Automatic, multi-grained elasticity-provisioning for the Cloud

Decision Process for On-demand Elasticity Report

Deliverable no.:5.1
Date: 28-05-2013

Celar is funded by the European Commission DG-INFSO
Seventh Framework Programme, Contract no.: 317790

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CPU	Computer Processing Unit
ID	Identifier
I/O	Input/ Output
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
TOSCA	Topology and Orchestration Specification for Cloud Applications
SYBL	Simple Yet Beautiful Language
VM	Virtual Machine
WP	Work Package
XML	Extensible Markup Language

Deliverable Title	**Decision Process for On-demand Elasticity Report**
Deliverable No.	**5.1**
Filename	CELAR_d5.1_finalrelease.docx
Author(s)	Georgiana Copil, Daniel Moldovan, Hung Duc Le, Hong-Linh Truong, Schahram Dustdar
Date	28.05.2013

Start of the project: 1.10.2013
Duration: 36 Months
Project coordinator organization: ATHENA RESEARCH AND INNOVATION CENTER IN INFORMATION COMMUNICATION & KNOWLEDGE TECHNOLOGIES (ATHENA)

Deliverable title: Decision Process for On-demand Elasticity Report
Deliverable no.: D5.1

Due date of deliverable: 31 May 2013
Actual submission date: 28 May 2013

Dissemination Level

X	PU	Public
	PP	Restricted to other programme participants (including the Commission Services)
	RE	Restricted to a group specified by the consortium (including the Commission Services)
	CO	Confidential, only for members of the consortium (including the Commission Services)

Deliverable status version control

Version	Date	Author
0.1	2 May 2013	Georgiana Copil, Daniel Moldovan, Hung Duc Le, Hong-Linh Truong, Schahram Dustdar
0.2	8 May 2013	Georgiana Copil, Daniel Moldovan, Hung Duc Le, Hong-Linh Truong, Schahram Dustdar
0.3	22 May 2013	Georgiana Copil, Daniel Moldovan, Hung Duc Le, Hong-Linh Truong, Schahram Dustdar
1.0	28 May 2013	Georgiana Copil, Daniel Moldovan, Hung Duc Le, Hong-Linh Truong, Schahram Dustdar

Abstract

This document presents an overview of the Decision Module. First we present elasticity decision requirements for applications supported by CELAR. We then analyze the Decision Module in detail, covering application structure, inputs and outputs, and runtime functionalities expected for elasticity decision making. Based on that, the architecture of the decision module is presented, describing various types of interaction among decision module components and between decision module components and other CELAR modules. We focus on essential parts like granularity of the decisions, mechanisms used for decision or the application model considered.

Keywords

Elastic adaptation, decision, elasticity, cloud computing, elastic control

1 Introduction

Cloud computing elasticity in its widely-cited definition given by the National Institute for Standards and Technology [NIST 2011] outlines "on-demand self-service" (automatic, without the need of human interaction) and "rapid elasticity" as being two of the five essential characteristics. Although on-demand provisioning of resources in a pay-as-you-go manner facilitates greatly the accomplishment of these two characteristics, the cloud application elasticity is still not well supported by contemporary techniques, and there are many operations related to elasticity when deploying or executing cloud-based applications (i.e. application configurations, and platform configurations depending on the application).

The CELAR platform proposes multi-perspective elastic provisioning for cloud applications, through both smart deployment and elastic adaptation of the application in the cloud computing environment, considering the user's application description and the requirements regarding the application. As shown in the architecture of the CELAR System in Figure 1 (in the Deliverable D1.1 [D1.1]), the CELAR System has three main parts: Application Management Platform, Cloud Information and Performance Monitor, and Elasticity Platform. The Elasticity Platform contains the Decision Module, the focus of WP 5, of which the initial architecture and decision processes are being detailed in this deliverable. Within the Application Management Platform, the application user[1] (1) describes his/her application structure at a high-level view, the elasticity "actions"/ "strategies" (i.e. scaling or configuration actions) that are possible within the described application structure and the elasticity requirements, and (2) submits the application for execution. The application is then deployed on the cloud and continuously controlled by the Elasticity Platform and monitored by the Cloud Information and Performance Monitor.

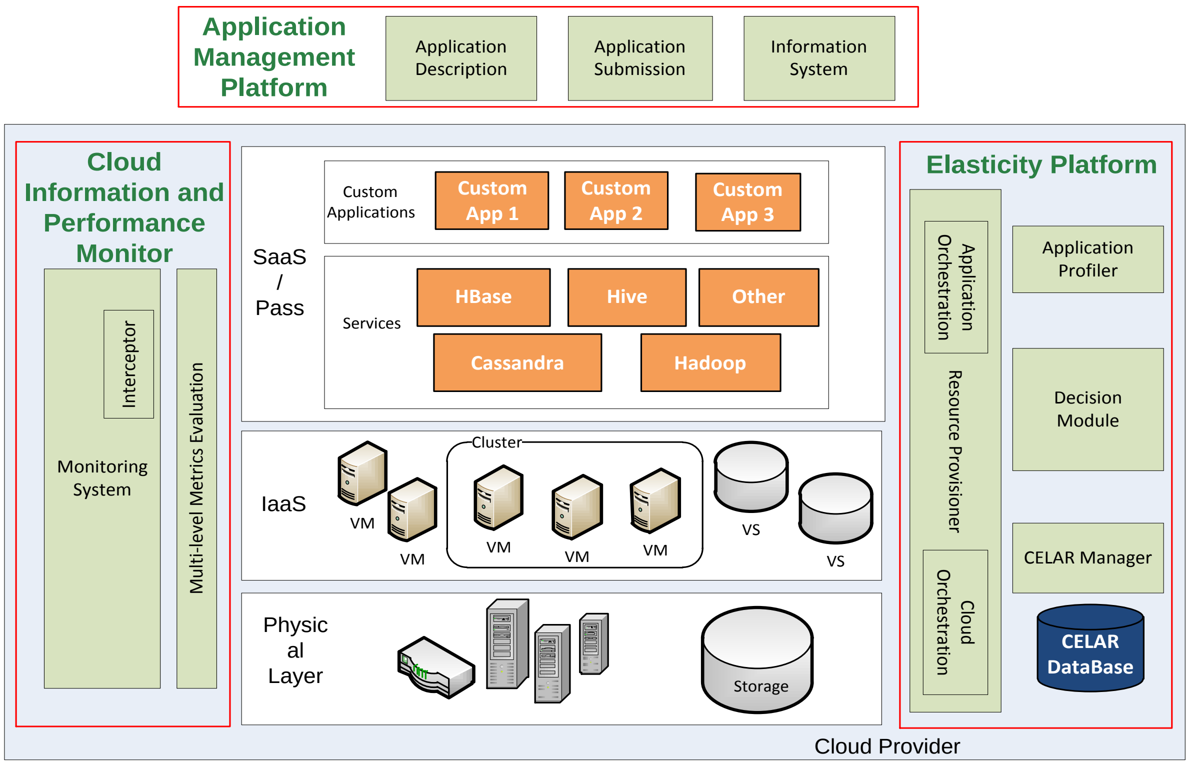


Figure 1: CELAR System Architecture

[1] For deliverables consistency, we will refer to the user using CELAR for elasticizing his/her application as "application user" in accordance with Deliverable D1.1 Section 2.3.

The requirements of cloud application elasticity imply that we need to support the automatic decision on adaptation measures for fulfilling the application user's elasticity requirements considering different properties like quality, load characteristics, estimated cost, and the measure in which the user wants to focus on. Currently, for application auto-adaptation the user is usually required to describe low-level, detailed requirements are too complex, although his/her major interest is usually the quality s/he expects and the cost s/he is willing to pay for the expected quality. Controlling cloud application elasticity in the contemporary view (i.e. the automatic scalability of resources) has been targeted by both research and industry. Several authors propose controllers for the automatic scalability/elasticity of entire applications [Yang 2009], just parts of the application (i.e. application data-end) [Lim 2010], or specific types of applications [Malkowski 2011]. Cloud providers offer tools for automatic scalability like Amazon's AutoScale [AutoScale], Paraleap AzureWatch for Windows Azure [AzureWatch] or IBM's SmartCloud [SmartCloud] initiative, automatically scaling resources depending on what the user specifies through low-level resource targeted policies. However, these approaches do not control the application on multiple levels taking into consideration the complex application structure, or the fact that elasticity is a complex multi-dimensional issue [Dustdar 2011].

The Decision Module is the core module of the Elasticity Platform and uses information received from all the other modules for deciding smart deployment of a newly received application and elastic control of deployed elastic applications. More importantly, the Decision Module uses the novel multi-dimensional view on elasticity which targets scaling the quality and cost of the applications rather than just resources, as shown in recent studies [Dustdar 2011].

The Decision Module takes real-time decisions for application adaptation to meet user elasticity requirements. The Decision Module facilitates an automatic adaptation process of the application to "outside" stimuli (e.g. workload, increasing cost or decreasing quality) without the need of user intervention, using the received description of application elasticity capabilities which show what are the actions available for the application adaptation and which structural part can be adapted. Moreover, not only real-time adaptation decisions are enforced but also smart deployment of the application, considering cloud providers services and estimated cost with respect to quality and performance. For generating control decisions considering the complex information on elasticity metrics and the complex application structure and run-time topology, we use an internal application model which aggregates structural and run-time data. We describe in more detail the research challenges and progress towards finding solutions in Section 4.

1.1 Purpose of this document

The aim of this document is to present an initial analysis of the decision process for on-demand elasticity, and to define the architecture of the Decision Module, the core component of the Elasticity Platform and the focus of WP5. We define Decision Module's functional and non-functional requirements, and describe its internal components and show initial solutions for constructing a generic, flexible and efficient Decision Module.

1.2 Document structure and relevant publications

The rest of the document is structured as follows: Section 2 contains the state of the art of elasticity control for cloud applications. Section 3 contains an analysis of the Decision Module actors, the use-cases driven by these actors, and the functional and non-functional requirements of the Decision Module. Section 4 focuses on the Decision

Module analysis, discussing issues that must be addressed for designing a fine-grained elasticity control component. In Section 5 we propose the Decision Module architecture considering the issues discussed in the analysis section. Section 6 gives short implementation guidelines and the last Section concludes the deliverable and outlines our future work.
This report is partially based on the works in [Copil 2013a], [Gambi 2013], [Copil 2013b].

2 State of the Art in Elasticity Control

Elasticity control of cloud applications is an important step towards fully benefiting from the on-demand, pay-per-use cloud computing model.

Schatzberg et al. [Schatzberg 2012] raise issues that appear in cloud elasticity control and outline that in cloud computing elasticity is an important area of research, which will facilitate the development of applications that would fully benefit from the advantages of cloud computing and from on-demand resources allocation.

The different perspectives of cloud services performance/cost/quality measurement are outlined by Li et al. [Li 2012] who proposes a list of categories of metrics which are used for evaluating cloud services. Their retrieved cloud service evaluation metrics are scattered over three aspects of cloud services: economics having as subdimensions cost and elasticity evaluation metrics, performance with subdimensions communication, computation, memory, storage evaluation metrics and security evaluation metrics. The abstract metrics are associated to measurable metrics for easier grasp of reality and for being able to actually compute the abstract metrics.

In what follows we present the existing research for cloud application elasticity control. The first section targets resource-oriented elasticity control, while in the second section we investigate research works focused either on quality or on cost in the cloud application scaling system design. Next, we present research that considered different application types (patterns) and research works targeting application analysis, related to our approach which targets application behavior analysis for different application types.

2.1 Resource elasticity control

Resource allocation and reallocation for cloud applications has been targeted in numerous research works. In contrast to this, we also consider quality and cost as dimensions of elasticity.

Moore et al. [Moore 2011] propose an inflate/deflate programming model, integrated within the application for coordinating with a central controller and deciding on whether the application should be provisioned more or less resources. Polo et al. [Polo 2011] present a resource-aware scheduling technique for MapReduce multi-job workloads, improving resource utilization across machines by observing completion time goals and leveraging job profiling information to dynamically adjust the number of slots on each machine and the workload placement across them. You et al. [You 2011] highlight that existing distributed storage solutions need to be able to scale even when containing large data sets, and describe Ursa, a scalable data management middleware system which scales to a large number of storage nodes and objects and aims to minimize latency and bandwidth costs during system reconfiguration, employing selection of data objects from hot-spot servers and topology-aware migration to minimize reconfiguration costs. Lim et al. [Lim 2010] propose an automatic controller targeting the application storage tier, also considering actuation delays, rebalancing and other problems which appear in this case. The controller not only decides on scaling horizontally at infrastructure level, but also handles rebalancing the data and synchronizing resource scaling with software adaptation by considering the lags introduced by each.

Lama et al. [Lama 2012] define AROMA, a system for automated resource allocation and configuration of Hadoop parameters in clouds, achieving quality of service goals while minimizing the incurred cost. The authors consider that jobs with similar resource consumption patterns face similar bottlenecks, and create clusters of similar jobs and

employ a support vector machine (SVM) model for predicting a job's performance for various combinations of resources, configuration parameters and inputs. Lee et all [Lee 2011] propose a data-analytics oriented cloud architecture for allocating resources to data analytics applications in the cloud, proposing a resource sharing approach that achieves high performance and fairness by evaluating how much progress each analytics job is making with the assigned resources and allocating new resources based on this evaluation.
Xiong et al. [Xiong 2011] propose a multi-level control approach for N-tier web applications, which are modeled as a tandem queues and controlled at application and container level for achieving the best resource allocation. Wang et al. [Wang 2011] show that software resource allocation (e.g. server thread pool size) impacts considerably system performance, both in under-allocation and over-allocation case.

