UNIVERSITÀ DEGLI STUDI DI MILANO

Scuola di Dottorato in Informatica
Dipartimento di Informatica
Informatica/XXX Ciclo

TESI DI DOTTORATO DI RICERCA

# Supporting Users in Cloud Plan Selection

INF/01 INFORMATICA

DOTTORANDO
**Ala Arman**

TUTOR
Prof. Pierangela Samarati

CORRELATORI
Prof. Sara Foresti, Dr. Giovanni Livraga

COORDINATORE DEL DOTTORATO:
Prof. Paolo Boldi

A.A. 2016/17

# ACKNOWLEDGMENTS

I would like to dedicate this thesis, first, to my mother, an angel I owe her all achievements in my life. Second, to my father who is the symbol of power, humbleness, hard-working, and patience in my life. Third, to my elder sister, Neda who has supported me in every pace of my life with constant love and endless encouragement. Fourth, to my younger sister Negah who has taught me lifetime lessons which could be abbreviated in one sentence "Do not give up!". The spirit and motivation she brought to me caused me to keep on going when I was ready to quit. Fifth, to my brother, Azim who always I can trust him in every storm that comes my way. Finally, to my niece, Ghazal who I can not love her more.

Moreover, I would like to dedicate my greatest appreciation to Prof. Pierangela Samarati who has been and will be my role model. Research-wise, working with her as a Ph.D. student, apart from learning invaluable knowledge, taught me how to be deep and precise in research, just as she does. Character-wise, her kind heart together with her incredible social intelligence taught me how to be humble when you are the most well-known scientist in your field, while maintaining your disciplines and standards. Lastly, I must mention that her always-open-door policy was my only life-saver solution whenever I found myself in a deadlock.

Also, I would like to deeply appreciate Prof. Sara Foresti who patiently supported me with her precise and constructive comments which significantly improved my technical and writing skills. In addition, I would like to thank Prof. Giovanni Livraga who always made time for me to answer my every single question.

In addition, I appreciate Prof. Sushil Jajodia, Prof. Roberto Di Pietro, and Prof. Mauro Conti for reviewing my thesis and providing me constructive comments which significantly enhanced the quality of my thesis.

Further, I would like to thank my friends namely, Abhinav Anand, Gerson Soares, Massimo Walter Rivolta, Aleksandar Rikalovic, and Morteza Ghasemi for all their support and advises who helped me a lot with a warm smile, each time that I asked for their favor. In particular, I would like to acknowledge Abhinav who has been like a brother to me since I met him for the first time. He was always there for me whenever I needed a friend to share my stories.

Finally, I would like to thank the employees of Department of Computer Science including, Lorena, Daniela, Mario, Mirko, Claudia, and Danio for their constant support.

# ABSTRACT

Cloud computing is a reference paradigm for deploying applications and for storing, managing, and processing large amounts of data. Today, thanks to significant efforts made on optimizing the technological and business aspects of Cloud computing, Cloud providers available on the market provide a broad range of cost-effective services over the Internet, characterized by availability, scalability, and reliability. Therefore, individuals and organizations, ranging from small to large enterprises, can move their IT asset to the Cloud for benefiting from the significant advantages of Cloud computing technology.

Moving applications to the Cloud, however, requires addressing different problems. First, applications should be assessed to see to what extent they are ready to be moved to the Cloud, considering the specific properties (e.g., dynamic, distributed) of such environments. Outsourcing applications that cannot be efficiently moved to the Cloud could limit the use of Cloud computing. Also, Cloud providers offer their services through plans that differ in terms of their characteristics. This variety ensures great advantages for users, enabling them to choose the plan that better suits their needs and economical availability. However, considering the different characteristics of Cloud plans and the heterogeneous requirements of applications (e.g., the number of replicas, CPU rates, security features), choosing a Cloud plan, among those offered by a (set of) Cloud provider(s), is a hard challenge. The situation can get more complicated when users wish to move multiple applications, at the same time, to the Cloud because each application can have different (and even contrasting) requirements. Therefore, in this scenario, it is necessary to properly combine the requirements of applications and choose the Cloud plan that satisfies them in the best possible way.

In this thesis, we provide models and tools to help users in evaluating applications that are moved to the Cloud and in selecting the most suitable Cloud plans among the available ones on the market. Considering the problems mentioned above, we provide two main contributions in this thesis.

The first contribution focuses on evaluating applications by assessing their modularity. The dynamic nature of Cloud environments implies that Cloud resources usually undergo frequent changes (e.g., resource join/leave/failure). Therefore, applications with poor modular design, due to high coupling and low cohesion among their components, cannot suitably adapt themselves to dynamic changes, and as a result, are not able to run effectively on the Cloud. Likewise, such applications cannot be broken down into components and usually are not efficient when they are distributed across Cloud environments. In this thesis, we then propose an approach aimed at estimating the modularity of applications to examine to what degree they can be easily moved to the Cloud.

The second contribution concerns plan selection. First, we propose a consensus-based plan selection approach aimed at choosing the plan that best balances the preferences of all the applications by reaching a trade-off among their requirements. Next, we focus on plan selection under uncertainty, by considering a scenario where a group of users with low technical skills and a limited budget wish to outsource multiple applications to the Cloud. In this respect, we propose an approach aimed at selecting a Cloud plan that respects budget limits according to the preferences, even imprecise, of applications provided by a set of users. Finally, we focus on supporting the business objectives of users when they move their applications to the Cloud. To do so, we consider a multiple-application scheduling scenario, where a virtual machine is selected for each application. Then, we propose an approach aimed at maximizing financial profit that is estimated for each application when it is executed on the Cloud, considering compensation mechanisms offered by service providers in their service level agreements (SLAs).

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1 INTRODUCTION

The rapid advancement in the popularity of Information and Communication Technology (ICT) coupled with the growing demand for storing, processing and, managing digital data have properly justified the expansion of Cloud computing technology. Today, Cloud users can benefit from the significant advantages of Cloud computing over the conventional methods of on-premise computing. First of all, Cloud computing is considered as a cost-effective computing paradigm [1] [2] because *1)* capital costs (e.g., hardware infrastructure, software licenses) can be ignored [3]; *2)* operating expenses (e.g., costs for administration and maintenance, server cooling) can be cut- off [4]; *3)* the elastic nature of Cloud resources and the existence of various "pay-per-use" [5] pricing models allow payment only for used resources and lead to significant economic savings [6] [5]. Second, thanks to the rapid delivery of Cloud services over the Internet, data and applications can be universally accessed, processed, and managed in a reliable and secure way. Third, due to an almost infinite amount of resources provided by Cloud computing technology [7], users can move their data and applications to the Cloud while being worry-free about the lack of required resources. Fourth, by delegating obligations associated with data and application management to Cloud providers, users are relieved tedious and low-level administrative tasks [8] (e.g., software update, backup, data replication). Therefore, in addition to eliminating the need for hiring administrative staff, users can focus on the core workflow of their business.

Although the technological and business benefits of Cloud computing services may be seductive, moving applications to the Cloud is a difficult challenge and involves addressing different key problems. First, we need to assess to what extent applications are ready to be easily outsourced to the Cloud, considering the specific properties (e.g., dynamic, distributed) of Cloud environments. In fact, the adoption of solutions for selecting Cloud plans that suitably meet the requirements of applications would not be useful if applications cannot run effectively on the Cloud. Second, Cloud plan selection for outsourcing applications is a difficult problem to address. There are two major reasons can be considered for this: *1)* nowadays, the Cloud computing market is enriched by several Cloud providers introducing different Cloud plans with heterogeneous characteristics; *2)* the significant progress in ICT has resulted in designing applications with high levels of complexity and various requirements (e.g., availability level, CPU rates, security guarantees). The consequences of adopting Cloud plans that do not fit the requirements of applications can be discussed from two points of views: *1)* if the characteristics of selected Cloud plans are higher than the requirements of applications that are moved to the Cloud, it could cause waste of Cloud resources and introduce unnecessary costs (e.g., virtual machine (VM) rental costs, software and hardware licenses) which is to the benefit of neither Cloud providers nor users; *2)* insufficiently meeting the requirements of applications that are moved to the Cloud decreases the satisfaction of users from Cloud computing technology and impedes its further adoption. Also, the intuitive approach for plan selection when multiple applications, at the same time, are moved to the Cloud is choosing a plan for each application. However, the problem of Cloud plan selection can get even more complicated when users are interested in choosing a single Cloud plan for multiple applications that are moved to the Cloud because each application can have different (and possibly contrasting) requirements. Therefore, to select a Cloud plan for a set of outsourcing applications, it is essential to properly combine their requirements in order to balance their satisfaction which is usually a controversial issue and needs to be carefully investigated in Cloud plan selection scenarios.

## 1.1 CONTRIBUTIONS OF THE THESIS

In general, this thesis follows two main lines of research. First, it focuses on evaluating to what extent applications can be easily moved to the Cloud, considering the distributed and dynamic nature of Cloud environments. Second, we provide novel solutions for selecting suitable Cloud plans, among those available on the Cloud market, considering several key problems involved in Cloud plan selection scenarios.

In the following, we provide a summary of the contributions of this thesis.

- *Modularity evaluation.* The dynamic nature of Cloud environments implies that Cloud resources change frequently due to management operations (e.g., resource churn) or their volatility (e.g., resource failure) [9]. Therefore, to be able to run effectively on the Cloud, outsourcing applications should be capable of dynamically adapting themselves to such frequent changes in Cloud environments [9]. Therefore, applications with poor modular design, which are difficult to reconfigure, are not able to run effectively on the Cloud [9]. Also, such applications, which cannot be broken into smaller parts, are not usually performant when they are distributed across Cloud infrastructures due to high coupling between their comprised components. We propose a software engineering approach to analytically evaluate the modularity [10] of applications to estimate to what extent they can be easily moved to the Cloud w.r.t. change flexibility and/or distributability. Our proposed solution operates in three main steps: *i)* defining an evaluation classification which includes different modularity attributes (e.g., coupling, cohesion) and their associated metrics (e.g., coupling between classes for coupling attribute), according to the context of problem (e.g., application type, execution context of application on the Cloud); *ii)* estimating modularity at micro (component) level, considering attributes and metrics explored in the first step; *iii)* evaluating modularity at macro (system) level, considering the obtained modularity at micro level (see Chapter 3).

- *Cloud plan selection.* The Cloud market is growing at a quick pace, offering a variety of opportunities to its users. Indeed, Cloud providers available on the market sell plans that differ in the services they offer, the quality of services they guarantee, and the price lists they apply. This variety provides great advantages for users, enabling them to choose the plan that better suits their needs and economical availability. However, selecting best-fit Cloud plans, considering the functional and non-functional requirements of applications as well as the characteristics of available Cloud plans, is not often an easy task and implies more than few trivial steps. Also, moving multiple applications to the Cloud, at the same time, makes plan selection even more complicated as each application can have different (and possibly contrasting) requirements. Therefore, fulfiling the requirements of some applications might leave other applications unsatisfied. Then, we provide innovative solutions aimed at choosing suitable Cloud plans for a set of applications, considering their requirements and the characteristics of Cloud plans (see Chapters 4, 5, and 6). The main features of the proposed solutions can be summarized as follows.

  **Consensus-based Cloud plan selection.** Balancing the satisfaction of the requirements of all applications is considered as a fundamental objective when multiple applications simultaneously are moved to the Cloud. We present an approach aimed at balancing the satisfaction of applications' requirements (e.g., availability level, CPU rates, security guarantees) by selecting a Cloud plan that is globally considered the most acceptable by all applications. It operates first by ranking the available Cloud plans (matching plan characteristics and application requirements) and then by selecting, through a consensus-based process, the one that is considered more acceptable by all applications.

  **Uncertainty management in Cloud plan selection.** Conventional techniques for selecting Cloud plans implicitly assume that users are familiar with the technical details of the requirements of their applications which might not always be the case as unskilled IT users might move their applications to the Cloud as well. Also, users might consider a limited budget for selecting a Cloud plan, among those available, each with possibly a different price. This problem can get even more complicated when a team of users/stakeholders contribute to a Cloud plan selection process. We propose an approach aimed at choosing an affordable Cloud plan for a set of applications when multiple users/stakeholders provide imprecise information about them. We first measure crisp importance for each application, based on the associated linguistic importance, expressed by each user, and then, measure the crisp preference of applications over each criterion (e.g., availability, performance), based on the linguistic preference of each application over the criterion and the obtained crisp importance of applications, using fuzzy techniques. Finally, we select the affordable plan, through a cost-benefit analysis process, that is the best fit for applications, considering the obtained preferences of applications over the set of criteria.

**Risk-aware application scheduling.** Cloud providers usually apply some compensation mechanisms when their promised qualities of services are not met. Therefore, to support the business objectives of users, such compensation mechanisms, which can have high impacts on the financial profit of applications when they are executed on the cloud, should be carefully considered in application scheduling scenarios. We propose an approach aimed at, by mapping each application to an available VM offered by multiple Cloud providers, maximizing financial profit that is estimated for each application, according to its importance. It mainly works in three phases. Considering each possible VM availability scenario, we first measure a penalty which is paid by its respective Cloud provider if the promised uptime of VM is not met, and then, estimate a financial profit for the current application to be scheduled if it is assigned to each available VM. Finally, through a risk analysis process, we assign each application to an available VM, according to the expected monetary value of application when it is mapped to each available VM.

## 1.2 ORGANIZATION OF THE THESIS

The remainder of the thesis is organized as follows.

**Chapter** 2 presents the state of the art of different approaches proposed for addressing the problems associated with Cloud scenarios discussed in this thesis.

**Chapter** 3 illustrates an approach to evaluate modularity for an application to estimate to what extent it can be easily moved to the Cloud w.r.t. change adaptability and/or distributability.

**Chapter** 4 discusses an approach for Cloud plan selection that best balances the satisfaction of the requirements of multiple applications by reaching a consensus among them.

**Chapter** 5 proposes a solution for Cloud plan selection by considering different imprecise opinions about applications that are moved to the Cloud, provided by multiple unskilled IT users with a limited budget for selecting Cloud plans.

**Chapter** 6 formulates Cloud plan selection as an application-to-VM assignment problem and uses quantitative risk analysis techniques to support the business objectives of users w.r.t. the financial profit of applications, considering service level agreement compensation mechanisms, offered by multiple Cloud providers.

**Chapter** 7 summarizes the contributions of this thesis and discusses the future work.

# 2

# RELATED WORKS

This chapter provides some the-state-of-art approaches related to topics covered in this thesis. Section 2.1 discusses the assessment of applications to see to what extent they can be easily moved to the Cloud w.r.t change flexibility and/or distributability as well as some related works in this area. Section 2.2 presents some basic concepts about Cloud plans and the importance of their selection together with some interesting related works in this area which are divided into two categories: *1) single-application context* which includes scenarios that a plan is selected for a single application (see Section 2.2.1) and *2) multiple-application context* which includes scenarios that a Cloud plan is selected for multiple applications with different and possibly contrasting requirements (see Section 2.2.2).

## 2.1 APPLICATION ASSESSMENT IN OUTSOURCING SCENARIOS

Today, the increasing popularity of Cloud computing due to providing almost unlimited resources which are highly available worldwide in an affordable fashion, compared to traditional on-premise computing, is inevitable. However, while the technological and economical advantages of Cloud computing technology may be seductive, several issues must be considered when users move their applications to the Cloud. First, applications are needed to be assessed to see to what extent they can be easily moved to the Cloud, considering the properties (e.g., dynamic, distributed) of Cloud environments. The objective of such assessment is customizing applications to improve their efficiency when they are executed on the Cloud. For example, in [11], a method is proposed to improve the efficiency of computation-intensive applications by decreasing the analysis time of term-weighting scenarios and keeping the quality of retrieved information in an acceptable level. As as a result, such applications can run more effectively when Cloud providers are short of resources (e.g., due to high workload, resource churn). In fact, even we adopt an approach to choose the best-fit plans(s) for satisfying the requirements of applications, it would not be a promising approach if they suffer from the lack of running efficiency on the Cloud. Therefore, the assessment of applications to see to what extent they are ready to be easily moved to the Cloud can be considered as a prerequisite for Cloud plan selection.

There are several aspects (e.g., performance, security, availability, maintenance cost) that are needed to be assessed when moving applications to the Cloud. In particular, considering the dynamic nature of Cloud environments, Cloud resources usually undergo several changes due to management operations (e.g., resource churn) or their volatility (e.g., resource failure) [9]. Therefore, applications that are moved to the Cloud need to appropriately adapt themselves to changes that they face to be able to run efficiently on the Cloud. Moreover, applications that are moved to the Cloud usually are distributed across Cloud infrastructure(s) [12], and as a result, they are required to get properly decomposed into smaller parts. Therefore, considering dynamic and distributed properties of Cloud environments, we need to evaluate applications w.r.t. change adaptability and/or decomposability to see to what extent they are ready to be moved to the Cloud. From the software engineering point of view, applications with modular design are more flexible to changes [13] compared to monolithic ones. Also, a modular application can be easily decomposed into smaller parts due to the low dependency between them, which as a result, suitably distributed across Cloud environment(s). Therefore, modularity can be considered as a metric for the change flexibility and/or distributability of applications. In the following section, we provide some basic concepts about application modular design, and then, we review some related works addressed the evaluation of application modularity.

### 2.1.1 MODULAR DESIGN EVALUATION

Modularity is defined as the ability of a system to decompose into a set of cohesive and loosely coupled modules/components [14]. An Application with a modular design comprises almost small and related objects, each fulfiling a clear and unique function [15]. In [16], five criteria defined for evaluating a modular design which enables us to infer the advantages of modular applications, presented as follows [10]:

- **Decomposability.** The architecture of a modular application can be decomposed to into smaller modules which enables their distribution across Cloud infrastructure(s).
- **Composability.** The modules of an application with a modular design can be reused to assemble new application(s). Such reusability allows for decreasing the required time and cost of developing new applications.
- **Understandability.** A module in a modular application is easily understandable as a standalone unit. Thus, it will be easier to build and simpler to change without having to know about and/or reference to other modules.
- **Continuity.** Small changes in a modular application will result in changes in individual modules rather than system-wide changes. As a result, the impact of

side-effects (e.g., changes in other modules, increasing dependency between modules) due to such changes will be minimized. Therefore, the more an application is modular, the more flexible it is to changes.

- **Protectability.** Considering an application with a modular design, if an error occurs within a module, the impact of side-effects induced by the error will be minimized.

Considering the continuity (change flexibility) and decomposability properties of modular design, modularity can be considered as a metric for evaluating the adaptation capability and/or distributability of applications. Several researches (e.g., [17], [18], [19], [20], [21]) studied the modularity of applications from a software engineering point of view. In the following, we will discuss some related works in this area.

**Modularity evolution assessment.** The authors in [17] considered the following measures for assessing the modularity of applications when a new version is released: *1) Coupling*, which is defined as a measure of interconnection among the components of an application [10]. Higher coupling values for an application reflect greater difficulties to change its components because a change in one component may have an impact on all other components that are coupled to it [22]; *2) Cohesion*, which is defined as a measure of the degree to which a component focuses on just one single task [10]. Higher cohesion values for an application reflect the higher division of functionalities among components, and as a result, higher change flexibility and decomposability; *3) Complexity,* which is revealed by coupling and cohesion (i.e., higher cohesion indicates lower complexity). Also, the following sub-measures considered for coupling, cohesion, and complexity.

- w.r.t. coupling, the following sub-measures are considered: *1) coupling between object classes [23]* which measures the number of other classes that are coupled to the assessed class; *2) response for a class [24]* which counts the number of methods that can be invoked in response to a message received by an object of the assessed class; *3) afferent coupling [25]* which is the number of classes in other packages depending on classes in the assessed package; *4) efferent coupling [25]* which counts the number of packages that classes in the assessed package depend upon; *v) coupling between methods [26]* which represents the total number of methods to which all the inherited methods are coupled.

- w.r.t. cohesion, the following sub-measures are taken into account: *1) lack of cohesion in methods (LCOM) [24] [27]* which represents the number of the pairs of methods that are not related through the sharing of the some of instance variables; *2) lack of Cohesion in methods (LCOM3) [28]* which is an improved version of LCOM and measures the cohesion of assessed class by considering the effective usage of instance variables.

- w.r.t. complexity, the following sub-measures are considered: *1) weighted method per class [24]* which measures the sum of the complexities of all class methods;

*2) McCabe's Cyclomatic Complexity [29]* which measures the number of different paths (decision points) in a method plus one.

