

Table 2. 5 : SLO violations of Device Configuration Service

Chapter 3

3. Camel Modelling Language Extensions

A family of DSLs called CAMEL[9], Cloud Application Modelling and Execution Language[1], was initially developed in the PaaSage[2] project [10] with the main goal of covering all necessary aspects related to the deployment and adaptive provisioning of Multi-Cloud applications. This family includes, among others, the Cloud Modelling Language (CLOUDML) for modelling the deployment topology of an application and the Scalability Rules Language (SRL) for specifying event patterns, scaling actions and scalability rules [11]. The Eclipse Modelling Framework (EMF) has been used in order to integrate all these DSLs into a coherent whole. In particular, an Ecore model (i.e., a meta-model) has been created to cover the abstract syntax of CAMEL. So, EMF provides the right tools for the generation of language abstract syntax via the use of meta-models. EMF also allows the generation of a Java class hierarchy representation of each meta-model based on its definition. In the context of this work we have extended the CAMEL Ecore meta-model in order to incorporate new classes and we have also updated CAMEL's deployment meta-model in order to incorporate all the changes which had to be made so that we could assign new classes to the remaining meta-models and to ensure that the deployment meta-models were compatible with these changes. We performed two extensions to CAMEL's sub DSL's. The first was in the scalability meta-model, which was renamed adaptation meta-model where we covered the adaptation aspect which maps to the modelling of adaptation tasks, rules and strategies. This extension was created for the modelling of advanced adaptation rules which included various kinds of adaptation tasks at different levels of abstraction. The second extension was in the execution meta-model where we modelled the adaptation histories. With the record of the historical information we can check the application's performance. Also it could be used to reason over the best deployments of an application's or its components [12]. This extension played a basic role in the dynamic selection algorithm of the most appropriate adaptation rule (Chapter 4). In the following parts, apart from the deployment meta-model update (Section 3.1), we will also indicate the extensions to the CAMEL scalability (Section 3.2) and execution meta-model (Section 3.3). Finally, we follow a number of use cases (Section 3.4) for the validation of this work.

[1] http://camel-dsl.org/

[2] https://paasage.ercim.eu/

3.1 CAMEL’s deployment meta model

An adaptation workflow is comprised of adaptation tasks that are executed in a certain order. The deployment meta-model covers the topology of an application in terms of its components. Thus, adaptation tasks map to application components. The main reason for updating the deployment meta-model was to cover all possible types of components which could be utilized in the definition of adaptation tasks. The classes which represent the internal and the external components were deleted and new subclasses inserted. The internal components were components owned by the system, and the external components were those owned by external systems. In the same way, internal and external service components were also deleted. These deletions were performed because through this update all such components were covered by the new subclasses of the Component class which were designed to implement more specific components over all the service model layers of the Cloud. So, in the Component class we introduced the subclass of a Microservice, i.e., of the smallest software unit that can work autonomously as an autonomous application software component and provide a certain functionality to other software components or applications. The PaaS can include the environment for a component plus the add-ons which could take the form of Microservices. Here the role of Microservices is to represent the add ons offered by the PaaS which are autonomous, pre-installed software components needed for the execution of the software components hosted in a PaaS environment. Finally, each PaaS relates to the requirements imposed over the respective environment in which the corresponding application component will be hosted. Another subclass of Component class is the VM class. This concept plays the role of a placeholder indicating the place where a certain component will be positioned. It represents an IaaS service in which an application component could be hosted. This placeholder also involves the set of resource requirements that need to be satisfied by this place / hosting component. Finally, there is the SaaS class, for the representation of an external component which provides a certain functionality over the internet. A SaaS component includes a unique registry ID and an aggregate of tasks which map to an application workflow. We can see the graphical representation of the Component and its subclasses below (Figure 3.1).

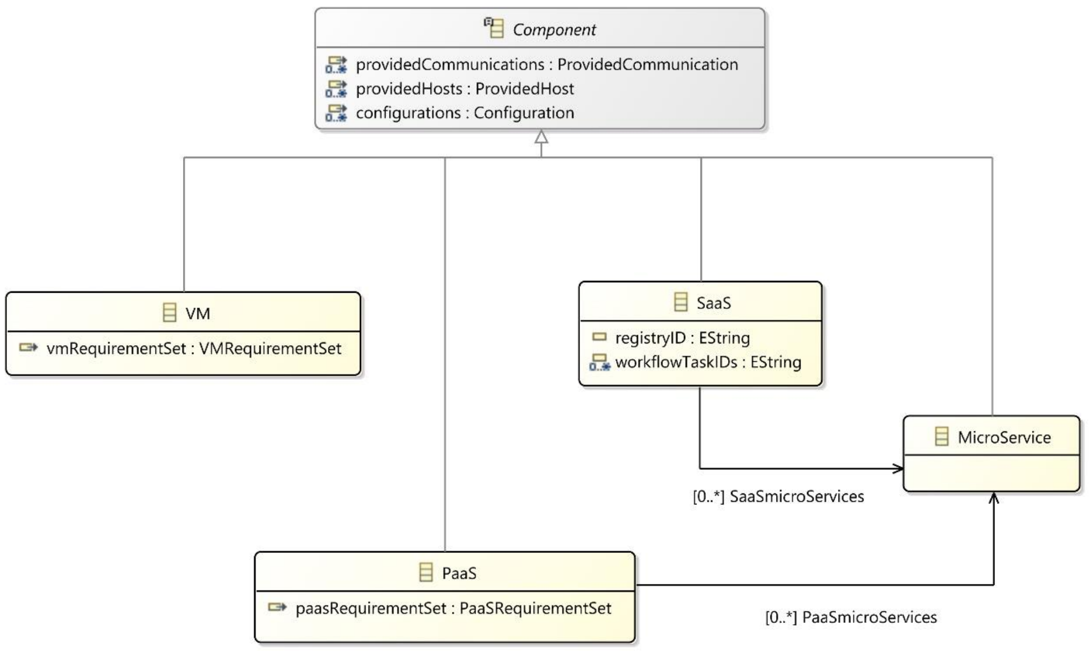

Component
providedCommunications : ProvidedCommunication
providedHosts : ProvidedHost
configurations : Configuration
VM
vmRequirementSet : VMRequirementSet
SaaS
registryID : EString
workflowTaskIDs : EString
MicroService
[0..*] SaaSmicroServices
PaaS
paasRequirementSet : PaaSRequirementSet
[0..*] PaaSmicroServices

Figure 3. 1 : Deployment Model Component Additions

To support the type-instance pattern and thus cater for the coverage of the models@runtime approach, CAMEL also covers the instance layer. The definition of respective instances of types (e.g., VMInstance having as type the VM class/concept) then covers the contents of this instance layer. So for each new type class defined in deployment meta-model, an instance class was also modelled.

3.2 CAMEL’s adaptation meta-model

CAMEL's scalability package has been developed to enable the modelling of scalability rules for the support of the adaptive provisioning of applications in order to retain a certain level of service. For this purpose, the Scalability Rules Language (SRL) has been developed. It enables the specification of noteworthy event patterns, determining the current problematic situation, that can lead to the triggering of scalability actions for enabling to change the application's configuration at runtime to address such a problematic situation. Apart from the scaling actions we need to define advanced adaptation rules and tasks which cover all the levels of abstraction. For the above reason we performed the first extension to CAMEL’s scalability meta-model. Through the new extension, the scalability was renamed adaptation meta-model. In this we introduce the Task class which has sub-classes the Application Task and the Adaptation Task. The latter is sub classed into two kinds of adaptation tasks: Single and Composite. Then, for the Single Adaptation Task we have specialization of adaptation actions at different levels of abstraction. Also, we have modelled the Adaptation Rule class which represents the mapping of a trigger event with an Adaptation Task. Lastly, we introduce Adaptation Strategies which can be considered as sets of Adaptation Rules which have in common the same event that triggers them. We let the system exploit the priorities of Adaptation Rules of an Adaptation Strategy in order to select the best possible one. The Adaptation Strategies needed in order to organize the adaptation rules in the context of the same triggering event. Bellow we will detail all the above parts of the extended adaptation meta- model.

3.2.1 Adaptation Rule

An Adaptation Rule maps an event to an adaptation task that needs to be executed in order to address the critical situation represented by that event. An Adaptation Rule has a distinct name; an event that triggers it; the adaptation task that should be enacted upon the event triggering, and a float variable that defines its priority. At this point, we should mention that this priority is regulated by the adaptation system based on all the alternative adaptation rules that can be enacted based on the same event (pattern) and the history of execution of all these rules. As such, an adaptation rule that is deemed to be able to better handle the current event (pattern) obtains the highest priority from those included in the same Adaptation Strategy.

3.2.2 Adaptation Strategy

A set of Adaptation Rules that are triggered by a certain Event or Event Pattern constitute an Adaptation Strategy. This means that the critical situation can be alternatively handled by the adaptation rules within the corresponding adaptation strategy. An Adaptation Strategy has its own unique id and contains a set of Adaptation Rules. The different Adaptation Rules with the same triggering event have different priority value at runtime. The value of priority is calculated by a mathematical formula and the analysis of this formula follows in Chapter 4. We can see the graphical representation of the Adaptation Rule and Adaptation Strategy in the following figure (Figure 3.2).

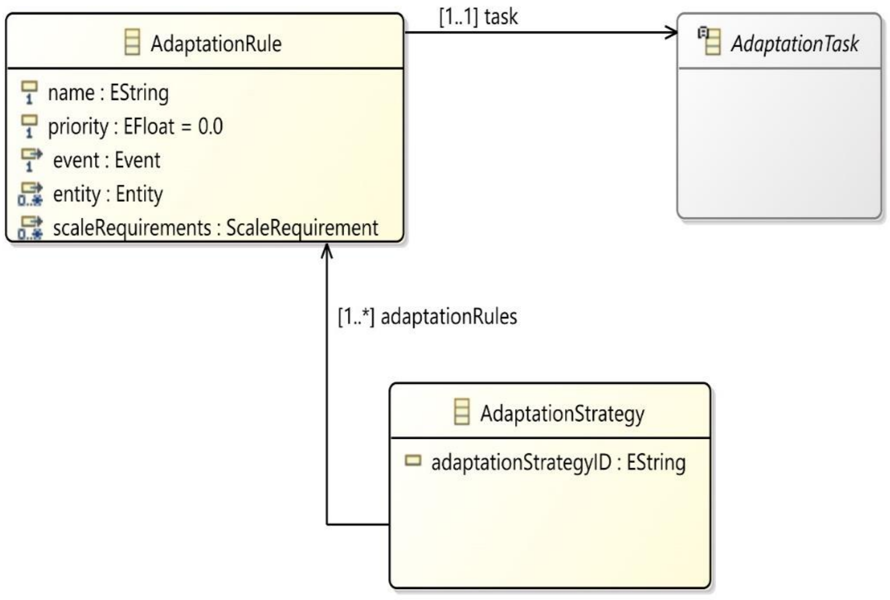


Figure 3. 2 : Adaptation Rule & Adaptation Strategy

3.2.3 Adaptation Workflow

An Adaptation Workflow controlling all the appropriate executing tasks which act in a critical situation. Below is the definition of basic classes that focus in adaptation-related tasks (either single or composite) and take part in the adaptation meta model package.

3.2.3.1 Task

A Task can be considered as a certain functionality which can be executed in the context of an application or an adaptation workflow. A Task has a unique id, in case it needs to be uniquely identified within the workflow it belongs to and a task name. As we can see in the figure below (Figure 3.3), a Task can be either an ApplicationTask or an AdaptationTask.

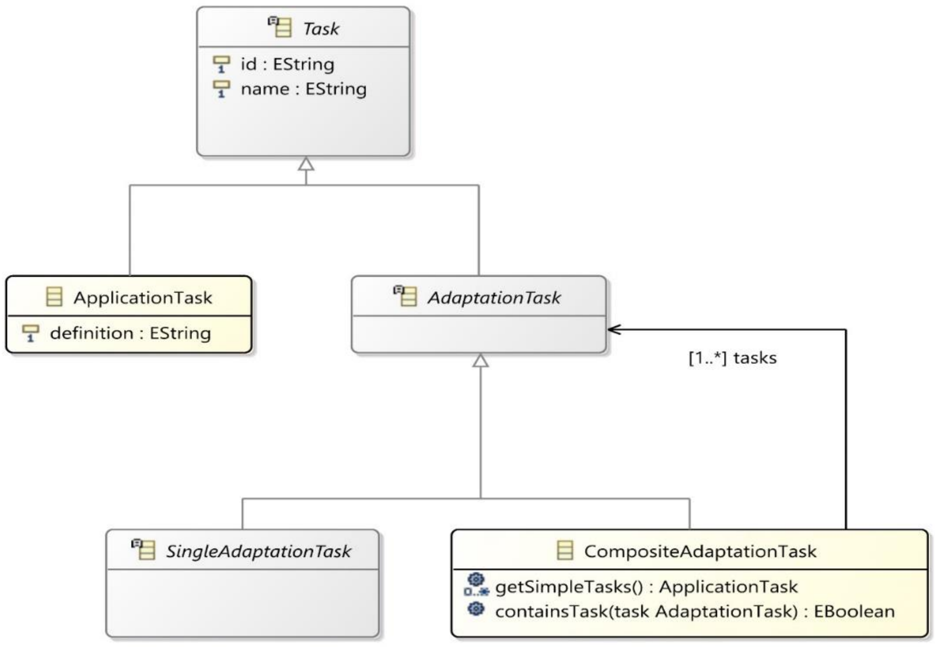


Figure 3. 3 : Task analysis

3.2.3.2 Application Task

In the case of application tasks we have a set of tasks which are mapped with software components and construct a workflow of an application. These software components can take the form of an application or a service (eg SaaS or Microservice). In order to model an application task, we describe the key elements that characterize it. For this purpose, an additional variable in the application task models the specification of the task (e.g., in terms of a standard language like BPMN).