2.2 Cost and quality in elasticity control

Elasticity has been outlined to be a complex property [Dustdar 2011] composed of quality elasticity, cost elasticity and resources elasticity. This section shows research on elasticity control which also considers cost and quality in the control process.
Truong et al. [Truong 2010] estimate the cost of application hosting on the cloud considering different sub-costs which may interfere during the lifetime of the application. Villegas et al. [Villegas 2012] propose a framework for conducting empirical research in different IaaS clouds, comparing different allocation and provisioning policies. The authors emphasize the importance of understanding the performance and cost associated with different provisioning or allocation policies, for being able to properly manage their application's workloads.
Gonzalez et al. [Gonzalez 2012] propose cloud infrastructure-level virtual machine management for increasing the VM availability. The authors also provide a study on how different properties of the cloud infrastructure affect the VM availability. Xu et al. [Xu 2010] highlight the complexity of selecting cloud services with lowest associated cost, especially when periodical discounts are offered by cloud providers, and present a service selection algorithm that takes into account time-sensitive intra and inter provider discounts, minimizing the services cost. Chaisiri et al. [Chaisiri 2012] focus on the complexity of selecting cloud services under different provisioning plans, such as reservation and on-demand, defining an optimal cloud resource provisioning algorithm that can provision resources used in multiple provisioning. Using deterministic equivalent formulation, sample-average approximation, and Benders decomposition, their proposed solution minimizes the total cost of resource provisioning in cloud computing environments.
In our work we will target the control of elasticity as a complex property composed of quality, cost and resources in contrast with works presented above which consider either quality or cost. We also construct a model for anticipating how quality, resources and cost affect each other.

2.3 Application-patterns in elasticity control

Patterns for cloud applications have begun being studied only recently. Fehling et al. [Fehling 2012] describe cloud application patterns and their associated abstract management flows for defining best practice rules corresponding to the different application patterns. They outline the fact that, for taking advantage of cloud computing properties, one needs to change the architectural principles of applications and focus more on queuing-based applications. Moreover, the applications need to be designed for

using as much as possible of the services offered by PaaS providers, and structured for enabling scalability on various application parts.
Although the majority of research works in cloud application control propose controlling only specific types of applications, several research works consider more application types for cloud application control. Calheiros et al. [Calheiros 2012] propose Aneka, a platform for facilitating the development and deployment of scalable applications on the cloud, supporting several application models. Aneka provides support for hybrid clouds and promotes the integration between desktop grids and clouds. Aneka also provides dynamic resource provisioning considering the jobs type, size and intensity and the cost. Juve et al. [Juve 2011] propose a system for automating the provisioning process for cloud-based applications. They consider two application models, one workflow application and one data storage case, and show how for these cases the applications can be deployed and configured automatically.
In our approach we argue that application patterns are contributing factors to application behavior. For supporting elasticity we will consider different application patterns and learn their impact on application behavior under different stress factors.

2.4 Cloud application elasticity behavior analysis

In cloud application control, application behavior in response to external stress factors such as workloads or actions enforced on the application is an important subject since it gives knowledge on the system that needs to be controlled. We now present several approaches on the analysis and prediction of cloud application behavior.
Marian et al. [Marian 2012] highlight that system monitoring is key to understanding system behavior, task which is increasingly complex in large scale system deployments, like the present-day, and provide a mechanism that extracts indexable kernel-level system descriptions, or signatures, which accurately capture the state of a system at a point in time, over which statistical analysis can be performed to analyze system behavior. Verma et al. [Verma 2010] analyze the impact of reconfiguration actions on the system performance. They observe that the live migration is affected by the CPU usage of the source virtual machine, both in terms of the migration duration and application performance. The authors conclude with a list of recommendations on dynamic resource allocation. Lu et al. [Lu 2011] profile physical resource utilization information of Virtual Machines (VMs), employing a directed factor graph to model the multivariate dependence relationships among different resources (CPU, memory, disk, network) across virtual and physical layers, improving the accuracy of the resource utilization information collected within guest VMs. Singh et al. [Singh 2011] propose Predico, a system able to predict the impact of workload changes on the behavior of data center applications, using a network of queues to analytically model the behavior of large distributed applications, and apply their approach to predict resource utilization and latency for two large applications. Wang et al. [Wang 2012] focus on determining performance fluctuations in n-tier systems especially during bursting workloads, fluctuations that can be generated or amplified by various factors, such as cache misses, garbage collection and inefficient scheduling policies.
The PRESS framework proposed by Gong et al. [Gong 2010] extracts patterns in application demands and considers these patterns for automatic resource allocation. A comparison with other algorithms shows that the proposed prediction algorithm has comparably less under- and over-allocation errors. Shen et al. [Shen 2011] propose the CloudScale framework which uses resource prediction for automating resource allocation according to service level objectives (SLOs) with minimum cost. Based on

resource allocation prediction, CloudScale uses predictive migration for solving scaling conflicts (i.e. there are not enough resources for accommodating scale-up requirements) and CPU voltage and frequency for saving energy with minimum SLOs impact.
Didona et al. [Didona 2012] introduced Transactional Auto Scaler, a system designed to accurately predict the performance of applications running on top of transactional in-memory data grids during scaling operations which employs a hybrid forecasting methodology relying on analytical modeling and machine learning. Kundu et al. [Kundu 2012] propose two algorithms, based on artificial neural network and support vector machines for constructing an application model correlating resource allocation with the performance obtained with the respective resource allocation. Matsunaga et al. [Matsunaga 2010] extend an existing classification tree algorithm called Predicting Query Runtime (PQR) for predicting spatiotemporal utilization of resources for cloud applications. The authors emphasize that the attributes chosen for the learning process play a considerable role in the results obtained, and that system performance attributes are mostly important for predicting execution time while application performance attributes are of great importance for all the considered scenarios.
As opposed to discussed research work on this area, we propose using the multiple level monitoring information and the application architectural and deployment patterns for constructing a dependency model which would reflect the application behavior under different external stress factors like application workloads or enforced actions. Using information contained in this dependency model, we control the application elasticity from the three main perspectives, quality cost and resources, and on multiple levels according to the application structure.

3 Decision Module Requirements

The Decision Module is a core part of the Elasticity Provisioning Platform which determines elasticity-enabling decisions for adapting the application to the current workload considering the application user's elasticity requirements and application elasticity behavior from cost, quality and resource perspectives. Considering the structure of the application, the Decision Module finds action plans for the smart deployment of an application or for elastic application adaptation. The Decision Module uses information from the CELAR DataBase and monitoring information from the Cloud Information and Performance Monitor, interacts with the CELAR Manager for receiving tasks coming from the other modules or for sending generated elasticity action plans to the other modules. These modules are part of the Elasticity Platform and are described in detail in Deliverable D3.1.

Similar to the use cases presented in the CELAR Deliverable D1.1 [D1.1] and Deliverable D3.1 [D3.1], in this Section we present the Use Cases for the Decision Module. The first part of this Section presents the main actors that interact with the Decision Module, the next part focusing on describing how the defined actors interact with the Decision Module. After introducing these concepts we describe functional and non-functional requirements for the Decision Module.

3.1 Actors

The Decision Module interacts with various components of the CELAR System, which are regarded as actors. Among them, two main actors are the Cloud Information and Performance Monitor and the CELAR Manager:

- *Cloud Information and Performance Monitor*: the Decision Module's decision process can take place either in time intervals or be event driven. For the event driven case, the event is triggered by the Cloud Information and Performance Monitor when out-of-the-ordinary behavior is detected on the monitored metrics (i.e. a sudden spike on the user's request rate).
- *CELAR Manager*: this actor is essential for Decision Module interaction and coordination with the other CELAR modules. The CELAR Manager sends to the Decision Module the message that a new application needs to be deployed, together with the identification number of the application (used to identify the application in the CELAR DataBase), for the Decision Module to find a smart deployment plan and send it back to the CELAR Manager together with an estimate on the costs. Throughout the application's execution, the Decision Module checks the fulfillment of application user's elasticity requirements and in case of elasticity requirements violation, the Decision Module generates an elasticity adaptation plan and sends it to the CELAR Manager together with application elasticity analysis and estimated cost. The CELAR Manager forwards the plan to the Resource Provisioner for enforcement and the adaptation plan, together with the application elasticity analysis and estimated costs to the Application Management Platform for informing the application user with regard to the status of his/her application.

3.2 Use cases

The following table shows the list of the Decision Module Use Cases.

Table 1: Decision Module Use Cases

Nb.	Use case name	Actor	Description
1	Trigger New Deployment Event	CELAR Manager	When a smart deployment strategy is needed, the CELAR Manager triggers a "`new application deployment needed`" event, and piggybacks the ID of the new application. The information is then taken from the CELAR Database (use case 2).
2	Trigger New Decision Cycle	Cloud Information and Performance Monitor	The Cloud Information and Performance Monitor sends an event to the Decision Module whenever it detects an abnormal application behavior reflected in the monitored metrics.
3	Get Monitoring Information	Cloud Information and Performance Monitor	The Decision Module gets fresh monitoring information from the Cloud Information and Performance Monitor either through a push or pull manner. This use case is the case in which Decision Module **pulls** fresh information from the Cloud Information and Performance Monitor.

The above use cases are presented in Figure 2 through a UML Diagram.

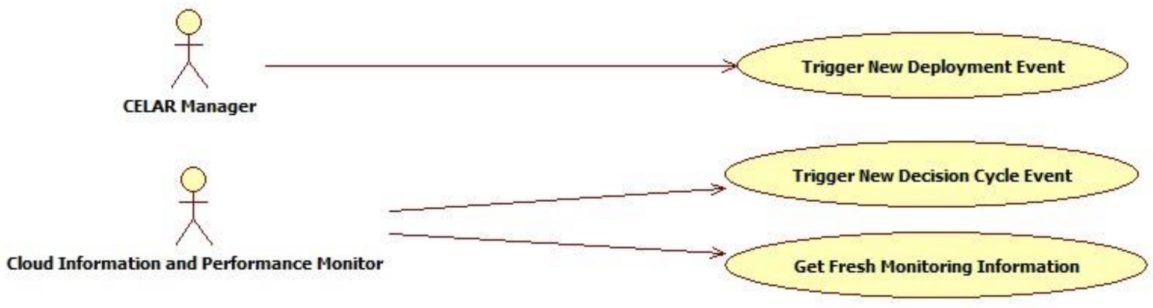

CELAR Manager
Trigger New Deployment Event
Cloud Information and Performance Monitor
Trigger New Decision Cycle Event
Get Fresh Monitoring Information

Figure 2: UML Diagram of the Decision Module Use Cases

3.3 Functional requirements

In this section we present the functional requirements of the Decision Module that derive from the above use cases and from the more generic functional requirements of the Elasticity Platform from Deliverable D3.1 [D3.1].

FR1 -- Structure monitoring information:
Monitoring data retrieved from different monitoring sources at different levels, from system to application level need to be mapped on the application structure with respect to the application vertical software components stack and their run-time topology, enabling fine grained analysis of application behavior.

FR2 -- Generate mappings between low level and application-level metrics:
The majority of existing monitoring solutions focus on system-level metrics, such as CPU utilization, which, while important, do not fully capture the actual behavior of cloud applications and cannot easily be mapped to application level metrics such as end user

experience. As cloud application users are usually interested only in end-user experience, a mechanism for defining application level metrics as a hierarchical composition of lower level metrics needs to be defined. The complex composite metrics are added to the already structured monitoring information, providing a mechanism which supports tracing of high-level metrics to underlying resource level ones, enabling the discovery of unexpected cloud application behavior. The unexpected application behavior, further analyzed can be a case of user requirements violation, being followed by a new decision process resulting in an elasticity adaptation plan.

FR3 -- Evaluate application user's elasticity requirements:
Using the structured monitoring information enriched with metrics resulted from the composition process above, the Decision Module evaluates application user's elasticity requirements in terms of cost, quality and resources. It considers the application structure, the different metrics associated to that structure, and application profiling information and decides whether the current application status is in accordance with the requirements specified by the application user. Moreover, if hints about the application workload characteristics are given by the application user, the decision module adapts the decision frequency and mechanisms used to the workload characteristics (e.g. if the workload has a certain periodicity, the decision will be adapted to it).

FR4 -- Generate action plan:
An action plan is generated in two cases: (i) for the case the CELAR Manager signaled that a smart deployment is needed for a new application, or (ii) for the case the analysis phase found violations on elasticity requirements, and an elastic adaptation is needed.
For the first case, the Decision Module uses pulled information from the CELAR DataBase concerning offered cloud services by infrastructure providers, application user's elasticity requirements and application profiling information and generates an action plan consisting of a deployment configuration fulfilling all his requirements (e.g. balancing between cost and quality).
In the second case, the Decision Module uses the analysis with regard to application user's elasticity requirements together with information regarding application profile, application elasticity capabilities and the metrics resulted from monitoring and generates an elasticity adaptation plan.