Then, the values of measures are obtained in four release versions of two open source projects (Camel and POI). Next, the modularity of each project is described by separately interpreting the metric values. The work in [17], while sharing the same idea of considering measures and sub-measures for modularity evaluation, is different from our proposed approach in Chapter 3 because we measure modularity at two levels (micro (component) level and macro (system) level). Also, we consider a different importance for each modularity metric, in contrast, to approach presented in [17] which assumes all measures to be equally important.

**Modularity improvement.** The distribution of classes among modules in an application is considered as an effective factor impacting modularity. In fact, classes with higher similarity (affinity) should be placed in the same module to improve modularity. In [30], a quantitative approach is proposed aimed at analyzing the modularity of applications, and then, providing a solution for improving them. To evaluate modularity, the authors in [30] propose a modularity factor, called MMF, based on the level of coupling among classes which indicates the affinity (similarity) among them. Then, assuming that the number of modules is known, a set of modularization solutions (i.e., the distribution of classes among modules) is proposed, using different cluster analysis and affinity-rating techniques. Next, a dispersion parameter (i.e., the maximum number of classes in a module) is defined to avoid concentrating a large number of classes on a small set of modules. Then, the best modularization solution is the one with high MMF and not too much dispersion. The metrics considered for evaluating modularity in [30] are limited to coupling, cohesion, and size (i.e., the number of classes, the number of modules) in contrast to our proposed method where it is possible to consider any modularity attribute and metric, according to problem context. Also, the authors ignore the different importance/relevance of modularity metrics in contrast to our approach. Moreover, the work in [30] does not evaluate modularity at system level which is different from our proposed method as we estimate modularity both at micro (component) level and at macro (system) level. In [18] a method is proposed aimed at providing heuristic advice on code modularity based on a heuristic design similarity measure. The proposed similarity measure, which calculates the similarity between two objects in an application, supports two services namely, clustering and maverick analysis. Clustering service identifies the set of related procedures and maverick analysis identifies procedures that appear to be in the wrong module. The work in [18] focuses on code modularity which is different from ours as we consider modularity measurement at micro (component) level and macro (system) level. Also, our proposed method is more general as it is possible to consider any modularity attribute and associated metric which provides suitable flexibility in choosing modularity attributes and metrics according to the context of problem (e.g., application architecture, application type, Cloud service change rate, the level of application distribution). Such modularity attributes and metrics could be defined by referring to existing guidelines and classifications (e.g., [31], [32]).

In [20], a method is proposed to evaluate a quality attribute (e.g., maintainability) considering associated quality characteristics (e.g., correctability, testability), each with respective metrics (e.g., fault rate and required effort for testability) [21] [33]. Inspired from [20], the authors in [21], which probably is the closest work to ours for modularity evaluation (see Chapter 3), proposed a method to measure modularity as a quality attribute. They also considered only maintainability and reusability as quality characteristics. In contrast to the work in [21], our work provides suitable flexibility in modularity evaluation by defining a generic modularity evaluation classification which modularity attributes (e.g., coupling, cohesion, size) and their associated metrics can be considered both at micro (component) level (e.g., coupling between components [24], for coupling attribute) and at macro (system) level (e.g., coupling between servers, for coupling attribute). Moreover, we introduced a metric at system level for evaluating modularity. It measures the number of use-cases for each application service which is obtainable by adopting UML analysis (e.g., considering a sequence diagram associated with the assessed application).

The second main issue addressed in this thesis is Cloud plan selection. In the next subsection, we provide some basic concepts about Cloud plans and some related works about their selection in outsourcing scenarios.

## 2.2 CLOUD PLAN SELECTION IN OUTSOURCING SCENARIOS

The growing popularity of Information and Communication (ICT) has resulted in developing applications with complex and different functional and non-functional requirements. Therefore, when applications are moved to the Cloud, several aspects should be considered to select Cloud services that suitably meet the requirements of applications. Exploring necessary criteria for fulfilling the requirements of applications can be one of the most important aspects of such consideration. To show the diversity and low-levelness of the requirements of outsourcing applications, we conducted a literature review to explore criteria that are usually considered for the requirements of applications, from both user and provider point of views. As depicted in Table 2.1, we divide the considered criteria into four main categories, including *Security*, *Cost*, and *Performance*, and *Support*, and then, for each category, we consider some sub-categories (e.g., *Elasticity* and *Latency*, considering *Performance* criterion).

| Criterion | Sub-criterion (1) | Sub-criterion (2) | Sub-criterion (3) | Comments |
|---|---|---|---|---|
| **Security** | Confidentiality | | | e.g., dependency on external third party [34], intrusion detection time |
| | Integrity | | | e.g., credential management overhead [35] |
| | Availability | | | e.g., load balancing [35], fault resolution time [36] |
| **Cost** | Pricing | Model | Resource | e.g., per-use, subscription, prepaid per-use |
| | | | Licence | e.g., hourly, one-time charge, free [37] |
| | | Variation | | e.g., discount on the per-use model [37] |
| | | Price changes | | e.g., increasing 15% instance/hour and storage price in 2 years [38] |
| | | Premium services | | e.g., fast connection [37] |
| | | Temporal replication of components [37] | | to handle spikes |
| | | Hidden extra charges | | e.g., ingress and egress bandwidth to predict in advance) [37] |
| | | Data transfer between Clouds [39] | | |
| | Staff skills [40] | Technical/Experience of IT professionals [41] | | |
| | | Negotiating & Engaging in technical discussions [41] | | |
| | Staff training [42] | knowledge/experience/skills of staff | | |
| | | Available documentation from the provider | | e.g., Windows Azure tutorials |
| | Operating cost [43] | Test [44] | | functional and non-functional properties [45] |
| | | (Re)-Configuration/Installation | | e.g., collecting (re)configuration information [45], GUI [46], database, code, connection [47], security policy [48], data encryption/fragmenation [36], third party tool installation [42] |
| | Maintenance | Update [44] | | e.g., patching OS, applications, and VMs with the last security updates |
| | | Upgrade | Upgrade frequency [49] | |
| | | | Test [44] | functional and non-functional properties (e.g., response time fault tolerance) [45] |
| | | | Configuration | e.g., database, code, connection [47] , security policy [48] |
| | | Monitoring | | e.g., usage bills, access logs [44] |
| | Time to market [44] | Technical/Experience skills [40] | | |
| | | Application complexity | | |
| **Performance** | Elasticity | Workload variation | Application type | e.g., data handling/exchanged amount [50], computation-intensive [50] |
| | | | Peak usage [50] | e.g., avg. duration of load peaks/ year, avg. # of load peaks/year |
| | | | # of users [50] | |
| | | | Geographic distribution of users [50] | |
| | | | Static content [51] | static contents (e.g., images, audio files) are better candidates to move to the Cloud |
| | | Scalability | Multi-tenancy awareness [37] | e.g., virtualization overhead |
| | | | Concurrency [37] | e.g., to support data replication between (VM) instances |
| | | | Scaling latency [37] | e.g., resource availability, application architecture |
| | | | Replication [52] | |
| | Latency | Distance between on-premise and off-premise layers of application [37] | | |
| | | Distance between application and user [37] | | |
| | | Distance between VM instances [37] | | e.g., in the same data center, region. |
| | | Traffic shaping [53] | | which leads to high latency |
| | | Availability of fast dedicated connections [37] | | |
| | | Available bandwidth | | |
| | | Input/Output operations per second [51] | | |
| **Support** | Consulting | | | e.g., consulting prior to application oustourcing [54] |
| | Maintenance | | | e.g., maintenance for deployed application accross several providers [39] |

**Table 2.1:** Example of criteria for the requirements of applications

As the complexity of outsourcing applications increases, Cloud providers are required to offer more heterogeneous Cloud plans to better satisfy the requirements of applications. Therefore, considering the requirements of applications and the characteristics of Cloud plans, selecting suitable Cloud services is usually far from straightforward. For example, a computation-intensive application (e.g., a signal processing application) which works on publicly available data, may require a Cloud service with high performance features (e.g., CPU rates, disk speed). However, another application that processes sensitive data is mostly interested in a Cloud service with high security features (e.g., encryption algorithms, authentication mechanisms). Cloud providers usually offer Cloud services in the form of *plans* that differ in the services they offer, the quality of services they guarantee, and the price lists they apply (e.g., a (set of) virtual machine (VM(s)), dedicated hosts, storages). This variety provides great advantages for users, enabling them to choose the plan that better suits their needs and economical availability. Today, there are three main types of Cloud plans can be considered in the Cloud market which is presented as follows.

- *Provider-defined.* The characteristics of these Cloud plans are defined by Cloud providers, and as a result, provide the least flexibility for users as they need to select plans among those offered by Cloud providers. The advantage here is that Cloud users can choose plans considering their available characteristics which can ease the selection process. However, users need to know about the low-level information (e.g., implementation and management details) of Cloud plans which are not usually revealed by Cloud providers [55].

- *User-defined.* The characteristics of these plans are defined by users considering the requirements of outsourcing applications, which as a result, they provide the maximum flexibility for users as they can ask for the most tailored plans for their applications. However, users need to be technically aware of the requirements of applications.

- *Hybrid.* Users can select these plans, offered by Cloud providers, and customize them according to the requirements of applications. However, the flexibility of such plans might suitably not meet the requirement of applications as Cloud providers usually do not allow for the full customization of a hybrid plan.

The availability of various Cloud plans with heterogeneous characteristics has both favorable and unfavorable consequences. On the one hand, the variety of Cloud plans gives users the freedom of selecting Cloud plans that better fit the requirements of applications. One the other hand, selecting Cloud plans, over other available ones, is considered as an essential challenge as it has clear consequences on the quality of provided services (e.g., a plan with frequent downtimes would cause considerable inconveniences to users trying to interact with the applications deployed over it). Therefore, apart from issues and difficulties in choosing the type of Cloud plans (e.g., the familiarity of users with Cloud scenarios, the allowed customization level of Cloud plans by Cloud providers), selecting suitable plans is a fundamental issue which should be carefully investigated. To deal with these issues, the Cloud computing literature have addressed various issues in Cloud plan selection scenarios which will briefly be discussed in sections 2.2.1 and 2.2.2.

### 2.2.1 CLOUD PLAN SELECTION IN A SINGLE APPLICATION CONTEXT

In this section, we will discuss some related works (e.g., [56], [57], [58], [59], [60]) that consider Cloud plan selection in a single application context.

**Service discovery and selection.** The authors in [56] consider a scenario characterized by a set of Cloud providers which can publish the characteristics of their services and a user which selects the one that better satisfies the requirements of her application. To do so, the authors use a framework called Resources Via Web Services (RVWS) [61] which allows web services to be stateful, and as a result, current information about the resources of services is kept locally in the WSDL documents of services. Therefore, resources do not need to be examined every time a WSDL document is called. For a service, two attributes are considered: *1) state,* which covers the current availability of a service and associated resources; *2) characteristics,* which represents the nonfunctional attributes of offered services (e.g., cost, QoSs). To support the discovery of available services, a dynamic broker is considered which allows publishing detailed information about services, keeping them updated, and finding suitable resources and services based on application requirements. To support service selection, the user provides three groups of requirements including, service, resource, and provider with the possibility of specifying their associated attributes on filtering. Finally, the dynamic broker finds matching services, resources, and providers.

**Cloud plan selection based on multi-level criteria.** The authors in [57] proposed a framework called Service Measurement Index Cloud (SMICloud) to select a suitable Cloud service for an application. The authors, as the requirements of an application, consider some attributes (e.g., performance, assurance), each with two levels of sub-attributes (e.g., serviceability and free support, considering assurance attribute). Also, they provide a flexible ranking model to support various types of attributes, including boolean (e.g., free support), range (e.g., the initiation time of a VM), unordered set (e.g., the number of supported platforms (portability)), and numerical. Then, the authors adopt a multi-criteria decision making (MCDM) technique called Analytical Hierarchical Process (AHP) to properly assign a different importance to each considered attribute and sub-attribute, and then, rank available Cloud services.

Another major line of work focuses on security issues in multi-Cloud scenarios, proposing solutions to protect integrity (e.g., [62, 63, 64, 65]) and confidentiality of accesses (e.g., [66, 67]) to data outsourced to multiple providers. While sharing with us a scenario characterized by multiple Cloud providers, these proposals are complementary to our provided methods and tools for Cloud plan selection in this thesis as they specifically focus on the enforcement of protection mechanisms. All the above-mentioned works operate in a single-application context. That is, Cloud plans are selected based on a single set of requirements. However, the problem of selecting Cloud plan selection can get even more complicated when multiple applications, at the same time, are moved to the Cloud because each application can have different (and possibly contrasting) requirements. For example, an application with unpredictable workload patterns is more interested in plans with efficient elasticity algorithms, while an "always-on" application (e.g., mission-critical web applications) cares more about the availability characteristics of plans (e.g., the number of replicas, failure recovery time). Therefore, users need to properly combine applications' requirements and choose plans that satisfy them in the best possible way. Since the focus of proposed models and tools in this thesis is on supporting Cloud plan selection in a multiple-application context, we will review some related works in this context in the following section.

### 2.2.2 CLOUD PLAN SELECTION IN A MULTIPLE-APPLICATION CONTEXT

The research community have addressed Cloud plan selection in a multiple-application context (e.g., [60], [68], [69]). In the following, we will briefly review some interested related works in this area.

**Supporting security requirements as constraints.** An interesting method proposed in [68] to support considering various security requirements when the best suite of services is selected. The authors define security requirements in term of three sets of constraints, including application-oriented, infrastructure-oriented, and global constraints. For example, an application-oriented constraint, called *Restrict*, implies that all available VM instances are required to place within a given community area (e.g., within EU countries), based on security and privacy policies, and government enforced obligations. Also, another application-oriented constraint, called *Distribute*, implies that if user replicates her application on two VM instances, to avoid single points of failure, they never should be located on the same physical host at the same time. *Forbid* is an example of infrastructure-oriented constraint which indicates that VM instances, owned by a user, cannot place on some specific physical hosts as they are considered for system-level services (e.g., access control engines). An infrastructure-oriented constraint, named *Count*, limits the number of VM instances on a physical host to avoid their performance degradation. Also, *resource capacity*, as a global constraint, states that the amount of resources, consumed by all VMs that are located on a physical host, cannot exceed the total capacity of physical host in any dimension (e.g., CPU, memory). After constraint definition phase, the authors map available VM instances on a set of physical hosts which are classified into some clusters. To do so, adopting a greedy heuristic-based algorithm, available clusters and physical hosts are analyzed and those can perform VM provisioning are identified, considering the defined constraints. The algorithm tries to map available VM instances to a cluster with the highest amount of available resources to reduce the load variance between clusters. Also, to reduce energy consumption costs, when a cluster with maximum resource availability is selected, each physical host within the cluster is analyzed to map as many VM instances as possible on that physical host. While the work in [68] shares with our work (see

Chapter 4) the idea of the satisfaction of multiple sets of user requirements (in terms of constraints), it also focuses on the reduction of the energy costs as well as the running physical hosts from the provider's point of view. Therefore, it is different from our scenario as it is not completely based on the consensus among the sets of user requirements on their satisfaction level.

**Composition-based Cloud provider selection.** In [69], the authors suggest a method for selecting the best set of Cloud providers, according to multiple sets of requirements. To do so, first, the authors, capture major relationships between user requirements and Cloud providers, while filtering the minor ones which are considered as noises, adopting Singular Value Decomposition Technique (SVD) [70]. Then, the centroid values of user requirements are calculated as the weighted sum of user requirements. Finally, a set of Cloud providers is selected based on measuring Cosine similarity between the obtained centroid values of requirements and the characteristics of Cloud providers. The work in [69] is different from ours (see Chapter 4) as it focuses on selecting a set of Cloud providers, while we consider the selection of a single Cloud plan.

In this thesis, considering Cloud plan selection in a multiple-application context, we followed two major lines of research: supporting the business objectives of users (see Chapter 6) and their uncertainty about the requirements of applications (see Chapter 5). Therefore, in the following subsections (Section 2.2.2.2 and Section 2.2.2.1), we will review some related works, considering these lines of works, respectively.

#### 2.2.2.1 CLOUD PLAN SELECTION UNDER UNCERTAINITY

Traditional techniques for selecting suitable Cloud plans for applications implicitly assume that users are aware of the technical details of the requirements of applications. However, it is not always the case as users, possibly without an IT background, can also be interested in moving their applications to the Cloud. Therefore, since such users may not have precise ideas about the requirements of applications, quantitative approaches might not precisely capture the uncertainty of users about applications' requirements, and as a result, select plans that do not adequately meet them. Supporting Cloud plan selection under uncertainty studied also in the past (e.g., [71], [72], [73], [74], [75]). In the following, we review some interesting works in this area.

**Supporting service composition for unskilled users.** A method in [71] proposed which tries to support users with vague ideas about the requirements of multiple applications by simplifying the process of the selection of a set of services. The main components of proposed architecture are as follows: *1) user portal* which presents all available services to users. Also, it provides some graphical interfaces for capturing user requirements; *2) translator* which translates Cloud service information to a provided web service modeling language format; *3) Cloud service repositories* which maintains Cloud services' information; *4) discovery and negotiation service* which maps requirements to resources, using a provided ontology-based discovery technique; *5) composition* which builds possible service compositions and excluding incompatible ones; *6) optimizer* which evaluates service composition candidates, considering user's QoS preferences;

*7) planning* which determines the order of deploying services; *8) image packaging* which builds the discovered services and meta-data into deployable packages (e.g., Amazon machine images); *9) deployment* which configures and sets up services (e.g., firewall configurations). The authors consider four composition criteria including, deployment cost, deployment time (i.e., required time for deploying a service on a VM), reliability, and compatibility. Considering deployment cost, the authors define some categories including, acquisition cost (e.g., licensing cost), ongoing cost (e.g., data transfer cost), and decommissioning cost (e.g., costs related to archiving and removing data at the end of application lifecycle). Considering reliability, the authors introduce a metric called SLA confidence level which measures how services are reliable, considering their associated SLAs and their performance history. The final objective is to find a fully-compatible service composition for a set of applications which minimizes deployment time and cost, and improves reliability. To do so, first, Pareto front composition solutions are identified using multi-objective algorithms, considering the preferences of users which are described by defining some high level "if-then" rules to build a fuzzy rule-based engine. The proposed fuzzy inference engine has three inputs including, deployment time, deployment cost, and composition reliability. The output of fuzzy inference engine is the desirability level of composition. Then, the best Pareto front composition solution is chosen by ranking the identified ones. The work in [71], which shares the idea of supporting Cloud plan selection for unskilled users, is different from ours as it provides a composition of services to meet the requirements of applications, while we consider the selection of one single Cloud plan. Also, the authors in [71], contrasting to us, focus on Cloud service selection in a single-user context (i.e., a single user contributes to the service selection process).

**QoS-based service composition optimization.** In [72], a method is proposed for the optimization of multiple service selection for a complex job which includes several simple jobs (sub-jobs). The proposed service composition optimization is carried out in four steps: *i) job analysis* which the QoS importance of each job is measured in this step. For this, the authors consider four categories of QoS indices including, market or manufacturing response time, cost, quality, and the guaranteed ability of contract. Also, for each QoS index, a set of sub-indices are considered (e.g., success rate and reliability, considering the guaranteed ability of contract). Then, QoS indices and their sub-indices are organized into a hierarchy in order to calculate the job-specific importance of all QoS indices and their sub-indices, adopting fuzzy AHP technique; *ii) service discovery* which in this step, through a semantic matchmaking process (e.g., semantic UDDI [76]), a list of alternative services is obtained for each sub-job; *iii) SLA negotiation* between service provider and consumer to get the volumes of each service's QoS indices; *iv) establish a multi-objective optimization model* for service composition, considering optimization objective and constraints (e.g., for each job, maximizing the performance/cost ratio of services, while ensuring that total time and price do not exceed time budget and price budget); *v) solve the model* to get the optimal solution, adopting an optimization algorithm. The work in [72], apart from service composition-based scenario, is different from ours in Chapter 5 as we focus on selecting a single Cloud plan when multiple users contribute to the Cloud plan selection process. Also, the authors in [72] assume that sub-jobs are to be equally important, in contrast to our scenario where each application can have a different imprecise importance.