3.2.3.3 Adaptation Task

An Adaptation Task acts at any level of abstraction. In this case, we have mapping between tasks with abstraction type of components (tasks, workflows, software components) that need to be enacted in a critical situation. There are two types of adaptation tasks. Firstly, the Single and, secondly, the Composite adaptation task, which actually maps to the definition of an adaptation workflow. A Single adaptation task describes an action on a set of one or more software components or on a certain application task or the whole application workflow. So we have a set of Single Adaptation Tasks that are shown schematically in the following figure (Figure 3.4) and these will be analyzed in the following.

Single Adaptation tasks are related to tasks that can be applied to a certain level covered in the Cloud (WfaaS, SaaS, PaaS, IaaS). The extensive description of the subclasses of Single Adaptation Task that have been defined is given below.

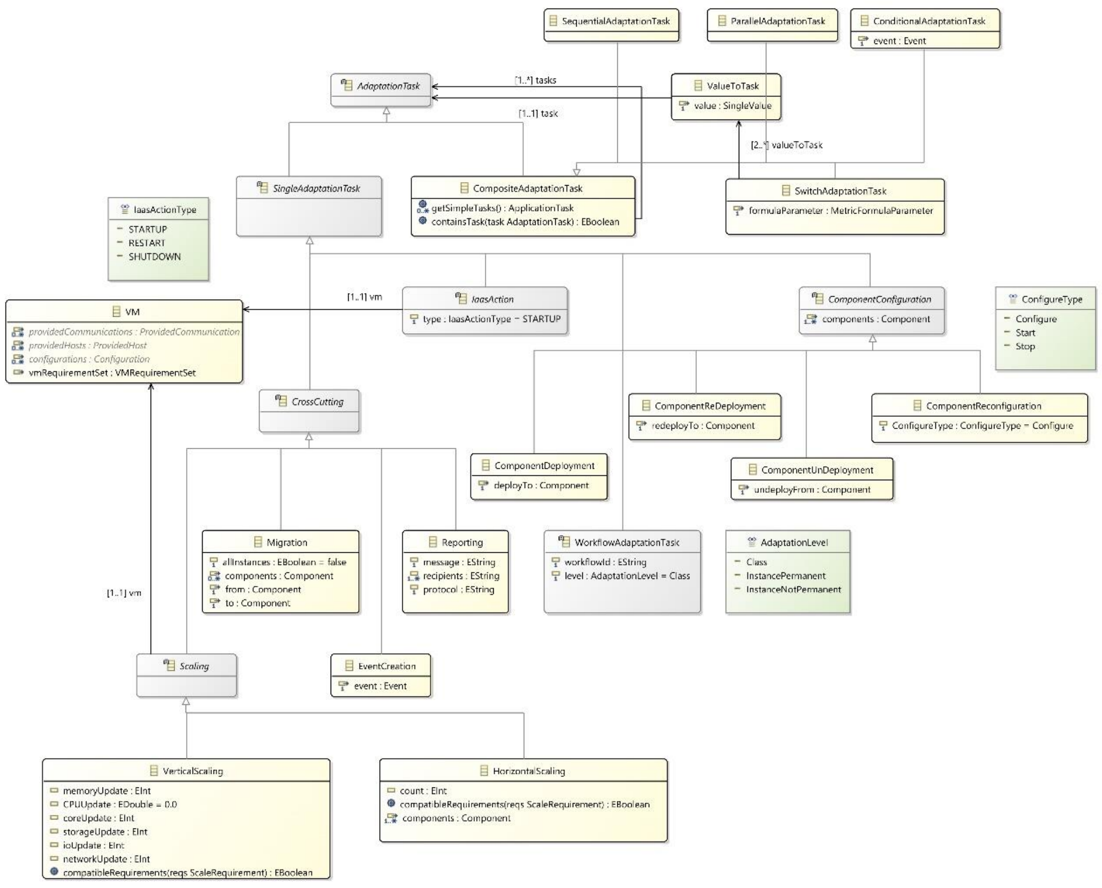


Figure 3. 4 : Single Adaptation Tasks

3.2.3.3.1 Single Adaptation Tasks

This section contains an extensive description of the different types of Single Adaptation Tasks that have been modelled in the adaptation meta-model.

A. Component Configuration

This is an abstract class that represents a component configuration task (Figure 3.5). This class contains the following subclasses:

a. Component Deployment

This task represents the deployment of one or more components over a target hosting component. Details about the configuration of the component(s) for its (their) proper deployment in that host can be found in its (their) own specification, where there is a containment association.

b. Component UnDeployment

This task can be used to both uninstall and delete one or more components. To support the execution of this task, we need to know both the components to be uninstalled and the hosting component in which these components have been deployed. Again, details about the component(s) undeployment can be found in its (their) specification.

c. Component Redeployment

This task represents the redeployment of one or more components over a target hosting component. To support the execution of this task, we need to know both the components to be redeployded and the hosting component in which these components would be deployed.

d. Component Reconfiguration

With this task we can run a specific set of configuration commands on one or more components. The set of configuration commands to be invoked are captured in the configuration type, i.e., a certain enumeration which includes as members, the commands of the components start, stop and configure.

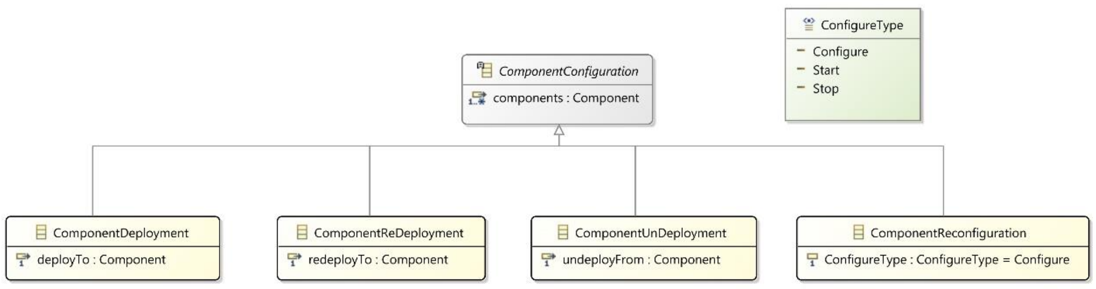


Figure 3. 5 : Component Configuration Graphical Representation

B. IaaS Action

Here is the definition of tasks related to adaptation actions that are invoked on infrastructural elements, such as VMs (Figure 3.6). More specifically:

a. StartUp

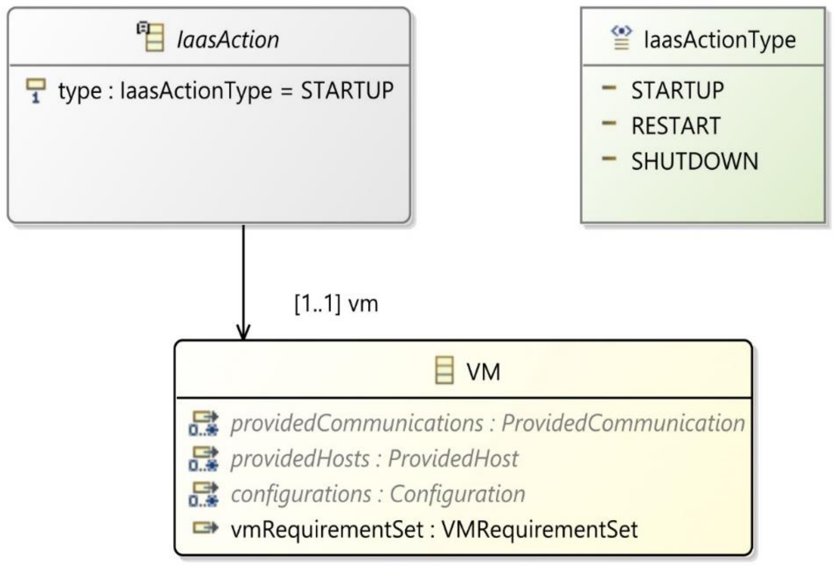
This task is dedicated to the start up of a certain VM which is referenced. It could be useful when a certain VM is found down and we thus need to start it up.

b. ShutDown

This task is dedicated to the shut down of a certain VM which is referenced. It could be useful when we need to stop the operation of a certain VM.

c. Restart

This task focuses on rebooting a certain VM. In fact, this task can be realized through the sequential execution of the above two tasks (first shutdown and then startup).



IaasAction
type : IaasActionType = STARTUP
IaasActionType
STARTUP
RESTART
SHUTDOWN
[1..1] vm
VM
providedCommunications : ProvidedCommunication
providedHosts : ProvidedHost
configurations : Configuration
vmRequirementSet : VMRequirementSet

Figure 3. 6 : IaaS Graphical Representation

C. Workflow Adaptation Task

This is an abstract task that is associated with the adaptation task indicating tasks able to adapt a certain workflow. In this abstract class we specify the id of the workflow and the type of the adaptation level. In an Workflow Adaptation Task we have the specification of the task adaptation level. The difference is that at the class level which is an enumeration that has three values; the changes are permanent and cover all instances of the respective component (e.g., application, workflow) targeted by the adaptation task, while at the instance level the changes are temporary and concern a certain instance of that component only. If the non_permanent at instance level case holds, then the change is applied only for one iteration in the loop. If it is permanent, then it will hold for all iterations of that loop (Figure 3.7). This class contains the following subclasses:

a. Workflow Recomposition

This task is related to how the tasks are organized within the workflow. With this task we can modify the content and structure of the workflow within a certain workflow region. It is useful in the case of the system or a user decide to change the structure of the elements in the workflow. Any adaptation task that is being modelled here, is actually executed by a system automatically. The modelling could be done manually, automatically or semi-automatically. The decision to adapt is actually transferred to the modelling, in other words, once you model a rule you bind the decision (when we should adapt) with the respective action (what is involved in the adaptation). Usually, the workflow region is defined between the current execution point and the last workflow element. We assume this by default and provide modelling elements in the respective cases which take the form of adaptation workflow tasks where the actual impact/replacement region needs to be specified by the user.

b. Task Modification

This is an abstract task that is associated with the modification of application tasks within the workflow referenced.

a. Task Addition

This task describes how to insert a new element into the workflow. So refer to ids of possible workflow elements that precisely specify the position of the task insertion. The needed information are the workflow id and the position of the new task between the others in the workflow.

b. Task Deletion

Similarly, to the case of task addition, we need to specify the id of the workflow being modified as well as the position in the workflow on which the respective action/task needs to be performed. That position witnesses the actual task to be deleted so there is no need for a direct reference to it. In case of a non-permanent change, task deletion can be regarded as task bypassing. As the task will be omitted from execution only for the current workflow instance affected by the critical situation

c. Task Replacement

Here is a description of replacing a task with another one within the same workflow. A task could appear multiple times within a workflow. However, we consider that each occurrence will map to a workflow task with the same name but different id. So potentially, we could use the ids in order to distinguish occurrences from one and the other at the replacement phase. For the execution of this task we need the new task of the replacement.

d. Service Replacement

Service Replacement is a task associated with replacing an entire service with another one. Here, in this task, we need to refer to these services as well as the place where the old service had been deployed. Another important piece of information is the service uri which fully identifies the new service. In case of SOAP services, it could also help to obtain their whole specification (in WSDL).