FR5 -- Learn adaptation action effects and elasticity adaptation action plans:
The Decision Module considers all information gathered in the past, and creates a dependency model for detecting correlations between application types, available actions and multiple level metrics. The model contains two main types of information: action effects on multiple level metrics depending on application type, and learned action plans that were considered successful for a specific type of application (an application type refers to application architectural and deployment patterns, see [Fehling 2011]).

3.4 Non-functional requirements

This section presents Decision Module non-functional requirements derived from above use cases and from the description of Elasticity Provisioning Platform requirements from Deliverable 3.1, and the means by which we achieve these requirements for the Decision Module.

NFR1 -- Abstraction
As specified in Deliverable D3.1, the Elasticity Provisioning Platform needs to be able to handle different application types. Since the Decision Module is the core of this platform, we intend to fulfill this non-functional requirement by enabling generic control for various application types, using abstract models that will be generic enough to map on different application patterns. The Decision Module uses an internal application model, which is constructed according to the application structure description received from the application user and fed with data from the profiler for creating a more accurate representation. The Decision Module uses this generic internal application model for modeling both application structure, real-time behavior and possible elasticity control actions, abstracting from details which are application specific and generating elasticity plans without knowing all the details of their implementation.

NFR2 -- Efficiency (time and resource utilization efficiency):
The Decision Module needs to take decisions in a timely manner. Considering that an instance of the Decision Module is deployed for each application on the CELAR Application Orchestrator server (Deliverable D1.1 Section 3.4.5), the module's resource usage in terms of CPU/memory usage is a less pressing problem. However, the time in which the deployment/adaptation decisions are generated and enforced needs to be as small as possible for minimizing the under/over-provisioning impact. Moreover, mechanisms designed need to take into consideration the possibility of receiving high amount of monitoring data (i.e. for the case of a complex application, which is deployed on many machines and for which a high number of metrics is monitored with high frequency) and its complexity.

NFR3 -- Robustness:
The Decision Module shall be able to cope with unexpected situations, like extremely unpredictable load variations, erroneous monitoring data, and rapid oscillations between under and over provisioning. Since the Decision Module controls the cloud application and it is one of the core parts which enable application elasticity, unexpected application behavior should not produce a failure in decision or worse, a termination of the Decision Module. Moreover, incomplete monitoring information will be considered when designing the Decision Module, and lack of profiling information is replaced by runtime application behavior analysis.

NFR4 -- Failure Management:
The Decision Module needs to handle cases where the actions are not successful, by taking measures like re-doing the action in case of the action failure, or undoing and re-doing the actions which preceded the failed action. We plan to enforce this non-functional requirement by monitoring the enforcement of the action plan, learning which actions are opposite to each other and having the capacity to undo and redo actions on-demand.

4 Decision Module Analysis

This Section presents a detailed analysis of the Decision Module's functionalities, expected inputs and produced outputs, as well as the issues to be encountered in the decision process. We first discuss possible elasticity requirements and on the application structure, and propose an internal model used as basis for the decision process and for reflecting the control actions. Next, we analyze both expected-to-be-received and expected-to-be-returned information (expected input/output analysis), outlining our assumptions and explaining the requirements. We end this Section by analyzing the possible runtime behavior of the decision module.

4.1 Application structure and elasticity requirements

Cloud applications are usually distributed among several virtual machines which may belong to different (virtual) clusters or even different cloud providers. Cloud application elasticity entails the automatic adaptation of its different components according to the demand and to the application user's elasticity requirements targeting quality and cost and their sub-classes. For instance, for a web application it may be the case that the business-end needs to be scaled when there are new computationally intensive jobs while the front-end and the data end have continuously stable loads. For obtaining highly granular control of cloud applications and being aware of what application component is being controlled, a composition-based model of the application is needed. Several works have proposed different ways to structure cloud applications, e.g., TOSCA [TOSCA] and CIMI [CIMI]. In our work, we define an *internal cloud application composition model* for controlling elasticity which uses concepts from existing cloud computing reference architectures (NIST CCRA [NIST 2011], IBM CCRA [IBM 2011]) and is generic enough, mapping different application types, while still complying with the proposed representations by different standardizing institutions. The model is used internally by the Decision Module for building a "current application model" by considering application structure and system-level information like how components/tiers of the application communicate at runtime or what is the type of relationship between application components/tiers.

4.1.1 Abstract application composition model

This model is needed in multi-level elasticity control for bridging the gap between the information about the system as seen by the cloud user or developer, which mainly consists of total costs and overall quality, and the data needed for accurately controlling the application which is generally lower level information. The application structure in Figure 3 is an *internal model* used for representing both application workflow elasticity requirements and structural aspects of the application. The proposed model is generic, aiming at supporting the structure of different kinds of applications (e.g. queue-based applications, web applications or batch jobs).

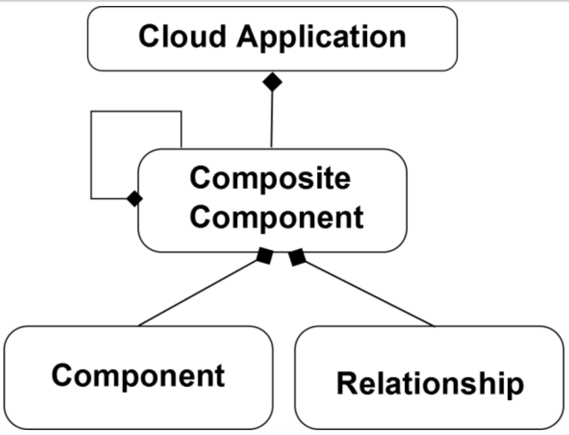


Figure 3: Abstract Composition Model

Our abstract model decomposes the cloud application into smaller sub-parts and consists of component, relationship, composite component, and cloud application. A component represents any kind of module or unit encapsulating functionality or data. This view is generic enough to facilitate the modeling of different types of applications and systems (i.e. web services, software package and web resource), each with different views on the units or modules the application is composed of. A group of components together with the relationships between them can be modeled as a composite component, a cloud application being composed of one or more complex components. The composite component is not physically represented at runtime; it is only a representational concept helping with the cloud application control.

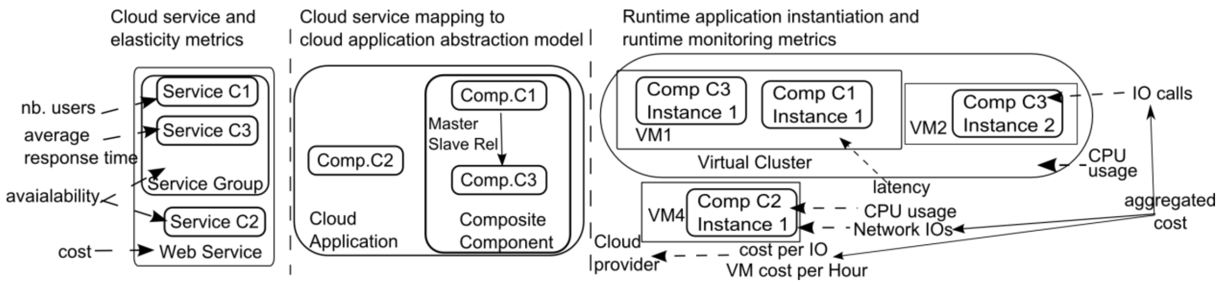


Figure 4: Mapping from User Perspective to System Perspective through Abstract Composition Model

Figure 4 shows a view on the cloud application from the structural and targeted metrics point of view. If we take the example of a web service (the left side of Figure 4), the cloud user views his/her web service as a set of services (in this case Service C1, Service C2, and Service C3), some of them grouped together for monitoring purposes (in this case Service Group which consists of Service C1 and Service C3). The metrics targeted in user's elasticity requirements in this stage are high level metrics, referring to the quality, cost and resources of services, of groups of services or even of the entire web service. At runtime, in the right part of the figure, service instances are deployed on virtual machines, in different virtual clusters or even different cloud providers, being viewed by the runtime control system through the light of an internal model (e.g. Service C1 becomes Component C1, possibly having more than one instances deployed on more than one virtual machines), and the accessible metrics are low level ones. The system to control such application needs to know how to aggregate metrics for obtaining the higher level ones which are targeted by the user, and how services are linked together for having the capacity to properly control them. For bridging this information gap between the user and runtime perspective we propose using the described cloud

application abstraction model, which facilitates the control system of the cloud application to take actions while being aware of the structure of the cloud application.
The cloud application abstraction model is associated with metrics information from two views. Firstly, the model is associated with metrics from the application user perspective that can view the cloud application as a web service and has a high level view concerning metrics. Next, the model is associated with metrics from the control system perspective which views cloud applications in a uniform manner using our abstract model and has an initial low level view concerning metrics, composed of information provided by the Cloud Information and Performance Monitor. These two views on metrics are mapped from one to another, aggregating low-level metrics for computing higher level ones. For instance, availability at application level would be computed from availability at each application part and the cost is aggregated from static information from the cloud provider on cost per I/O and VM cost, and the run-time application topology and loads.

4.1.2 Elasticity requirements specification

Elasticity requirements described by the application user through the Application Management Platform are pulled by the Decision Module from the CELAR DataBase. These requirements are structured according to the application model, depending on which part of the application they are connected to. For this description we developed SYBL, a language for elasticity requirements specification.
SYBL (Simple Yet Beautiful Language) is a directive-based language proposed for enabling multiple level elasticity control of cloud applications. SYBL has a <Monitoring, Constraints, Strategies> structure (see Figure 5) for the description of elasticity specifications in monitoring preferences, the constraints needed for specific metrics and the strategies suggested by the application user for fixing the encountered problems. Each elasticity requirement specified through SYBL is also referred to as a SYBL directive. More details on SYBL elasticity requirements can be found in [Copil 2013a].

```
Constraint := constraintName : CONSTRAINT ComplexCondition
Monitoring := monitoringName : MONITORING varName=MetricFormula
Strategy := strategyName : STRATEGY WHEN ComplexCondition :action(parameterList)|
          strategyName : STRATEGY WAIT ComplexCondition|
          strategyName : STRATEGY STOP|
          strategyName : STRATEGY RESUME

MetricFormula := metric | number | metricFormula MathOperator metric | metricFormula
          MathOperator number
ComplexCondition := Condition | ComplexCondition BitwiseOperator Condition|
          (ComplexCondition BitwiseOperator Condition)
Condition := metric RelationOperator number| number RelationOperator metric |
          Violated(name)|Fulfilled(name)
MathOperator := + | - | * | /
BitwiseOperator := OR | AND | XOR | NOT
RelationOperator := <|>|>=|<=|==|!=
```

Figure 5: SYBL in BNF

The application user can either detail strategies or just specify high-level elasticity requirements, depending on the functionalities offered by the Application Management Platform, and on the type of application user (i.e. the application user could have only high level quality or cost-related requirements on the overall application, while the application expert/developer can specify lower level elasticity requirements, specific to

a component or complex component –e.g. the CPU usage should be lower than 80%, the response time lower than 3 ms.).
To better grasp possible elasticity requirements, we present some examples depending on the level at which the specification is described, as well as on the user type (for describing elasticity requirements we make use of the CELAR user hierarchy defined in Deliverable D1.1 Section 2.3) or elasticity type. Elasticity depends on the type of the user and the granularity at which he/she wants to specify the application's elasticity. On one hand, the purpose of elasticity requirements varies from controlling costs to achieving higher quality or even specifying demands on the relation between cost, resources and quality, for example:

- *Cost-related elasticity requirements*: An application owner may specify that when the total cost is higher than 800 Euro for a number of clients, there should be a scale-in action for keeping costs in acceptable limits.
- *Quality-related elasticity requirements*: An application user may need to monitor different quality parameters, which should be in acceptable limits. For instance, an application owner can specify constraints on the response time depending on the number of users currently accessing the provided software. An application developer could specify that the result from a data analytics algorithm must reach a certain data accuracy under a cost constraint without caring how many resources should be used for executing the code of that algorithm.
- *Elasticity requirements on the relation between cost and quality:* An application owner could specify its pricing schema or price computation policies, for example that when availability is higher than 99% for a predefined period of time the cost should increase by 10%.

On the other hand, the user needs different granularities at which s/he can specify elasticity requirements, the user having elasticity requirements for the entire application or for specific application parts. This results in the need of having specifications enforced at different levels, as opposed to usual approaches in resource allocation and reallocation such as controlling only resources for the whole application or component level [Moran 2011], [Han 2012]. To this end, the following elasticity controls at different levels will be supported:

- *Component level elasticity requirements*: the user could specify different requirements based on the component type, i.e. the nature of requirements for the computation engine that are different from the requirements for the front-end component. The user can control the cost associated with a component, which means aggregating the cost of processes, storage and communication associated to the component but residing on different virtual machines.
- *Relationship level elasticity requirements:* the application user could need to specify the type of relation between components (e.g. master-slave, peer-to-peer, or map reduce), the type of network connection one should ensure between two components, the communication intensity (for determining which should be the components location), etc.
- *Composite Component level elasticity requirements*: the application user could specify requirements on a group of components connected by relationships, i.e. the cost over that complex component should be within specific limits, or that the complex component gathers components which should have really high availability.
- *Application level elasticity requirements*: elasticity requirements can be applied on the overall availability of the whole application, imposing aggregating data about

the different components of the application and the communication between them. The cost for the application at a defined level refers to the cost of components usage, communication between components and storage.