**Hierarchical Cloud service selection.** In [73], a Cloud trust evaluation system is proposed for selecting the best Cloud service, considering numerical (e.g., CPU speed) and linguistic (e.g., security policy, network security) requirements. To do so, the authors generate a hierarchical fuzzy inference system which includes four components: *1) web interface* which provides users an interface to submit their functional and non-functional requirements; *2) discovery service* which retrieves services that can meet functional requirements (e.g., the number of CPU cores, memory amount), static QoS requirements (e.g., security, policy), and business policies; *3) Cloud benchmark service* which constantly monitors the performance of Clouds by running benchmark services; *4) trust evaluation service* which returns the best service, according to considered criteria. The proposed hierarchical fuzzy inference system is composed of connected atomic typical fuzzy inference modules, defined for considered criteria. The inputs of upper-level modules in the proposed hierarchical fuzzy inference system are the outputs of lower level modules. The inputs of leaf interface modules are obtained from the services' past benchmark results. The system dynamically generates fuzzy "If-Then" rules for each module, according to user requirements and a pessimistic rule generation strategy (i.e., the output trust level is acceptable only if all input variables satisfactory). In this manner, the proposed hierarchical evaluation framework ranks available services, according to their evaluated trust values, user requirements, and the services' benchmarking results. The work in [73], while shares with our work in Chapter 5 the idea of selecting a Cloud service/plan, is different from our scenario as we consider multiple users who contribute to the process of selecting a Cloud plan for multiple applications.

**Hybrid Cloud service selection.** In [74], the authors propose a fuzzy decision-making framework for selecting Cloud services. The authors introduce a fuzzy Cloud ontology to model the relationship between service concepts (e.g., network management) and service properties (e.g., service response time) as well as the relationship between service concepts. Also, the proposed fuzzy Cloud ontology supports the calculation of similarity between Cloud service concepts and the query of service compositions. Then, the authors choose $k$-top services having the highest similarities with the submitted query. Next, adopting fuzzy AHP, the importance of non-functional properties (e.g., QoSs, price) is calculated. Finally, adopting fuzzy TOPSIS, candidate services are rated. The final ranking of services is based on the linear combination of service similarity with the submitted query and service ratings. The work in [74], which focuses on the selection of Cloud services in a single-user context, is different from ours proposed in Chapter 5 as we consider selecting a Cloud plan for multiple applications with different and possibly contrasting preferences over a set of criteria (e.g., availability, performance) in a multiple-user context.

**Group-based Cloud service selection.** In [75], a multi-attribute group decision-making tool is proposed to help users for selecting Cloud providers. To do so, a committee of decision makers (DMs) is formed and asked to provide their qualitative assessments about Cloud providers for subjective attributes (e.g., technology, environment) and quantitative assessments for objective attributes (e.g., cost). The subjective and objective assessment of attributes is calculated using linguistic weighted arithmetic averaging (LWAA) [77] operator and the statistical variance of quantitative assessments, respectively. Then, based on a weighted combination method, the subjective and objective preferences of attributes are combined to obtain their integrated preferences. Next, considering the integrated attribute preferences, the objective (adopting an improved TOPSIS method [78]) and subjective (adopting a Delphi-AHP method [79]) importance of DMs are measured. Then, the integrated importance of each DM is calculated as the linear combination of subjective and objective importances. Finally, Cloud providers are scored and raked, according to the aggregated decision opinions of DMs, considering the preferences of attributes and the importance of DMs. The work in [75], while shares with our work in Chapter 5 the idea of selecting a Cloud plan/provider in a group-based context, is different from ours as we consider the selection of a Cloud plan for multiple applications, each with possibly a different importance.

### 2.2.2.2 BUSINESS-ORIENTED CLOUD PLAN SELECTION

The growing popularity of Cloud computing technology lies in several reasons. Among them, the business-effectiveness of Cloud plans is considered as one of the important ones. In fact, for a majority of Cloud users, the cost-effectiveness of Cloud solutions is one of the main motivations for outsourcing their data and applications to the Cloud.

As we mentioned before in Section 2.2, a Cloud plan can be any type of customized Cloud service such as a (set of) VM(s). In VM provisioning scenarios, when multiple applications are moved to the Cloud, they usually are scheduled by mapping each one to a VM, among a set of available VMs. Here, a fundamental challenge is supporting the business objectives of users in application scheduling scenarios. In particular, when applications to be outsourced are business-critical ones (e.g., e-commerce applications), users expect not only minimizing the risk of the financial loss of applications but also efficiently supporting their financial profit, when they are executed on the Cloud. Then, it is essential to carefully investigate the process of the scheduling of such applications to suitably support the business objectives of users w.r.t. the financial profit of applications. In the literature, there are several business-oriented methods to support VM provisioning scenarios (e.g., [80], [81], [82], [83], [84]). In the following, we review some interesting related works in this area.

**SLA-based resource provisioning in virtualized Cloud datacenters.** In [80], the authors proposed a method for improving resource provisioning in virtualized Cloud data centers w.r.t. the profit of Cloud providers. The proposed method includes four key components: *1) admission control* which decides whether the requested VM for an application can be allocated and the QoS requirements can be met if the requested VM is allocated; *2) VM manager* which initiates a VM and allocates it to a physical host having the required capacity; *3) job scheduler* which schedules applications on newly initiated VMs; *4) SLA manager* which monitors current SLAs for each application. Also, the authors consider two types workloads: *i) transactional* workloads (e.g., web applications with time-varying workloads); *2) non-interactive* workloads which there is no communication between tasks. For transactional workloads, the authors consider three types of penalties for SLA violation including, fixed, proportional, and delay-independent. The fixed penalty, as its name clearly states, is applied whenever a Cloud provider fails to meet current resource capacity demand. The delay-independent penalty is proportional to delay incurred by a Cloud provider in returning the resource capacity. The proportional penalty is proportional to delay that is incurred by a Cloud provider in returning some resource capacity, considering times that the resource capacity was requested and allocated. For non-interactive workload, the authors consider a penalty considering the required number of CPU cycles and a deadline to provide such amount of resource. The authors used artificial neural network techniques to forecast the future workload (VMs' CPU utilization) and predict the amount of available resources in future. Based on such prediction, it is decided if a new application can be accepted for assigning to available VM(s) or not. Finally, for resource provisioning, the priority is given to applications with lower penalty rates. The work in [80], which shares with our work presented in Chapter 6 the idea of mapping applications to VMs by considering SLA penalty rates to support the profit of Cloud providers, is orthogonal to our scenario as we focus on supporting the business objective of users w.r.t. the financial profit of applications.

**Auction-based VM provisioning.** In [81], a method is proposed to efficiently allocate resources in a combinatorial auction system and reduce SLA penalty cost. The authors define SLA in terms of a deadline for the execution of a job. They consider a scenario in which a set of users request computing resources in the form of VM instances. Each user requests resources for a job by submitting a bid including, the number of VM instances for each type of VM and price that she is willing to pay for using the requested bundle of VMs. Cloud provider runs an auction mechanism periodically and users bid for VM bundles for a unit of time. If a user needs VM instances for more than one unit of time, she needs to bid periodically. The bidding process is continued until either its deadline is exceeded or the job execution is completed. To maximize the profit of Cloud provider by reducing penalty cost, the probability of deadline violation is computed, and then, the profit of Cloud provider is estimated based on the deadline violation probability obtained in the previous step, the revenue of VMs, their running cost, and SLA violation penalty cost. The authors conclude that jobs with impending deadlines are more likely to be determined as auction winners because the probability of allocating resources to them is higher as they incur less SLA penalty for Cloud providers. The proposed work in [81] focuses on maximizing the profit of Cloud provider, in contrast to our scenario which tries to maximize the financial profit of applications, owned by users.

**Profit optimization in VM provisioning scenarios.** In [83], the authors propose a constraint-based approach to support the overall profit optimization of Cloud providers in VM provisioning scenarios. To do so, they consider a scenario which consists of a set of virtual clusters (VC). Each VC is defined as a set of VMs, running on private resources and possibly some VMs rented from public Cloud providers. Also, each VC is associated with a specific application type which is managed by a corresponding programming framework (e.g., for batch application types, it is possible to use Oracle Grid Engine (OGE)). Each VC hosts a subset of applications. Also, each application runs on a set of VMs, including private and public ones. The proposed profit estimation policy is called when a new request to deploy an application is received and no resource is available on the associated VC to run the application. To meet the objective of optimization policy (i.e., maximizing the overall profit of Cloud provider), the authors try to avoid getting resources from public Cloud providers by giving private resources to a new request. Such resources could be obtained from private resources that are already assigned to running applications. However, the performance of such applications could be decreased, and as a result, guaranteed QoSs in their SLAs could be violated. To avoid such situations, the authors ensure, as a constraint, that profit achieved from a request for hosting a new application would be more than the penalties for impacted applications. Also, taking resources from already hosted applications could impact the reputation of Cloud provider. To avoid such situations, the authors consider a constraint which implies that the percentage of impacted applications should be less or equal to a predefined threshold. The authors propose two ways to provide resources for a new request. First, resources are borrowed from running applications during their execution which could impact on their performance with a cost called bid which is the sum of penalties for the impacted applications. The second way is waiting until some resources are released by some running applications which impacts on the resource request with a waiting time. When a new resource request to host an application arrives, among the VM prices of public Cloud providers, the cheapest one is selected. Then, all available VCs provide their waiting time and a bid, according to the percentage of impacted applications. Then the smallest bid that the associated percentage of impacted applications is less than a predefined threshold is selected. Next, the cost of shortest waiting time that the percentage of impacted application is less than the predefined threshold is calculated. Further, the cost of selected bid and waiting time is compared, the minimum one is selected, and added to the cost of running application on the private resources. Finally, the cost of private resources and public resources is compared, and the resource type (public or private) with the minimum cost is chosen to host the new application.

## 2.3 CHAPTER SUMMARY

In this chapter, we discussed some related works in the literature that addressed issues discussed in this thesis. We started with reviewing some related works that evaluate the modularity of applications to see to what extent they can be easily moved to the

Cloud w.r.t. adaptation capability and/or decomposability. Then, we focused on related works that addressed Cloud plan selection in a single-application context which a set of requirements is considered for selecting Cloud plans. Next, we presented some related works for Cloud plan selection in a multiple-application context with a special focus on supporting the uncertainty of users about the requirements of applications as well as the business objectives of users.

# 3

# MODULAR DESIGN EVALUATION IN OUTSOURCING SCENARIOS

Cloud computing environments are highly dynamic because Cloud resources continuously evolve, according to changes that occur in such environments. Therefore, to run effectively on the Cloud, outsourcing applications should be able to adapt themselves to dynamic frequent changes in Cloud environments. Moreover, since applications can be distributed across several Cloud environments, they are required to suitably be decomposed into smaller parts. In this chapter, we focus on the evaluation of applications to see to what extent they can be easily moved to the Cloud w.r.t. change adaptability and/or decomposability, considering the dynamic and distributed nature of Cloud environments.

## 3.1 INTRODUCTION

Today, considering the significant benefits of Cloud computing, compared to traditional on-premise computing, more and more organizations as well as individuals are moving their applications to the Cloud. However, due to the specific properties of Cloud environments, applications should be carefully assessed to see to what extent they are ready to be moved to the Cloud. In this chapter, we consider the dynamic and distributed properties of Cloud environments to evaluate applications which will be discussed in detail in the following.

*Dynamic.* Cloud computing environments are highly dynamic and unpredictable [85] due to several reasons. To mention a few, *1) resource management* (e.g., virtual machine (VM) migration, VM consolidation) is highly dynamic to properly meet the elasticity requirements imposed by available pay-as-you-go pricing models [86]; *2)fault tolerance strategies*, which imply the possibility of system accommodation to faults arising in the course of operation [87], should have dynamic features [88]; *3) introducing new laws*

*and regulations* (e.g., security and privacy regulations), *standards* (e.g., REST, SOAP), *services*, and *updates* increase the dynamicity of Cloud environments; *4) service level agreements* (SLAs) can change frequently, according to changes in the quality of services (QoSs) (e.g., response time, real-time throughput [89]), guaranteed by Cloud providers, penalty mechanisms when SLA is violated, service price). Therefore, the dynamic nature of Cloud environments implies that applications need to be flexible enough to properly accommodate themselves to variant changes in such environments. Therefore, applications that are not properly configurable and customizable cannot run effectively in Cloud environments [9].
*Distributed.* The essence of Cloud computing technology is inherited from distributed computing [90]. Distributed Cloud data centers across the world include many computing physical machines [91] [92]. Also, thanks to the virtualization technology, each physical machine can host several VMs, each with possibly different characteristics (e.g., CPU rates, memory, price). Therefore, considering the distributed nature of Cloud environments, applications are usually distributed across several VMs [93] and even several Cloud providers [94]. In addition, in some outsourcing scenarios, applications are needed properly distributed across public and private Clouds [95]. That is, some applications' components are needed to be executed locally, while other ones should be executed on the Cloud. Therefore, applications are required to be properly decomposable into smaller parts with separate functionalities to be efficiently distributed across Cloud environments.

Considering the second objective of this thesis to support users in selecting suitable Cloud plans for applications, we need to, first, evaluate applications to see to what extent they can be easily moved to the Cloud. Otherwise, even selecting suitable Cloud plans for applications would not be efficient, and as a result, it might hinder further Cloud computing adoption. In this chapter, we provide an approach aimed at evaluating the modularity of applications to estimate to what degree they are ready to be moved to the Cloud w.r.t. their change adaptability and/or decomposability. In fact, modular applications, due to having functional independent components (i.e., components with "single-minded" function and "aversion" to excessive communication with other components [10]), are properly reconfigurable and distributable. In the following sections, we present our proposed method for evaluating the modularity of applications in detail.

### 3.1.1 CHAPTER OUTLINE

This chapter is structured as follows. Section 3.2 provides some basic concepts on the problem and its definition. Section 3.3 presents our approach for the defined problem. Finally, we provide chapter summary and concluding remarks in Section 3.4.

## 3.2 BASIC CONCEPTS AND PROBLEM DEFINITION

To evaluate modularity for an application *App*, we consider a bottom-up approach. That is, we estimate the modularity of application *App*, first, at micro (component) level, denoted as $Q^{(c)}$, and then, at macro (system) level, denoted as $Q^{(s)}$. Application *App* includes a set $E^{(c)} = \{e^{(c)}, \dots, e_n^{(c)}\}$ of component-level entities (e.g., $E^c = \{e_1^{(c)}, e_2^{(c)}, e_3^{(c)}\}$ in our running example). Each component-level entity $e_i^{(c)} \in E^{(c)}$, which hereafter simply called *component*, is composed of $h$ classes maintained in a *class set* $\mathbb{C}_i = \{\mathbb{c}_{i,1}, \dots, \mathbb{c}_{i,h}\}$.

As the first step to estimate the modularity of application $App$, it is important to consider necessary modularity attributes, according to the context of problem (e.g, application type, QoS guarantees). Such attributes are maintained in an *attribute set* $\mathbb{A} = \{\mathbb{a}_1, \dots, \mathbb{a}_m\}$ (e.g., $\mathbb{A} = \{$*Coupling,Cohesion,Size*$\}$ in our running example). We note that in the literature, coupling, cohesion, and size are known as three main modularity attributes, which as a result, are suitable candidates to be considered for the set $\mathbb{A}$ of attributes in our running example. In the following, we will present these attributes in more detail.

- *Coupling.* This attribute is defined as a measure of interconnection among the components of application [10]. Higher coupling values for an application reflects a greater difficulty to change its components because a change in one component may have an impact on other components that are coupled to it [22]. Moreover, a high coupling for an application indicates its low efficiency when it is distributed across Cloud environments due to the tight dependency between its components. Therefore, coupling is considered as an attribute with a negative impact on modularity.

- *Cohesion.* This attribute is defined as a measure of the degree to which a component focuses on just one single task [10]. High cohesion for an application reflects the suitable division of functionalities among components, and as a result, proper change flexibility and decomposability in dynamic Cloud environments. Therefore, cohesion is considered as an attribute with a positive impact on modularity.

- *Size.* This attribute is one of the common indicators for maintenance effort and re-configurability [96]. In fact, high size for an application reflects its high complexity when it is re-configured. Therefore, application size is considered as an attribute with a negative impact on modularity.

Also, for each attribute $\mathbb{a} \in \mathbb{A}$, some metrics are considered both at component level and at system level, which respectively maintained in a *component metric set* $M^{(c)} = \{m_1^{(c)}, \dots, m_p^{(c)}\}$ and a *system metric set* $M^{(s)} = \{m_1^{(s)}, \dots, m_q^{(s)}\}$. Figure 3.1 shows a classification for modularity evaluation in our running example which clearly shows the considered attributes and their associated metrics. We note that modularity attributes and metrics can be defined considering several parameters such as application architecture (e.g., peer-to-peer, client-server), application type (e.g., real-time, web-based), and the execution context of application (Cloud service change rate, the level of application distribution), etc. by referring to existing guidelines and classifications (e.g., [31], [32]). Figure 3.2, inspired from [21], shows the static relation between modularity and the associated attributes in the set $\mathbb{A}$ of attributes as well as the respective metrics in the sets of component metrics $M^{(c)}$ and system metrics $M^{(s)}$. In

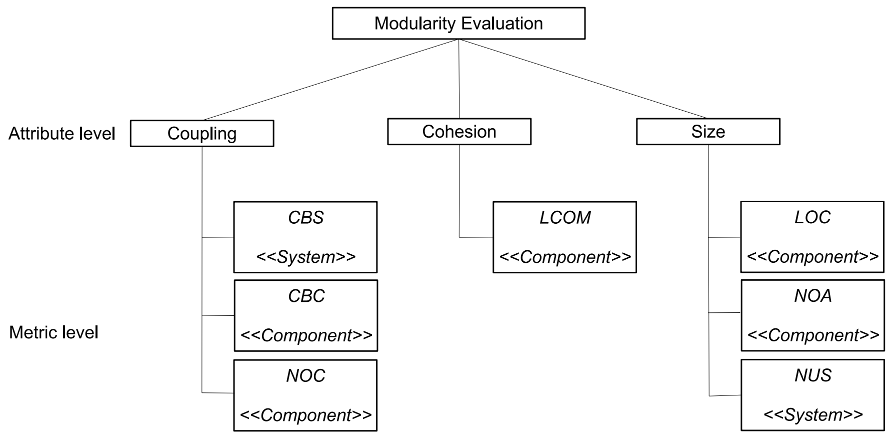

**Figure 3.1:** Example of modularity evaluation classification

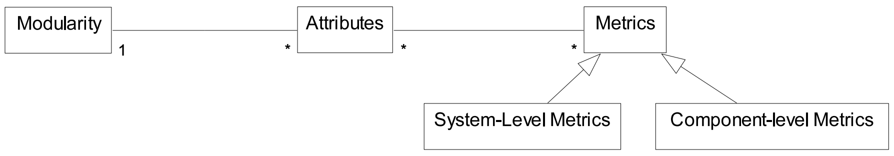

**Figure 3.2:** Modularity evaluation meta-model

our running example, the sets of component metric and system metric respectively are $M^{(c)} = \{CBC,NOC,LCOM,LOC,NOA\}$ and $M^{(s)} = \{CBS,NUS\}$, as depicted in Table 3.1 and Table 3.2. We will discuss about these metrics later in this chapter (see Section 3.3), in more detail.

Figure 3.3, inspired from [21], shows our two-level modularity evaluation model in our running example. We note that, according to Figure 3.3, we consider component-level modularity $Q^{(c)}$ as an external metric for system-level modularity. The motivation behind such consideration lies in providing more flexibility in evaluating system-level modularity. In fact, when the whole application is moved to the Cloud, micro-level modularity may not have a high relevance in evaluating change flexibility and/or decomposability as we are more focused on the evaluation of application-level modularity by considering system-level entities and their associated metrics. Therefore, by considering component-level modularity as a metric for system-level modularity, we can manage its impact on system-level modularity.