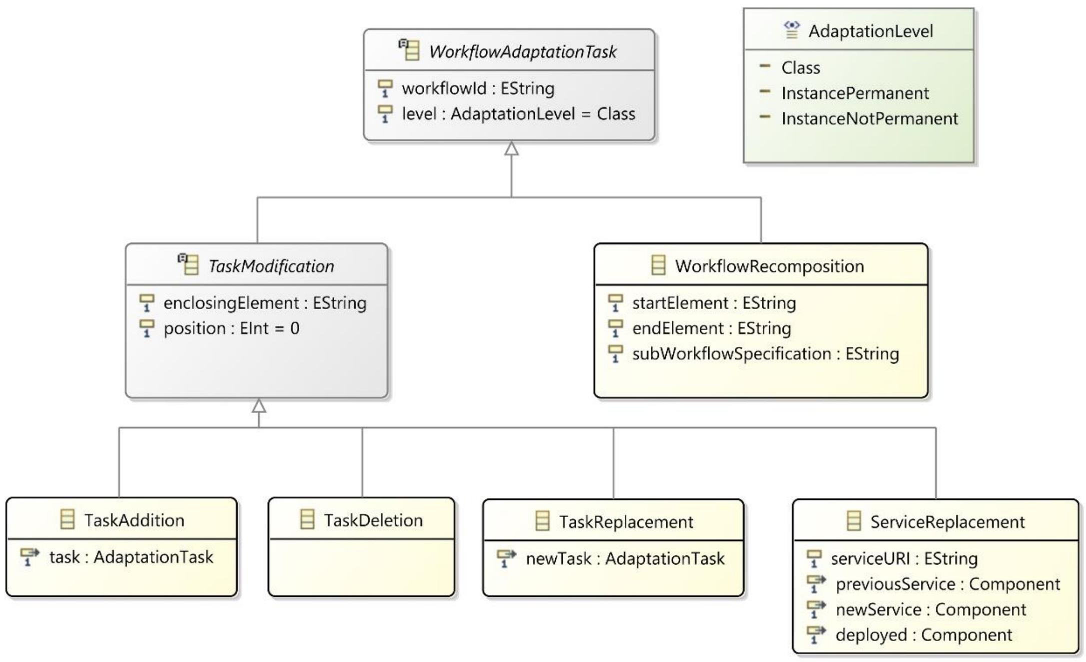


Figure 3. 7 : WfaaS Tasks Representation

D. CrossCutting

In this class, we have the case of tasks which are cross-cutting to all the others. These can be used for reporting events/messages to certain recipients as well as generating events which could be used to trigger adaptation workflows. So, these tasks alert the system with the critical situations that have been identified. They are also needed for warning and alerting the admin/expert when situations occur that require further investigation. It could be also possible that such users are always informed about any single piece of adaptation action / workflow that is executed. In that case, these users not only get informed about the critical situation but also how this situation was attempted to be addressed. In the cross cutting events we have also the Migration and the Scaling because these are tasks for all the layers of the Cloud model (Figure 3.8). More specifically:

a. Reporting

This task is responsible for reporting a message to a certain set of recipients based on a certain protocol. An example for this reporting task is the report to the administrator of a system for a particular subject over email.

b. Event Creation

This is a task dedicated to the creation of a new event in order to alert the system about a critical situation. For example, when reaching the scalability limits of a certain component, an event creation alerts the system about it.

c. Migration

Migration means moving one or more software components from one hosting component to another one. Such an action could be offered by a PaaS provider. It could be also part of the management platform of a certain organization. For this task, we specify the set of components that will be transferred, the initial hosting component where these components have been already deployed, the target hosting component on which these components will be transferred and a Boolean annotation which shows if the migration is made for all the instances of the transferred components or only for a particular one. If the system decides which should be the most optimal target hosting component, then such a component should not be specified by the modeler. In this latter case, the place of deployment depends on the requirements of the components to be transferred. Each adaptation task (single or composite) is executed by the adaptation system. The required installation of the components in IaaS as well as all the actions needed to make the service operational are taken care of by the one who offers the adaptation system / framework which could be considered as a PaaS provider. We mention below all possible migration cases:

- PaaS → PaaS
  One PaaS provider could support the migration of a certain component from one PaaS environment to another one. The PaaS environment might or might not be provided by different providers. In the latter case, we could imagine the possibility that we need to upgrade the PaaS service within the same PaaS provider). In the case of different PaaS providers (original and target) we change the Cloud. While in the case of same PaaS provider, the Cloud is not changed.

- PaaS → IaaS
  Here we have the migration of a PaaS environment owned by PaaS to an IaaS environment owned by us. It requires to exploit an IaaS abstraction tool or the IaaS interface of the target provider to support the migration. The adaptation system (or service if we consider that each action maps to a certain service) decides about the realization of the migrated service.

- IaaS → IaaS
  Migration here has to be done by adapting one or more software components based on the interfaces and facilities offered by the corresponding IaaS providers (origin and target). If we change provider, we also change the Cloud. If not, then the Cloud remains unaffected.

- IaaS → PaaS
  Here, we have the transfer of the components in an environment ready for operation, as the use of a PaaS enables this possibility. However, the undeployment of components is the responsibility of the action executor which could exploit the facilities / interfaces of the origin Cloud provider. First of all, we should have the insurance that the service in the new environment is operational and then we perform the undeployment in the origin Cloud. If the migration does not succeed, we are still left with the previous deployment of the application.

d. Scaling

a. HorizontalScaling

This task is relative to increasing or decreasing the number of instances of a certain component which is deployed on respective instances of a certain VM. The arguments of this task is the number of instances and corresponding components. When you attempt to increase the number of instances of that component, you have a scale-out and then the number in the count argument, which is the proposed number of instances, is positive while in the opposite case you have a scale-in and then the number in the count is negative.

b. VerticalScaling

Actually, this action attempts to adjust the capabilities of the VM on which a certain component is deployed. Upgrading or downgrading such capabilities results in a scale-up or scale-down, respectively. When we need additional resources (of possibly different types), then we can request the precise increase in the amount of resources of the respective VM type with a scale up action. On the other hand, a scale-down leads to decreasing the amount of resources of the VM type referenced. When implementing we assume that when the update values over the VM/container hosting components increase, we have a scale-up; when decreased instead, we have a scale-down.

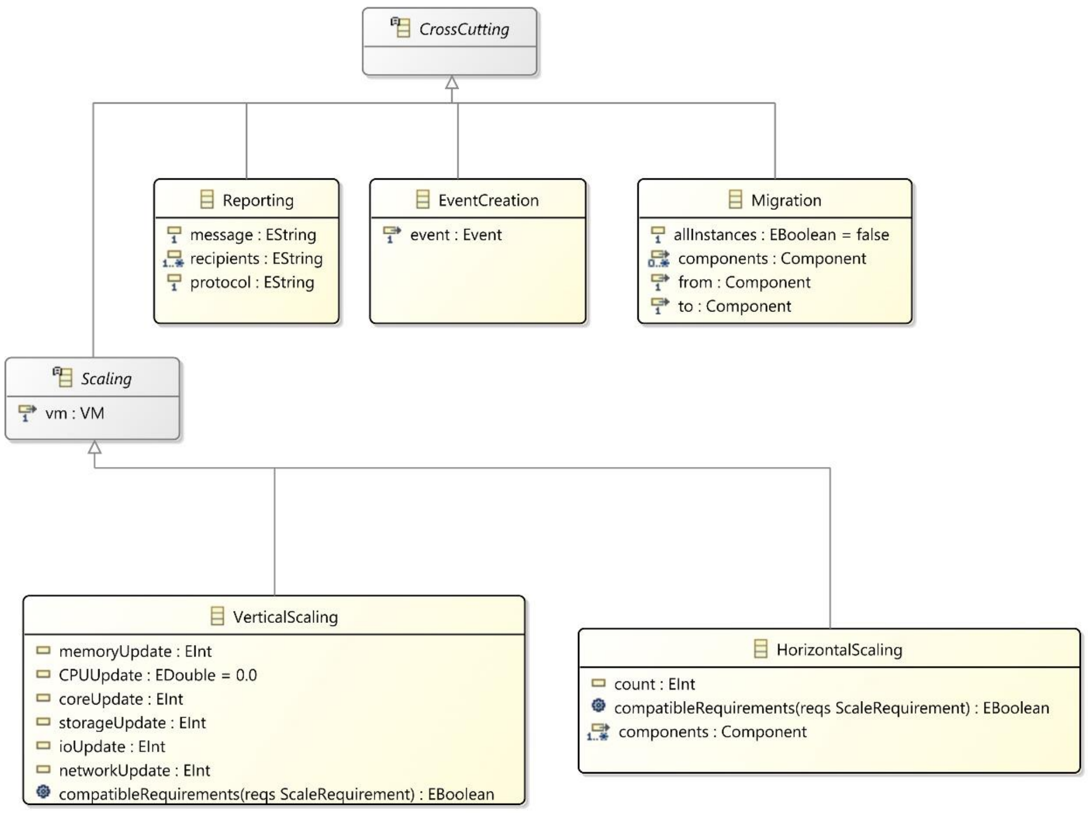


Figure 3. 8 : Cross Cutting Graphical Representation

3.2.3.3.2 Composite Adaptation Task

A composite Adaptation Task can represent an Adaptation Workflow and it maps to a hierarchical tree structure where at the leaves are placed Simple Adaptation Tasks. This structure describes the performance such as the specific order with which tasks should be executed. Each composite adaptation includes a set of adaptation tasks. Such a modelling can eventually lead to the production of hierarchical tree structures. Figure 3.9 depicts the adaptation meta-model, the composite adaptation task and its sub-classes. The sub-classes are analyzed below.

We define types of composite adaptation tasks. Each sub-class of composite adaptation task maps to a concrete type that corresponds to a well-known and used workflow (control flow) construct. These types can be mixed with each other as:

A. Sequential Adaptation Task

This task should be specified when we need to describe a sequential workflow / execution of adaptation tasks.

B. Parallel Adaptation Task

This task should be specified when a set of adaptation tasks needs to be executed in parallel.

C. Switch Adaptation Task

This task concerns the selection of an adaptation task from two or more alternative tasks. This selection depends on the respective value of a certain metric formula parameter (i.e., of a metric or a certain formula over a metric set). This means that each adaptation task alternative is mapped to a different value of that metric or metric formula. Also we have the property **Value To** in order to support a kind of mapping between the values of a metric formula parameter and alternative tasks.

D. Conditional Adaptation Task

This task models a conditional composition of adaptation tasks in an if-then-else fashion where the occurrence of the respective event leads to the execution of the first adaptation task referenced while the non-occurrence of this event leads to the execution of the second task referenced.

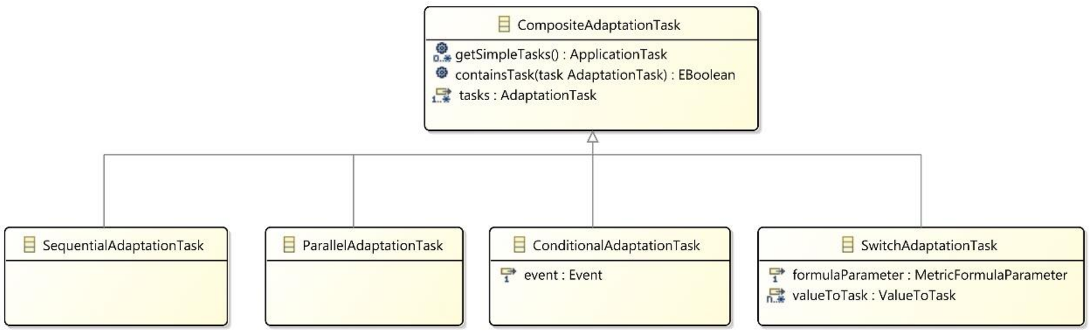

CompositeAdaptationTask
getSimpleTasks() : ApplicationTask
containsTask(task AdaptationTask) : EBoolean
tasks : AdaptationTask
SequentialAdaptationTask
ParallelAdaptationTask
ConditionalAdaptationTask
event : Event
SwitchAdaptationTask
formulaParameter : MetricFormulaParameter
valueToTask : ValueToTask

Figure 3. 9 : Composite Adaptation Tasks

3.3 CAMEL’s execution meta model

Apart from the deployment package update and the extension of the adaptation package of CAMEL, we have performed an extension also for the execution package. The initial goal of the execution model, apart from the capturing of the application history, was also to enable the analysis of this history in order to support deployment reasoning. Now, this goal is extended in order to cover the dynamic calculation of the priority of adaptation tasks and corresponding adaptation rules. Respectively, we will analyze the modifications at the execution meta model. The execution meta-model is the part of CAMEL's ecore model which defines the needed class mapping to the application runtime measurements. One of the defined classes in the execution meta-model is ExecutionContext which contains information related to a certain execution of the application. It covers one task execution session. New sessions are covered by different execution contexts. In the same package, there is also the Measurement class with its subclasses which are: Application Measurement, Internal Component Measurement, Communication Measurement, VM Measurement and PaaS Measurement. Another important class is the SLO Assessment class which represents the evaluation of an SLO in the context of a produced measurement. Lastly there is the RuleTrigger class which encapsulates all the needed information related to the triggering of an Adaptation Rule. This class has been updated in order to connect with the Adaptation Rule class. Previously, it pointed to a scalability rule but now it points to an adaptation rule. In the execution meta-model, we have introduced new attributes within the TaskRealization class needed for the computation of the adaptation rules priority. In the following, we will analyze all these new attributes.