To support the aforementioned requirements, we model elasticity properties into a "resources-cost-quality" representation seen in Figure 6 on the left part, the right part of the figure showing possible sub-dimensions of cost, quality or resources, which at their turn could be further decomposed. The user should be given the opportunity of specifying the application's behavior, most specifically in what direction it should automatically scale in different cases in the space defined by the three axes (resource, cost and quality). Many properties from each of the elasticity dimensions are strongly interdependent, an elasticity property belonging to one axis being a multi-dimensional function of properties belonging to the other two axes.

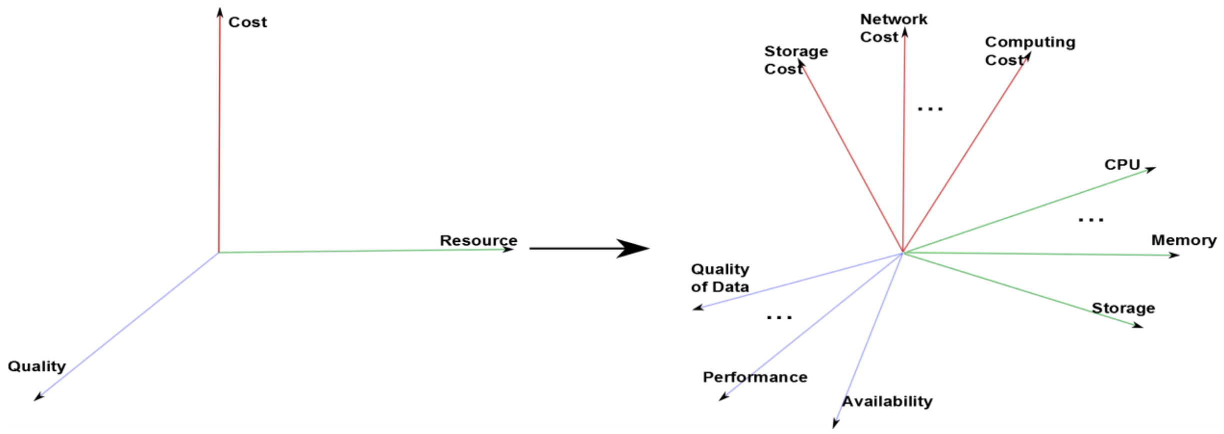


Figure 6: Multi-dimensional Representation of Elasticity

Depending on the user type (i.e. application owners usually pose application level requirements at high level concerning cost and quality while application developers may have lower level requirements also concerning resources), the application user can specify elasticity requirements on the form of constraints on considered metrics or even strategies if available. If only high-level elasticity requirements are available, the decision module can use learned information and information about how the metrics are aggregated at different levels to transfer the constraints into strategies referring to lower level metrics.

4.2 Information analysis

This Section gives an overview of the input expected by the Decision Module and on the output content and structure. The Decision Module takes as input the application structure, the application elasticity capabilities described by the application user through the Application Management Platform (received by the Decision Module in TOSCA representation as node operations), the relationships between components at runtime (i.e. master-slave, peer to peer, etc.), the elasticity requirements and the monitoring information containing metrics values; it outputs an elasticity adaptation action plan containing a sequence of actions which would adapt the application to current loads and elasticity requirements.

4.2.1 Input

4.2.1.1 Expected monitoring information

Monitoring information is received by the Decision Module from the Cloud Information and Performance Monitor. For analyzing cloud application behavior, we need a monitoring system capable of describing the monitored information source with respect to the monitored application software stack, monitoring the vertical hierarchy of the software stack levels, and their horizontal distribution over the application run-time topology (Figure 7).

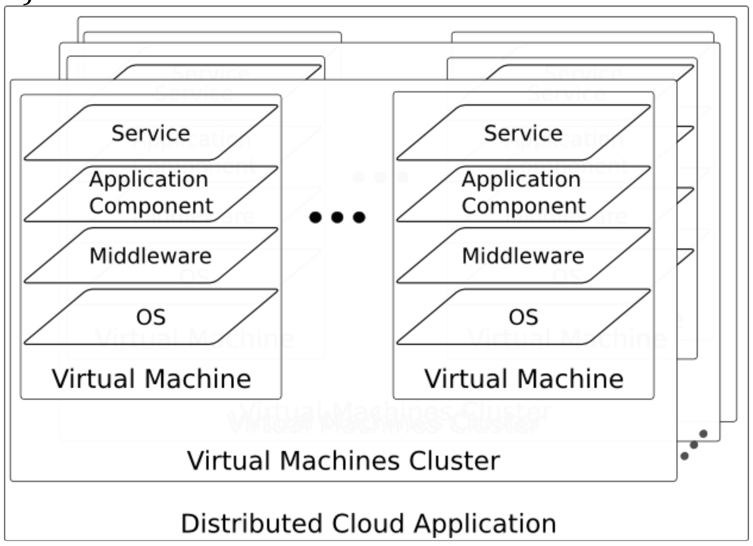


Figure 7: Cloud Application Monitoring Information Model

Monitoring data used by the Decision Module should contain enough information as to identify both the monitored application component and from where in the application deployment topology the data is retrieved. The monitoring data is internally structured using our informational model containing vertical monitoring levels ordered according to the monitoring information generality, starting from the most generic Operating System (OS) level (e.g. CPU idle/usage, free memory, and packets sent/received) to Service level(e.g. response time, latency, and throughput). These levels form the basic building block of any cloud application and give from coarse- to fine-grained information about the application behavior, from system information to the behavior of a particular client process. The horizontal monitoring model levels are individual Virtual Machine (VM) monitoring, and VM Cluster monitoring, providing a horizontal view over the application deployment topology, allowing the description of both individual VMs and VM Clusters containing VMs which perform related tasks.
For the data structure of the monitored metrics received from the Cloud Information and Performance Monitor, we envision a simple structure containing the timestamp, metric name, value, measurement unit, the monitoring level targeted by the metric, the minimum monitoring interval in milliseconds (minimum data refresh period) and the ID of the monitored application part (component, composite component or application) targeted by the metric.
Using supplied data source descriptions, cloud application monitoring data can be collected using both a push and pull based mechanism, covering a large spectrum of usage scenario. The pull-based approach enables adaptive data collection, having the ability to control the data sources querying time intervals, depending on the system behavior. Such adaptive collection can be used to keep the monitoring data traffic to a minimum, which might be important in a cloud environment where data traffic might be

priced based on the quantity of transferred data. A pull-based mechanism provides maximum control over the data collection process and is suited for systems that behave within some predictable boundaries, for which stable time intervals can be detected in order to schedule data source queries periodically. On the other hand, such a system can easily become a problem when critical events occur within the monitored service, periodical querying introducing unavoidable delays in detecting such events or even not reporting them at all.
To address this second scenario, push-based data collection is also supported, in which in the case of a critical event, the Cloud Information and Performance Monitor can send directly to the Decision Module a list of monitored metrics using the agreed format. After receiving the metrics list, the Decision Module uses the metric component IDs to map them onto the internal model, after which it can query according to the affected component other data sources for extra information, and then analyze the application elasticity requirements in order to determine the nature and source of the critical event.

4.2.1.2 Application static information

Application static information concerns descriptive information with regard to the application, useful for both a smart deployment of the application on the cloud considering elasticity requirements and for automatic adaptation of the application deployment topology according to the application loads. We describe static information below, structured on information type:

- *Application structure* is received by the Application Management Platform from the application user, placed in the CELAR DataBase and used by the Decision Module in the process of finding a smart deployment or elastic adaptation action plan.
- *Application elasticity capabilities* are descriptions of what actions can be enforced for the application, and the level at which the respective action can be enforced (i.e. application level, or action for a complex component or even at component level). A scale out action at component level (for this example we consider one component is deployed on one VM) would mean creating a new VM and subscribing to the component's load balancer, while a scale out action at complex component level can mean creating an entire new cluster (i.e. creating a new load balancer for the components that compose the cluster, creating new components, making the proper configurations for the components to subscribe to the newly created load balancer, and subscribing to the cluster's load balancer). Application elasticity capabilities do not target only horizontal scaling (e.g. adding/removing VMs), but also vertical scaling (adding/releasing resources like memory or storage) and configuration actions (changing the application or component configuration).
- *Application system (run-time) information* is information concerning runtime behavior of application components, like relationships among components (i.e. client-server, master-slave, and peer-to-peer). This type of information is gathered from multiple sources: an initial hint can be provided by the user, a more accurate description resulting from the profiler. This information will be continually refined by the Decision Module considering application evolution in time.
- *Application elasticity requirements* are described by the application user with the help of the Application Management Platform (see CELAR Architecture in Figure 1).

4.2.2 Input pre-processing

The input described above is processed by the Decision Module by integrating all the necessary information into the application model based on the compositional module of Figure 3. The monitoring information is mapped according to the application different structural levels (i.e. at component level, the cost could be the sum between the IO cost, the number of VMs necessary for hosting the component, the communication cost among these VMs, etc.), being used by the Decision Module when learning how different actions impact the metrics at different levels (i.e. a scale-out, depending on its level has an impact both on application level metrics and on component level metrics), and when generating new action plans.

4.2.2.1 Monitoring information associated withthe application structure

Monitoring information retrieved from the Cloud Application and Performance Monitor needs to be processed for composing higher level metrics which are normally targeted by application users in their requirements (i.e. the application user would normally be interested in application performance and data quality rather than, for example the CPU usage). The metrics must also be mapped onto the application structure to enable in-detail analysis of the behavior of application components.

While having metrics from different application levels is the first step towards application elasticity control, simple, directly collected metrics might not be sufficient for a complete elasticity analysis, as there might exist metrics that cannot be monitored directly or which are defined as a composition of other metrics. One example is the cost of running a cloud service, which usually depends both on direct measurable factors such as service running time, network data transfer or used storage size, and factors influenced by the cloud pricing policy, such as cost per hour per subscription scheme, cost per transferred data, or cost for storage size and type.

The second issue is mapping collected metrics on the application structure, as to support fine-grained analysis of application behavior. We provide a mechanism for aggregating monitoring data in a hierarchical structure bound to the logical application structure.

Towards describing cloud application properties that are not easy to measure directly, and mapping the collected metrics on the application structure, we define a mechanism for specifying metric mapping rules. Rules can be defined over one or more metrics existent at one monitoring level. When defined over a set of metrics, a rule generates a new composite metric. Each rule specifies the application structure level at which simple and composite metrics will be mapped. Optionally, a rule can also specify additional levels to which the metric is propagated. This propagation allows hierarchical structuring of metrics, enabling the definition of mapping rules over metrics resulted from rules applied at lower monitoring levels. The metrics mapping mechanism provides a proper information model for the service behavior analysis phase.

Using the metrics mapping mechanism, values from existing directly monitored metrics can be combined with values from other metrics or static values (such as cost), and be used to obtain new metrics representing anything from a simple measurement unit conversion, to metrics spanning more service components (such as service overall CPU usage) and metrics that contain cloud provider data (such as overall service storage cost). Although usually through metric composition the resulting metric loses expressiveness detail and depth compared to the source metrics, it gains perspective, providing a broader view over the system. The composite broader metric used in combination with the deeper source metrics provides a better overview over the behavior of a system than just using narrow but deep metrics.

Through this mapping stage, the Decision Module structures the metrics received from the Cloud Information and Performance according to the application structure (i.e. application, component, relationship level). Considering that elasticity requirements would target high level metrics (e.g. quality of data, performance, and cost) and the control is enabled at resource-level (e.g. virtual machines allocated, storage) the composite metrics provide the mechanism for tracing high level to low-level/resource-level metrics. This facilitates a fine-grained analysis of the application behavior, at all levels of the application structure (see model from Figure 3), and a low level control using the available mechanisms (IaaS or PaaS level actions).

4.2.2.2 Application structure mapping to internal structure

From the reasoning perspective, having a description of the application which relies on the model that is used in the decision process is of high importance for the decision process. The decision process needs to use an internal application model fed with all the necessary information, in which most of the applications or even all applications can be represented. This would enable the decision process to be generic, to be independent on the application type and to avoid the overhead of continuously pulling the information from multiple sources.

Following the abstraction model of Figure 8, the application can be represented in an XML structure which can be translated from TOSCA [TOSCA] or other description languages. The decision module takes the application structure or topology from the CELAR DataBase, together with the system information consisting of relationships between the components and groups (the upper part of the figure, showing information on `Node 1`, `Node 2` and `Node 3`), and transforms it into the internal application structure, the `Node 1` becoming `Comp C1`, the connection between `Node 3` and `Group 1` being represented as a separate entity, relationship `Rel CC1-C3`.