Also, not all metrics in the sets of component metrics $M^{(c)}$ and system metrics $M^{(s)}$ can be assumed to be equally important for modularity evaluation. For example, metric *CBC* can have a higher importance compared to *NOC* and *LOC*. To express the importance/relevance of each component-level metric $m_j^{(c)} \in M^{(c)}$ and system-level metric $m_j^{(s)} \in M^{(s)}$, they are associated with a *weight*, where higher weights model higher importance/relevance of the metric for modularity evaluation. Formally, component metric set $M^{(c)}$ and system metric set $M^{(s)}$ are associated with a *component-level weight vector* $W^{(c)}[1,\ldots,|M^{(c)}|]$ and a *system-level weight vector* $W^{(s)}[1,\ldots,|M^{(s)}|+1]$, respectively, where $W^{(c)}[j]$ and $W^{(s)}[k]$ are the weights of metrics $m^{(c)}[j]$ and $m^{(s)}[k]$, respectively. To enable comparison among component metric weights and system metric

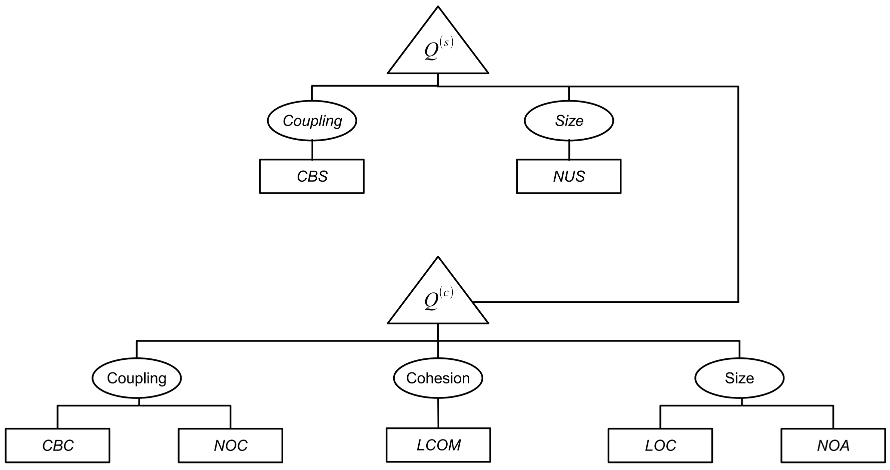

**Figure 3.3:** Two-level modularity evaluation model

| | $\mathbf{M^{(c)}}$ | **Impact on Modularity** |
|---|---|---|
| **Coupling** | Coupling Between Components *(CBC)* [97] | Negative |
| | Number Of Children *(NOC)* [24] | Negative |
| **Cohesion** | Lack of COhesion in Methods *(LCOM)* [24] [27] | Negative |
| **Size** | Lines Of Code *(LOC)* [98] | Negative |
| | Number Of Attributes *(NOA)* [99] | Negative |

**Table 3.1:** Example of component metric set $M^{(c)}$

| | $\mathbf{M^{(s)}}$ | **Impact on Modularity** |
|---|---|---|
| **Coupling** | Coupling Between Servers *(CBS)* | Negative |
| **Size** | Number of Usecases for a Service *(NUS)* | Negative |

**Table 3.2:** Example of system metric set $M^{(s)}$

weights, we assume that vectors $W^{(c)}$ and $W^{(s)}$ are normalized (i.e., $\sum_{j=1}^{|M^{(c)}|} W^{(c)}[j] = 1$ and $\sum_{k=1}^{|M^{(s)}|+1} W^{(s)}[k] = 1$). We note that, as we discussed before, we consider modularity at component level as a metric for system-level modularity. Therefore, $W^{(s)}[|M^{(s)}| + 1]$ represents the weight of component-level modularity $Q^{(c)}$. Table 3.3 shows component-level weight vector $W^{(c)} = [0.40, 0.05, 0.20, 0.05, 0.30]$ and system-level weight vector $W^{(s)} = [0.50, 0.30, 0.20]$. For example, $W^{(c)}$ states that metrics *NOC* and *LOC* have the same relative importance (0.05 each), while metric *CBC* is more relevant (0.40). Also, $W^{(s)}[3] = 0.20$ represents the weight of component-level modularity $Q^{(c)}$ in evaluating system-level modularity.

|  | $\mathbf{W}^{(c)}$ |
|---|---|
| *CBC* | 0.40 |
| *NOC* | 0.05 |
| *LCOM* | 0.20 |
| *LOC* | 0.05 |
| *NOA* | 0.30 |

|  | $\mathbf{W}^{(s)}$ |
|---|---|
| *CBS* | 0.50 |
| *NUS* | 0.30 |
| $\mathbf{Q}^{(c)}$ | 0.20 |

**Table 3.3:** Example of component-level weight vector $W^{(c)}$ and system-level weight vector $W^{(s)}$

## 3.3 PROPOSED APPROACH FOR MODULARITY EVALUATION

In [20], a method is proposed to estimate software quality attributes (e.g., maintainability), considering their associated quality characteristics (e.g., correctability, testability), each with respective metrics (e.g., fault rate and required effort for testability) [21] [33]. Based on [20], in [21], a solution is proposed for evaluating the modularity of an application, considering only maintainability and re-usability, as modularity attributes. Inspired from these works, our proposed method, first, estimates modularity at component level $Q^{(c)}$ (see Section 3.3.1), and then, at system level $Q^{(s)}$ (see Section 3.3.2). We note that a higher modularity for an application reflects its higher change flexibility and/or decomposability, and as a result, its higher efficiency when it is moved to the Cloud w.r.t. the dynamic and distributed properties of Cloud environments.

### 3.3.1 MODULARITY EVALUATION AT COMPONENT LEVEL

Figure 3.4, which is an extended version of Figure 1. in [100], shows a class diagram associated with application App as well as its comprised components. To estimate component-level modularity $Q^{(c)}$, for each component $e_i^{(c)} \in E^{(c)}$, we need to measure the values of metrics in component metric set $M^{(c)}$, associated with modularity attributes in attribute set $\mathbb{A} = \{Coupling, Cohesion, Size\}$. To enable comparison between all measured metric values at component level, we normalize them. Formally, suppose that $v_i^{(c)}[j]$ is the raw value of metric $m_j^{(c)} \in M^{(c)}$ for component $e_i^{(c)} \in E^{(c)}$. Then, the associated normalized value $\mathbb{V}_i^{(c)}[j]$ is calculated as:

$$\mathbb{V}_i^{(c)}[j] = \begin{cases} \frac{v_i^{(c)}[j] - v_{min}}{v_{max} - v_{min}}, & \text{if } v_i^{(c)}[j] \neq v_{min} \wedge v_i^{(c)}[j] \neq v_{max} \\ 0, & \text{if } v_i^{(c)}[j] = v_{min} \\ 1, & \text{if } v_i^{(c)}[j] = v_{max} \end{cases} \tag{3.1}$$

where $v_{min}$ and $v_{max}$ are the minimum and maximum raw values of $m_j^{(c)} \in M^{(c)}$ for component $e_i^{(c)} \in E^{(c)}$, respectively. Normalized values $\mathbb{V}_i^{(c)}[j]$ for component $e_i^{(c)} \in E^{(c)}$ are maintained in a *normalized component-level vector* $\mathbb{V}_i^{(c)}[1, \ldots, |M^{(c)}|]$. Also, if each metric $m_j^{(c)} \in M^{(c)}$ at component level has a negative impact on modularity, the associated normalized measured value $\mathbb{V}_i^{(c)}[j]$ is subtracted from 1. Therefore, the value $V_i^{(c)}[j]$ of metric $m_j^{(c)} \in M^{(c)}$ with a negative impact on modularity for component $e_i^{(c)} \in E^{(c)}$ is defined as:

$$V_i^{(c)}[j] = \begin{cases} 1 - \mathbb{V}_i^{(c)}[j], & \text{if } m_j^{(c)} \text{ has a negative impact on modularity} \\ \mathbb{V}_i^{(c)}[j], & \text{otherwise} \end{cases} \tag{3.2}$$

where such values $V_i^{(c)}[j]$ of metrics for component $e_i^{(c)} \in E^{(c)}$ are maintained in a *component-level vector* $V_i^{(c)}[1, \ldots, |M^{(c)}|]$. Back to our running example, considering Table 3.1, metrics associated with *Coupling* and *Size* attributes (*CBC*, *NOC*, *LOC*, *NOA*, *CBS*, *NUS*), have negative impacts on modularity. Therefore, their associated normalized metric values are subtracted from 1 (see Sections 3.3.1.1 and 3.3.1.3). Also, metrics associated with *Cohesion* attribute (e.g., tight class cohesion (TCC), loose class cohesion (LCC) [101], and Class Cohesion (CC) [102]) have positive impact on modularity. However, in our running example, metric *LCOM*, as its name clearly states, measures the lack of cohesion for a component, and thus, has a negative impact on modularity. As a result, the normalized measured values of *LCOM* metric are subtracted from 1 (see Section 3.3.1.2).

Also, the overall value $V^{(c)}[j]$ of metric $m_j^{(c)} \in M^{(c)}$ is calculated as the average of the values $V_i^{(c)}[j]$ of metric $m_j^{(c)}$ measured for each component $e_i^{(c)} \in E^{(c)}$, that is:

$$V^{(c)}[j] = \frac{\sum_{i=1}^{|E^{(c)}|} V_i^{(c)}[j]}{|E^{(c)}|} \tag{3.3}$$

where such overall values for component-level metrics are maintained in a *overall component-level vector* $V^{(c)}[1, \ldots, |M^{(c)}|]$. In the following, we will present our approach for the evaluation of component-level modularity.

#### 3.3.1.1 EVALUATION OF COMPONENT-LEVEL COUPLING METRICS

Let us start with the evaluation of component-level metrics associated with attribute *coupling*. To do so, considering Figure 3.1, we need to evaluate *CBN* and *NOC* metrics which is presented in detail in the following.

**Coupling Between Components (*CBC*).** This metric is considered detrimental to modular design [103] which indicates the degree of mutual interdependence between components. Therefore, the more the coupling between components is less, the more they are modular, and consequently, the easier they are to change and decompose. Metric *CBC* measures, for a component, the number of other components that are coupled with the component. To measure the value of this metric, we use a metric called coupling between objects (CBO) [23] which measures, for a class, the number of non-inheritance relations with other classes [104]. An object of a class is coupled to another class if methods or instance associated with the class are used by the other one [97]. Then, we define *CBC* for a component as the sum of coupling *CBO* between the classes

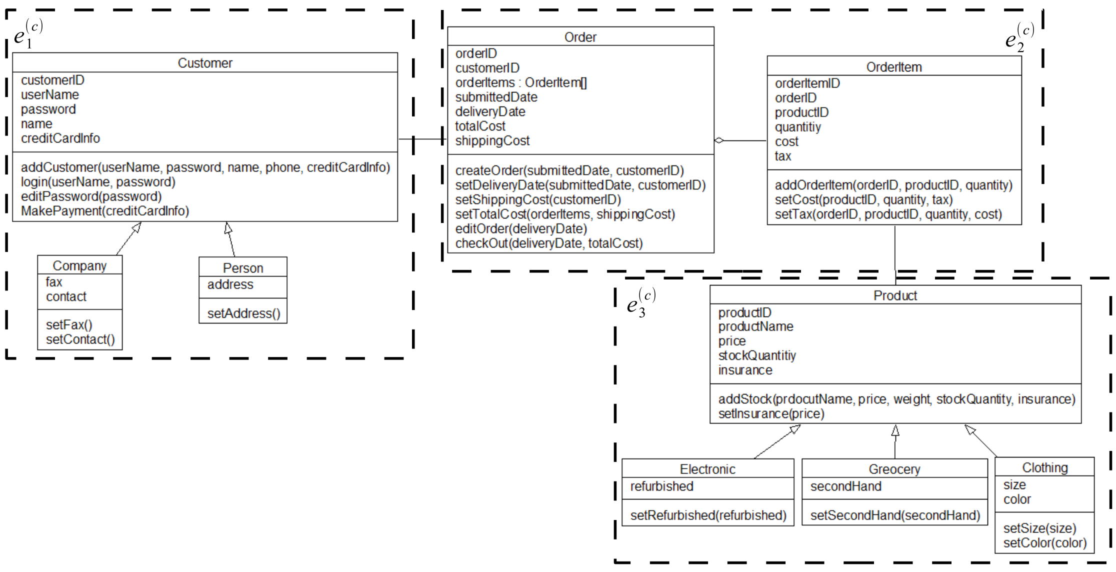
$e_1^{(c)}$
Customer
customerID
userName
password
name
creditCardInfo
addCustomer(userName, password, name, phone, creditCardInfo)
login(userName, password)
editPassword(password)
MakePayment(creditCardInfo)
Company
fax
contact
setFax()
setContact()
Person
address
setAddress()
Order
orderID
customerID
orderItems : OrderItem[]
submittedDate
deliveryDate
totalCost
shippingCost
createOrder(submitedDate, customerID)
setDeliveryDate(submittedDate, customerID)
setShippingCost(customerID)
setTotalCost(orderItems, shippingCost)
editOrder(deliveryDate)
checkOut(deliveryDate, totalCost)
$e_2^{(c)}$
OrderItem
orderItemID
orderID
productID
quantitiy
cost
tax
addOrderItem(orderID, productID, quantity)
setCost(productID, quantity, tax)
setTax(orderID, productID, quantity, cost)
$e_3^{(c)}$
Product
productID
productName
price
stockQuantitiy
insurance
addStock(prdocutName, price, weight, stockQuantity, insurance)
setInsurance(price)
Electronic
refurbished
setRefurbished(refurbished)
Greocery
secondHand
setSecondHand(secondHand)
Clothing
size
color
setSize(size)
setColor(color)

**Figure 3.4:** Class diagram and component architecture for application *App*

of component. Formally, suppose that, $CBO_{i,j}$ denotes the coupling *CBO* between class $c_{i,j} \in \mathbb{C}_i$ in component $e_i^{(c)}$ and classes in other components $E^{(c)} \backslash \{e_i^{(c)}\}$. Then, the raw *CBC* value $CBC_i$ of component $e_i^{(c)} \in E^{(c)}$ is measured as follows:

$$CBC_i = \sum_{j=1}^{|\mathbb{C}_i|} CBO_{i,j} \tag{3.4}$$

Back to our running example, considering Figure 3.4, Table 3.4 shows the raw value $e_i^{(c)}$, normalized value $\mathbb{V}_i^{(c)}[CBC]$, the value $V_i^{(c)}[CBC]$ of metric *CBC* for each component $e_i^{(c)} \in E^{(c)}$, and the associated overall value $V^{(c)}[CBC]$. For example, considering component $e_2^{(c)} \in E^{(c)}$ and the associated classes (*Order* and *OrderItem*), the values of metric *CBO* for classes *Order* and *OrderItem* are $CBO_{e_2^{(c)},Order} = 1$ (due to coupling between class *Order* and Class *Customer* in component $e_1^{(c)} \in E^{(c)}$) and $CBO_{e_2^{(c)},OrderItem} = 1$ (due to coupling between class *OrderItem* and class *Product* in component $e_3^{(c)} \in E^{(c)}$), respectively. Therefore, according to Formula 3.4, the raw value $CBC_2$ of metric *CBC* for component $c_2 \in E^{(c)}$ is calculated as $1 + 1 = 2$ which is the maximum metric *CBC* value among other components in $E^{(c)}$. Therefore, the normalized value $\mathbb{V}_2^{(c)}[CBC]$ of metric *CBC* for component $e_2^{(c)} \in E^{(c)}$ is equal to 1, according to Formula 3.1. Moreover, since metric *CBC* has a negative impact on modularity, to measure the value $V_i^{(c)}[CBC]$ of metric *CBC* for component $e_i^{(c)} \in E^{(c)}$, the associated normalized value $\mathbb{V}_i^{(c)}[CBC]$ is subtracted from 1 (see Section 3.3.1.1 and Formula 3.2). Therefore, considering component $e_2^{(c)} \in E^{(c)}$, $V_2^{(c)}[CBC] = 1 - 1 = 0$. Also, according to Formula 3.3, the overall value $V^{(c)}[CBC]$ of metric *CBC* is calculated as $^{1+0+1}/_{3} = 0.666$.

| | $\mathbf{CBC_i}$ | $\mathbb{V}_i^{(c)}$[CBC] | $\mathbf{V}_i^{(c)}$[CBC] | $\mathbf{V}^{(c)}$[CBC] |
|---|---|---|---|---|
| $c_1$ | 1 | 0 | 1 | |
| $c_2$ | 2 | 1 | 0 | 0.666 |
| $c_3$ | 1 | 0 | 1 | |

**Table 3.4:** Raw values $CBC_i$, normalized values $\mathbb{V}_i^{(c)}[CBC]$, values $V_i^{(c)}[CBC]$ of metric *CBC* for each component $e_i^{(c)} \in E^{(c)}$, and associated overall value $V^{(c)}[CBC]$

| | $\mathbf{NOC_i}$ | $\mathbb{V}_i^{(c)}$[NOC] | $\mathbf{V}_i^{(c)}$[NOC] | $\mathbf{V}^{(c)}$[NOC] |
|---|---|---|---|---|
| $c_1$ | 2 | 0.66 | 0.34 | |
| $c_2$ | 0 | 0 | 1 | 0.673 |
| $c_3$ | 3 | 1 | 0 | |

**Table 3.5:** Raw values $NOC_i$, normalized values $\mathbb{V}_i^{(c)}[NOC]$, values $V_i^{(c)}[NOC]$ of *NOC* metric for each component $e_i^{(c)} \in E^{(c)}$, and associated overall value $V^{(c)}[NOC]$

**Number Of Children (*NOC*).** This metric measures, for a component, the total number of classes inherited from other classes in the component. As the number of the children of a class grows, increasing the number of methods and instance variables that the class is coupled to them is more probable [105]. Therefore, the more the number of the children of a class is, the less it is flexible to change. To evaluate metric *NOC* at component-level, we consider class-level *NOC* which measures, for a class, the number of classes inherited from the class. Then, the value of metric *NOC* at component level is measured as the sum of class-level *NOC* values. Formally, suppose that $NOC_{i,j}$ denotes the value of metric *NOC* for class $\mathbb{c}_{i,j}$ in component $e_i^{(c)} \in E^{(c)}$. Then, the raw value $NOC_i$ of metric *NOC* for component $e_i^{(c)} \in E^{(c)}$ is measured as follows:

$$NOC_i = \sum_{j=1}^{|\mathbb{C}_i|} NOC_{i,j} \tag{3.5}$$

Back to our running example, considering Figure 3.4, Table 3.5 shows the raw value $NOC_i$, normalized value $\mathbb{V}_i^{(c)}[NOC]$, the value $V_i^{(c)}[NOC]$ of *NOC* metric for each component $e_i^{(c)} \in E^{(c)}$, and the associated overall value $V^{(c)}[NOC]$. For example, considering component $e_1^{(c)} \in E^{(c)}$, the values of metric *NOC* for classes *Customer*, *Person*, and *Company* are $NOC_{e_1^{(c)},Customer} = 2, NOC_{e_1^{(c)},Person} = 0$, and $NOC_{e_1^{(c)},Company} = 0$, respectively. Therefore, according to Formula 3.5, the raw value $NOC_1$ of metric *NOC* for component $e_1^{(c)} \in E^{(c)}$ is $2 + 0 + 0 = 2$. Also, according to Formula 3.1, the associated normalized value $\mathbb{V}_1^{(c)}[NOC] = {}^{2-0}/_3 = 0.66$. Since metric *NOC* has a negative impact on modularity, to measure the value $V_1^{(c)}[NOC]$ of metric *NOC* for component $e_i^{(c)} \in E^{(c)}$, the associated normalized value $\mathbb{V}_i^{(c)}[NOC]$ is subtracted from 1. Therefore, considering component $e_1^{(c)} \in E^{(c)}$, $V_1^{(c)}[NOC] = 1 - 0.66 = 0.34$. Also, the overall value $V^{(c)}[NOC]$ of metric *NOC* is calculated as ${}^{0.34+1+0}/_3 = 0.673$.

### 3.3.1.2 EVALUATION OF COMPONENT-LEVEL COHESION METRICS

In this section, we consider the evaluation of component-level metrics associated with attribute *cohesion*. To do so, considering Figure 3.1, we need to measure the value of *LCOM* metric which is presented in the following.

**Lack of COhesion in Methods (*LCOM*).** To evaluate this metric, we first consider class-level *LCOM* which measures how well the methods of a class are related to each other [106]. The value $\mathrm{LCOM}_{i,j}$ of metric *LCOM* for class $\mathbb{c}_{i,j}$ in component $e_i^{(c)} \in E^{(c)}$ is defined as [24]:

$$\mathrm{LCOM}_{i,j} = \begin{cases} |\mathcal{P}| - |\mathcal{Q}|, & \text{if } |\mathcal{P}| > |\mathcal{Q}| \\ 0, & \text{otherwise.} \end{cases}$$

where $\mathcal{Q}$ is the set of the pairs of methods sharing at least one used instance variable and $\mathcal{P}$ is the set of the pairs of methods that do not share any used instance variables. Considering a class, the more the value of metric *LCOM* is, the more the class is complex and less cohesive. We note that the value of metric *LCOM* is undefined for classes with no or only one method [107] and for classes with no instance variables [102].