3.3.1 Adaptation Histories Records

The Task Realization Class was introduced in order to keep all the needed information for the execution history records needed for the computation of the final priority of an Adaptation Rule. In this class, we have a characteristic name for the task realization, the corresponding Adaptation Task, the start time and the end time of the task execution, and two counters where the first one (upTimes) measures the number of times this task was available and the second (pingTimes) the number of pings performed in the context of availability checking during the task execution. This class also contains two Boolean, one concerning whether the execution of the adaptation task was error/bug-free (executionFault) and another one focusing on indicating whether the execution of this task was able to successfully address (executionSuccess) the respective event that triggered it. In the Figure 3.10 we can see both the RuleTrigger and the TaskRealisation which are the classes where the extension of the execution package focused plus their relations with other classes of the execution meta-model.

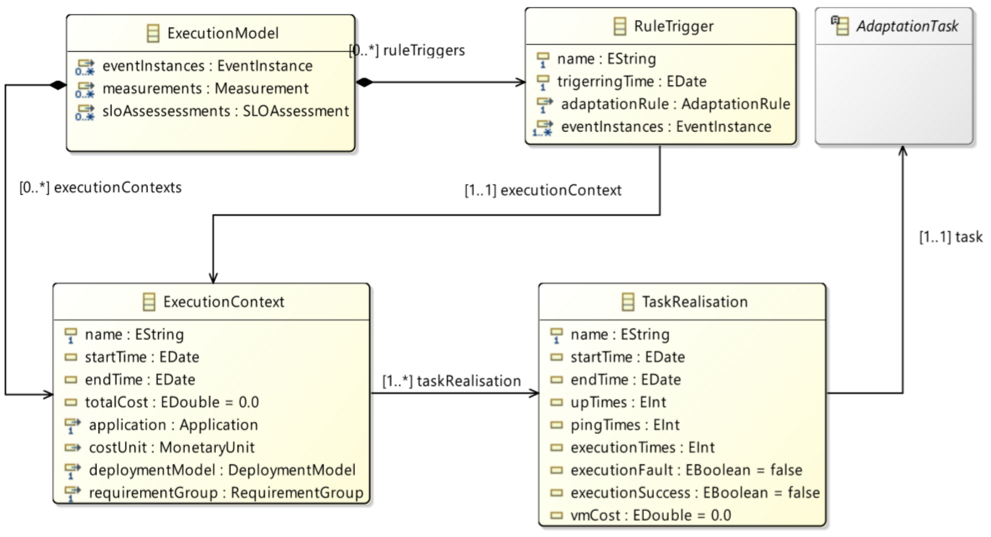


Figure 3. 10 :Task Realization Class in execution meta-model

3.4 Adaptation Scenarios Example

In this section, we will use the Traffic Management use case from Chapter 2 in order to explain how the extended CAMEL can be used to cover the modelling of specific adaptation scenarios; specifically the adaptation rules suiting them. So, we assume that some violations in the traffic management system could be detected. These violations relate to some particular metrics, like availability, uptime, response time of the called services etc, that could be the reason of an SLO violation. We will describe a set of adaptation scenarios and we assume that the reason of adaptation in them is particular SLO violations similar to the violations according to our use case example which is analyzed in the Chapter 2 (Section 2.7). In the following scenarios we describe the triggering events which cause the violations without to analyzing the corresponding metric conditions that cause the violations.

3.4.1 Adaptation Scenario 1 - Migration

There is an under-estimation of the resource requirements of the Monitoring Service and the private infrastructure does not have enough resources to cover these requirements. So, there is a need to move to a public Cloud instead. For this reason, the adaptation task of Migration will be activated by the corresponding adaptation rule.

More specific:

The triggering event could be: event_A = “Not enough memory is available” for the VM hosting the Monitoring Service and the corresponding task could be the single adaptation task:

task_A { Migration (Monitor Service, private/municipal Cloud component, public Cloud component,allInstances = true) }.

The respective adaptation rule could then be:

rule_A = event_A → task_A

3.4.2 Adaptation Scenario 2 - Component Replacement

Some changes in the direction of the roads in the particular area of the city center leads to the creation of a new version of the software component of the Monitoring Service. So there is another service that could be used to replace the old one. In this sense, in the Traffic Management Application, the software component of the Monitoring Service should be replaced with the new service which is named Super Monitoring Service.

More specific:

The triggering event could be: event_B = “Permanent deterioration of the performance” for the Monitoring Service and the corresponding single adaptation task could be:

task_B {Service Replacement (Monitor Service, Super Monitor Service)}.

The respective adaptation rule could then be:

rule_B = event_B → task_B.

3.4.3 Adaptation Scenario 3 - Cross Cutting

A temporary damage to the traffic lights of the zone_A in the city of Heraklion makes it necessary to inform the drivers about this damage through a report-based event via electronic road signs to the drivers.

More specific:

The triggering event could be: event_C = “temporally unavailable traffic lights” in the zone_A, and the single adaptation task could be:

task_C {Reporting(“traffic lights damage”, “Device Configuration Service”,”electronic road signs”)}.

The respective adaptation rule could then be:

rule_C = event_C → task_C.

3.4.4 Adaptation Scenario 4 - IaaS & Horizontal Scaling

Actually, the response time of the Assessment Service surpasses the respective SLO threshold such that there is an SLO violation related with a memory allocation failure. An event is triggered by this violation to activate the corresponding adaptation rule.

More specific:

The triggering event could be: event_D = “memory allocation failure” for the Assessment Service. The single adaptation task could be:

task_D {Horizontal Scaling (Hosting Components, Number Of Instances, Assessment Service)}.

And the adaptation rule could be:

rule_D = event_D → single adaptation task_D.

3.4.5 Adaptation Scenario 5 - IaaS & Vertical Scaling

When the Monitoring Service is running, the system monitoring shows that the main memory (in the respective VM) is no longer sufficient. The increased system resources needed for the VM hosting the Monitoring Service requires a memory growth. For this reason, the system monitoring would create a memory allocation warning event. As such, the solution to this problem would be the activation of a corresponding adaptation rule for vertical scaling.

More specific:

The triggering event could be: event_E = “memory allocation failure” for the Monitor Service. The adaptation single adaptation task could be:

task_E {Vertical Scaling (Monitor Service VM, memory Update, core Update, storage Update, io Update, network update, Scale up)}.

And the adaptation rule could be:

rule E = event_E → task_E.

3.4.6 Adaptation Scenario 6 - Workflow

Apart from single adaptation tasks, the system can also perform composite adaptation tasks. So, we assume that we have an application workflow related to the instance_A of Traffic Management Application instances (Chapter 2 section 2.3).

Due to an accident occurring in zone_A, the Medical Forces should be summoned so that they reach the point of accident and transport the injured to the hospital. However, the Rescue Forces should also reach the point to clear the area of objects produced by the collision. In this scenario, we have the parallel activation of the above application tasks which are produced by the Assessment Service and compose the plan with the actions to be performed by Device Configuration Service. So, we have an application workflow with the participation of two stakeholders, the Medical Forces and the Rescue Forces. These stakeholders are activated by the corresponding application tasks (Chapter 2.1) These tasks run in parallel in the application workflow. We can see an indicated figure (Figure 3.11) below.

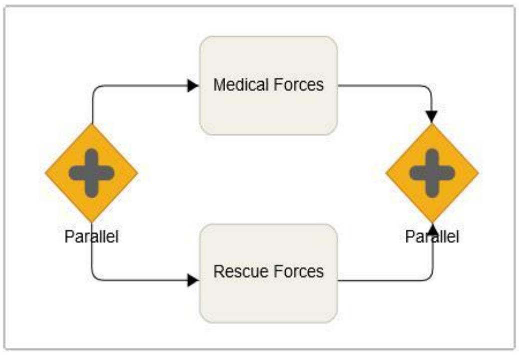


Figure 3.11 Application Execution Plan

The execution of the above tasks that should be performed under normal conditions are not carried out as an event pattern is triggering and an adaptation rule takes action to address a critical situation that is caused.

The event pattern includes two events that are:

- event_F = “run time violation” for the Assessment Service.
- event_G = “input mismatch” for the Device Configuration Service.

Thus the event pattern is: Event Pattern_ I {event_F , event_G }

The adaptation tasks that react to this event pattern are:

- single adaptation Task_F → Restart (Assessment Service).

- single adaptation Task_G → Reconfigure (Device Configuration Service).
- composite adaptation Task_H → (single adaptation Task_F, single adaptation Task_G).

The final composite adaptation task contains the adaptation tasks for the mapping events of the Event Pattern_I. So, the adaptation rule for the final complex adaptation task could be:

Adaptation rule K = {Event Pattern_I→ composite adaptation Task_H}

Chapter 4

4.Adaptation Rules Priority and Dynamic Selection Algorithm

The content of this chapter related with the analysis of the mathematical formula for the computation of the priority and the selection of the most appropriate adaptation rule. Firstly we analyze the mathematical formula of the computation of the priority and we make the correlation between the priority of the adaptation tasks and the adaptation rules (Section 4.1). Then we analyze all the quality attributes and the types of the computation of their utilities (Section 4.2). Sequentially we have the introduction of the dynamic selection algorithm of the most appropriate adaptation rule (Section 4.3) and two cases these algorithms being used (Section 4.4).

4.1 Mathematical Formula and Adaptation Tasks Correlation

For the selection of the most appropriate adaptation rule in order to increase performance we introduce a mathematical formula for the computation of adaptation rule priority. An adaptation rule is comprised of one adaptation task. This can be single or composite. In the latter case, it can include other adaptation tasks which can be single or composite (Chapter 3). Therefore, the computation of an adaptation rule priority depends on the computation of the adaptation task priority.

An execution context is exported by the CAMEL's execution meta model (Chapter 3) for every adaptation task whether this is a single adaptation task or a composite. Thus, in the following, we will first detail all the quality attributes of the mathematical formula for the calculation of the adaptation task priority, then we will present this formula and explain the procedure for applying it over the rules of an adaptation strategy. Finally, we will present the selection formula for the most appropriate adaptation rule in the context of a triggering event as a dynamic adaptation rule selection algorithm. The most appropriate adaptation rule in this algorithm is that with the highest priority.

4.2 Formula Quality Attributes and Utilities

The suggested mathematical formula computes the overall utility of an adaptation rule according to the sum of certain metrics (Figure 4.2) related to the quality of the corresponding adaptation tasks. We have particular quality attributes which are measured by respective metrics, and we define utility functions for them in order to calculate the utility of an adaptation task per quality attribute. The value result after the use of the utility function is a number point 0 to 1. The normalization of the numerical value of quality attributes is the reason for the use of the corresponding utility function. For the triggering events and event patterns, the adaptation Task Realization class of the execution meta-model records a set of execution data (Section 3.4). The mathematical formula is implemented on three levels of computation. On the first level we use this data as input in order to compute the value of each quality attribute. On the second level we use each exported value of the previous step as input and we implement a utility function in order to compute the utility of every quality attribute for each adaptation task. On the third level of computation we implement a method function and we use the set of the utilities of the quality attributes computed at the previous level, and the weight that users define for each of the quality attributes as input in order to compute the last utility value of the corresponding adaptation task which is equal to its priority. On the third level of computations the monotonicity of each quality attribute affects the result. The ultimate goal of the formula is the computation of the priority of the corresponding adaptation tasks, and by extension, of each adaptation rule.

As such, we can estimate the quality of an adaptation task through the analysis of the following quality attributes: execution time, availability, failure rate, cost of execution and success rate. We come up with the computation of the utilities of the quality attributes by the values of the corresponding metrics. The whole approach of the three levels of computation and the selection of the most appropriate adaptation rule with the use of the proposed mathematical formula are both implemented as an adaptation rules selection algorithm. The analysis of the corresponding mathematical formula quality attributes and the overall priority computation follow in the next subsections.

4.2.1 Quality Attributes Analysis

In this subsection, we will provide details of the **first level of computation**. In particular, we will explain the semantics of each attribute and then clarify how it is computed from the information expressed via CAMEL in the execution model of the application. For the needs of the next level of computation, we will analyze the corresponding quality attributes (Figure 4.1) and their overall values.