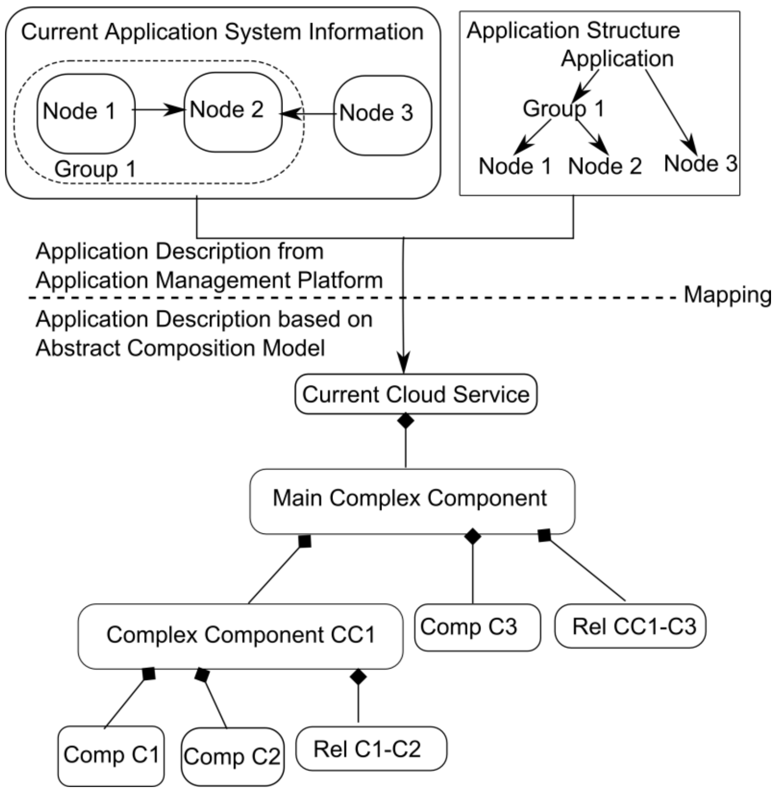


Figure 8: Mapping User's Application Description to Abstract Composition Model

Furthermore, the Decision Module associates the elasticity requirements with each part of the application structure, and can generate elastic adaptation plans while being aware of the application structure and of the requirements of the application user on each application part or group of application parts.
The elasticity requirements can be translated from the specifications as received from the Application Management Platform into SYBL constructs, also using the Decision Module history for specifying strategies for constraints which are specified by the application user and that are with high probability the same (i.e. in 99% of the cases the action plan for when a constraint is violated is the same). By taking this approach, we can save the time in searching for elasticity adaptation plans. SYBL constructs can be associated with the part of the application for which the requirements are described, and specific actions can be prescribed for the targeted application part.

4.2.3 Output towards other CELAR modules

The Decision Module returns towards other CELAR modules through the CELAR Manager output regarding the action plan and an action plan elasticity report resulted from the Decision Module analysis. The elasticity report contains information on how elastic the application is, answering to questions such as "how did the application behave when applying these actions?" or "how did the targeted metrics evolve?".
The generated action plans are also sent to other CELAR modules for enforcement. The action plans are generated as pairs of (`Action`, `Enforcement Time`) or (`Action`, `Actions To Be Finished`), where `Action` represents what needs to be done and `Enforcement Time` describes the time at which the action should be executed, and `Actions To Be Finished` represent the actions that need to be enforced successfully for enforcing the currently specified action. An example of action plan item for the second type of specification would be `(Reconfigure Component C1, (Scale out for C1))`, thus specifying that the current action to be enforced is the reconfigure action for the component C1, but not before the scale out action is finished successfully.
The action plan enforcement progress is continuously monitored by the Decision Module and in case of an action enforcement failure or of unexpected results a new sequence of actions is generated considering the updated knowledge regarding action effect or the effect of combinations of actions.

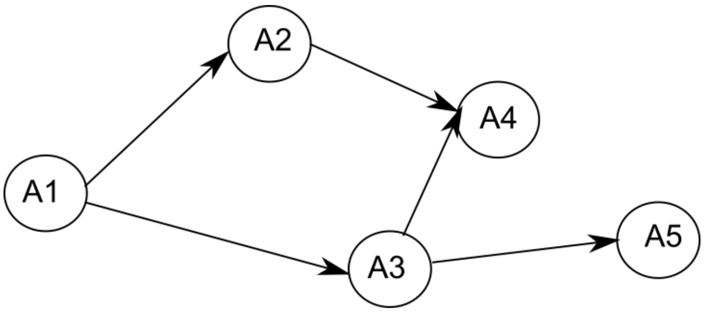


Figure 9: An Action Plan Represented as a Directed Graph

As shown in Figure 9, the action plan can be expressed as a directed graph where the time in which each action should be taken is not known or not necessarily important; what counts is the sequence in which the actions have to be enforced and which represents the dependency between them. In the directed graph, the graph nodes are actions and the dependency between them plays the role of temporal consequence, `A1->A2` meaning that `A2` can start being enforced once the A1 action is finished being

enforced. The relationship between `A4` and `A2` and `A3` states that both actions need to be finished for being able to start enforcing `A4`.
Therefore, the Decision Module does not just enforce action plans, but also checks whether they are successful and takes appropriate measures in case in which they are not, like sending new action plans to be enforced instead of the rest of the unsuccessful plan.

4.3 Runtime

The Decision Module is a module deployed for each application on the CELAR Application Orchestrator VM (see Section 3.4.5 from Deliverable D1.1), together with all the CELAR modules responsible for the application's adaptation. The Decision Module takes decisions either periodically or event driven (each time the Cloud Information and Performance Monitor triggers an event due to abnormal metric values), depending on its configuration.

4.3.1 Using SYBL elasticity directives for multiple level elasticity control

The Decision Module enforces control on multiple levels, taking into account the structure of the application. Figure 10 shows a simple flow depicting how the Decision Module can use the elasticity directives resulted from the elasticity requirements specified by the application user for generating elasticity adaptation action plans.
In the first phase, the directives are evaluated and the currently valid SYBL directives (their triggering conditions being evaluated to true) are extracted from the SYBL directives set (see SYBL description in Section 3.1.2). On the resulted set, conflicting directives can appear for the fact that users with different perspectives can specify elasticity requirements at different levels. For instance, directives specified at cloud application level can be in conflict with ones at component level: the application owner might want low costs while the application designer specifying elasticity requirements at component level might want high quality at any price. For solving the conflicting issues, for metrics targeted by the pair of conflicting directives the intersecting range is computed and a new directive is generated to replace the old ones (e.g. if one wants price to be lower than 800 and the other one specifies availability higher than 99.5 with the condition of having the price lower than 1200, we choose 800-1200 as an acceptable price range). Using the set of directives to be enforced, the next step is to generate the action plan containing the correct sequence of actions for enforcing the constraints and strategies specified, and send it for enforcement to the Resource Provisioner through the CELAR Manager.

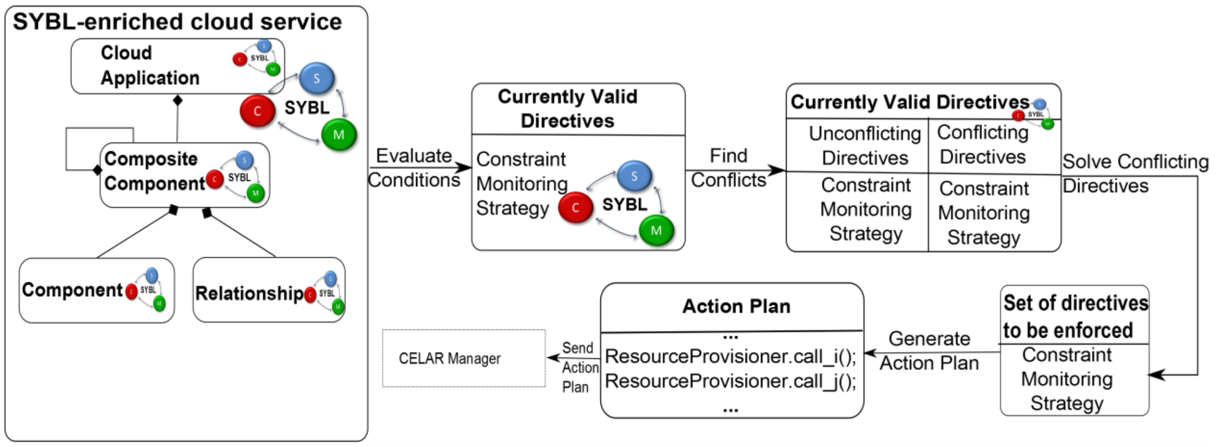

SYBL-enriched cloud service
Cloud Application
SYBL
Composite Component
Component
Relationship
Evaluate Conditions
Currently Valid Directives
Constraint
Monitoring
Strategy
SYBL
Find Conflicts
Currently Valid Directives
Unconflicting Directives
Conflicting Directives
Constraint Monitoring Strategy
Constraint Monitoring Strategy
Solve Conflicting Directives
Set of directives to be enforced
Constraint Monitoring Strategy
Generate Action Plan
Action Plan
...
ResourceProvisioner.call_i();
ResourceProvisioner.call_j();
...
Send Action Plan
CELAR Manager

Figure 10: Elasticity Control: from Directives to Action Plans

Conflicting directives are defined as directives which should be enforced at a given time, and which specify as desirable non-identical states. For instance, the elasticity constraint that requires a cost smaller than 300 Euro when availability is less than 80% is not opposed to, for example, the elasticity constraint which states that cost should be between 400 and 600 Euro when availability is higher than 99.5%.
We identify three types of conflicts: (i) conflicts between constraints and strategies from the same level of the cloud application, (ii) conflicts which appear between constraints and strategies belonging to different levels of the cloud application and (iii) conflicting directives targeting correlated metrics. For all types of directives conflicts, we first search for specified priorities which may solve the conflicts and show which directive should be enforced. The first type of conflicts is solved easily by our control runtime system with the help of the priority construct, through which the user specifies the hierarchy of priorities in the current specified directives. In the second type, the solution to the problem is not that simple due to security concerns (i.e. controls from component level are usually not aware of controls from cloud application region). When the latter type of conflict appears, we compute an intersection of the constraints' desired metric states and create a new directive which includes the requirements resulted metric states. If the intersection is null, a choice needs to be made between the requirements to enforce or enforce none, based on Decision Module settings. The user can specify for instance that the directives from a higher level have a higher priority, or that the directives from component level have highest priority since the user specifying elasticity requirements at component level has a better global perspective than the usual developer, while he is also aware of what happens in the code at lower levels. The third type of conflicts can appear between directives targeting different metrics, for instance in case the cost decreases the quality will also decrease, so constraints targeting high quality are in conflict with constraints targeting low costs. The elasticity control engine needs to know when enforcing one directive which other metrics it affects, and take appropriate measures for enforcing the other directives as well. We also plan to anticipate the effect one directive may have upon all the other metrics (not just the one specified in the directive).

4.3.2 Control types at different structural levels of the application

The elasticity capabilities are described by the application user through the Application Management Platform, and their impact on the elastic application behavior is extremely important but cannot be asked from the application user. Therefore, the Decision Module needs to know how different actions will impact the application behavior.
For being able to estimate the action effects even from the beginning of application deployment, we design a two-fold learning process two-fold: firstly, learning action effect patterns for application patterns, and secondly, continuous refinement of the generic learned model to the application behavior throughout the application hosting on the cloud. The first part of the learning process is a continuous process which associates expected application behavior in terms of metrics evolution with application patterns. When a new application is deployed, we need to be able to estimate depending on its architectural and deployment pattern the effect different control actions would have on the application. This initial information is provided by the user in case s/he can give hints on application behavior and by the Profiler module.
On the other hand, we need to have an even more fine-grained model which is continuously improving throughout the run-time of the application, learning from application's reactions to incoming workloads or to adaptation actions from different control levels. We will therefore have a global, static dependency model which can give

estimations on how the application behaves based on the patterns to which it subscribes and on the adaptation actions available. The second dependency model will be an instantiation of the global model for the current application (architectural and deployment pattern, available actions, etc.) which is continuously refined considering the application metrics throughout the application execution.

4.3.3 Considered algorithms for generating action plans

We plan to consider and evaluate several algorithms for generating adaptation action plans, taking into account application types, structure and possible application patterns. Moreover, we can enable decision mechanisms for data and networking as opposed to computation elasticity control.

Scientific applications require highly accurate decision making, and are not suitable for high adaptation frequencies (*low adaptation frequency algorithm)*, that would output optimal or close to optimal solutions even if the decision takes more time (in this case even exhaustive search is not out of the question). On the other hand, applications which are to be hosted for a high period of time (e.g. web applications), can have bursty workloads and *responsive algorithms* are necessary in which fast decisions need to be taken, which need to also map into fast enforcements (e.g. adding resources or increasing the quality should happen in the shortest possible time) or at least consider the time dimension of the enforcement process (e.g. heuristic-based algorithms, approximations or greedy algorithms). Moreover, these workloads can also have patterns over time (e.g. high/low workloads on holidays, weekends etc.), so we need to have algorithms addressing this anticipation and elastic adaptation according to the discovered patterns (*pattern-based algorithms),* for instance different types of mechanisms based on learning algorithms or clustering algorithms.