*Example.* Class *Order* in component $e_2^{(c)} \in E^{(c)}$ has 6 methods including, *createOrder*, *setDeliveryDate*, *setShippingCost*, *setTotalCost*, *editOrder*, and *checkOut*. There are 4 pairs of methods that at least share one instance variable with other methods in class *Order*, maintained in set $\mathcal{Q}$.

$$\mathcal{Q} = \{(\textit{createOrder,setDeliveryDate}),(\text{createOrder,setShippingCost}), \\ (\text{setDeliveryDate,setShippingCost}),(\text{editOrder,checkOut})\}$$

For example, methods *createOrder* and *setDeliveryDate* share one instance variable (*CustomerID*). Also, there are 11 pairs of methods that do not share any instance variable which are maintained in set $\mathcal{P}$ as follows.

$$\mathcal{P} = \{(\textit{createOrder,setTotalCost}),(\text{createOrder,editOrder}),(\text{setDeliveryDate}, \\ \text{setTotalCost}),(\text{setDeliveryDate,editOrder}),(\text{setShippingCost,setTotalCost}), \\ (\text{setShippingCost,editOrder}),(\text{setTotalCost,editOrder}),(\text{createOrder,checkOut}), \\ (\text{setDeliveryDate,checkOut}),(\text{setShippingCost,checkOut}),(\text{setTotalCost,checkOut})\}$$

Therefore, $|\mathcal{Q}| = 4$ and $|\mathcal{P}| = 11$. As a result, the value $\mathrm{LCOM}_{c_2,\mathrm{Order}}$ of metric *LCOM* for class *Order* in component $e_2^{(c)} \in E^{(c)}$ is equal to $|\mathcal{P}| - |\mathcal{Q}| = 11 - 4 = 7$.

Then, the raw value of metric *LCOM* for a component can be measured as the sum of the values $\mathrm{LCOM}_{i,j}$ of metric *LCOM* for each class $\mathbb{c}_{i,j}$ in component $e_i^{(c)} \in E^{(c)}$. Formally, suppose that, $\mathrm{LCOM}_{i,j}$ denotes the value of metric *LCOM* for class $\mathbb{c}_{i,j}$ in

| | $\mathbb{c}_{i,j}$ | $\lvert\mathcal{P}\rvert$ | $\lvert\mathcal{Q}\rvert$ | **$LCOM_{i,j}$** | **$LCOM_i$** | $\mathbb{V}_i^{(c)}$**[LCOM]** | $V_i^{(c)}$**[LCOM]** | $V^{(c)}$**[LCOM]** |
|---|---|---|---|---|---|---|---|---|
| **$c_1$** | *Customer* | 2 | 4 | 0 | 0 | 0 | 1 | 0.666 |
| **$c_2$** | *Order* | 11 | 4 | 7 | 7 | 1 | 0 | |
| | *OrderItem* | 0 | 3 | 0 | | | | |
| **$c_3$** | *Product* | 0 | 1 | 0 | 1 | 0 | 1 | |
| | *Clothing* | 1 | 0 | 1 | | | | |

**Table 3.6:** *LCOM* values $LCOM_{i,j}$ for each class $\mathbb{c}_{i,j}$ in component $c_i \in C$, raw values $LCOM_i$, normalized values $\mathbb{V}_i^{(c)}[LCOM]$, values $V_i^{(c)}[LCOM]$ of metric *LCOM* for each component $e_i^{(c)} \in E^{(c)}$, and associated overall value $V^{(c)}[LCOM]$

component $e_i^{(c)} \in E^{(c)}$. Then, the raw value $LCOM_i$ of metric *LCOM* for component $e_i^{(c)} \in E^{(c)}$ can be measured as:

$$LCOM_i = \sum_{j=1}^{|\mathbb{C}_i|} LCOM_{i,j} \tag{3.6}$$

Back to our running example, considering Figure 3.4, Table 3.6 shows the value $LCOM_{i,j}$ of metric *LCOM* for each class $\mathbb{c}_{i,j} \in \mathbb{C}_i$ in component $e_i^{(c)} \in E^{(c)}$. Also, it shows the raw value $LCOM_i$, normalized value $\mathbb{V}_i^{(c)}[LCOM]$, the value $V_i^{(c)}[LCOM]$ of metric *LCOM*, for each component $e_i^{(c)} \in E^{(c)}$, and the associated overall value $V^{(c)}[LCOM]$. For example, considering component $e_2^{(c)} \in E^{(c)}$, the values $LCOM_{e_2^{(c)},\mathit{Order}}$, $LCOM_{e_2^{(c)},\mathit{OrderItem}}$ of metric *LCOM* for classes *Order* and *OrderItem* are 7 and 0, respectively. Therefore, according to Formula 3.6, the raw value $LCOM_2$ of metric *LCOM* for component $e_2^{(c)} \in E^{(c)}$ is $7+0=7$. Also, the associated normalized value $\mathbb{V}_2^{(c)}[LCOM]$ is equal to 1, according to Formula 3.1. Since metric *LCOM* has a negative impact on modularity, to measure the value $V_i^{(c)}[LCOM]$ of metric *LCOM* for each component $e_i^{(c)} \in E^{(c)}$, the associated normalized value $\mathbb{V}_i^{(c)}[LCOM]$ is subtracted from 1. Therefore, for component $e_2^{(c)} \in E^{(c)}$, $V_2^{(c)}[LCOM] = 1-1 = 0$. Also, according to Formula 3.3, the overall value $V^{(c)}[LCOM]$ for metric *LCOM* is calculated as $1+0+1/3 = 0.666$. We note that, as we mentioned before, metric *LCOM* is not defined for classes with no or only one method as well as classes with no instance variables. Therefore, considering our running example, we did not include these classes (i.e., *Company*, *Person*, *Grocery*, *Electronic*) in Table 3.6, for the sake of brevity.

#### 3.3.1.3 EVALUATION OF COMPONENT-LEVEL SIZE METRICS

In this section, we consider the evaluation of component-level metrics associated with attribute *size*. To do so, considering Figure 3.1, we need to measure the values of metrics *LOC* and *NOA* which is presented in the following.

**Lines Of Code (LOC).** This metric counts, for a class, all code lines, excluding blank and comment ones [27]. Then, the value of metric *LOC* for a component can be obtained as the sum of the values of metric *LOC* measured for classes in the component.

| | $\mathbb{c}_{i,j}$ | $\mathbf{LOC}_{i,j}$ | $\mathbf{LOC}_i$ | $\mathbb{V}_i^{(c)}$**[LOC]** | $\mathbf{V}_i^{(c)}$**[LOC]** | $\mathbf{V}^{(c)}$**[LOC]** |
|---|---|---|---|---|---|---|
| $\mathbf{c_1}$ | *Customer* | 250 | 290 | 0 | 1 | 0.545 |
| | *Company* | 30 | | | | |
| | *Person* | 10 | | | | |
| $\mathbf{c_2}$ | *Order* | 800 | 1100 | 1 | 0 | |
| | *OrderItem* | 300 | | | | |
| $\mathbf{c_3}$ | *Product* | 500 | 585 | 0.364 | 0.636 | |
| | *Electronic* | 20 | | | | |
| | *Grocery* | 20 | | | | |
| | *Clothing* | 45 | | | | |

**Table 3.7:** *LOC* values $LOC_{i,j}$ for each class $\mathbb{c}_{i,j}$ in component $e_i^{(c)} \in E^{(c)}$, raw values $LOC_i$, normalized values $\mathbb{V}_i^{(c)}[LOC]$, values $V_i^{(c)}[LOC]$ of metric *LOC* for each component $e_i^{(c)} \in E^{(c)}$, and associated overall value $V^{(c)}[LOC]$

Formally, suppose that, $LOC_{i,j}$ denotes the value of metric *LOC* for class $\mathbb{c}_{i,j}$ in component $e_i^{(c)} \in E^{(c)}$. Then, the raw value $LOC_i$ of metric *LOC* for component $e_i^{(c)} \in E^{(c)}$ can be measured as:

$$LOC_i = \sum_{j=1}^{|\mathbb{C}_i|} LOC_{i,j} \tag{3.7}$$

Back to our running example, considering Figure 3.4, Table 3.7 shows the values $LOC_{i,j}$ of metric *LOC* for each class $\mathbb{c}_{i,j} \in \mathbb{C}_i$ in component $e_i^{(c)} \in E^{(c)}$. Also, it shows the raw values $LOC_i$, normalized values $\mathbb{V}_i^{(c)}[LOC]$, the value $V_i^{(c)}[LOC]$ of metric *LOC* for each component $e_i^{(c)} \in E^{(c)}$, and the associated overall value $V^{(c)}[LOC]$. For example, component $e_2^{(c)} \in E^{(c)}$ includes two classes namely, *Order* and *OrderItemID*. The values $LOC_{e_2^{(c)},\mathit{Order}}$, $LOC_{e_2^{(c)},\mathit{OrderItem}}$ of metric *NOC* are 800 and 300, respectively. Therefore, according to Formula 3.7, the raw value $LOC_2$ of metric *LOC* for component $e_2^{(c)} \in E^{(c)}$ is $800 + 300 = 1100$. Then, according to Formula 3.1, the associated normalized value $\mathbb{V}_2^{(c)}[LOC] = 1$. Also, since metric *LOC* has a negative impact on modularity, to measure the value $V_i^{(c)}[LOC]$ of metric *LOC*, the associated normalized value $\mathbb{V}_i^{(c)}[LOC]$ is subtracted from 1. Therefore, for component $e_2^{(c)} \in E^{(c)}$, $V_2^{(c)}[LOC] = 1 - 1 = 0$. Also, according to Formula 3.3, the overall value $V^{(c)}[LOC]$ for metric *LOC* is calculated as $1{+}0{+}0.636/3 = 0.545$.

**Number OF Operations (NOA).** This metric, at class level, measures the number of attributes in a class. Then, the value of *NOA* metric for a component can be measured as the sum the values of metric *NOA* for classes in the component. Formally, suppose that, $NOA_{i,j}$ denotes the value of *NOA* metric for class $\mathbb{c}_{i,j}$ in component $e_i^{(c)} \in E^{(c)}$. Then, the raw value $NOA_i$ of metric *NOA* for component $e_i^{(c)} \in E^{(c)}$ can be calculated as:

| | $\mathbb{c}_{i,j}$ | $\mathbf{NOA}_{i,j}$ | $\mathbf{NOA}_i$ | $\mathbb{V}_i^{(c)}$[NOA] | $V_i^{(c)}$[NOA] | $V^{(c)}$[NOA] |
|---|---|---|---|---|---|---|
| $c_1$ | *Customer* | 6 | 9 | 0 | 1 | 0.666 |
| | *Company* | 2 | | | | |
| | *Person* | 1 | | | | |
| $c_2$ | *Order* | 7 | 13 | 1 | 0 | |
| | *OrderItem* | 6 | | | | |
| $c_3$ | *Product* | 5 | 9 | 0 | 1 | |
| | *Electronic* | 1 | | | | |
| | *Grocery* | 1 | | | | |
| | *Clothing* | 2 | | | | |

**Table 3.8:** *NOA* values $\mathrm{NOA}_{i,j}$ for each class $\mathbb{c}_{i,j}$ in component $e_i^{(c)} \in E^{(c)}$, raw values $\mathrm{NOA}_i$, normalized values $\mathbb{V}_i^{(c)}[\mathrm{NOA}]$, values $V_i^{(c)}[\mathrm{NOA}]$ of metric *NOA* for each component $e_i^{(c)} \in E^{(c)}$, and associated overall value $V^{(c)}[\mathrm{NOA}]$

$$\mathrm{NOA}_i = \sum_{j=1}^{|\mathbb{C}_i|} \mathrm{NOA}_{i,j} \tag{3.8}$$

Back to our running example, considering Figure 3.4, Table 3.8 shows the values $\mathrm{NOA}_{i,j}$ of metric *NOA* for each class $\mathbb{c}_{i,j}$ in component $e_i^{(c)} \in E^{(c)}$. Also, it shows the raw values $\mathrm{NOA}_i$, the normalized values $\mathbb{V}_i^{(c)}[\mathrm{NOA}]$, the values $V_i^{(c)}[\mathrm{NOA}]$ of metric *NOA* for each component $e_i^{(c)} \in E^{(c)}$, and the associated overall value $V^{(c)}[\mathrm{NOA}]$. For example, component $e_2^{(c)} \in E^{(c)}$ includes two classes namely, *Order* and *OrderItemID*. The values $\mathrm{NOA}_{e_2^{(c)},\mathit{Order}}$, $\mathrm{NOA}_{e_2^{(c)},\mathit{OrderItem}}$ of metric *NOA* are 7 and 6, respectively. Therefore, according to Formula 3.8, the raw value $\mathrm{NOA}_2$ of metric *NOA* for component $e_2^{(c)} \in E^{(c)}$ is $7 + 6 = 13$. Also, the associated normalized value $\mathbb{V}_2^{(c)}[\mathrm{NOA}] = 1$, according to Formula 3.1. Since metric *NOA* has a negative impact on modularity, to measure the value $V_i^{(c)}[\mathrm{NOA}]$ of metric *NOA* for component $e_i^{(c)} \in E^{(c)}$, the associated normalized value $\mathbb{V}_i^{(c)}[\mathrm{NOA}]$ is subtracted from 1. Therefore, considering component $e_2^{(c)} \in E^{(c)}$, $V_2^{(c)}[\mathrm{NOA}] = 1 - 1 = 0$, according to Formula 3.2. Also, according to Formula 3.3, the overall value $V^{(c)}[\mathrm{NOA}]$ for metric *NOA* is calculated as $^{1+0+1}/_{3} = 0.666$.

### 3.3.1.4 MEASURING COMPONENT-LEVEL MODULARITY

We have measured, so far, the values of each component-level metric $m_j^{(c)} \in M^{(c)}$ for each component $e_i^{(c)} \in E^{(c)}$. Now, considering component-level weight vector $W^{(c)}$, we evaluate modularity $0 \leqslant Q^{(c)}[i] \leqslant 1$ at component level as:

$$Q^{(c)} = \sum_{j=1}^{|M^{(c)}|} V^{(c)}[j] \cdot W^{(c)}[j] \tag{3.9}$$

Back to our running example, considering component-level weight vector $W^{(c)} = [0.40, 0.05, 0.20, 0.05, 0.30]$ and overall component-level vector $V^{(c)} = [0.666, 0.673, 0.666, 0.545, 0.666]$, according to Formula 3.9, component-level modularity $Q^{(c)}$ for application App is calculated as $0.666 \cdot 0.40 + 0.673 \cdot 0.05 + 0.666 \cdot 0.2 + 0.545 \cdot 0.05 + 0.666 \cdot 0.30 = 0.660$.

### 3.3.2 MODULARITY EVALUATION AT SYSTEM LEVEL

To evaluate modularity at system level, we need to measure the value of each system-level metric $m_k^{(s)} \in M^{(s)}$ (e.g., $M^{(s)} = \{CBS, NUS\}$ in our running example). To enable comparison between measured values for each metric in $M^{(s)}$, we normalize them. Formally, suppose that $E_k^{(s)}$ is the set of system-level entities (e.g., services and servers in our running example) w.r.t. system-level metric $m_k^{(s)} \in M^{(s)}$. Also, suppose that, $v_i^{(s)}[k]$ is the raw value of system-level metric $m_k^{(s)} \in M^{(s)}$ for system-level entity $e_i \in E_k^{(s)}$. Then, the associated normalized value $\mathbb{V}_i^{(s)}[k]$ of $m_k^{(s)}$ for entity $e_i \in E_k^{(s)}$ is calculated as:

$$\mathbb{V}_i^{(s)}[j] = \begin{cases} \frac{v_i^{(s)}[k] - v_{min}}{v_{max} - v_{min}}, & \text{if } v_i^{(s)}[k] \neq v_{min} \wedge v_i^{(s)}[k] \neq v_{max} \\ 0, & \text{if } v_i^{(s)}[k] = v_{min} \\ 1, & \text{if } v_i^{(s)}[k] = v_{max} \end{cases} \quad (3.10)$$

where $v_{min}$ and $v_{max}$ are minimum and maximum raw values of metric $m_k^{(s)} \in M^{(s)}$ for system-level entity $e_i^{(s)} \in E_k^{(s)}$, respectively. Normalized measured values $\mathbb{V}_i^{(s)}[k]$ for entity $e_i^{(s)} \in E_k^{(s)}$ are maintained in a *normalized system-level vector* $\mathbb{V}_i^{(s)}[1, \ldots, |M^{(s)}|]$. Also, if system-level metric $m_k^{(s)} \in M^{(s)}$ has a negative impact on modularity, to measure the value $V_i^{(s)}[k]$ of metric $m_k^{(s)}$ for entity $e_i^{(s)} \in E_k^{(s)}$, the associated normalized measured value $\mathbb{V}_i^{(s)}[k]$ is subtracted from 1. That is:

$$V_i^{(s)}[k] = \begin{cases} 1 - \mathbb{V}_i^{(s)}[k], & \text{if } m_k^{(s)} \text{ has a negative impact on modularity} \\ \mathbb{V}_i^{(s)}[k], & \text{otherwise} \end{cases} \quad (3.11)$$

where such values are maintained in a *system-level vector* $V_i^{(s)}[1, \ldots, |M^{(s)}|]$. Back to our running example, both metrics (i.e., *CBS* and *NUS*) in the set $M^{(s)}$ of system metrics have negative impacts on modularity as they are associated with *Coupling* and *Size* attributes, respectively (see Figure 3.1). Therefore, to evaluate the modularity of system-level entities w.r.t. *CBS* and *NUS*, their associated normalized values are subtracted from 1. Also, the overall value $V^{(s)}[k]$ of metric $m_k^{(s)} \in M^{(s)}$ is calculated as the average of the values $V_i^{(s)}[k]$ of metric $m_k^{(s)} \in M^{(s)}$, measured for each system-level entity $e_i^{(s)} \in E_k^{(s)}$, that is:

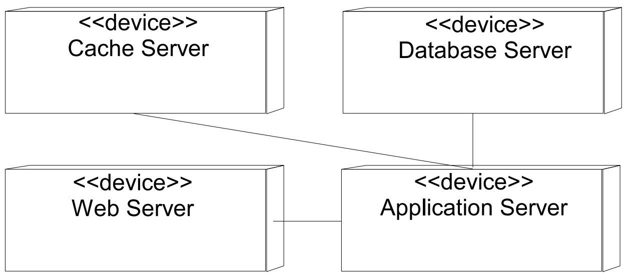

**Figure 3.5:** Deployment diagram associated with application App

| | $\mathbf{CBS_i}$ | $\mathbb{V}_i^{(s)}$**[CBS]** | $V_i^{(s)}$**[CBS]** | $V^{(s)}$**[CBS]** |
|---|---|---|---|---|
| ***Web Server*** | 1 | 0 | 1 | |
| ***Application Server*** | 3 | 1 | 0 | 0.750 |
| ***Cache Server*** | 1 | 0 | 1 | |
| ***Database Server*** | 1 | 0 | 1 | |

**Table 3.9:** Raw values $CBS_i$, normalized values $\mathbb{V}_i^{(s)}[CBS]$, values $V_i^{(s)}[CBS]$ of metric *CBS* for each server $e_i^{(s)} \in E_{CBS}$, and associated overall value $V^{(s)}[CBS]$

$$V^{(s)}[k] = \frac{\sum_{i=1}^{|E_k|} V_i^{(s)}[k]}{|E_k|} \tag{3.12}$$

where such overall values are maintained in a *overall system-level vector* $V^{(s)}[1, \ldots, |M^{(s)}| + 1]$. We note that, as we discussed in Section 3.2, we consider component-level modularity $Q^{(c)}$ as a metric for evaluating modularity at system level. Therefore, $V^{(s)}[|M^{(s)}| + 1]$ in overall system-level vector is component-level modularity $Q^{(c)}$. In the following, we will present our approach for evaluating the values of system-level metrics.