4.2.1.1 Execution Time

The execution time metric of an adaptation task is modelled in the execution meta-model as execution start time and execution end time in the Task Realization class. The more an adaptation task takes to execute, the less suitable it might be for adaptation. The adaptation task requiring the least execution time is the most appropriate if we assume that the cost of its execution depends on the execution time. For the execution time quality attribute value, we compute the mean execution time of adaptation task i, by dividing the sum of the raw execution times of the adaptation task by the number N which is the number of times it has been executed. By raw execution, we mean the subtraction of the executionStartTime from the executionEndTime for every index j of executions. More formally:

$$\mathbf{MeanExecution\ Time}_i = \frac{\sum_{j=1}^{N} \left(\mathbf{executionEndTime}_{i,j} - \mathbf{executionStartTime}_{i,j}\right)}{N}$$

In the following analysis the value of the Execution quality attribute is of equal value to the Mean Execution time which is computed by the above formula and whose unit of measurement is time. Also, in the remaining quality attributes computation formulas, all references to i, j, and N share the same semantics as the above. More specifically:

- i = the adaptation task
- j = the index of executions
- N= the number of execution times for i

4.2.1.2 Availability

Another quality attribute is availability. In the execution meta-model, two counters in the Task Realization class are implemented, one for the upTimes and the other for the pingTimes needed for the computations of the availability quality attribute value. In this case, we compute the availability of adaptation task i by dividing the number of times the adaptation task was available by the number of times it was pinged. More formally:

$$\mathbf{Availability}_i = \sum_{j=1}^{N} \frac{\mathbf{upTimes}_{i,j}}{\mathbf{pingTimes}_{i,j}}$$

The task is pinged only while being executed such that the respective upTimes are independent of the execution times and are defined during execution. In the following analysis the value of the Availability quality arises from the above mathematical formula of computation, which is represented as a percentage.

4.2.1.3 Failure Rate

The result of the execution of an adaptation task can vary. Sometimes, after the task execution, the respective result might not be the expected one. This could occur when, for example, the task execution stops with an error and an exception is thrown. This situation would then signify that the corresponding adaptation rule has failed. If such failures often occur for a certain task, such knowledge should be utilized to prevent executing this task in the near future in the context of a respective adaptation rule. Thus, for the computation of task Failure Rate, we need to compute the failure rate by dividing the sum of the faulty executions by the total number of execution times for this task. In the execution meta model we have the execution fault counter implemented in the Task Realization class for the computations of the Failure Rate quality attribute. The following formula denotes the computation of task Failure Rate:

$$\mathrm{FailureRate} = \frac{\sum_{j=1}^{N} \mathrm{executionFault}_{i,j}}{N}$$

In the following analysis the value of the Failure Rate quality attribute arises from the above mathematical formula and it too is represented as a percentage.

4.2.1.4 Successability

When the execution of the task finishes, another important aspect to consider is the Successability of the executed task. In other words, we need to know if the task execution was successful, that is, if it indeed responded to the cause which called for it. If the respective event that caused the triggering of the corresponding adaptation rule was successfully handled, this means that the adaptation task execution was successful. The result of this quality attribute is not the opposite of the result of the Failure Rate because here we focus on the success of the action and not on the success of the execution of the corresponding task. The execution of a task can be completed but the reason for the triggering of this task might not be satisfied. This attribute helps to avoid vulnerable loops of tasks. These tasks can result from the value of the rest of the quality attributes. For the computation of task Successability, we need the number of times the task had successfully achieved its goal of addressing the current event; and the total number of times this task was executed. In the CAMEL's execution meta-model we have a SuccessfulExecution counter implemented in the Task Realization class for the purposes of computing the Successability quality attribute. For the computation of the Successability value we need to divide the sum of successful execution times by the execution times of the adaptation task. More formally:

$$\mathbf{Successability}_i = \frac{\sum_{j=1}^{N} \mathbf{SuccessfulExecution}_{i,j}}{N}$$

In the following analysis, the value of the Successability quality attribute arises from the above mathematical formula, and is represented as a percentage.

4.2.1.5 Cost

Another requisite quality attribute is the cost of the requisite VM resources required for an adaptation task execution. The cost depends on the VM offering that is utilized for the hosting of the adaptation task. Given that the major Cloud providers have resource bases in US, the most common cost measuring unit is the US Dollar. The Dollar is thus used for the computation of cost in our analysis. We assume that the pricing policies are dictated by the provider and could refer to the use of VM per hour. Consequently, the chief concern in making our computations is the cost of the VM that hosts the adaptation task. In order to simplify the computation of the cost quality attribute, we assume that each task maps to a unique VM. Subsequently, what is required is the average execution time which is computed in the Execution quality attribute. If the Mean execution time is calculated in a different time unit than a second, we would need a unit conversion to be inserted in the formula. In the execution meta model we have the Cost counter implemented in the Task Realization class. The average cost of the adaptation task is computed by multiplying the cost of the hosting VM of the adaptation task by the Mean Execution Time which is equal to the value of the Execution Time quality attribute. More formally:

$$\text{Cost}_i = \text{VM cost}_i \cdot \text{MeanExecutionTime}_i$$

In the following analysis the value of the Cost quality attribute arises from the above mathematical formula, and the unit of measurement is dollars per time.

Quality Attributes
Execution
Availability
FailureRate
Successability
Cost

Table 4. 1 : Quality Attributes

Metrics
ExcecutionStartTime
ExcecutionEndTime
UpTimes
PingTimes
ExecutionTimes
ExecutionFault
SuccessfulExecution
vmCost

Table 4. 2 : Metrics

4.2.2Utility Function per each Quality Attribute

On the **second level of computation** we have the implementation of a certain form of a generic utility function. All the above quality attributes could be positively or negatively monotonic (Figure 4.3). For a positively monotonic metric, as the value of the metric increases the respective utility also increases (e.g e^x). In the case of a negatively monotonic metric, the opposite is observed(e.g e^{-x}). For each of the quality attributes, there is a minimum ($\min(\text{value})$) and a maximum value ($\max(\text{value})$) in the context of the (same) triggering event across all adaptation tasks that can address this event. For each quality attribute, we use the generic utility function in order to compute its utility value. The value annotation in the following mathematical formula is equal to the value of each quality attribute which has been computed in the first step of computation. The utility function for each quality attribute takes a certain form which depends on its monotonicity. This leads to the following generic forms of (linear) utility functions (1,2):

Positive Monotonic

$$\mathrm{uf}(\mathrm{value}) = \frac{\mathrm{value} - \min(\mathrm{value})}{\max(\mathrm{value}) - \min(\mathrm{value})} \qquad (1)$$

Negative Monotonic

$$\mathrm{uf}(\mathrm{value}) = \frac{\max(\mathrm{value}) - \mathrm{value}}{\max(\mathrm{value}) - \min(\mathrm{value})} \qquad (2)$$

Quality attributes	**Monotonic**
Execution	Negative
Availability	Positive
FailureRate	Negative
Successability	Positive
Cost	Negative

Table 4. 3 : Quality attributes monotonic characteristics

4.2.3 Adaptation Task Priority Function

In t**he final level of computation,** we analyze a method for final priority value of the corresponding adaptation rule. As we have already mentioned, the computation of an adaptation rule priority depends on that of the adaptation task priority. As far as the historical records of the adaptation tasks in the execution model are concerned, we handle the case of single and composite adaptation tasks in almost the same way. The only difference is that the cost of the composite task cannot be directly ascertained, and we need to compute it from its respective sub-tasks. This difference is taken into account in the computations performed in order to ascertain the cost quality attribute value in the first level of computation. We can thus make the following assertion:

$$\mathbf{AdaptationRulePriority_i = AdaptationTaskPriority_i}$$

Where i is the index of both the adaptation rule and its mapping adaptation task.

In order to denote the relative importance of each metric for the end-user, we rely on the Analytical Hierarchy Process (AHP) [13]. The result of this process is an assignment of weights to all of these quality attributes, indicating their relative importance, and whose sum should be equal to one. We also follow the Single Additive Weighting (SAW) technique [14] which maps to the utility for each attribute, which is equal to the weighted sum of the application of the global value derived for each quality attribute on its utility function. More formally:

$$\mathbf{AdaptationTaskPriority_i} = \sum_{q=1}^{Q} \mathbf{w}_q \cdot \mathbf{uf}_q\left(\mathbf{value}_{\mathbf{i},q}\right)$$

Where i is the adaptation task, q is the quality attribute, Q is the set of the quality attributes, w is the weight of each metric and uf is the utility function of the quality attribute value of the adaptation task.

4.3 Dynamic Selection Algorithm

All the computation levels are implemented in a dynamic selection adaptation rule algorithm. In the following subsections we will analyze dynamic selection algorithm by analyzing the adaptation rule priority formula, which is the core of the algorithm.

4.3.1 Adaptation Rule Priority Formula

If all three levels of computation are concluded, we can choose the adaptation rule with the highest priority. Hence, if we try to implement the whole process as an algorithm, the last step is to introduce the formula for the selection of the most appropriate adaptation rule.

The priority for an adaptation rule is analogous to the priority of the corresponding adaptation task in the context of the same triggering event (Section 4.2.3). An adaptation strategy contains the set of the adaptation rules which have been triggered in the past for the same triggering event or event pattern (Chapter 3). The role of the following computation formulas is to select the rule with the highest possible priority from the set of adaptation rules which exist in the adaptation strategy of a triggering event.

More formally :

$$\mathrm{AdaptationRule}_s = \max\left(\mathrm{AdaptationRulePriority}_{\mathrm{r}} \right)$$

Where s is the selected adaptation rule and r is the number of adaptation rules in the corresponding adaptation strategy.

4.3.2 Selection Algorithm

In this subsection we introduce the proposed dynamic selection algorithm for the selection of the adaptation rule with the higher priority value. This algorithm use the implemented mathematical formula of adaptation rule priority and its goal is the selection of the most proper adaptation rule in the context of a triggering event. In order to define the dynamic selection algorithm we should first analyze the separate parts of the mathematical formula functions. Thus, we will analyze each of the mathematical formula levels of computation with a psedo-code algorithm in order to make an introduction at the definition of the dynamic selection algorithm.

- In the **first level of computation** we have the calculation of the quality attributes values. For this reason the function in Figure 4.1 is introduced :

```
INPUT : Task Realization Object,Quality_Attributes
OUTPUT : Quality Attributes values for a task realization input

function: get Quality Attributes Values {
    for every quality_attribute in Quality_Attributes{
      QA_values[quality_attribute] = quality_attribute.calculateValue(Task Realization Object);
    }
    return QA_values;
}
```

Figure 4. 1 : Get Quality Attributes Values function

With this function, we calculate the value for each of the five Quality Attributes of a specific task. The complexity of the algorithm equals to O (n) as the iteration equals to the number of quality Attributes that is constant and the functions of the mathematical formula where called in the iteration are linear.

- In the **second level of computation** we have the utility function calculation. So the function in Figure 4.2 is introduced:

```
   INPUT : QA_values ,Quality_Attributes, Tasks, curr_Task
   OUTPUT : Uf_values for each quality attribute for the current task

 function: get Quality Attributes Utilities{
    for every task in Tasks{
    for every quality_attribute in Quality_Attributes{
              max_Value[quality_attribute] = get.Max(task.QA_values[quality_attribute]]);
              min_Value[quality_attribute] = get.Min(task.QA_values[quality_attribute]]);
   }
   }
   for every quality_attribute in Quality_Attributes{
    curr_task.utility = calculate.Utility(task.QA_values[quality_attribute],
                          max_Value[quality_attribute],min_Value[quality_attribute]);
         Uf_values[quality_attribute] = curr_utility;
   }
    return Uf_values;
 }
```

Figure 4. 2 : Get Quality Attributes Utilities function

With this function, we calculate the utilities for each of the 5 Quality Attributes of a specific task. For this calculation we need first to compute the minimum and the maximum values for each of the Quality Attributes of the tasks which have the same adaptation strategy with the corresponding task. Thus, the complexity of this function is O(n^2).