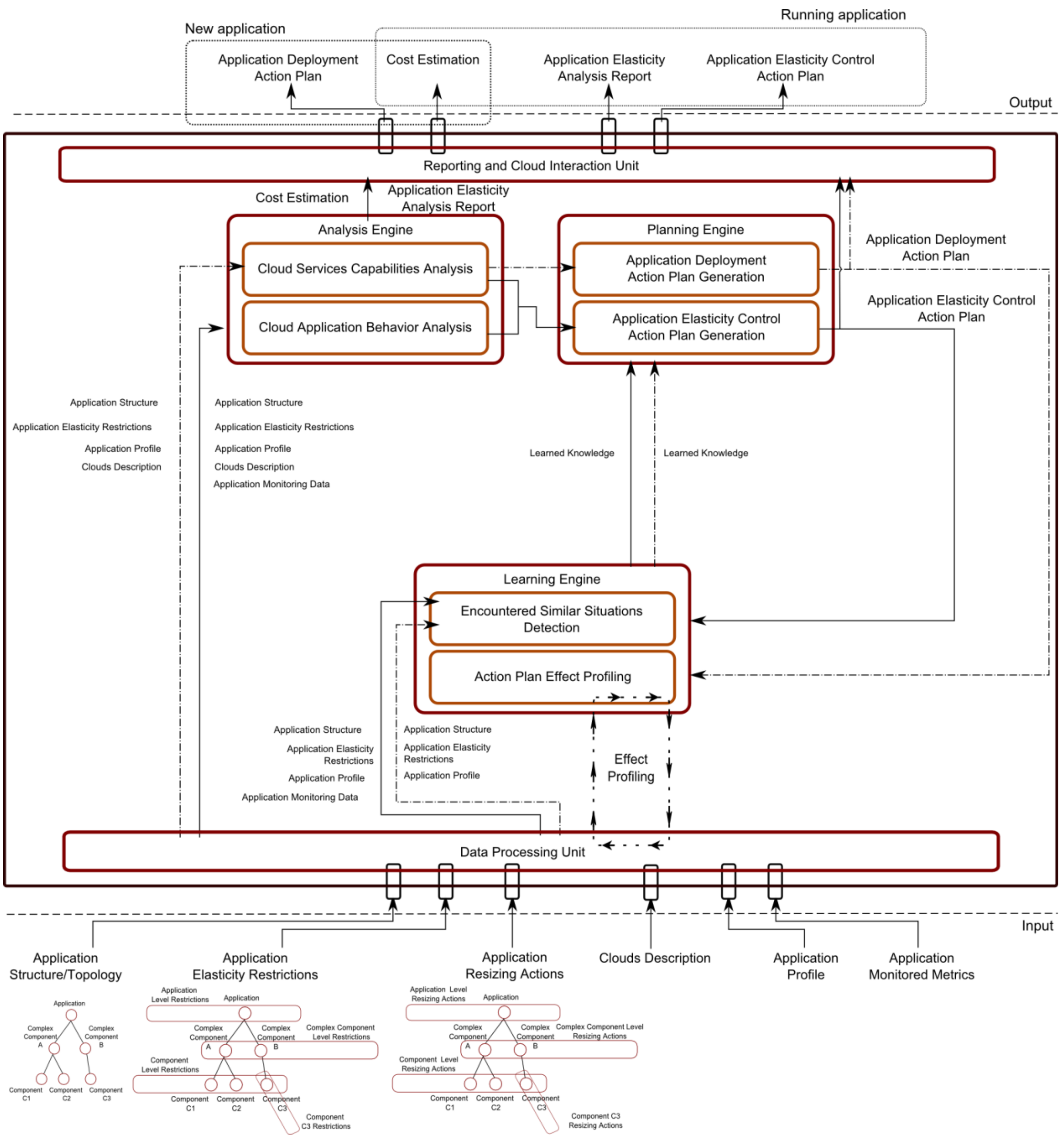
Furthermore, different application parts can map to different of the above enumerated algorithms. For instance, presentation layer components or business components would require patterns-enabled algorithms or responsive algorithms, while a complex component of the same application consisting of a data analytics program which needs to be run on a map reduce structure would entail a different type of adaptation algorithm (low adaptation frequency algorithms).

5 Decision Module Architecture

The Decision Module is a core component of the CELAR Elasticity Platform, used for reasoning on the application metrics and generating elasticity adaptation action plans for respecting the elasticity requirements specified by the CELAR user.

5.1 Decision module conceptual architecture

Figure 11 shows a conceptual view on the Decision Module. The Decision Module is composed of three core components: the *Learning Engine,* the *Analysis Engine* and the *Planning Engine,* and two components for interacting with the CELAR modules: the *Data Processing Unit* and the *Reporting and Cloud Interaction Unit*.



New application
Running application
Application Deployment Action Plan
Cost Estimation
Application Elasticity Analysis Report
Application Elasticity Control Action Plan
Output
Reporting and Cloud Interaction Unit
Cost Estimation
Application Elasticity Analysis Report
Analysis Engine
Cloud Services Capabilities Analysis
Cloud Application Behavior Analysis
Planning Engine
Application Deployment Action Plan Generation
Application Elasticity Control Action Plan Generation
Application Deployment Action Plan
Application Elasticity Control Action Plan
Application Structure
Application Elasticity Restrictions
Application Profile
Clouds Description
Application Structure
Application Elasticity Restrictions
Application Profile
Clouds Description
Application Monitoring Data
Learned Knowledge
Learned Knowledge
Learning Engine
Encountered Similar Situations Detection
Action Plan Effect Profiling
Application Structure
Application Elasticity Restrictions
Application Profile
Application Monitoring Data
Application Structure
Application Elasticity Restrictions
Application Profile
Effect Profiling
Data Processing Unit
Input
Application Structure/Topology
Application Elasticity Restrictions
Application Resizing Actions
Clouds Description
Application Profile
Application Monitored Metrics
Application
Complex Component A
Complex Component B
Component C1
Component C2
Component C3
Application Level Restrictions
Application
Complex Component
Complex Component
Complex Component Level Restrictions
A
B
Component Level Restrictions
Component C1
Component C2
Component C3
Component C3 Restrictions
Application Level Resizing Actions
Application
Complex Component
Complex Component
Complex Component Level Resizing Actions
A
B
Component Level Resizing Actions
Component C1
Component C2
Component C3
Component C3 Resizing Actions

Figure 11: Decision Module Conceptual Architecture

The *Data Processing Unit* finds in the CELAR DataBase (see CELAR architecture Figure 1) static information describing the application (i.e. application structure, application elasticity requirements, application elasticity capabilities, application profile) and the services offered by the cloud providers (cloud description) and dynamic information on the application evolution (application monitored metrics). The received static and dynamic information is sent towards the *Learning Engine* and the *Analysis Engine.* The *Learning Engine* detects for similar situations and action effects through the history, and feeds information relevant to the current application to the *Planning Engine.* The *Analysis Engine* evaluates metrics corresponding to each of the application parts (i.e. component level, complex component, application, relationships) against the elasticity requirements which is forwarded to the *Planning Engine* and computes application elasticity (composed of cost, quality and resources elasticity) which is forwarded to the *Reporting and Cloud Interaction Unit* which sends it to the CELAR Manager (which forwards it to the Application Management Platform for informing the CELAR user). The *Planning Engine* processes all information received and generates a sequence of actions, called **action plan,** for fulfilling the preferences of the CELAR user. The **action plan** is sent to the CELAR Manager, forwards it to the Resource Provisioner for enforcement.

5.1.1 Data Processing Unit

The Data Processing Unit has as main scope processing data coming from other CELAR modules, translating the data into structures used by the rest of the internal Decision Module components. The data processing unit executes information mapping on three main parts: (i) the application information is mapped to the Decision Module internal model described in Section 3.2.2.1, (ii) the elasticity requirements are mapped into SYBL descriptions for enabling the control and reasoning based on SYBL directives and (iii) the monitoring data is aggregated according to the application structure and the underlying resource structure as described in Section 3.2.2.2.

5.1.2 Learning Engine (Cloud Application Behavior Prediction Engine)

The Learning Engine uses the information from the Data Processing Unit for estimating application behavior as response to external stress factors such as workload or adaptation actions. We are interested not only in how the metrics targeted by the application user are affected, but also in how much time it takes for the action to be enforced. Initial work published in [Gambi 2013] applies change-point detection for learning the amount of time necessary for an action to be enforced in an un-intrusive manner. This information is valuable for the Decision Module due to the real time nature of the control, and the mechanism can be extended for capturing information on expected evolution on action enforcement for various metrics as well as for metrics correlation, using learning approaches like Bayes networks or reinforcement learning.
The Learning Engine uses historic information for correlating application patterns to action effect patterns for constructing a dependency model which helps estimating action effects even for applications recently deployed.

5.1.3 Analysis Engine

The Analysis Engine performs a two-phase analysis over cloud application monitoring data: (i) determines if the cloud application behavior violates user supplied elasticity restrictions and (ii) performs deep inspection of application behavior towards tracing the source of the violation as to support action plan generation.
The first analysis phase is designed as a lightweight process, analyzing independently in parallel the supplied elasticity restrictions against the aggregated monitoring model.

This step determines as fast as possible if there is any elasticity restriction violation, and overlaps the analysis result over the monitoring model, paving the way towards the next analysis phase.
The second analysis phase is a heavier process, aimed at detecting elasticity requirements violations by inspecting both directly monitored and aggregated metrics (in the case restrictions are defined using metric aggregation). The analysis is performed over the monitored data model, vertically trough the service software stack and horizontally over the deployment topology, determining which underlying software component violates the application elasticity restrictions and from where in the deployment topology the component the violation originates.
The result of the second analysis process is sent to the Planning Engine, which is in charge of determining adaptation strategies that would bring the application back in a situation in which all elasticity restrictions are respected.

5.1.4 Planning Engine

The Planning Engine uses information from the Analysis Engine and from the Learning Engine for deciding on strategies which have the form of action plans. The Learning Engine is firstly consulted for checking if there were similar situations in history of current application or if for this application type there exists a predefined "recipe"/ sequence of actions which would help fulfilling the elasticity requirements. If this is the case, the actions learned are mapped to the current application characteristics, and then sent for enforcement. If not, the Planning Engine will use the appropriate type of algorithm depending on the configurations and on the part of the application that requires adaptation (information received from the Analysis Engine) for generating an action plan which is bound to the current application model.

5.1.5 Reporting and Cloud Interaction Unit

The Reporting and Cloud Interaction Unit is responsible for mapping the action plan generated by the Planning Engine into a format accepted by external CELAR modules, non-dependent on the internal application model. It also sends the data generated by the Analysis Engine which describes the application behavior in terms of relation between metrics and elasticity requirements for the analyzed period of time.

5.2 Interactions within Decision Module

For the case when the CELAR Manager triggers an event of new application deployment, the Data Processing Unit forwards to the Analysis Engine information concerning application structure, elasticity requirements, elasticity capabilities, profiling information and cloud static information (steps 8, 9, 10 from Figure 12). The Analysis Engine asks for previous knowledge on the same type of application, or on similar application with similar elasticity requirements. The Analysis Engine also maps the application structure to an internal module, which is further presented in Section 4.1 (step 14 from Figure 12). The Analysis Engine abstracts and gathers all the received information into the abstraction internal model, and sends to the Planning Engine the request for a deployment strategy. The deployment strategy/plan, together with the elasticity report resulted from the analysis and planning steps are sent to the Reporting and Cloud Interaction Unit which forward them to the CELAR Manager. The elasticity report consists of estimated costs, estimated application elastic behavior (i.e. application reaction to external actions in terms of high and lower level metrics).

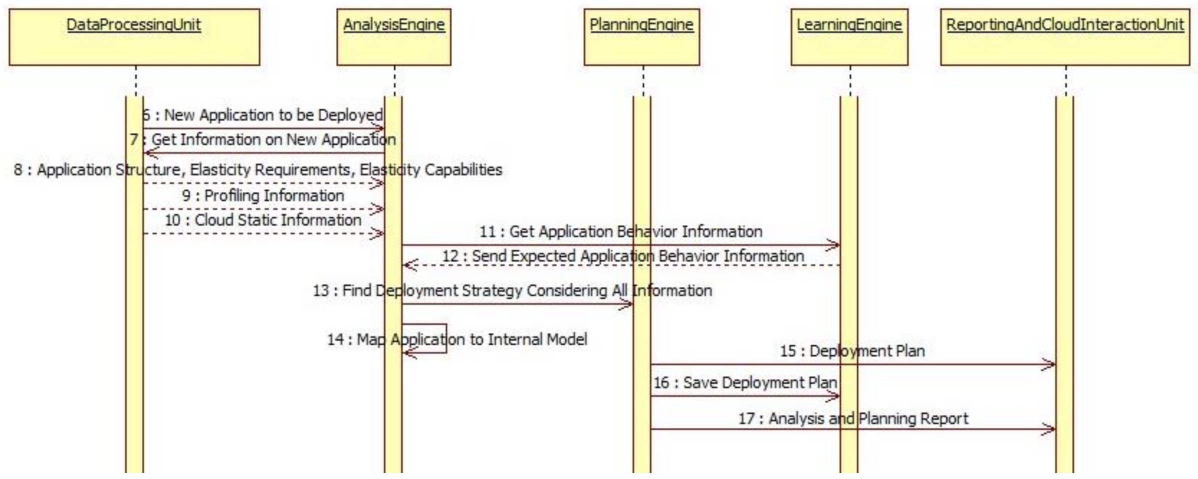

DataProcessingUnit
AnalysisEngine
PlanningEngine
LearningEngine
ReportingAndCloudInteractionUnit
6 : New Application to be Deployed
7 : Get Information on New Application
8 : Application Structure, Elasticity Requirements, Elasticity Capabilities
9 : Profiling Information
10 : Cloud Static Information
11 : Get Application Behavior Information
12 : Send Expected Application Behavior Information
13 : Find Deployment Strategy Considering All Information
14 : Map Application to Internal Model
15 : Deployment Plan
16 : Save Deployment Plan
17 : Analysis and Planning Report

Figure 12: Application Deployment

Figure 13 shows the elastic adaptation case sequence diagram, showing what steps are taken for finding an action plan for the elastic adaptation of the deployed application. The Analysis Engine continually gets monitoring information from the Data Processing Unit (step 10 from Figure 13), verifies if the elasticity requirements are fulfilled (step 11 from Figure 13) and sends to the planning engine the requirements that are found being violated (step 12 from Figure 13). The planning engine uses the Learning Engine for finding information on estimated actions effects (step 16 from Figure 13) and uses it for generating an action plan (step 17 from Figure 13). The action plan is sent to the Reporting and Cloud Interaction Unit for enforcement (step 18 from Figure 13). The action plan is stored by the learning engine. Throughout action plan execution, the enforcement is monitored and the action effect is learned. If the action plan does not have the expected effect, the decision module will generate a different action plan.