**Coupling Between Servers (CBS).** This metric, which can be the extension of *CBN* metric in [20], measures coupling between servers including physical and virtual servers. Considering the set $E_{CBS}^{(s)} = \{$*Web server,Cache server,Application Server,Database Server*$\}$ of system-level entities w.r.t. metric *CBS* in our running example, Figure 3.5 depicts relations between servers in application App in a deployment diagram. The raw value $CBS_i$ of metric *CBS* for a server $e_i^{(s)} \in E_{CBS}^{(s)}$ can be measured as the sum of connections between server $e_i$ and other servers in deployment diagram associated with the assessed application. For example, as depicted in Table 3.9, the raw value $CBS_{Application\ Server}$ of metric *CBS* for server *Application Server* is equal to 3 as it is coupled to three other servers including, *Web Server, Cache Server*, and *Database Server*. Moreover, considering Formula 3.10, the associated normalized value $\mathbb{V}_{Application\ Server}^{(s)}[CBS]$ is equal to 1. Also, since metric *CBS* has a negative impact on modularity, to measure the value $V_i^{(s)}[CBS]$ of metric *CBS* for server $e_i^{(s)} \in E_{CBS}^{(s)}$, the associated normalized value $\mathbb{V}_i^{(s)}[CBS]$ is subtracted from 1, according to Formula 3.11. For example, considering server *Application Server*, $V_{Applciation\ Server}^{(s)}[CBS] = 0$. Also, according to Formula 3.12, the overall value

| | $\mathbf{NUS_i}$ | $\mathbb{V}_i^{(s)}$**[NUS]** | $\mathbf{V}_i^{(s)}$**[NUS]** | $\mathbf{V}^{(s)}$**[NUS]** |
|---|---|---|---|---|
| ***View*** | 15 | 1 | 0 | |
| ***Stock Management*** | 2 | 0.071 | 0.928 | |
| ***User Management*** | 2 | 0.071 | 0.928 | |
| ***Authentication*** | 1 | 0 | 1 | 0.612 |
| ***Order Management*** | 11 | 0.714 | 0.286 | |
| ***Repository*** | 12 | 0.785 | 0.215 | |
| ***Payment*** | 2 | 0.071 | 0.928 | |

**Table 3.10:** Raw values $NUS_i$, normalized values $\mathbb{V}_i^{(s)}[NUS]$, and values $V_i^{(s)}[NUS]$ of metric *NUS* for each service $e_i^{(s)} \in E_{NUS}$, and associated overall value $V^{(s)}[NUS]$

$V^{(s)}[CBS]$ of metric *CBS* is $1+0+1+1/4 = 0.750$.

**Number of Use-cases for a Service (NUS).** This metric is proposed, to the best of our knowledge, for the first time. It measures, for a service, the total number of usecases handled by the service. The more the value of metric *NUS* for a service is, the more the service is complex, and as a result, the less it is modular. Back to our running example, to measure the value of metric *NUS* for each service in the set $E_{NUS}^{(s)} = \{$*View,Stock Management,User Management,Authentication,Order Management,Repository,Payment*$\}$ of system-level entities w.r.t. metric *NUS*, we consider sequence diagram associated with application App as depicted in Figure 3.6. Considering a service, we define a *use-case* as a message that the service is received from other services, including itself, in sequence diagram associated with the assessed application. For example, considering Figure 3.6, the raw value $NUS_{\textit{Stock Management}}$ of metric *NUS* for service *Stock Management* is equal to 2 because there are two massages received by service *Stock Management* (*Manage Add Product, Manage View Stock Information Request*). Table 3.10 shows the raw values $NUS_i$, normalized values $\mathbb{V}_i^{(s)}[NUS]$, values $V_i^{(s)}[NUS]$ of metric *NUS* for each service $e_i^{(s)} \in E_{NUS}^{(s)}$, and the associated overall value $V^{(s)}[NUS]$. For example, considering service *Order Management*, the raw value $NUS_{\textit{Order Management}}$ of metric *NUS* is equal to 11 as there are 11 usecases for this service. Also, the associated normalized value $\mathbb{V}_{\textit{Order Management}}^{(s)}[NUS]$ is equal to 0.714, according to Formula 3.10. Moreover, since metric *NUS* has a negative impact on modularity, to measure the value $V_i^{(s)}[NUS]$ of *NUS* metric for service $e_i^{(s)} \in E_{NUS}^{(s)}$, the associated normalized value $\mathbb{V}_i^{(s)}[NUS]$ is subtracted from 1. For example, considering service *Order Management*, $V_{\textit{Order Management}}^{(s)}[NUS] = 1 - 0.714 = 0.286$. Also, according to Formula 3.12, the l value $V^{(s)}[NUS]$ of metric *CBS* is $0+0.928+0.928+1+0.286+0.215+0.928/7 = 0.612$.

Therefore, overall system-level vector in our running example is $V^{(s)} = [0.750, 0.612, 0.660]$ which, for instance, $V^{(s)}[3] = 0.660$ represents component-level modularity $Q^{(c)}$ (see Section 3.3.1.4). Then, considering overall system-level vector $V^{(s)}$ and system-level weight vector $W^{(s)}$ (see Table 3.3), we can evaluate the value $0 \leqslant Q^{(s)} \leqslant 1$ of modularity at system level, according to the following formula:

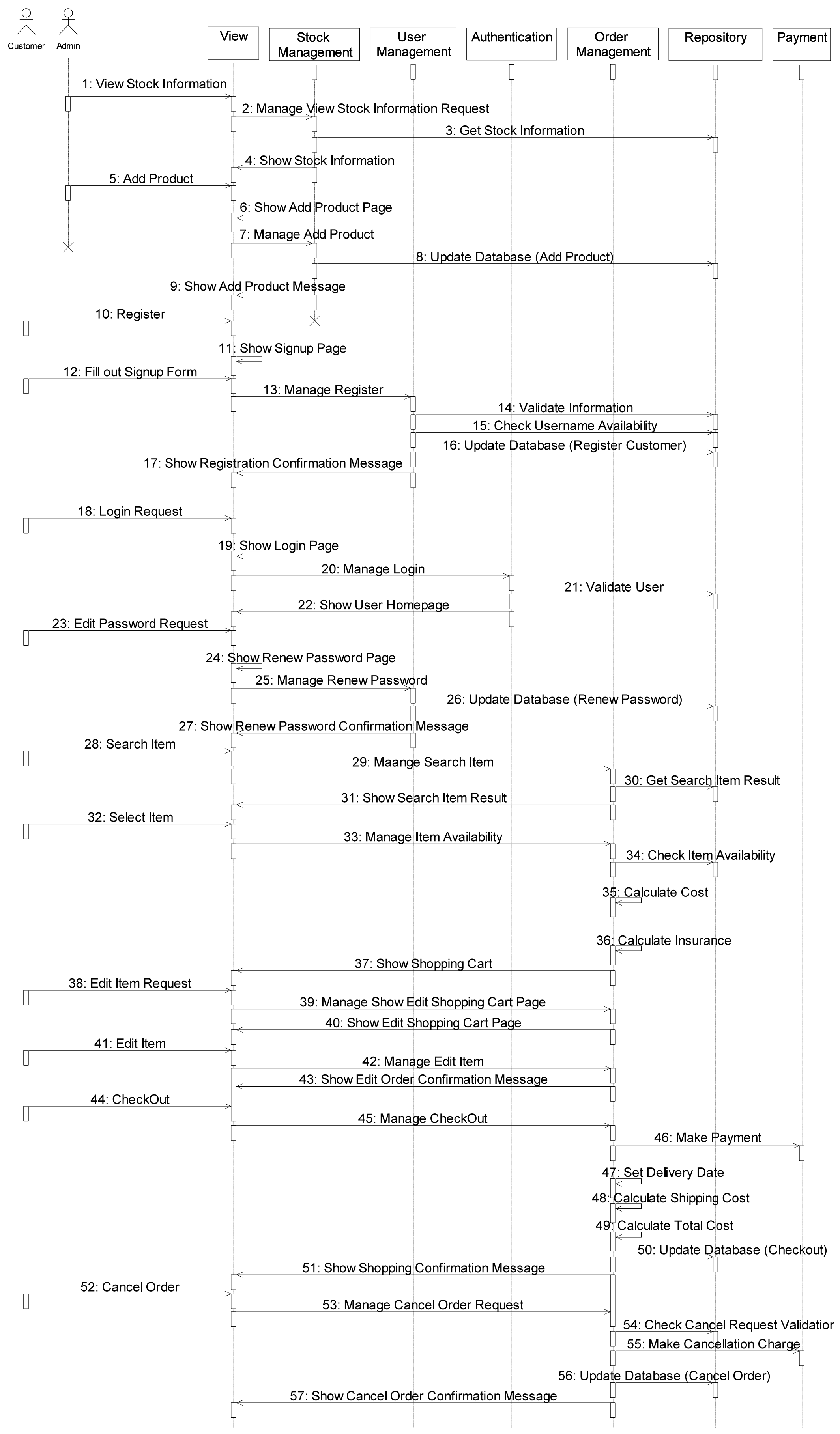
Customer
Admin
View
Stock Management
User Management
Authentication
Order Management
Repository
Payment
1: View Stock Information
2: Manage View Stock Information Request
3: Get Stock Information
4: Show Stock Information
5: Add Product
6: Show Add Product Page
7: Manage Add Product
8: Update Database (Add Product)
9: Show Add Product Message
10: Register
11: Show Signup Page
12: Fill out Signup Form
13: Manage Register
14: Validate Information
15: Check Username Availability
16: Update Database (Register Customer)
17: Show Registration Confirmation Message
18: Login Request
19: Show Login Page
20: Manage Login
21: Validate User
22: Show User Homepage
23: Edit Password Request
24: Show Renew Password Page
25: Manage Renew Password
26: Update Database (Renew Password)
27: Show Renew Password Confirmation Message
28: Search Item
29: Maange Search Item
30: Get Search Item Result
31: Show Search Item Result
32: Select Item
33: Manage Item Availability
34: Check Item Availability
35: Calculate Cost
36: Calculate Insurance
37: Show Shopping Cart
38: Edit Item Request
39: Manage Show Edit Shopping Cart Page
40: Show Edit Shopping Cart Page
41: Edit Item
42: Manage Edit Item
43: Show Edit Order Confirmation Message
44: CheckOut
45: Manage CheckOut
46: Make Payment
47: Set Delivery Date
48: Calculate Shipping Cost
49: Calculate Total Cost
50: Update Database (Checkout)
51: Show Shopping Confirmation Message
52: Cancel Order
53: Manage Cancel Order Request
54: Check Cancel Request Validatior
55: Make Cancellation Charge
56: Update Database (Cancel Order)
57: Show Cancel Order Confirmation Message

**Figure 3.6:** Sequence diagram associated with application App

$$Q^{(s)} = \sum_{k=1}^{|M^{(s)}|+1} V^{(s)}[k] \cdot W^{(s)}[k] \tag{3.13}$$

Higher values for $Q^{(s)}$ indicate higher levels of modularity for the assessed application. Back to our running example, considering $V^{(s)} = [0.750, 0.612, 0.660]$ and $W^{(s)} = [0.50, 0.30, 0.20]$, system-level modularity $Q^{(s)}$ for application App is calculated as $0.750 \cdot 0.50 + 0.612 \cdot 0.30 + 0.660 \cdot 0.20 = 0.690$. Considering the modularity of an application, there are two possible options for the application that is not ready to be moved to the Cloud, w.r.t. change flexibility and/or distributability: *i)* if the cost of the reconfiguration of application is affordable for the application owner, it can be reconfigured by application developers to improve the modularity of application; *ii)* otherwise, moving the application to the Cloud is ignored. Such investigation, which is beyond the objectives of our study presented in this chapter, can be considered as a future line of research.

## 3.4 CHAPTER SUMMARY

In this chapter, we provided a solution to estimate the modularity of an application to see to what extent it can be easily moved to the Cloud, w.r.t. change flexibility and/or decomposability. To do so, we considered a "bottom-up" approach which evaluates modularity, first, at component-level, and then, at system-level. We note that we provided a generic classification including attributes that are considered important for an application, together with their associated metrics. Such classification notably improves the flexibility of solution as attributes and metrics for modularity evaluation can be selected based on several considered parameters (e.g., application type, application architecture, the execution context of the application on the Cloud). Also, apart from using well-known attributes and metrics for evaluating modularity, we proposed a new system-level size metric which measures the total number of use-cases for a service.

# 4 CONSENSUS-BASED CLOUD PLAN SELECTION

An important task when an application is moved to the Cloud is finding a Cloud plan, among those available ones offered by a (set of) Cloud provider(s), according to the requirements of an application and the characteristics of available Cloud plans. Also, if a user wishes to move multiple applications, at the same time, to the Cloud, plan selection can be even more complicated by the fact that different applications can have different (and possibly contrasting) requirements.

This chapter defines an approach enabling users to select a Cloud plan that best balances the satisfaction of the requirements of multiple applications. Our solution operates by first ranking the available plans for each application (matching plan characteristics and application requirements) and then by selecting, through a consensus-based process, the one that is considered more acceptable for all applications.

## 4.1 INTRODUCTION

Cloud computing represents today the reference paradigm for deploying applications and for storing, managing, and processing large amounts of data. Thanks to the advantages in providing an illusion of an infinite amount of resources by offering cost-effective elastic services which are universally accessible on-demand, more and more private and public organizations as well as individuals are moving their data and applications to the Cloud [7] [68]. Cloud providers sell Cloud plans that differ in the services they offer, the quality of services they guarantee, and applied pricing models. This variety provides great advantages for users, enabling them to choose the plan that better suits their needs and economical availability.

Moreover, when multiple applications, at the same time, are moved to the Cloud, the problem of Cloud plan selection can even get more complicated. In fact, each application can have a different "ideal" Cloud plan, according to its requirements. For example, applications operating with sensitive data will mostly care about security (e.g., encryption algorithms, security auditing), while applications running data-intensive computations on publicly available or non-confidential data will be more interested in performance (e.g., CPU and disk speed, network latency). Therefore, since each application can have different and possibly contrasting requirements, a single Cloud plan that satisfies the requirements of all applications might not exist. As a result, the user needs to properly combine applications' requirements and choose a plan that satisfies them in the best possible way. A naive approach to choose the most suitable plan would consist in identifying the best plan for each application, and then, selecting the one chosen by the majority of the applications. However, such an approach would risk leaving the requirements of some applications completely unsatisfied.

This chapter defines an approach aimed at balancing the satisfaction of requirements of multiple applications. To this purpose, our solution first produces, for each application, a ranking of the available plans according to the requirements of applications. It then selects the plan that is globally considered the most acceptable by all applications. To implement these two steps, we put forward the idea of jointly adopting a multi-criteria decision-making technique (TOPSIS [108]) to rank applications, and a consensus-based voting technique (Borda count [109]) to choose a plan that is ranked high by all applications. The combined adoption of these techniques enables the user to choose a Cloud plan that better balances the requirements of all the applications, reaching a trade-off among their (possibly contrasting) needs.

### 4.1.1 CHAPTER OUTLINE

This chapter is structured as follows. Section 4.2 presents a reference scenario for our problem together with some basic concepts on the problem. Section 4.3 illustrates an approach for selecting Cloud plans based on the consensus between the applications by jointly adopting TOPSIS and Borda Count. Section 4.4 presents a pseudo-code algorithm for our proposed solution. Finally, Section 4.5 presents concluding remarks.

## 4.2 BASIC CONCEPTS AND PROBLEM DEFINITION

We consider a scenario characterized by a user wishing to outsource to Cloud a set $A = \{a_1, \dots, a_n\}$ of applications. To this aim, she needs to find the most suitable among a set $P = \{p_1, \dots, p_m\}$ of plans offered by a set of Cloud providers. Each plan $p \in P$ might have different characteristics with respect to a set $C = \{c_1, \dots, c_l\}$ of criteria that the user considers of interest for the set $A$ of applications. For instance, $C$ can include criteria such as the guaranteed availability, the charged costs, or the security guaranteed by the providers. In the definition of $C$, the user can refer to existing guidelines and classifications (e.g., [110]), combined with her personal needs.

Since plans in $P$ differ in the characteristics of the offered services, we assume the user to rate the degree to which a criterion $c_i \in C$ is "satisfied" by a plan $p_j \in P$. Intuitively, this degree expresses how much the services offered by $p_j$ are close to an ideal scenario that maximizes the satisfaction of $c_i$ (e.g., a Cloud provider offering its services for free would have the maximum rating for the *Cost* criterion). Each plan $p_j$ is

| | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ |
|---|---|---|---|---|---|
| **Availability** | 0.40 | 0.50 | 0.90 | 0.40 | 0.20 |
| **Performance** | 0.50 | 0.60 | 0.97 | 0.30 | 0.30 |
| **Security** | 0.60 | 0.70 | 0.80 | 0.20 | 0.40 |
| **Costs** | 0.50 | 0.40 | 0.10 | 0.60 | 0.70 |
| ... | ... | ... | ... | ... | ... |
| **Backup** | 0.40 | 0.60 | 0.30 | 0.30 | 0.20 |
| **StorageSpace** | 0.50 | 0.30 | 0.40 | 0.30 | 0.20 |
| **MobileSupport** | 0.60 | 0.80 | 0.30 | 0.40 | 0.50 |

**Table 4.1:** Example of rating vectors $R_1, \dots, R_5$

then associated with a *rating vector* $R_j[1, \dots, l]$, where $0 \leqslant R_j[i] \leqslant 1$ represents the rating of $p_j$ with respect to criterion $c_i$, where higher ratings represent better satisfaction of the criterion. For instance, a plan $p_j$ providing more than 10 synchronized replicas, sophisticated authentication mechanisms and encryption algorithms, high CPU rates and network bandwidth, but applying expensive price lists, will have a high rating w.r.t. security, availability, and performance, and a low rating w.r.t. cost. Table 4.1 illustrates an example of rating vectors $R_1, \dots, R_5$ for plans $p_1, \dots, p_5$ respectively, over different criteria. For instance, $p_5$ is rated lower for criterion *Availability* ($R_5[\textit{Availability}] = 0.20$) than for criterion *Costs* ($R_5[\textit{Costs}] = 0.70$).

The set $C$ of criteria is defined by the user considering all the requirements of all her applications in $A$. Indeed, as mentioned in Section 4.1, not *all* criteria in $C$ may be relevant to *all* applications in $A$. We denote with $C_k \subseteq C$ the set of criteria relevant to application $a_k \in A$. For instance, with reference to our running example, the set $C_1$ of criteria relevant to $a_1$ is $C_1 = \{\textit{Availability}, \textit{Performance}, \textit{Security}, \textit{Cost}\}$, and the set $C_2$ relevant to $a_2$ is $C_2 = \{\textit{Backup}, \textit{StorageSpace}, \textit{MobileSupport}\}$. Also, given an application $a_k \in A$ with its set $C_k$ of relevant criteria, not all criteria $c_i \in C_k$ can be assumed to be equally important to $a_k$. For instance, with reference to the example above, $a_1$ might value *Security* more than *Availability*. A natural way to express the requirements of an application $a_k \in A$ consists in associating a weight to each criterion in $C_k$. In fact, this permits to model applications having different (and possibly disjoint) relevant criteria, with different relevance for different applications. Considering an application $a_k \in A$, the importance of each criterion $c_i \in C_k$ is modeled by associating $c_i$ with a *weight*, where higher weights model higher importance of the criterion for $a_k$. Formally, the requirement for an application $a_k \in A$ is expressed as a *weight vector* $W_k[1, \dots, |C_k|]$, where $W_k[i]$ represents the weight (i.e., the relative importance) of criterion $c_i$ for application $a_k \in A$. To enable comparison among the weights, we assume the weight vectors to be normalized (i.e., $\sum_{i=1}^{|C_k|} W_k[i]=1$, $k = 1, \dots, n$). Table 4.2 illustrates two example of weight vectors for two applications, $a_1 \in A$ and $a_2 \in A$, where $C_1 = \{\textit{Availability}, \textit{Performance}, \textit{Security}, \textit{Costs}\}$ and $C_2 = \{\textit{Backup}, \textit{StorageSpace}, \textit{MobileSupport}\}$. For instance, the weight vector $W_2 = [0.30, 0.40, 0.30]$ of application $a_2 \in A$ states that criteria *Backup*

|  | $W_1$ |
|---|---|
| **Availability** | 0.04 |
| **Performance** | 0.02 |
| **Security** | 0.04 |
| **Costs** | 0.90 |

|  | $W_2$ |
|---|---|
| **Backup** | 0.30 |
| **StorageSpace** | 0.40 |
| **MobileSupport** | 0.30 |

**Table 4.2:** Example of weight vectors for applications $a_1$ and $a_2$ over different criteria

and *MobileSupport* have the same relative importance (0.30 each), while criterion *StorageSpace* is more relevant (0.40).

Given an application $a_k \in A$ and a set $P$ of plans, the user can identify the plan $p \in P$ that best matches the requirements of $a_k$ by using classical multi-criteria decision making approaches (e.g., [111]). Because of the heterogeneity of the requirements of the applications, however, the plan maximizing the satisfaction of the requirements of an application $a_k \in A$ may not be the plan maximizing the satisfaction of the requirements of another application $a_x \neq a_k$. It would instead be desirable to combine the requirements of all the applications, to select a plan that satisfies all of them in the best possible way. A simple solution would choose the plan that better satisfies the majority of the application requirements. However, such a trivial approach may select a solution that fully satisfies the requirements of applications $A \setminus \{a_k\}$ while not satisfying the requirements of $a_k$ at all. This solution might then be considered not desirable as it would strongly penalize application $a_k \in A$. To prevent such a situation, we propose to adopt a consensus-based approach aimed at choosing the plan that balances the preferences of all the applications, hence enabling the user to determine a solution that provides a good trade-off in the satisfaction of the requirements of all her applications.