- At the **third level of computation** we first calculate the priority of an adaptation task and then the priority of the corresponding adaptation rule. So the corresponding function follows in the Figure 4.3.

```
 INPUT : Adaptation Rule, Weights, Quality_Attributes, Adaptation Strategy
   OUTPUT : Adaptation Task Priority


   function:getAdaptation Task Priority{
      Quality_AttributesValues = get Quality Attributes Values (Adaptation Rule.Adaptation Task);
      Quality_AttributesUtilities = get Quality Attributes Utilities(Quality_AttributesValues,
                        Quality_Attributes,Adaptation Strategy.Tasks, Adaptation Rule.Adaptation Task);
    sum =0 ;
    for every quality_attribute in Quality_Attributes{
        sum = sum+Quality_AttributesUtilities[quality_attribute] * Weight(quality_attribute);
    }
    return sum;
}
```

Figure 4. 3 : Get Adaptation Task Priority function

With this function, we calculate the priority of an adaptation task. The complexity of this function is analogous with the complexity of the internal functions that are called. Thus, the complexity is equal with O(n^2). Finally, we have the dynamic selection rule algorithm (Figure 4.4) which use the above functions in order to achieve its goal.

```
INPUT :  event (or event pattern),Weights,Quality_Attributes
OUTPUT : most Appropriate Adaptation Rule


function: Adaptation Rule Selection{
    strategy= getAdaptation Strategy(event);
    rules = getAdaptation Rules(strategy);
    Selected_Rule="";
    max_priority = 0;
    for every rule in rules{
        rule_priority = get Adaptation Task Priority(rule.task,Weights,Quality_Attributes,strategy);
        if (rule_priority>=max_priority){
            max_priority= rule_priority;
            Selected_Rule= rule;
        }
    }
    return Selected_Rule;
}
```

Figure 4. 4 : Get Adaptation Rule Priority function

With this function, we find the most appropriate adaptation rule. In this function we call all the previous functions. Nevertheless, an adaptation rule is composed by a set of adaptation tasks. Thus, there is another loop and the complexity is equal to O(n^3). In general, complexity also depends on how records are stored and how we retrieve the data from the records. At this point, there is certainly room for improvement and research.

4.4 Use Cases with Adaptation Rules Priority

At the following subsections we have an adaptation scenario with an adaptation rule selection and then two adaptation numerical examples with the set of computations that have been analyzed in this Chapter.

4.4.1 Adaptation Scenario for the Adaptation Rule selection

The dynamic selection adaptation rule algorithm can be applied to all cases where a triggering event is exported, and the most appropriate adaptation rule should react. There are many scenarios that can be defined on the basis of the use case that we have analyzed in Chapter 2. An indicative example is broken down in the scenario which follows.

We assume that the Monitoring Service has availability problems. To address this issue, event_A is triggered and through the adaptation strategy of the corresponding event, two previously mapped different adaptation rules respond. The first is AdaptationRule_A, which is required to overcome transience by restarting the Monitoring Service. The second is AdaptationRule_B, which can be employed to overcome permanent errors by re-deploying the component on the same VM.

More specifically:

- event_A = down (Monitoring Service)
- Adaptation Rule_A = event_A → Restart (MonitorService)
- Adaptation Rule_B = event_A → Reconfiguration (MonitorService)

At this point, the proposed dynamic adaptation rule selection algorithm is executed, and after having been applied to all the levels of computations of the proposed mathematical formula, arrives at the selection of the adaptation rule demonstrating the highest priority. If we assume that the priority of Adaptation Rule_A is 0.57 and that the priority of Adaptation Rule_B is 0.43, then, the first Adaptation Rule is selected in order to address event_A. More detailed examples of the proposed mathematical formula of the adaptation rule priority value, on all the levels of the computations, follow in the next subsection, as well as the selection of the most appropriate adaptation rule.

4.4.2 Numerical examples with priority computations of Adaptation Rules

In the following examples we focus on priority computations. We don't use just a single specific scenario to apply our calculations, but we define abstract adaptation rules, events and adaptation tasks, all of which can be formed in different scenarios and use cases like the one presented in the previous subsection. In the following computations we assume that the VM cost is 0.01 $ / hour or 0.000027 $ for each single adaptation task, meaning that the mapped services with the corresponding single adaptation tasks are hosted on different instances of the same VM offering.

4.4.2.1 Abstract Scenario 1- single adaptation tasks

Here we assume that we have an event named event_A, and three single adaptation tasks mapped to this event which are SAT1, SAT2 and SAT3. Thus, the corresponding adaptation rules mapping to this triggering event are:

Adaptation Rule_1 = event_A → SAT1
Adaptation Rule_2 = event_A → SAT2
Adaptation Rule_3 = event_A → SAT3

1st level of computations

On the first level of computation we compute the quality attribute values (Section 4.2.1) for each adaptation task separately for each adaptation rule. Thus, it is necessary to compute the metrics received from the recorded histories. It is assumed that the respective histories of the adaptation tasks are as follows:

<u>Adaptation Task SAT1 (Table 4.4)</u>

- This task has been used for the same event three times (Execution Times = 3).
- By the subtraction of the recorded StartExecutionTime by the recorded EndExecutionTime time the indicative execution time records of the task are:
  First execution time: 4 sec.
  Second execution time: 5 sec.
  Third execution time: 6 sec.
- The task was pinged three times during its execution (ping times = 3); in two of them the task was up (up Times = 2).
- In one of the three execution times this single adaptation task resulted in an error. So, the number of faults is equal to one. (Execution Faults = 1).
- The adaptation task was able to address the event successfully in 2 of the three execution times. (SuccessfulExecution = 2).
- Needed VM Resources : 1 VM.

Adaptation Task SAT2 (Table 4.5)

- This task has been used twice for the same triggering event (Execution Times = 2).
- By the subtraction of the recorded StartExecutionTime by the recorded EndExecutionTime time the indicative execution time records of the task are:
  First execution time: 2 sec.
  Second execution time: 1 sec.
- The task was pinged three times during its execution (ping times = 3); and all of them where up. (up Times = 3).
- In all the execution times this single adaptation task has no errors in its result. So, the number of faults is equal to zero. (Execution Faults = 0).
- The adaptation task was able to address the event successfully all the execution times. (SuccessfulExecution = 2)
- Needed VM Resources : 1 VM.

Adaptation Task SAT3 (Table 4.6)

- This task has been used twice for the same triggering event (Execution Times = 2).
- By the subtraction of the recorded StartExecutionTime by the recorded EndExecutionTime time the indicative execution time records of the task are:
  First execution time: 10 sec.
  Second execution time: 15 sec.
- The task was pinged three times during its execution (ping times = 3); and in two of them it was down (upTimes = 1).
- In all the execution times this single adaptation task had 1 error in its results. So, the number of faults is equal to one. (Execution Faults = 1).
- The adaptation task was able to address the event successfully 1 of the 2 execution times (SuccessfulExecution = 1).
- Needed VM Resources : 1 VM.

Quality attributes value computation

- Adaptation Task SAT1

ATTRIBUTE	**VALUE**
Execution	15 / 3 = 5 sec
Availability	2 / 3 = 0.66

Failure Rate	1 / 3 = 0.33
Successability	2 / 3 = 0.66
Cost	1 VM * 0.000027 $ * 5sec = 0.000135 $

Table 4. 4 : Quality Attributes SAT1

- Adaptation Task SAT2

ATTRIBUTE	VALUE
Execution	3 / 2 = 1.5 sec
Availability	3 / 3 = 1
Failure Rate	0/ 2 = 0
Successability	2 / 2 = 1
Cost	1 VM * 0.000027 $ * 1,5 sec = 0.0000405 $

Table 4. 5 : Quality Attributes SAT2

- Adaptation Task SAT3

ATTRIBUTE	VALUE
Execution	25 / 2 = 12.5 sec
Availability	1 / 3 = 0.33
Failure Rate	1/ 2 = 0.5
Successability	1/ 2 = 0.5

Cost	1 VM * 0.000027 $ * 12,5 sec = 0.0003375 $

Table 4. 6 : Quality Attributes SAT3

2nd level of computation

The second level of computation concerns the utility function calculation for each of the quality attributes. Based on the first level of computation, we have the following max and min values for every quality attribute in the context of the same event.

ATTRIBUTE	max value	min value
Execution	12.5 sec	1.5 sec
Availability	1	0.33
Failure Rate	0.5	0
Successability	1	0.5
Cost	0.0003375 $	0.0000405 $

Subsequently we apply the utility function (Section 4.2.2) so as to compute the utility value for each of the quality attributes. An analytical table of the results of the computations follows.

ATTRIBUTE	SAT1 utility	SAT2 utility	SAT3 utility
Execution (negative monotonic)	0,68	1	0
Availability (positively monotonic)	0,49	1	0

Failure Rate (negatively monotonic)	0.34	1	0
Successability (positively monotonic)	0,32	1	0
Cost (negatively monotonic)	0.68	1	0

3rd level of computation
Weights can be defined by the end users in order to give priority to the quality attributes. If a user decides that the availability quality attribute has a higher priority, the corresponding weight would be higher than the others. In our use case we assume that the end user decides that all the quality attributes have the same weight. The number of quality attributes is 5 so the weight of each one is ⅕ = 0.2. At this point we apply the mathematical method of the adaptation task priority (Section 4.2.3) for each of the adaptation tasks.

Priority (SAT1) = 1/5*(0.68+0,2+1+0,32+0.68) = 0,5
Priority (SAT2) = 1/5*(1+1+1+1+1) = 1
Priority (SAT3) = 1/5*(0+0+0+0+0) = 0

Selection Algorithm

Hence, the priority of each SATi equals that of its mapped adaptation rule i. Thus, Adaptation Rule_1 priority is equal to 0.576, Adaptation Rule_2 priority is equal to 0.8, and Adaptation Rule_3 priority is equal to 0. Therefore, Adaptation Rule_2 is better than Adaptation Rule_1 which is better than Adaptation Rule_3. So the best choice for triggering event_A is Adaptation Rule 2.

4.4.2.2 Abstract Scenario 2- single & composite adaptation tasks

We assume that we have three adaptation rules being triggered by the same event event_B. The first one maps to a composite adaptation task which is the CAT1 and contains two single adaptation tasks which are the single adaptation tasks, SAT5 and SAT6. The control flow of the composite adaptation task is sequential. The second adaptation rule maps to a single adaptation task which is the SAT4, and the third to a single adaptation task which is the SAT7.Thus, the corresponding adaptation rules mapping to this triggering event are:

Adaptation Rule_4 = event_B → CAT1(SAT5, SAT6)
Adaptation Rule_5 = event_B → SAT4
Adaptation Rule_6 = event_B → SAT7

1st level of computation
As we have already mentioned in the previous example it is necessary to compute the quality attributes of the metrics received from the recorded histories. It is assumed that the respective histories of the adaptation tasks are as follows:

Adaptation Task CAT1 (Table 4.7)

- This task has been used for the same event twice (Execution Times = 2).
- By the subtraction of the recorded StartExecutionTime by the recorded EndExecutionTime time the indicative execution times records of the task are:
  First execution time: 10 sec.
  Second execution time: 15 sec.
- The task was pinged three times during its execution (ping times = 3); in two of them the task was up (up Times = 2).
- At the first execution SAT5 produces an error as result. So, the number of faults is one (Execution Faults = 1).
- The adaptation task was able to address the event successfully in all the triggering times (SuccessfulExecution = 2).
- Needed VM Resources : 2 VM

Adaptation Task SAT4 (Table 4.8)

- This task has been used twice by the same triggering event (Execution Times = 2).
- By the subtraction of the recorded StartExecutionTime by the recorded EndExecutionTime time the indicative execution times records of the task are:
  First execution time: 7 sec.
  Second execution time: 9 sec.
- The task was pinged three times during its execution (ping times = 3); in two of them the task was up (up Times = 2).
- At the executions produced only one time an error as result. So, the number of faults is one (Execution Faults = 1).
- The adaptation task was able to address the event successfully in all the triggering times. (SuccessfulExecution = 2)
- Needed resources 1 VM.

Adaptation Task SAT7 (Table 4.9)

- This task has been used three times by the same triggering event.
- By the subtraction of the recorded StartExecutionTime by the recorded EndExecutionTime time the indicative execution times records of the task are:
  First execution time: 10 sec.
  Second execution time: 20 sec.
  Third execution time: 30 sec.
- The task was pinged two times during its execution (ping times = 2); in one of them the task was up (up Times = 1).
- At the last execution produced an error as result. So, the number of faults is 1.
- The number of faults is one (Execution Faults = 1).
- The adaptation task was able to address the event successfully only one time (SuccessfulExecution = 1)**.**
- Needed resources 1 VM.

<u>Quality attributes value computation</u>

- Adaptation task CAT1

ATTRIBUTE	VALUE
Execution	25 / 2 = 12.5 sec
Availability	2 / 3 = 0.66
Failure Rate	1 / 2 = 0.5
Successability	2 / 2 = 1
Cost	2 VM * 0.000027 $ * 12.5 sec = 0.000675$

Table 4. 7 : Quality Attributes CAT1

- Adaptation task SAT4

ATTRIBUTE	VALUE
Execution	18 / 2 = 8 sec

Availability	2 / 3 = 0.66
Failure Rate	1 / 2 = 0.5
Successability	2 / 2 = 1
Cost	1 VM * 0.000027 $ * 8 sec = 0.000216 $

Table 4. 8 Quality Attributes SAT4

- Adaptation task SAT7

ATTRIBUTE	VALUE
Execution	60 / 3 = 20 sec
Availability	1 / 2 = 0.5
Failure Rate	1 / 3 = 0.33
Successability	1 / 3 = 0.33
Cost	1 VM * 0.000027 $ * 20 sec = 0.00054 $

Table 4. 9 : Quality Attributes SAT7

2nd level of computation

The second level of computation concerns the utility function calculation (Section 4.2.2) for each of the quality attributes. Based on the first level of computation, we have the following max and min values for every quality attribute in the context of the same event.