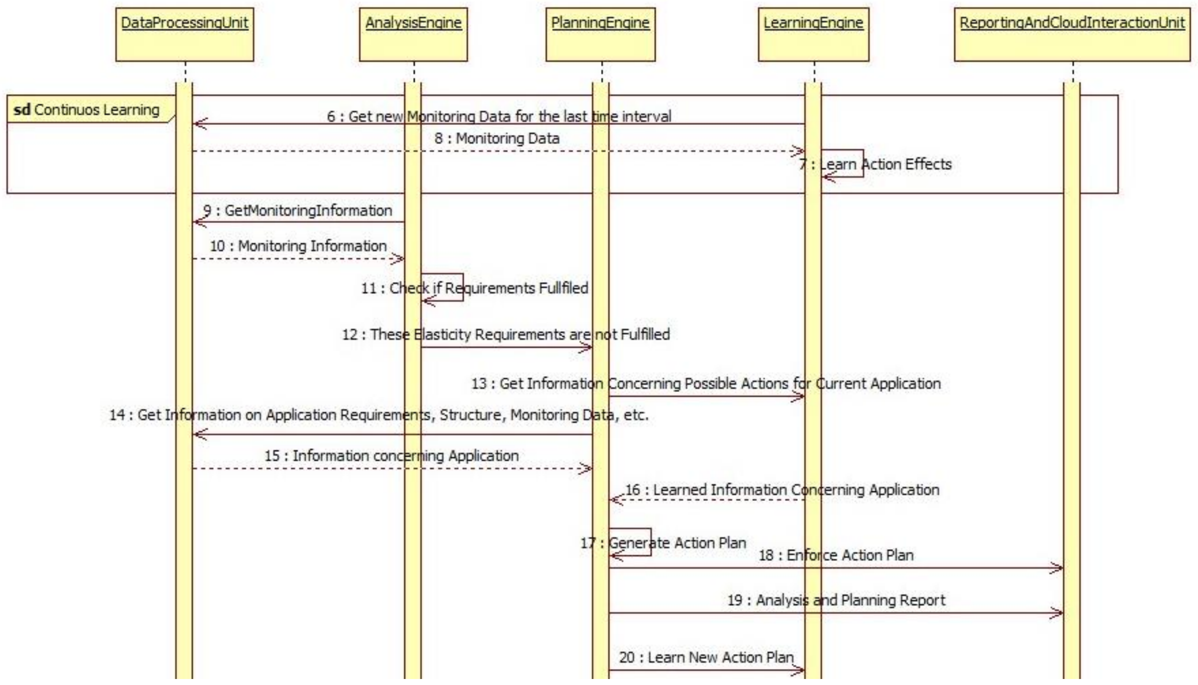

DataProcessingUnit
AnalysisEngine
PlanningEngine
LearningEngine
ReportingAndCloudInteractionUnit
sd Continuos Learning
6 : Get new Monitoring Data for the last time interval
8 : Monitoring Data
7 : Learn Action Effects
9 : GetMonitoringInformation
10 : Monitoring Information
11 : Check if Requirements Fullfiled
12 : These Elasticity Requirements are not Fulfilled
13 : Get Information Concerning Possible Actions for Current Application
14 : Get Information on Application Requirements, Structure, Monitoring Data, etc.
15 : Information concerning Application
16 : Learned Information Concerning Application
17 : Generate Action Plan
18 : Enforce Action Plan
19 : Analysis and Planning Report
20 : Learn New Action Plan

Figure 13: Elasticity Action Plan Generation

5.3 Interactions with other CELAR Modules

The Decision Module interacts with CELAR DataBase for obtaining all the necessary information (i.e. application monitoring information, application profile, elasticity requirements, application structure or application capabilities) and with the CELAR Manager for finding when it needs to provide a smart deployment strategy and for sending the generated plans (either for deployment or for elasticity control/adaptation) together with the cost estimation. The cost estimation together with the action plan and the elasticity analysis report are further sent by the CELAR Manager to the Application Management Platform for informing the application user. The generated action plan is also sent by the CELAR Manager for enforcement to the Resource Provisioner.

Table 2: Interaction with CELAR Modules

Input From			**Output To**	
Component	Input Type		**Component**	Output Type
CELAR Manager	New Application Added Event		**CELAR Manager**	Application Deployment Action Plan
				Application Elasticity Control Action Plan
CELAR DataBase	Application Profile			Cost Estimation
	Application Elasticity Requirements and Application Structure/Topology			Application Elasticity Analysis Report
Cloud Information and Performance Monitor	Application Monitoring			
	VM Level Monitoring			

Figure 14 shows the sequence of steps to be followed by the two CELAR modules which the Decision Module interacts with. Firstly, in case of new application submission to the CELAR System, the CELAR Manager sends the Decision Module the event of new application submission and application's ID. In the next steps, the Decision Module interrogates the CELAR DataBase for the application structure, requirements, profile, and elasticity capabilities and generates a new deployment plan (step 5-9 of sequence diagram shown in Figure 14). The generated plan together with application elasticity report is sent to the CELAR Manager, which in turn passes the received plan to the Resource Provisioner for elasticity enforcement, and the plan together with the elasticity report to the Application Management Platform.

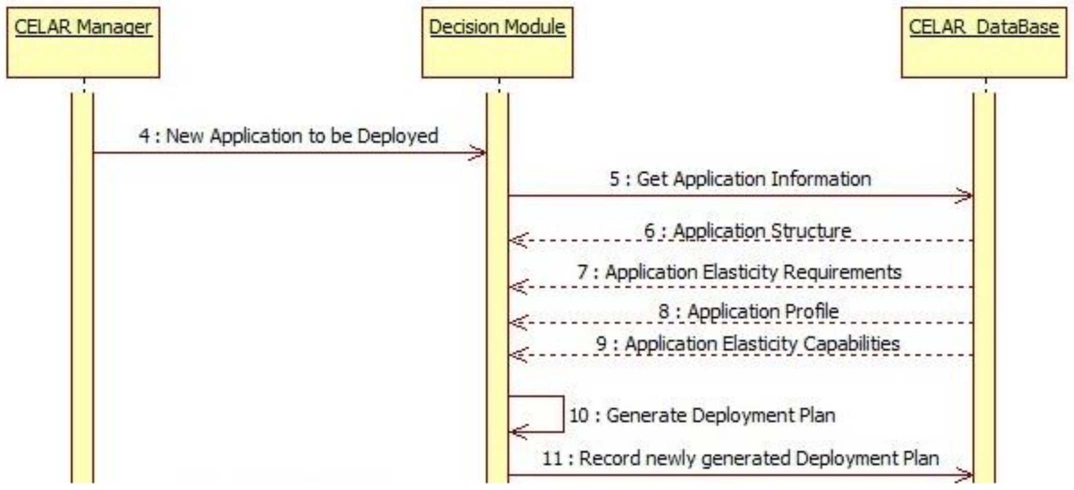

CELAR Manager
Decision Module
CELAR DataBase
4 : New Application to be Deployed
5 : Get Application Information
6 : Application Structure
7 : Application Elasticity Requirements
8 : Application Profile
9 : Application Elasticity Capabilities
10 : Generate Deployment Plan
11 : Record newly generated Deployment Plan

Figure 14: Application Deployment

The sequence diagram in Figure 15 depicts the sequence of steps necessary for elastic adaptation actions. The decision module gets fresh monitoring information from the Cloud Information and Performance Monitor on the monitored metrics of the application. The decision module checks whether the elasticity requirements are fulfilled, and in case of violation, it generates an elastic adaptation action plan and sends it to the CELAR Manager for enforcement together with the elasticity report, which are forwarded to the Application Management Platform. The plan is also recorded in the CELAR DataBase.

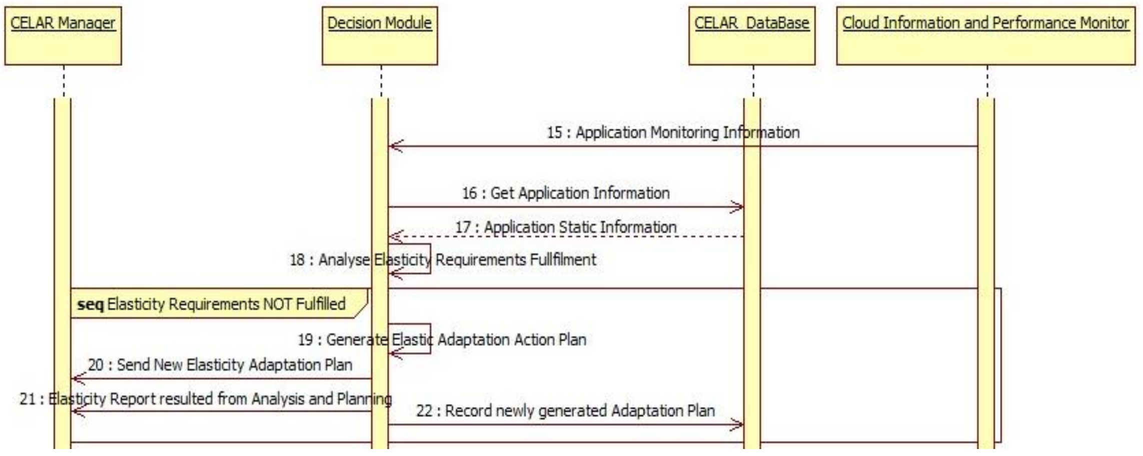

CELAR Manager
Decision Module
CELAR DataBase
Cloud Information and Performance Monitor
15 : Application Monitoring Information
16 : Get Application Information
17 : Application Static Information
18 : Analyse Elasticity Requirements Fullfilment
seq Elasticity Requirements NOT Fulfilled
19 : Generate Elastic Adaptation Action Plan
20 : Send New Elasticity Adaptation Plan
21 : Elasticity Report resulted from Analysis and Planning
22 : Record newly generated Adaptation Plan

Figure 15: Elasticity Action Plan Generation

6 Towards the Implementation of the Decision Module

Following the above design, we plan to implement a Decision Module able to interact with other CELAR modules in order to take elasticity adaptation decisions. The described components of the Decision Module are to be implemented using techniques and mechanisms which were outlined in this deliverable description, detailing for the implementation process the techniques used.

Throughout the implementation process several decisions need to be taken which may also involve other CELAR modules:

- *Input/ Output structure* – the structure of the input and the output need to be agreed for facilitating inter-module communication (e.g. whether the application structure is described using TOSCA [TOSCA], a different description language or even CELAR-defined description).
- *Communication mechanisms* – how the Decision Module will communicate with other modules, what protocols and languages are used, etc.
- *Algorithms/mechanisms used for decision*, depending on application type, structure, etc. – as described in the Analysis section (Section 3.3), we can have different decision mechanisms depending on the part of the application we focus and on the application type (e.g. scientific, gaming, etc.).

The first version of the Decision Module prototype will be described in deliverable D5.2, based on the initial design detailed in the current document.

7 Conclusions

In this document we have provided a detailed analysis and design description of the Decision Module. We have detailed the actors involved in the Decision Module, use-cases resulted and the functional and non-functional requirements for the Decision Module. We described a model for the application model which is to be used as a basis for the Decision Module's reasoning process, and presented the Decision Module architecture together with the interactions between internal components and interactions with other CELAR modules. Based on this deliverable, the implementation progress on the Decision Module will be further documented into deliverables D5.2 and D5.3.