## 4.3 CONSENSUS FOR CLOUD PLAN SELECTION

Our approach to choose the plan that best fits the user requirements operates in two steps (see Figure 4.1): *1) rank*, for each application, the providers on the basis of their compliance with the application requirements; *2) reach a consensus* in the choice of the provider that better suits the application requirements, based on the rankings obtained in the first step. In the following, we present our approach, based on TOPSIS for computing rankings, and Borda count for reaching a consensus.

### 4.3.1 RANKING CLOUD PLANS FOR AN APPLICATION

The first step of our solution aims at producing a ranking of the plans in $P$ for each application $a_k \in A$. Such a ranking reflects the satisfaction of the requirements of $a_k$ by the different plans, being the first plan in the ranking the one better satisfying all the requirements of $a_k$.

To rank the plans for an application, we propose to adopt traditional multi-criteria decision making (MCDM) techniques. In fact, MCDM approaches effectively identify,

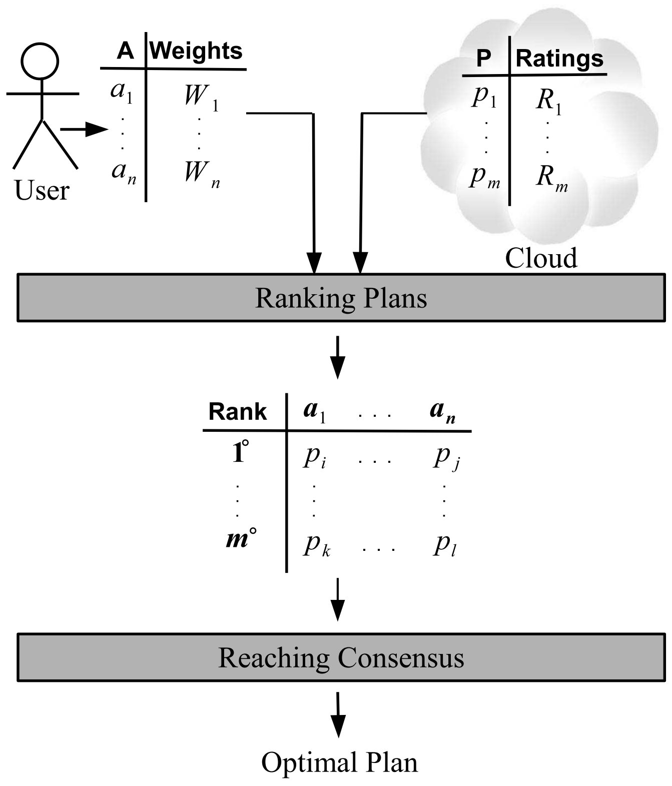

**Figure 4.1:** Working of the approach

in a pool of alternative solutions, the one that optimizes a set of objective functions (i.e., application requirements in our terminology). Among several MCDM techniques, a possible approach relies on adopting TOPSIS [108] as it showed to provide good results when applied to Cloud scenarios, traditionally characterized by many alternatives compared to the number of criteria [112].

Given an application $a_k \in A$, a set P of alternative solutions (plans, in our scenario), a set $C_k$ of criteria relevant to $a_k$, the weights $W_k$ assigned by $a_k$ to the criteria in $C_k$, and the ratings $R_j[i], i = 1, \ldots, |C_k|, j = 1, \ldots, |P|$ assigned to plan $p_j \in P$ for criterion $c_i$, TOPSIS produces a ranking of the alternatives in P, ordering them according to how well they satisfy the criteria (from the best to the worst). To produce such a ranking, TOPSIS evaluates the distance of each plan in P from the *ideal* and *anti-ideal* solutions, ranking higher those plans that are closer to the ideal solution and farthest from the anti-ideal solution. Intuitively, the ideal solution $p_k^+$ for $a_k$ is a plan (which may not belong to P) that satisfies in the best possible way all the criteria relevant to $a_k$. On the contrary, the anti-ideal solution $p_k^-$ for $a_k$ is a plan (which may not belong to P) that satisfies in the worst possible way the criteria relevant to $a_k$. For each application $a_i \in A$in $A$, TOPSIS works in three steps: *i)* it first computes a weighted decision matrix, based on weights and ratings; *ii)* it then identifies the ideal and anti-ideal solutions; and *iii)* finally, it ranks the plans based on their distance from the ideal and anti-ideal solutions.

In the remainder of this section, we illustrate more in details the working of TOPSIS in our scenario. For simplicity, in the following, we refer our discussion to one

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ |
|---|---|---|---|---|---|
| **Availability** | 0.40 | 0.50 | 0.90 | 0.40 | 0.20 |
| **Performance** | 0.50 | 0.60 | 0.97 | 0.30 | 0.30 |
| **Security** | 0.60 | 0.70 | 0.80 | 0.20 | 0.40 |
| **Costs** | 0.50 | 0.40 | 0.10 | 0.60 | 0.70 |

**Table 4.3:** Decision matrix $\mathbb{R}_1$ for application $a_1 \in A$

application only ($a_k \in A$), with the note that the process described is executed for all applications in $A$.

**Weighted decision matrix.** To determine the weighted decision matrix for each application $a_k \in A$, TOPSIS uses a *decision matrix* $\mathbb{R}_k$, with a row for each criteria $c \in C_k$ and a column for each plan $p \in P$. Basically, the decision matrix for application $a_k \in A$ is composed of the rating vectors $R_j$ (restricted to the criteria $C_k$ relevant to $a_k$): each cell $\mathbb{R}_k[i][j]$ in the decision matrix represents the rating $R_j[i]$ assigned to plan $p_j \in P$, for criteria $c_i \in C_k$. Table 4.3 illustrates the decision matrix for $a_1$, obtained from the rating vectors in Table 4.1 restricted to the first four criteria (i.e., those relevant for $a_1$). The original TOPSIS proposal normalizes the decision matrix, to guarantee that values in different cells can be properly compared. Since the rating values assigned to plans are already a-dimensional values between $0$ and $1$, in our scenario, it is not necessary to normalize the decision matrix $\mathbb{R}_k$.
To properly take into consideration the importance of the different criteria in $C_k$ for the considered application $a_k \in A$, the decision matrix $\mathbb{R}_k$ is composed with vector $W_k$ (i.e., with the weights assigned to each criteria to reflect the application needs). Each cell in the *weighted decision matrix* $\mathbb{D}_k$ for application $a_k \in A$ is computed as the product $\mathbb{D}_k[i][j] = \mathbb{R}_k[i][j] \cdot W_k[i]$ of the rating obtained by plan $p_j \in P$ for criterion $c_i$, and the weight of criterion $c_i$ for application $a_k \in A$. Table 4.4 illustrates the weighted decision matrix for application $a_1 \in A$ of our running example. For instance, $\mathbb{D}_1[\mathit{Availability}][p_1]$ is obtained as $\mathbb{R}_1[\mathit{Availability}][p_1] \cdot W_1[\mathit{Availability}]=0.4 \cdot 0.04=0.016$. Note that the weighted decision matrix permits to identify, for each criterion $c_i$ singularly taken, the best and the worst plan, which correspond to the highest and lowest values in the row representing $c_i$. As an example, the best plan w.r.t. the *Security* criterion for application $a_1 \in A$ is $p_3 \in P$, while the worst is $p_4 \in P$.

**Ideal and anti-ideal solutions.** Based on the weighted decision matrix $\mathbb{D}_k$, TOPSIS is able to identify both the ideal and the anti-ideal solutions $p_k^+$ and $p_k^-$ for application $a_k \in A$. For the ideal solution $p_k^+$, the weighted rating for criterion $c_i$ (denoted $D_k^+[i]$) is the maximum weighted rating obtained by a plan in $P$ for $c_i$ (i.e., $D_k^+[i]=\mathit{max}\{\mathbb{D}_k[i][j] : p_j \in P\}$, line 13). For instance, the ideal solution for application $a_1 \in A$, considering the weighted decision matrix in Table 4.4, has weighed ratings $D_1^+=[0.036, 0.019, 0.032, 0.630]$. Similarly, for the anti-ideal solution $p_k^-$, the weighted rating for criterion $c_i$ (denoted $D_k^-[i]$) is the minimum weighted rating obtained by a plan in $P$ for $c_i$ (i.e., $D_k^+[i]=\mathit{min}\{\mathbb{D}_k[i][j] : p_j \in P\}$, line 14). For instance, the anti-ideal

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_1^+$ | $p_1^-$ |
|---|---|---|---|---|---|---|---|
| **Availability** | 0.016 | 0.020 | 0.036 | 0.016 | 0.008 | 0.036 | 0.008 |
| **Performance** | 0.010 | 0.012 | 0.019 | 0.006 | 0.006 | 0.019 | 0.006 |
| **Security** | 0.024 | 0.028 | 0.032 | 0.008 | 0.016 | 0.032 | 0.008 |
| **Costs** | 0.450 | 0.360 | 0.090 | 0.540 | 0.630 | 0.630 | 0.090 |
| $\mathbf{dist}_j^+$ | 0.182 | 0.271 | 0.540 | 0.096 | 0.035 | | |
| $\mathbf{dist}_j^-$ | 0.360 | 0.271 | 0.039 | 0.450 | 0.540 | | |
| $\mathbf{S_1}$ | 0.665 | 0.500 | 0.068 | 0.824 | 0.939 | | |

**Table 4.4:** Weighted decision matrix $\mathbb{D}_1$, ideal solution $p_1^+$, anti-ideal solution $p_1^-$, distances $\text{dist}_j^+$ and $\text{dist}_j^-$ of each plan $p_j$ from $p_1^+$ and $p_1^-$, and relative closeness $S_1$ of each plan to the ideal solutions for application $a_1 \in A$

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ |
|---|---|---|---|---|---|---|---|
| **1°** | $p_5$ | $p_3$ | $p_5$ | $p_1$ | $p_3$ | $p_4$ | $p_3$ |
| **2°** | $p_4$ | $p_2$ | $p_4$ | $p_2$ | $p_2$ | $p_5$ | $p_2$ |
| **3°** | $p_1$ | $p_1$ | $p_1$ | $p_5$ | $p_1$ | $p_1$ | $p_1$ |
| **4°** | $p_2$ | $p_4$ | $p_2$ | $p_3$ | $p_5$ | $p_2$ | $p_5$ |
| **5°** | $p_3$ | $p_5$ | $p_3$ | $p_4$ | $p_4$ | $p_3$ | $p_4$ |

**Table 4.5:** Example of rankings of the plans for each application

solution for application $a_1 \in A$, considering the weighted decision matrix in Table 4.4, has weighted ratings $D_1^+$=[0.008, 0.006, 0.008, 0.090].

**Ranking.** To produce a ranking, TOPSIS then computes the Euclidean *distance* of each plan $p_j \in P$ from the ideal $p_k^+$ and anti-ideal $p_k^-$ solution in an $l$-dimensional space (with $l$ the number of criteria in $C_k$). Then, it computes the relative closeness of each plan $p_j$ to the ideal solutions as $\frac{\text{dist}_j^-}{\text{dist}_j^+ + \text{dist}_j^-}$, where $\text{dist}_j^+$ and $\text{dist}_j^-$ are the distance of $p_j \in P$ from $p_k^+$ and $p_k^-$, respectively, where such closeness values are maintained in a score vector $S_k$). For instance, the relative closeness values of the plans in P to $p_1^+$ for application $a_1 \in A$ considering the weighted decision matrix in Table 4.4, is $S_1$=[0.665, 0.500, 0.068, 0.824, 0.939]. The higher the value $S_k$, the better the plan satisfies the requirements of application $a_k \in A$. Then, TOPSIS produces a ranking of the plans for application $a_k \in A$ by ordering them in decreasing order of $S_k$. For instance, with reference to our running example, the ratings in Table 4.1, and the weights in Table 4.2, the ranking of plans produced by TOPSIS for application $a_1 \in A$ is $\langle p_5, p_4, p_1, p_2, p_3 \rangle$ (see column $a_1$ in Table 4.5).

### 4.3.2 REACHING CONSENSUS AMONG THE APPLCIATIONS

The second step of our solution aims at choosing a Cloud plan that balances the preferences of all user applications. A straightforward approach to maximize requirements

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | Tot |
|---|---|---|---|---|---|---|---|---|
| $p_1$ | 3 | 3 | 3 | 5 | 3 | 3 | 3 | 23 |
| $p_2$ | 2 | 4 | 2 | 4 | 4 | 2 | 4 | 22 |
| $p_3$ | 1 | 5 | 1 | 2 | 5 | 1 | 5 | 20 |
| $p_4$ | 4 | 2 | 4 | 1 | 1 | 5 | 1 | 18 |
| $p_5$ | 5 | 1 | 5 | 3 | 2 | 4 | 2 | 22 |

**Table 4.6:** Borda scores assigned to each plan by each application

satisfaction would adopt a majority voting, that is, it would choose Cloud plan ranked first by most applications. For instance, consider a scenario characterized by a set $A = \{a_1, \ldots, a_7\}$ of applications and a set $P = \{p_1, \ldots, p_5\}$ of plans, where the rankings computed by TOPSIS for each application are illustrated in Table 4.5. The plan that would win with the majority voting approach would be $p_3$. However, as already noted, this solution might be not desirable as $p_3 \in P$ is ranked last by three applications $a_1$, $a_3$, and $a_6$, which would then be strongly penalized.

We then propose to adopt a consensus-based voting technique, that permits to choose an alternative that is acceptable for a broad set of voters (applications, in our scenario), rather than simply counting majority. While noting that there are different approaches that can be applied (e.g., [113], [114]), we consider - as an example - the Borda count method [109]. In our Cloud scenario, alternatives correspond to plans and the applications play the role of voters. To express its vote, each application $a_k \in A$ associates a Borda score $B_k[j]$ with each plan $p_j \in P$. Such a score reflects the rankings computed by TOPSIS (or, more in general, by the chosen MCDM technique) by assigning $m = |P|$ points to the first ranked plan, 1 to the last ranked plan, and $m+1-x$ to the $x$-th ranked plan. The overall Borda score $\mathtt{Borda}(p_j)$ of a plan $p_j \in P$ is then obtained by summing the scores assigned to the plan by each application, that is, $\sum_{k=1}^{n} B_k[j]$. The plan with the highest Borda score is the one that is chosen by the user, since it has the consensus of all the applications. As an example, Table 4.6 illustrates the Borda scores assigned by each application to each plan and the overall score of each plan. It is interesting to see that the chosen plan is $p_1 \in P$, which is ranked first by one application only, while $p_3 \in P$ is only the third choice, even though it is ranked first by three applications.

## 4.4 ALGORITHM FOR THE PROPOSED CONSENSUS-BASED CLOUD PLAN SELECTION APPROACH

Given the problem of choosing a Cloud plan, among those available ones offered by a (set of) Cloud provider(s), we provide a pseudocode algorithm for our consensus-based approach. The algorithm, reported in Figure 4.2, takes as input a set $A$ of applications, a set $P$ of Cloud plans, a set $C$ of criteria, weight vectors $W_1, \ldots, W_n$, and rating vectors $R_1, \ldots, R_m$. Also, it returns plan $p \in P$ as an optimal plan based on the consensus between applications.

First, the algorithm obtains a decision matrix $\mathbb{R}_k$ for each application $a_k \in A$ (lines 2–5). As we mentioned before, each cell $\mathbb{R}_k[i][j]$ in the decision matrix represents the rating $R_j[i]$ assigned to plan $p_j \in P$ for criteria $c_i \in C_k$. Then, for each application $a_k \in A$, a weighted decision matrix $\mathbb{D}_k$ is obtained (lines 6–9). Each cell in $\mathbb{D}_k$ for application $a_k \in A$ is computed as the product $\mathbb{D}_k[i][j] = \mathbb{R}_k[i][j] \cdot W_k[i]$ of the rating obtained by plan $p_j \in P$ for criterion $c_i$, and the weight of criterion $c_i$ for application $a_k \in A$. Then, to produce a ranking for each application $a_k \in A$, the Euclidean *distance* of each Cloud plan $p_j \in P$ from the ideal $p_k^+$ and anti-ideal $p_k^-$ solution in an $l$-dimensional space (with $l$ the number of criteria in $C_k$) is measured, according to TOPSIS (lines 10–14). Then, the relative closeness of each plan $p_j$ to the ideal solutions as $\frac{\mathtt{dist}_j^-}{\mathtt{dist}_j^+ + \mathtt{dist}_j^-}$, where $\mathtt{dist}_j^+$ and $\mathtt{dist}_j^-$ are the distance of $p_j \in P$ from $p_k^+$ and $p_k^-$, respectively (lines 15–19). Next, for each application $a_k \in A$, the algorithm defines a list $O_k$ of size $|P|$ to maintain plans in ranked order for $a_k$, and then, inserts plans in $O_k$ in decreasing order of $S_k[j]$ (lines 21–22).

Then, to produce a global ranking based on the consensus between applications, for each application $a_k \in A$, a vector $B_k$ of size $|P|$ is considered (lines 24). Next, for each plan $p_j \in P$ and application $a_k \in A$, a Borda score $B_k[j]$ is calculated (lines 25–27). Further, a Borda score $\mathtt{Borda}(p_j) = \sum_{k=1}^{n} B_k[j]$ for each plan $p_j \in P$ is obtained (lines 28–29). Finally, plan $p \in P$ is returned as the optimal plan, which is the plan with the highest Borda score $\mathtt{Borda}(p)$ (line 30).

```
INPUT
$A = \{a_1, \dots, a_n\}$ /* set of applications */
$P = \{p_1, \dots, p_m\}$ /* set of plans */
$C = \{c_1, \dots, c_l\}$ /* set of criteria */
$W_1, \dots, W_n$ /* weight vectors */
$R_1, \dots, R_m$ /* rating vectors */
OUTPUT
$p \in P$ /* optimal plan */
MAIN
/* Step 1: rank plans for each application in $A$ */
1:  for each $a_k \in A$ do
2:    let $\mathbb{R}_k$ be the decision matrix of size $|C_k| \times |P|$
3:    for each $i = 1, \dots, |C_k|$ do
4:      for each $j = 1, \dots, |P|$ do
5:        $\mathbb{R}_k[i][j] := R_j[i]$ /* fill $\mathbb{R}_k$ with values in the rating vectors */
6:    let $\mathbb{D}_k$ be the weighted decision matrix of size $|C_k| \times |P|$ for $a_k$
7:    for each $i = 1, \dots, |C_k|$ do
8:      for each $j = 1, \dots, |P|$ do
9:        $\mathbb{D}_k[i][j] := \mathbb{R}[i][j] \cdot W_k[i]$ /* fill $\mathbb{D}_k$ with weighted ratings */
10:   let $D_k^+$ be the weighted rating vector of size $|C_k|$ for $p_k^+$
11:   let $D_k^-$ be the weighted rating vector of size $|C_k|$ for $p_k^-$
12:   for each $i = 1, \dots, |C_k|$ do
13:     $D_k^+[i] := max\{\mathbb{D}_k[i][j] \mid j = 1, \dots, |P|\}$ /* $p_k^+$ rating for $c_i$ */
14:     $D_k^-[i] := min\{\mathbb{D}_k[i][j] \mid j = 1, \dots, |P|\}$ /* $p_k^-$ rating for $c_i$ */
15:   let $S_k$ be a vector of size $|P|$ for $a_k$ /* to store closeness values */
16:   for each $j = 1, \dots, |P|$ do
17:     let $dist_j^+$ be the distance between $p_j$ and $p_k^+$
18:     let $dist_j^-$ be the distance between $p_j$ and $p_k^-$
19:     $S_k[j] := \frac{dist_j^-}{dist_j^+ + dist_j^-}$ /* closeness between $p_j$ and ideal solutions */
20:   let $O_k$ be a list of size $|P|$ to contain plans in ranked order for $a_k$
21:   for each $j = 1, \dots, |P|$ do
22:     insert $p_j$ in $O_k$ in decreasing order of $S_k[j]$
/* Step 2: reach consensus */
23: for each $a_k \in A$ do
24:   let $B_k$ be the Borda vector of size $|P|$ for $a_k$
25:   for each $j = 1, \dots, |P|$ do
26:     let $x$ be the position of $p_j$ in $O_k$ /* TOPSIS ranking of $p_j$ */
27:     $B_k[j] := |P| + 1 - x$ /* Borda score for plan $p_j$ and application $a_k$ */
28: for each $j = 1, \dots, |P|$ do
29:   $Borda(p_j) := \sum_{k=1}^{n} B_k[j]$ /* Borda score for plan $p_j$*/
30: return $p \in P$ s.t. $\nexists p' \in P, p' \neq p : Borda(p') > Borda(p)$
```

**Figure 4.2:** Algorithm for selecting the consensus-based optimal plan

## 4.5 CHAPTER SUMMARY

This chapter presented a solution enabling Cloud users to choose a plan, among those available ones offered by a (set of) Cloud provider(s), which is based on reaching a consensus between applications. In our approach, each application individually ranks the available Cloud plans depending on how well they satisfy the application requirements. The choice on the final plan is then taken adopting a consensus-based approach on the different rankings. The proposed solution provides a tool which guarantees that the selected Cloud plan is globally considered the most acceptable by all applications, according to a consensus among them.