ATTRIBUTE	max utility	min utility
Execution	20 sec	8 sec
Availability	0.66	0.5
Failure Rate	0.5	0.33
Successability	1	0.33
Cost	0.000675 $	0.000216 $

Subsequently we apply the utility function (Section 4.2.3) so as to compute the utility value for each of the quality attributes. An analytical table of the results of the computations follows.

ATTRIBUTE	CAT1 utility	SAT4 utility	SAT7 utility
Execution (negative monotonic)	0,63	1	0
Availability (positive monotonic)	1	1	0
Failure Rate (negative monotonic)	0	0	1
Successability (positive monotonic)	1	1	0
Cost (negative monotonic)	0	1	0,29

3rd level of computation

We assume that the end user decides that all the quality attributes have the same weight. The number of quality attributes is 5 so the weight of each one is ⅕ = 0.2. At this point we apply the mathematical method of the adaptation task priority(section) for each of the adaptation tasks.

Priority(CAT1) = ⅕ *(0,63+ 1 + 0 + 1 + 0) = 0,526
Priority(SAT4) = ⅕ *(1+1+0+1+1) = 0,8
Priority(SAT7) = ⅕ *(0+0+1+0+0,29) = 0.258

<u>**Selection Algorithm**</u>

Like the previous example the Adaptation Rule_4 priority is equal with 0.526 the Adaptation Rule_5 priority is equal with 0.8 and the priority of Adaptation Rule_6 is equal with 0.258. Thus, the Adaptation Rule_5 is selected as it has the higher priority.

Chapter 5

5. Related Work

In this chapter we will analyze the related work in scalability rule modelling (Chapter 5.1); adaptation rule modelling (Chapter 5.2) and dynamic adaptation of services or applications in order to maintain a certain service/quality level across different abstraction levels (Chapter 5.3). There will be no analysis over approaches which record Cloud application execution histories as such approaches do not yet exist.

5.1 Scalability Rule Modeling

The proposed adaptation meta-model involves the original scalability part of CAMEL's SRL sub DSL [11]. However, there is a set of approaches that aim to introduce scaling adaptation models with scalability adaptation rules. Most other languages correlate only one single scalability metric with one single scaling action. Some of such languages have been developed in European projects like [15] and [16].

- In [15] is presented a formal Service Definition Language to support service deployment and Automated Service Lifecycle management for service provisioning and dynamic scalability.

- In [16] we have the introduction of a toolkit targeting the Cloud service and infrastructure providers. The innovations behind the toolkit are aimed at optimizing the whole service life cycle, including service construction, deployment, and operation, on a basis of aspects such as trust, risk, eco-efficiency and cost.

Other languages have been developed for use in commercial products (eg AWS, amazon web services). We will then mention some of these:

- A Cloud elasticity language has been proposed in [17] in order to express simple scalability rules. This language includes elements like the scope, the metric condition and sliding window, the scalability limit as well as scaling action details (e.g., scale type). Complex metrics, event patterns and composite scalability rules cannot be expressed by this language.

- The SYBL scalability rule language is a novel language for controlling elasticity in Cloud applications and have been proposed in [18]. CAMEL is more expressive than this language in terms of specifying more complex conditions and complete metric definitions. In CAMEL, the adapted objects have a full reference while SYBL only references the object to be adapted via an identifier.

- Amazon’s CloudFormation [3] is exploited for modelling horizontal scalability policies. The conditions in this language are only related to resource metrics while the scaling actions are only correlated with a pre-configured VM image, that must be manually mapped to the appropriate application component. This makes the situation more difficult for the customers due to the provider lock-in.

Thus, one major characteristic of CAMEL is that it is more expressive than most scalability rule languages because it can define more features for the modeling process. CAMEL's instrument is the ability to fully identify an object and give it a multitude of properties useful for the modeling process. For example, some of CAMEL's offers are the advanced event with the specification of events patterns. This is why CAMEL is not as simple as other modeling languages.

Nevertheless, according to the above, one of the main criteria that makes CAMEL better in modeling of scaling rules is the complexity that modeling objects can have. With the new expansion that has been made in the context of this work, scalability actions are replaced with adaptation tasks. In essence, the adaptation tasks are a superset of scaling actions. At the next section, we mention some adaptation approaches that deal with the modelling of adaptation rules which go beyond the scope of scaling.

5.2 Adaptation Rule Modeling

Most Cloud adaptation modelling approaches are limited to the resource level where resource-related adaptation actions result from computing the difference between the current state and future state of an application model. Previous approaches made toward Cloud application adaptation modelling were virtually non-existent; most focusing only in part on model adaptation rules; the remaining lack the capability to specify adaptation actions on all possible levels and do not enable the modelling of either adaptation workflows or more advanced (or composite) adaptation actions that take the form of adaptation workflows. Based on this analysis, our proposition is ahead of the current state-of-the-art. Nevertheless, there are some related adaptation modeling approaches which we mention in the following.

- In [19] a cross-layer monitoring and adaptation approach of multilayer systems was proposed. In this approach, a language was specified in order to

[3] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

give to the system experts the ability to specify the system layers and their elements. For each system layer, there is a runtime model depicting the current system state. Overall we can see the whole system state as each runtime model covers part of this state as it includes the state of some of the system elements and not all. If a violation occurs, a manual or semi-automatic adaptation takes place and can affect all the model layers.

- In [20] model-based approach for adapting Cloud application topologies was proposed. Such an approach does not directly model adaptation rules. Instead, there are two Open Cloud Computing Interface models, the first implementing the current state topology of the Cloud application and the second the desired state. The adaptation steps are determined after this comparison takes place. This case does not directly model adaptation rules.

- In [21] there is an introduction of a conceptual model for adaptation inside a Cloud environment. This model covers two different types of adaptation. The first one relates to Cloud application specific adaptations and the second to Cloud resource-specific adaptations. The main difference with our approach is that it does not account the dependencies in a cross Cloud environment.

- The adaptation workflow is based on the direct and indirect relations among the Cloud entities. In [22] and [23] an evolution of the models@runtime pattern is presented. In the context of that work adaptation plans are specified as well as a runtime environment to enact them. The adaptation plan specification relies on a novel DSL which enables designing adaptation plans as workflows. In contrast to our meta-model, this DSL is not rich enough to cover the necessary actions in all possible layers while it does not capture all basic (adaptation) workflow control constructs as in our work.

Our modelling has the potential to be more expressive and could be more complete in terms of the scenarios that it can cover. Also it is the only one that supports dynamicity, auto; semi-auto and manual adaptation and used through cross layer and Multi-Cloud environments.

5.3 Priority Computation

At this section we compare approaches which follow the concept of the adaptation of a service or application in order to maintain a certain service/quality level. Thus the actions that should be performed rely on conditions on metrics. These conditions are the cause of the triggering of the execution of an adaptation action. In the context of this work the adaptation rules have a particular structure and the actions that they perform are correlated with tasks which are mapped by services. So the task scheduling process in Cloud computing environments is associated with the computation process in order to find the adaptation rule with the higher priority. In the following we will mention some approaches correlated with the context of this work.

- In [24] is presented the ECMAF, a monitoring and adaptation framework that follows a rule based approach. There are adaptation strategies consisting of event patterns that are mapped to adaptation workflows that can be executed in order to address a problematic situation. Here is used a logic based mining approach [8] to mine adaptation rules with the use of service execution history. This is a cross-layer adaptation approach introduced in [25] which do not cover all the Cloud- based levels.

- In [26] is an approach for the Web Service adaptation and evolution. In this work there is a formulation of some service parameters and their relationship with adaptation behavior of a service based system. Thus, a Fuzzy Inference System (FIS) is adopted for capturing overall QoS and selecting adaptation strategies using fuzzy rules. The overall QoS is computed by the QoS parameters and the efficient selection of the adaptation strategies inferred by overall QoS, importance of QoS and cost of service substitution. This approach has differences with our work. We compute the overall priority value for each adaptation rule and not directly for tasks. Also there are differences in computation formulas.

- In [27] there is an approach for selecting the best possible Cloud service composition that relies on user requirements. The selection of the best possible Cloud service composition affects the provisioning phase, as the more distant from optimality is the selected solution, the more adaptation actions will be enacted. Through this approach we have the optimal composition of different types of Cloud services by simultaneously satisfying various types of user requirements. These types, not concurrently supported by any Cloud application design tool, include quality, deployment, security, placement and cost requirements. The AHP [13] is used in order for the users to participate in the final result by giving the weight to each of the requirements. The main difference with our work is that we compute the priority of adaptation rules and not directly the priority of services or services compositions. Also there are differences at the formulas and the metrics that take part in.

- In the context of the work in [28] we have a Cloud-based architecture for the lifecycle management of the whole Cloud service lifecycle. This architecture also takes into consideration energy-efficiency matters. Special focus is put on intra-layer self-adaptation through the scheduling of adaptation actions over different Cloud layers. Thus, this is achieved through SaaS, PaaS and IaaS intra-layer self-adaptation in isolation. The overall architecture is capable of adapting to meet the energy goals of applications on a per layer basis. In [29] we have the use of an adaptive energy-aware algorithm for maximizing energy efficiency and minimizing the SLA violations rate in Cloud data centers. Actually this algorithm is responsible for calculating the combination of VMs that will lead to a consolidated solution. In our work we specify the quality attributes in order to cover a great variety of metrics for the adaptation rules priority cross the different abstraction levels. This approach focuses mainly on IaaS and PaaS levels of abstraction.

Because there are no approaches with adaptation rules priority computations, we focus on several approaches related with the efficient dynamic task scheduling. As we have mentioned before an adaptation rule is mapped with a service or application by the adaptation task. The similarity with our approach relies on the proposed dynamic selection algorithm of the adaptation rule with the highest priority. All the mathematical formulas which take place in the proposed algorithm can be compared to other works that describe formulas to calculate the best selection of an application, a service or a task

- In [30] we have an introduction to a priority-based queuing model designed to evaluate the services leased by the Cloud service provider. In the queue, general service time and response time for arriving requests and pending requests are stored. The services are considered to be SaaS, PaaS or IaaS and the computations in the Queuing model use a Markovian arrival rate. The proposed analytical model schedules the Cloud services in order to result in maximum profit.

- In [31] is introduced an approach related with the mapping of the Cloud resources with the corresponding tasks in order to process the customer requests. The priority of task execution is a critical issue in the task scheduling process and is computed according to the most important parameters that can meet user requirements. An important aspect is the dynamic computation of the priority value which is adopted in this approach. A Dynamic Priority-Queue (DPQ) approach based on a hybrid multi-criteria decision making (MCDM) and Differential Evolution (DE) is presented. Also a hybrid meta-heuristic algorithm based on Particle Swarm Optimization (PSO) and Simulated Annealing (SA) is introduced. As in the previous approach here we have some similarities in the computation formulas of the most proper task for execution with the dynamic selection of the adaptation rule with the higher priority.

- In [32] an efficient and dynamically scheduling algorithm is proposed. This algorithm combines a set of features in order to provide an efficient allocation of tasks. Therefore, analyzing the impact of the different pricing models on scheduling algorithm will lead to choosing the right pricing model that will not affect the cost. This paper proposes developing a scheduling algorithm that combines these features to provide an efficient mapping of tasks and improve Quality of Service (QoS).

- In [33] we have a Markov decision process model designed to minimize the task scheduling time and optimize load balancing as a scheduling goal. So, actually we have a Cloud workflow scheduling algorithm which incorporates a Markov decision process model and attempts to minimize task scheduling time and optimize the load balancing through the use of reinforcement learning techniques. The set of scheduling schemes is a Pareto optimal solution set, which can select the optimal scheduling scheme according to the user's preference. The most suitable of the schemes is chosen according to the users' preferences.

- In [34] we have the introduction of a new dynamic auto-scaling method that automatically adjusts thresholds depending on the execution environment status observed by advanced multi-level monitoring systems. In this way, multi-level monitoring information that includes both infrastructure and application-specific metrics helps the service providers accomplish satisfactory adaptation mechanisms for the various runtime conditions. The more the dynamicity is enhanced, the greater is the support of the adaptation improvements on both application performance and resource utilization aspects.