References

[AutoScale] Amazon AutoScale, http://aws.amazon.com/autoscaling/

[AzureWatch] ParaleapAzureWatch for Windows Azure: https://www.paraleap.com/

[Calheiros 2012] Rodrigo N. Calheiros, Christian Vecchiola, DilebanKarunamoorthy, and RajkumarBuyya."The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds".*Future Generation Computing Systems* 28, 6 (June 2012), 861-870. http://dx.doi.org/10.1016/j.future.2011.07.005

[Chaisiri 2012] Sivadon Chaisiri, Bu-Sung Lee, Dusit Niyato, "Optimization of Resource Provisioning Cost in Cloud Computing," IEEE Transactions on Services Computing, vol. 5, no. 2, pp. 164-177, 2012 http://doi.ieeecomputersociety.org/10.1109/TSC.2011.7

[CIMI] DMTF. Cloud Infrastructure Management Interface (CIMI).http://dmtf.org/standards/cloud

[Copil 2013a] Georgiana Copil, Daniel Moldovan, Hong Linh-Truong, and Schahram Dustdar. "SYBL: an Extensible Language for Controlling Elasticity in Cloud Applications". *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing - CCGRID2013*, Delft, the Netherlands, May 14-16, 2013. [ACCEPTED] http://www.infosys.tuwien.ac.at/research/viecom/papers/SYBL_ccgrid2013.pdf

[Copil 2013b] Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, Scahahram Dustdar, "Multiple Levels Elasticity Control of Cloud Services", May 2013, under submission.
http://www.infosys.tuwien.ac.at/prototypes/elasticitycontrol/papers/multiLevelControl.pdf

[D1.1] User Requirements and System Architecture V1, Deliverable, CELAR Project, https://wiki.celarcloud.eu/doku.php?id=CELAR_Project:Deliverables:Deliverables_1.X:D1.1

[D3.1] Architecture Definition of the Elasticity Provisioning Platform, Deliverable, CELAR Project,
https://wiki.celarcloud.eu/doku.php?id=CELAR_Project:Deliverables:Deliverables_3.X:D3.1

[Didona 2012] Diego Didona, Paolo Romano, Sebastiano Peluso, and Francesco Quaglia. "Transactional auto scaler: elastic scaling of in-memory transactional data grids". *In Proceedings of the 9th international conference on Autonomic computing (ICAC '12)*. ACM, New York, NY, USA, 125-134. DOI=10.1145/2371536.2371559 http://doi.acm.org/10.1145/2371536.2371559

[Dustdar 2011] Schahram Dustdar, Yike Guo, Benjamin Satzger, Hong Linh Truong. „Principles of Elastic Processes". *IEEE Internet Computing* 15(5): 66-71 (2011)

[Fehling 2011] Christoph Fehling, Frank Leymann, Ralph Mietzner, Walter Schupeck , "A Collection of Patterns for Cloud Types, Cloud Service Models, and Cloud-based

Application Architectures," University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, Germany, University of Stuttgart, Institute of Architecture of Application Systems, *Technical Report Computer Science 2011/05*, May 2011. http://www.iaas.uni-stuttgart.de/institut/mitarbeiter/fehling/TR-2011-05%20Patterns for Cloud Computing.pdf

[Fehling 2012] Christoph Fehling, Frank Leymann, Jochen Rütschlin, David Schumm. "Pattern-Based Development and Management of Cloud Applications."*Future Internet* 4, no. 1: 110-141.

[Gambi 2013] Alessio Gambi, Daniel Moldovan, Georgiana Copil, Hong Linh-Truong, SchahramDustdar. "On Estimating Actuation Delays in Elastic Computing Systems". *8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)* - San Francisco, USA, May 20-21, 2013. [ACCEPTED] http://www.infosys.tuwien.ac.at/research/viecom/papers/seams13-id46-p-16142-preprint.pdf

[Gong 2010] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. "PRESS: PRedictive Elastic ReSource Scaling for cloud systems". *International Conference on Network and Service Management (CNSM), 2010*, vol., no., pp.9,16, 25-29 Oct. 2010 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5691343&isnumber=5691186

[Gonzalez 2012] Andres J. Gonzalez, Bjarne E. Helvik. "System management to comply with SLA availability guarantees in cloud computing". *2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom),* vol., no., pp.325,332, 3-6 Dec. 2012 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6427508&isnumber=6427477

[Han 2012] Rui Han, Li Guo, Ghanem, M.M., Yike Guo. "Lightweight Resource Scaling for Cloud Applications," *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid),* vol., no., pp.644,651, 13-16 May 2012 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6217477&isnumber=6217397

[IBM 2011] IBM Cloud Computing Reference Architecture. https://www.opengroup.org/cloudcomputing/uploads/40/23840/CCRA.IBMSubmission.02282011.doc

[Juve 2011] Gideon Juve and Ewa Deelman."Automating Application Deployment in Infrastructure Clouds," *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on* , vol., no., pp.658,665, Nov. 29 2011-Dec. 1 2011 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6133211&isnumber=6133119

[Kundu 2012] Sajib Kundu, Raju Rangaswami, Ajay Gulati, Ming Zhao, and Kaushik Dutta. 2012. Modeling virtualized applications using machine learning techniques.

SIGPLAN Not. 47, 7 (March 2012), 3-14. DOI=10.1145/2365864.2151028 http://doi.acm.org/10.1145/2365864.2151028

[Lama 2012] Palden Lama and Xiaobo Zhou. "AROMA: automated resource allocation and configuration of mapreduce environment in the cloud". In *Proceedings of the 9th international conference on Autonomic computing (ICAC '12)*. ACM, New York, NY, USA, 63-72. DOI=10.1145/2371536.2371547 http://doi.acm.org/10.1145/2371536.2371547

[Lee 2011] Gunho Lee, Byung-Gon Chun, and H. Katz. "Heterogeneity-aware resource allocation and scheduling in the cloud". In *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing (HotCloud'11)*. USENIX Association, Berkeley, CA, USA, 4-4.

[Li 2012] Zheng Li, Liam O'Brien, He Zhang and Rainbow Cai. "On a Catalogue of Metrics for Evaluating Commercial Cloud Services".*2012 ACM/IEEE 13th International Conference on Grid Computing (GRID),* pp.164,173, 20-23 Sept. 2012 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6319167&isnumber=6319148

[Lim 2010] Harold C. Lim, Shivnath Babu, and Jeffrey S. Chase."Automated control for elastic storage". In *Proceedings of the 7th international conference on Autonomic computing* (ICAC '10). ACM, New York, NY, USA, 1-10. DOI=10.1145/1809049.1809051 http://doi.acm.org/10.1145/1809049.1809051

[Lu 2011] Lei Lu, Hui Zhang, Guofei Jiang, Haifeng Chen, Kenji Yoshihira, and Evgenia Smirni. "Untangling mixed information to calibrate resource utilization in virtual machines". In *Proceedings of the 8th ACM international conference on Autonomic computing (ICAC '11)*. ACM, New York, NY, USA, 151-160. DOI=10.1145/1998582.1998606 http://doi.acm.org/10.1145/1998582.1998606

[Malkowski 2011]Simon J. Malkowski, Markus Hedwig, Jack Li, Calton Pu, and Dirk Neumann. "Automated control for elastic n-tier workloads based on empirical modeling". In *Proceedings of the 8th ACM international conference on Autonomic computing* (ICAC '11). ACM, New York, NY, USA, 131-140. http://doi.acm.org/10.1145/1998582.1998604

[Marian 2012] Tudor Marian, Hakim Weatherspoon, Ki-Suh Lee, and Abhishek Sagar. "Fmeter: extracting indexable low-level system signatures by counting kernel function calls". In *Proceedings of the 13th International Middleware Conference (Middleware '12)*, Priya Narasimhan and Peter Triantafillou (Eds.). Springer-Verlag New York, Inc., New York, NY, USA, 81-100

[Matsunaga 2010] Andréa Matsunaga and José A. B. Fortes."On the Use of Machine Learning to Predict the Time and Resources Consumed by Applications".In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing* (CCGRID '10). IEEE Computer Society, Washington, DC, USA, 495-504. 98 http://dx.doi.org/10.1109/CCGRID.2010.98

[Moore 2011] Ryan W. Moore and Bruce R. Childers."Inflation and deflation of self-adaptive applications".In *Proceedings of the 6th International Symposium on Software*

Engineering for Adaptive and Self-Managing Systems (SEAMS '11). ACM, New York, NY, USA, 228-237.http://doi.acm.org/10.1145/1988008.1988041

[Moran 2011] Daniel Morán, Luis M. Vaquero, Fermín Galán. "Elastically Ruling the Cloud: Specifying Application's Behavior in Federated Clouds". *2011 IEEE International Conference on Cloud Computing (CLOUD)*, vol., no., pp.89, 96, 4-9 July 2011 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6008697&isnumber=6008659

[NIST 2011] NIST Cloud Computing Reference Architecture. September 2011, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[Polo 2011] Jordà Polo, Claris Castillo, David Carrera, Yolanda Becerra, Ian Whalley, Malgorzata Steinder, Jordi Torres, and Eduard Ayguadé. "Resource-aware adaptive scheduling for mapreduce clusters". In *Proceedings of the 12th ACM/IFIP/USENIX international conference on Middleware (Middleware'11)*, Fabio Kon and Anne-Marie Kermarrec (Eds.). Springer-Verlag, Berlin, Heidelberg, 187-207. DOI=10.1007/978-3-642-25821-3_10 http://dx.doi.org/10.1007/978-3-642-25821-3_10

[Schatzberg 2012] Dan Schatzberg, Jonathan Appavoo, Orran Krieger, and Eric Van Hensbergen."Why Elasticity Matters". *Technical Report BUCS-TR-2012-006*, Computer Science Department, Boston University, April 15, 2012.

[Shen 2011] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. 2011. CloudScale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing* (SOCC '11). ACM, New York, NY, USA, Article 5 , 14 pages. DOI=10.1145/2038916.2038921 http://doi.acm.org/10.1145/2038916.2038921

[Singh 2011] Rahul Singh, Prashant Shenoy, Maitreya Natu, Vaishali Sadaphal, and Harrick Vin. "Predico: a system for what-if analysis in complex data center applications". In *Proceedings of the 12th ACM/IFIP/USENIX international conference on Middleware (Middleware'11)*, Fabio Kon and Anne-Marie Kermarrec (Eds.). Springer-Verlag, Berlin, Heidelberg, 123-142. DOI=10.1007/978-3-642-25821-3_7 http://dx.doi.org/10.1007/978-3-642-25821-3_7

[SmartCloud] IBM SmartCloud Initiative http://www.ibm.com/cloud-computing/us/en/index.html

[TOSCA] OASIS Committee Specification, "Topology and Orchestration Specification for Cloud Applications Version 1.0 18 March 2013", http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.html.

[Verma 2010] Akshat Verma, Gautam Kumar, and Ricardo Koller. 2010. "The cost of reconfiguration in a cloud". In *Proceedings of the 11th International Middleware Conference Industrial track* (Middleware Industrial Track '10). ACM, New York, NY, USA, 11-16. DOI=10.1145/1891719.1891721 http://doi.acm.org/10.1145/1891719.1891721

[Villegas 2012] David Villegas, Athanasios Antoniou, Seyed Masoud Sadjadi, and Alexandru Iosup.“An Analysis of Provisioning and Allocation Policies for Infrastructure-as-a-Service Clouds”.In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)* (CCGRID '12). IEEE Computer Society, Washington, DC, USA, 612-619. DOI=10.1109/CCGrid.2012.46 http://dx.doi.org/10.1109/CCGrid.2012.46

[Wang 2011] Qingyang Wang, Simon Malkowski, Deepal Jayasinghe, Peng Cheng Xiong, Calton Pu, Yasuhiko Kanemasa, Motoyuki Kawaba, and Lilian Harada.“The Impact of Soft Resource Allocation on n-Tier Application Scalability”.*2011 IEEE InternationalParallel & Distributed Processing Symposium (IPDPS)* vol., no., pp.1034,1045, 16-20 May 2011 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6012911&isnumber=6012804

[Wang 2012] Qingyang Wang, Yasuhiko Kanemasa, Motoyuki Kawaba, and Calton Pu. “When average is not average: large response time fluctuations in n-tier systems”. In *Proceedings of the 9th international conference on Autonomic computing (ICAC '12)*. ACM, New York, NY, USA, 33-42. DOI=10.1145/2371536.2371544 http://doi.acm.org/10.1145/2371536.2371544

[Xiong 2011] Pengcheng Xiong, Zhikui Wang, Simon Malkowski, Qingyang Wang, Deepal Jayasinghe, and Calton Pu. “Economical and Robust Provisioning of N-Tier Cloud Workloads: A Multi-level Control Approach”. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems* (ICDCS '11). IEEE Computer Society, Washington, DC, USA, 571-580.http://dx.doi.org/10.1109/ICDCS.2011.88

[Xu 2010] Lei Xu and Brendan Jennings. „A Cost-Minimizing Service Composition Selection Algorithm Supporting Time-Sensitive Discounts”. In *Proceedings of the 2010 IEEE International Conference on Services Computing (SCC '10)*. IEEE Computer Society, Washington, DC, USA, 402-408. DOI=10.1109/SCC.2010.76 http://dx.doi.org/10.1109/SCC.2010.76

[Yang 2009] Jie Yang, Jie Qiu, and Ying Li. 2009.“A Profile-Based Approach to Just-in-Time Scalability for Cloud Applications”.In *Proceedings of the 2009 IEEE International Conference on Cloud Computing* (CLOUD '09). IEEE Computer Society, Washington, DC, USA, 9-16. DOI=10.1109/CLOUD.2009.87 http://dx.doi.org/10.1109/CLOUD.2009.87

[You 2011] Gae-won You, Seung-won Hwang, and Navendu Jain. “Scalable load balancing in cluster storage systems”. In *Proceedings of the 12th International Middleware Conference (Middleware '11)*. International Federation for Information Processing, Laxenburg, Austria, Austria, 100-119