There are differences in the techniques that are used for the task scheduling which are identified in computation process of the tasks with the higher priority both in the computational formulas structure and the corresponding quality metrics. According to our approach the adaptation rules are responsible for the triggering of the corresponding adaptation tasks. Thus, by the measurements related to the adaptation task execution we compute the priority of the adaptation rules. Adaptation histories play a fundamental role in the adaptation rules priority. Although our approach is considered as state of the art as far as it concerns the historical records of tasks executions, we can say that some task scheduling approaches have structure similarities (eg ECMAF). Another important aspect in our work is that the users can decide on the weights of each of the quality attributes needed for the priority computation. In such cases like ours, approaches like AHP can be valuable.

Chapter 6

6. Conclusion and Future Work

6.1 Conclusion

In the context of this work, we have created two extensions of the CAMEL language mapping to two of its meta-models, the adaptation and the execution. In the extension for CAMEL’s adaptation meta-model we introduce adaptation tasks, adaptation rules and strategies. Adaptation rules match an event or event pattern, representing an occurrence of a critical situation, with adaptation workflows, which specify the concrete adaptation actions to be performed for addressing this situation, while adaptation strategies are necessary both for organizing the set of adaptation rules in the context of the same event or event pattern that triggers them, and for representing the application’s adaptive behavior. The extension for CAMEL’s execution meta-model was introduced in order to capture and record the adaptation history of Multi-Cloud applications. An adaptation history of each application recorded particular sensor measurements which are exploited for the computation of the quality attributes that participate in the priority formula to be calculated and come from previous executions of the adaptation rules selected. Thus, the captured information was used in order to derive important knowledge useful for the future use of adaptation actions. This provided for the selection of the most appropriate adaptation rule according to the current problematic situation expressed in the form of an event (pattern). Thus, this selection relies on the computation of the adaptation rule priority through the use of a mathematical formula. Finally, we introduce a dynamic selection algorithm needed for the selection of the adaptation rule with the higher priority.

The goal of this thesis is to optimize the adaptation of the applications across multiple Clouds and different abstraction levels. All the introduced elements of the CAMEL's extensions help in order to achieve this goal. The workflows are specified in language-agnostic manner. Language-agnostic specification is a real benefit if it can be assorted with the transformation logic into the language of the workflow engine to be exploited for the enactment of the workflows specified. Apart from advanced adaptation rules, we have the coverage of multiple levels, the grouping of adaptation rules and the language- & implementation-agnostic specification of workflows that concerns the fact that we do not restrict the adaptation tasks to their realizations leaving it free for the adaptation system to choose the best possible realization at runtime.

As a result of the above analysis we conclude that with the new extensions of the CAMEL modeling language we optimize the management of the applications in Multi-Cloud environments. The language became more expressive and supports both of the cross layer adaptation to all the levels of the Cloud (IaaS, PaaS, SaaS, WaaS), and the dynamic selection of the adaptation rule with the highest priority.

6.2 Future Work

The main drawback of the current work is that it is not validated under normal conditions. Also there is a room for improvement of the modelling of the participated classes and the optimization of the dynamic selection algorithm for adaptation rules. This will supply directions which attempt to resolve these drawbacks:

- The proposed extensions could be validated by different use cases. The use cases which have been analyzed are generic in order to demonstrate the usage of the content of this work. The adaptation rule strategies and histories could be analyzed by more complex use cases under normal conditions and for a set of connections between Cloud Providers and Clients. This approach can describe workflows of adaptation actions that correspond to real SLO violations.
- The main worry of the dynamic selection algorithm is if it works properly as expected and if it takes the right decisions. The adaptation rule priority selection formula could thus be validated with real-time running services, and evaluated through resultant performance and suitability of the chosen adaptation rule.
- In addition, a more specialized approach to the performance metrics and the mathematical formula used for the computation of priorities could be made. More measurements must be carried out so that the result can be more accurate. This would result in the optimization of the proposed algorithm of the dynamic selection of adaptation rules.
- A great margin of improvement exists in the class of Component itself, so that more specialized component items can be created e.g the Container component. This will increase the expressivity of the language so that it can represent a wide variety of features.

Bibliography

[1] Mell, P. & Grance, T. (2011). The NIST definition of cloud computing.

[2] Puthanl, D., Sahooy, B.P.S., Mishraz, S. & Swainz, S. (2015) Cloud Computing Features, Issues and Challenges:A Big Picture. In IEEE International Conference on Computational Intelligence and Networks, pp. 7-18. Doi**:** 10.1109/CINE.2015.31

[3] Zhang, Q., Cheng, L. & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. In University of Waterloo, Waterloo, Canada, pp. 7-18. Doi: 10.1007/s13174-010-0007-6

[4] Zeginis, C. & Plexousakis, D. (2010). Monitoring the QoS of Web Services using SLAs. In Institute of Computer Science, Heraklion, Crete, Greece.

[5] Zeginis, C. (2009). Monitoring the QoS of Web services using SLAs – Computing metrics for composed services. Master Thesis, Greece, Heraklion, March 2009.

[6] Kritikos, K., Zeginis, C., Griesinger, F., Seybold, D. & Domaschka, J. (2017). A Cross-Layer BPaaS Adaptation Framework. In FiCloud, Prague, Czech Republic. In IEEE Computer Society, pp. 241–248. Doi: 10.1109/FiCloud.2017.12

[7] Zeginis, C., Kritikos, K., Garefalakis, P., Konsolaki, K., Magoutis, K. & Plexousakis, D. (2013). Towards Cross-Layer Monitoring of Multi-Cloud Service-Based Applications. In: Lau KK., Lamersdorf W., Pimentel E. (eds) Service-Oriented and Cloud Computing. ESOCC 2013. Lecture Notes in Computer Science, vol 8135. Springer, Berlin, Heidelberg. Doi: 10.1007/978-3-642-40651-5_1610.

[8] Zeginis C., Kritikos K. & Plexousakis D. (2014). Event Pattern Discovery for Cross-Layer Adaptation of Multi-cloud Applications. In: Villari M., Zimmermann W., Lau KK. (eds) Service-Oriented and Cloud Computing. ESOCC 2014. Lecture Notes in Computer Science, vol 8745. Springer, Berlin, Heidelberg.

[9] Rossini, A. (2015). Cloud Application Modelling and Execution Language

(CAMEL) and the PaaSage Workflow. Conference: ESOCC 2015: 4th European Conference on Service-Oriented and Cloud Computing, At Taormina, Italy, Volume: CCIS, volume 567, pp. 437-439.

[10] Bsila, A., Ferry, N., Horn, G., Kirkham T., Malawski, M., Parlavantzas, N., Pérez, C., Rouzaud-Cornabas, J., Rossini, A., Romero, D., Rossini, A., Solberg, A. & Song, H. (2014). PaaSage: Model Based Cloud Platform Upperware.

[11] Kritikos, K., Domaschka, J. & Rossini, A. (2014). SRL: A Scalability Rule Language for Multi-cloud Environments. In Conference: IEEE 6th International Conference on Cloud Computing Technology and Science. Doi: 10.1109/CloudCom.2014.170

[12] Kritikos, K., Magoutis, K., & Plexousakis, D. (2016). Towards Knowledge-Based Assisted IaaS Selection. In IEEE International Conference on Cloud Computing Technology and Science (CloudCom). Doi: 10.1109/CloudCom.2016.0073

[13] Saaty, T. (1980). Analytic Hierarchy Process. McGraw-Hill, New York.

[14] Hwang, C. & Yoon, K. (1981). Multiple Criteria Decision Making. Lect. Notes Econ. Math., 1981.

[15] Gal´an, F., Vaquero, L. M., Clayman, S., Toffetti, G., & Henriksson, D. (2009). Deliverable D4.1, D4.2 and D4.3 – Scientific Report. Reservoir project deliverable.

[16] Rumpl, A., Rasheed, H., Waeldrich, O., & Ziegler, W. (2010). Service Manifest: Scientific Report. Optimis project deliverable.

[17] Moore, L. R., Bean, K., & Ellahi, T. (2013). A Coordinated Reactive and Predictive Approach to Cloud. Elasticity. In CLOUD COMPUTING. IARIA..

[18] Copil, G., Moldovan, D., Truong, H. L., & Dustdar, S. (2013). SYBL: An Extensible Language for Controlling Elasticity in Cloud Applications. In CCGrid, pp. 112–119. IEEE Computer Society. Doi: 10.1109/CCGrid.2013.42

[19] Song, H., Raj, A., Hajebi, S., Clarke, A., & Clarke,S. (2013). Model-based cross-layer monitoring and adaptation of multilayer systems. Science China Information Sciences, volume 56(8), pp. 1–15.

[20] Erbel, J. M., Korte, F., & Grabowski, J. (2018). Comparison and Runtime Adaptation of Cloud Application Topologies based on OCCI. In CLOSER.

[21] Marquezan, C. C., Wessling, F., Metzger, A., Pohl, K.,Woods, C., & Wallbom, K. (2014). Towards exploiting the full adaptation potential of cloud applications. In PESOS, Proceedings of the 6th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems, pp. 48-57. Doi: 10.1145/2593793.2593799

[22] Lushpenko, M., Ferry, N., Song, H., Chauvel, F., & Solberg, A. (2015). Using adaptation plans to control the behavior of models@runtime. CEUR Workshop Proceedings, volume 1474, pp. 11–20.

[23] Blair, G., Bencomo, N., & France, R. B. (2009). Models@run.time. IEEE Computer Society Press Los Alamitos, CA, USA, volume 42(10), pp 22–27. Doi: 10.1109/MC.2009.326

[24] Zeginis, C., Konsolaki, K., Kritikos, K. & Plexousakis, D. (2011). ECMAF: an event-based cross-layer service monitoring and adaptation framework. In ICSOC Workshops, ser. Lecture Notes in Computer Science, vol. 7221, Paphos, Cyprus: Springer, pp. 147–161. Doi: 10.1007/978-3-642-31875-7_15

[25] Zeginis, C. (2014). Cross - layer monitoring and adaptation of multi - cloud service - based applications. Dissertation, Greece, Heraklion, October 2014.

[26] Pernici, B. & Siadat, H. (2011). Selection of Service Adaptation Strategies Based on Fuzzy Logic. In IEEE World Congress on Services, Washington, DC, USA. Doi: 10.1109/SERVICES.2011.33.

[27] Kritikos, K. & Plexousakis, D. (2015). Multi-cloud Application Design through Cloud Service Composition. In Conference: IEEE 8th International Conference on Cloud Computing, 2015. Doi: 10.1109/CLOUD.2015.96

[28] Djemame, K, Kavanagh, R , Armstrong, D et al. (6 more authors). (2017). Energy Efficiency Support through Intra-Layer Cloud Stack Adaptation. In: Lecture Notes in Computer Science. 13th International Conference on Economics of Grids, Clouds, Systems and Services (GECON 2016), 20-22 Sep 2016, Athens, Greece. Springer Verlag , pp. 129–143. Doi: 10.1007/978-3-319-61920-0_10

[29] Djemame, K., Bosch, R., Kavanagh, R., Alvarez, P., Ejarque, J., Guitart, J. & Blasi, L. (2017). Paas- IaaS Inter Layer Adaptation in Energy aware Cloud Environment. In: IEEE Transactions on Sustainable Computing, Vol: 2, pp. 127-139. Doi: 10.1109/TSUSC.2017.2719159

[30] Jaiganesh, M., Ramadoss, B., Vincent, Antony, Kumar, A. & Mercy, S. (2015). Performance Evaluation of Cloud Services with Profit Optimization. Department of Information Technology, PSNA College of Engg. and Tech, Dindigul, Tamilnadu, India. Doi: 10.1016/j.procs.2015.06.003

[31] Ben, Alla, H., Ben, Alla, S. & Ezzati, A. (2017). A Priority Based Task Scheduling in Cloud Computing Using a Hybrid MCDM Model. In: Sabir E., García Armada A., Ghogho M., Debbah M. (eds) Ubiquitous Networking. UNet 2017. Lecture Notes in Computer Science, vol 10542. Springer, Cham. Doi: 10.1007/978-3-319-68179-5_21

[32] Almezeini, N. & Harez, A. An Enhanced Workflow Scheduling Algorithm in Cloud Computing. In CLOSER 2016 - 6th International Conference on Cloud Computing and Services Science, King Saud University, Riyadh, Saudi Arabia.

[33] Jiahao W., Zhiping P., Delong C., Qirui L., Jieguang H. (2018). A Multi-object Optimization Cloud Workflow Scheduling Algorithm Based on Reinforcement Learning. In: Huang DS., Jo KH., Zhang XL. (eds) Intelligent Computing Theories and Application. ICIC 2018. Lecture Notes in Computer Science, vol 10955. Springer, Cham. Doi: 10.1007/978-3-319-95933-7_64

[34] Taherizadeh, S. & Stankovski, V. (2018). Dynamic Multi-level Auto-scaling Rules for Containerized Applications. The Computer Journal, Volume 62, Issue 2, pp. 174–197. Doi: 10.1093/comjnl/bxy043