# 5

# SUPPORTING CLOUD PLAN SELECTION UNDER UNCERTAINTY

In Chapter 4, we focused on the consensus between multiple applications to select a Cloud plan. Indeed, in Chapter 4 it was assumed that the user is aware about the technical requirements of applications and the characteristics of Cloud plans. However, it might be not always the case as unskilled users might be interested in moving their applications to the Cloud. Also, users/stakeholders might consider a limited budget for selecting a Cloud plan, among those available, each with possibly a different price. The problem of Cloud plan selection can get even more complicated when multiple users/stakeholders contribute to a Cloud plan selection process. To deal with these issues, in this chapter, we propose an approach aimed at choosing a Cloud plan when a set of unskilled IT users/stakeholders are interested in moving multiple applications to the Cloud, while suitably satisfying the budget constraints of users for selecting a Cloud plan. Our solution operates by first measuring the importance of applications and then by calculating their aggregated preferences, over a set of criteria, using fuzzy techniques. Finally, we select, through a cost-benefit analysis process, the affordable Cloud plan that is considered the best fit for all applications.

## 5.1 INTRODUCTION

Cloud computing is one of the most revolutionary advances in the history of computing. Thanks to the significant benefits of Cloud-based solutions regarding the delivery of elastic services, which can be accessed universally in a cost-effective fashion, more organizations and individuals are relying on external Cloud providers for storing and processing their data and applications [115][116]. Today, Cloud providers suggest different plans, with various interesting characteristics, to compete in the highly competitive Cloud market. Such diversity allows users to choose plans that efficiently satisfy both functional and nonfunctional requirements of their applications.

However, the availability of several Cloud plans in the market is a dual-edged sword. On the one hand, the variety of Cloud plans, which notably increases their flexibility to meet user requirements, may be seductive. On the other hand, choosing a suitable Cloud plan when multiple applications, at the same time, are moved to the Cloud, is an essential challenge as different applications might have different (and possibly) contrasting requirements. For example, while computation intensive applications (e.g. image/signal processing applications) that manage data with no security protection (e.g., publicly available or non-confidential data) require plans with high performance features (e.g., CPU rates, disk speed), applications that process sensitive data are mostly interested in plans with high security features (e.g., encryption algorithms, authentication mechanisms) to meet their requirements. Therefore, due to the heterogeneous requirements of applications, a single Cloud plan that meets all of them might not exist. It is then important to properly combine the requirements of applications to select a Cloud plan that satisfies them in the best possible way.

The problem of Cloud plan selection can get even more complicated when multiple users, possibly without an IT background, contribute to the selection process as they might not have precise ideas about the requirements of applications and the characteristics of Cloud plans. For example, they might be uncertain about the required number of synchronized replicas or backup schedule for an application to be outsourced. In this case, she may prefer to state "high" as the relevance of *availability* criterion for the application. In such context, traditional quantitative approaches for helping users in selecting a Cloud plan, which implicitly assume the user to be familiar with the technical requirements of applications and the characteristics of Cloud plans, are not considered as efficient. Such approaches might not precisely capture the uncertainty of users about application requirements, and as a result, select plans that do not adequately meet them.

In this chapter, we propose an approach aimed at choosing the Cloud plan, among those available from a Cloud provider that best satisfies imprecise information associated with applications to be outsourced, expressed by multiple users with a limited budget for selecting a Cloud plan. To do so, our approach, first, calculates the importance of applications, and then, their aggregated preferences over each considered criterion, using a *fuzzy technique* (weighted triangular average (WTA) [117]). It then selects, through a *cost-benefit analysis* process, the affordable plan that best fits the aggregated preferences of applications. The proposed approach provides a flexible tool which efficiently manages Cloud plan selection scenarios that include dealing with information, provided by a set of unskilled IT users, while properly meets their budget constraints for selecting Cloud plans.

### 5.1.1 CHAPTER OUTLINE

This chapter is organized as follows. Section 5.2 presents our problem. Section 5.3 illustrates our proposed solution. Section 5.4 provides a pseudo-code algorithm for the proposed solution. Finally, Section 5.5 presents chapter summary and concluding remarks.

## 5.2 BASIC CONCEPTS AND PROBLEM DEFINITION

We consider a scenario, as depicted in Figure 5.1, characterized by a set $U = \{u_1, \ldots, u_h\}$ of users who wish to outsource a set $A = \{a_1, \ldots, a_n\}$ of applications to the Cloud. In the proposed scenario, users can be considered as the representatives of their respective organization who are responsible for selecting one single Cloud plan, among a set $P = \{p_1, \ldots, p_m\}$ of plans, offered by a set of Cloud providers, for all applications in $A$. Each plan is associated with a *price vector* $\mathbb{P}[1, \ldots, m]$, where $\mathbb{P}[j]$ denotes the price of plan $p_j$. For example, $\mathbb{P} = [70, 55, 45, 45, 35, 30]$, where $\mathbb{P}[2] = 55\$$ is the price of plan $p_2$. Also, users in $U$ consider a *budget* $b$ which represents their financial constraint for selecting a Cloud plan. For example, budget $b = 60\$$ implies that the users can afford plans with prices less than or equal to the budget (60$).

Each plan $p \in P$ might have different characteristics (w.r.t. a set $C = \{c_1, \ldots, c_l\}$ of criteria) that are considered necessary for the set $A$ of applications. For example, $C$ can include criteria such as availability, performance, security, and elasticity, guaranteed by the providers (i.e., $C = \{$*Availability, Performance, Security, Elasticity*$\}$). Since users in $U$ may not come from an IT background, they may not be aware of criteria that are necessary for each application $a_i \in A$. Therefore, to simplify the process for users in $U$, we consider a trusted third auditor (TTA), which provides consultation services to users. We assume that the TTA is trusted by both users in $U$ and the Cloud provider(s). Then, the TTA defines the set of criteria $C$ for the applications. In the definition of $C$, the TTA can consider existing references and classifications (e.g., [110]) as well as individual requirements for each application in $A$.

To enable comparison between the plans in $P$, we assume that each plan $p_j \in P$ is rated by the TTA, considering each criterion $c_k \in C$, denoted as $p_j \rightsquigarrow R_j$ in Figure 5.1. This rating indicates the degree to which a criterion $c_k \in C$ is "satisfied" by a plan $p_j \in P$. That is, it instinctively shows how much the offered services by a plan are close to an optimal scenario that maximizes the fulfilment of criterion $c_k$ (e.g., a plan offering its services in an "always-available" manner would have the maximum rating for *Availability* criterion). Formally, each plan $p_j$ is associated with a *rating vector* $R_j[1, \ldots, l]$, where $0 < R_j[k] \leqslant 1$ denotes the rating degree of plan $p_j$, considering criterion $c_k$. For example, a plan providing a low number of synchronized replicas, high CPU rates, simple encryption, and load balancing algorithms, will have a high rating w.r.t. *Performance* and low ratings for *Availability*, *Security*, and *Elasticity*. Table 5.1 shows the example of rating vectors $R_1, \ldots, R_6$ for plans $p_1, \ldots, p_6$, over criteria defined in $C$. As an example, plan $p_2$ is rated lower for *Availability* ($R_2[\mathit{Availability}] = 0.720$) compared to *Performance* ($R_2[\mathit{Performance}] = 0.780$).

Moreover, each user might consider a different importance for each application. A natural way to consider the importance of applications is associating a weight to each application $a_i$, by each user. However, as we discussed in Section 5.1, it is not often an easy task when users are uncertain about exact and specific requirements that are considered for applications, including their importance. In other words, if users, possibly without IT skills, are interested in considering some importance for application $a_i$, compared to application $a_j$, they may not be able to quantitatively express it (e.g.,

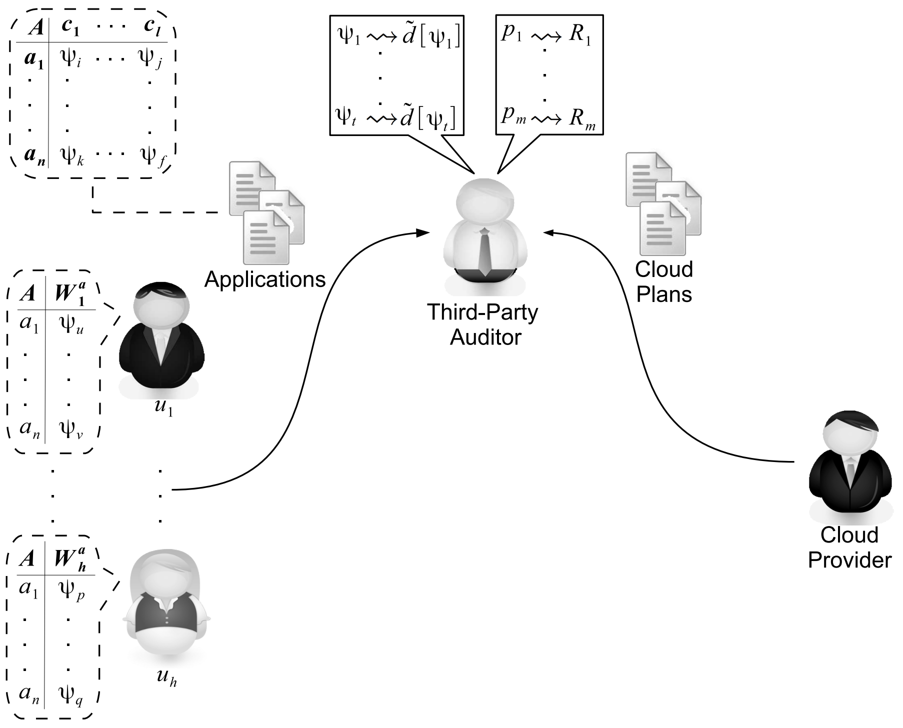
$\psi_1 \rightsquigarrow \tilde{d}[\psi_1]$
$\psi_t \rightsquigarrow \tilde{d}[\psi_t]$
$p_1 \rightsquigarrow R_1$
$p_m \rightsquigarrow R_m$
Applications
Third-Party Auditor
Cloud Plans
Cloud Provider
$u_1$
$u_h$

**Figure 5.1:** The Reference scenario

| | *Availability* | *Performance* | *Security* | *Elasticity* |
|---|---|---|---|---|
| **$R_1$** | 0.900 | 0.800 | 0.850 | 0.950 |
| **$R_2$** | 0.720 | 0.780 | 0.620 | 0.750 |
| **$R_3$** | 0.740 | 0.650 | 0.450 | 0.640 |
| **$R_4$** | 0.910 | 0.570 | 0.400 | 0.580 |
| **$R_5$** | 0.240 | 0.650 | 0.520 | 0.700 |
| **$R_6$** | 0.200 | 0.300 | 0.350 | 0.400 |

**Table 5.1:** The example of rating vectors $R_1, \ldots, R_6$

the importance of application $a_i$ is twice more than the importance of $a_j$). Then, they may prefer to linguistically express the importance of applications. For example, *Alice* believes that the importance of application $a_4$ is important, while *Bob* finds it as normal. Therefore, *Alice* and *Bob* may express the importance of application $a_4$ as "High" and "Medium", respectively, which are not precise enough to reason about it. To alleviate this concern, each user $u_v \in U$ is associated with a *linguistic importance vector* $W^a_v$, where $W^a_v[i] \in \Psi$ is the linguistic importance of application $a_i$. We note that, $\Psi = \{\psi_1, \ldots, \psi_t\}$ is a *linguistic set* that is totally ordered under the relation $<$ (i.e., $\psi_1 < \psi_2 < \ldots < \psi_t$). Back to our running example, $\Psi$ = {*Very Low (VL), Low (L), Medium (M), High (H), Very High(VH)*}, where, as an instance, *"VH"* indicates higher importance compared to *"H"*. Table 5.2 shows the linguistic importance vectors $W^a_{Alice}, W^a_{Bob}, W^a_{Carol}$ for *Alice*, *Bob*, and *Carol* in our running example, respectively. For

| | *Alice* | *Bob* | *Carol* |
|---|---|---|---|
| **$W^u$** | 0.7 | 0.4 | 0.2 |
| | | **$W^a$** | |
| **$a_1$** | *H* | *L* | *H* |
| **$a_2$** | *VL* | *M* | *M* |
| **$a_3$** | *M* | *H* | *M* |
| **$a_4$** | *H* | *M* | *VH* |

**Table 5.2:** User weight $W^u$ and linguistic importance vectors $W^a_{Alice}$, $W^a_{Bob}$, $W^a_{Carol}$

| | *Availability* | *Performance* | *Security* | *Elasticity* |
|---|---|---|---|---|
| **$a_1$** | *VL* | *H* | *VL* | *H* |
| **$a_2$** | *L* | *H* | *VH* | *L* |
| **$a_3$** | *H* | *M* | *H* | *M* |
| **$a_4$** | *VL* | *H* | *M* | *VH* |

**Table 5.3:** Linguistic criteria matrix $\mathbb{L}$

instance, $W^a_{Alice}[a_4] = H$ and $W^a_{Bob}[a_4] = M$ are the linguistic importance of application $a_4$, expressed by *Alice* and *Bob*, respectively.

In addition, given an application $a_i$ and a set of criteria $C$, not all criteria $c_k \in C$ can be assumed to be equally relevant to application $a_i$, from the view of users. For example, suppose that for application $a_1$, which is not an "always-on" application, due to the high frequency of request variations, *Elasticity* criterion is more relevant than *Availability* criterion. Therefore, for each application $a_i$, to consider the relevance of criterion $c_k$, it is associated with a *linguistic weight* $\psi \in \Psi$. Formally, the extent of the relevance of criterion $c_k \in C$ for application $a_i \in A$ is expressed in a *linguistic criteria matrix* $\mathbb{L}$ of size $n \times l$, with a row for each application $a_i \in A$ and a column for each criterion $c_k \in C$. Each cell $\mathbb{L}[i][k] \in \Psi$ represents the linguistic weight of criterion $c_k$ for application $a_i$. Table 5.3 presents the linguistic criteria matrix $\mathbb{L}$ in our running example. For example, *Availability* criterion for application $a_1$ ($\mathbb{L}[a_1][Availability] = M$) is less relevant than for application $a_3$ ($\mathbb{L}[a_3][Availability] = H$).

In order to capture the uncertainty of each linguistic variable $\psi \in \Psi$, it is associated, by the TTA, with a triangular fuzzy number (TFN) as one of the most popular and widely used types of fuzzy numbers due to its representation and computational simplicity [118]. Formally, translated TFNs are maintained in a vector $\tilde{d}[1, \ldots, t]$, where $\tilde{d}[\psi]$ is the associated TFN of linguistic variable $\psi$, denoted as $\psi \rightsquigarrow \tilde{d}[\psi]$ in Table 5.1. We note that it is possible to consider different TFN vectors for the importance of applications (see columns in Table 5.2) as well as a different TFN vector for each criterion $c_k \in C$ (see columns in Table 5.3). However, to keep the model simple, we skip considering such TFN vectors. Table 5.4 shows the TFN vector $\tilde{d}$ in our running example. For instance, $\tilde{d}[M]$, which is the associated TFN with the linguistic variable "*M*" is $(0.1, 0.3, 0.75)$ (i.e., $\tilde{d}[M] = (0.1, 0.3, 0.75)$). In order to provide a graphical illustration of associations between the linguistic variables in $\Psi$ and TFNs in $\tilde{d}$, Figure 5.2 presents

| $\psi$ | $\tilde{d}$ |
|---|---|
| Very Low (*VL*) | $(0, 0, 0.1)$ |
| Low (*L*) | $(0, 0.1, 0.3)$ |
| Medium (*M*) | $(0.1, 0.3, 0.75)$ |
| High (*H*) | $(0.3, 0.75, 1)$ |
| Very High (*VH*) | $(0.75, 1, 1)$ |

**Table 5.4:** Linguistic set $\Psi$ and TFN vector $\tilde{d}$

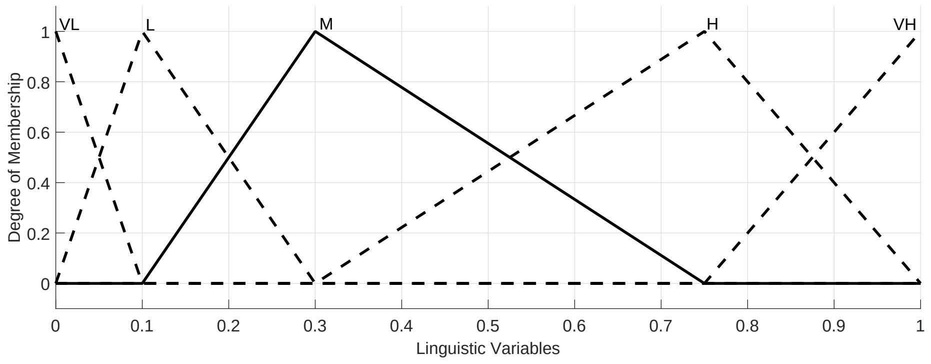

**Figure 5.2:** Membership functions for TFNs in $\tilde{d}$

the membership functions for each TFN in $\tilde{d}$. For example, the associated membership function of TFN $(0.1, 0.3, 0.75)$ is plotted with continuous lines.

Moreover, opinions that each user $u \in U$ holds about the relevance of each criterion $c_k \in C$ for each application $a_i \in A$, might have a different influence (e.g., according to the organizational role of user, his/her experience) in Cloud plan selection process which can be defined in several ways (e.g., by the organization manager). For example, considering the set $U = \{Alice, Bob, Carol\}$ of users, the opinions of *Alice*, which has a higher role in her organization compared to *Bob*, have more influence than those of *Bob* in choosing a suitable plan. To consider the extent of the influence of each user $u_v$ in Cloud selection process, s/he is associated with a *weight* $0 < W^u[v] \leqslant 1$, where higher weights model higher influence of user $u_v$ in the plan selection process. Such user weights are maintained in a *user weight vector* $W^u[1, \dots, h]$. Considering the user weight vector $W^u = [0.7, 0.4, 0.2]$ in our running example, the influence of the opinions of *Alice* ($W^u[Alice] = 0.7$) in Cloud plan selection process is more than *Bob* ($W^u[Bob] = 0.4$) and *Carol* ($W^u[Carol] = 0.2$).

Given an application $a_i$ and a set $P$ of plans, a user $u \in U$ can choose the best fit Cloud plan $p \in P$ considering the requirements of application $a_i$, by using classical multi-criteria decision making approaches (e.g., [111]). However, the problem can get more aggravated when a set of unskilled IT users, with a limited budget for selecting a Cloud plan, contribute in the process of choosing a Cloud plan for multiple outsourcing applications as it could significantly increase the rate of high-level imprecise information, provided by users. To overcome these challenges, we propose an approach aimed at choosing an affordable plan considering imprecise information provided by a set of users which may not have precise ideas about the requirements of applications.

## 5.3 PROPOSED SOLUTION FOR CLOUD PLAN SELECTION

Our methodology to choose the best fit affordable plan, based on imprecise information about applications in $A$, provided by users in $U$, operates in three main steps: *i) measure crisp importance* for each application $a_i \in A$, based on the linguistic importance of $a_i \in A$, expressed by each user in $U$; *ii) measure crisp weight* for each criterion $c_k \in C$, based on the crisp importance of applications, obtained in the first step, and the linguistic weight of criterion $c_k \in C$ ; *iii) select a plan* that is the best fit for applications, considering the obtained aggregated weight of each criterion and budget $b$ of users for selecting a Cloud plan.

Initially, to handle the impreciseness of linguistic criteria matrix $\mathbb{L}$ (see Table 5.3) and linguistic importance vectors $W^a_1, \ldots, W^a_h$ (see Table 5.2), we obtain *fuzzy criteria matrix* $\widetilde{\mathbb{L}}$ and *fuzzy importance vectors* $\widetilde{W^a_1}, \ldots, \widetilde{W^a_h}$, respectively. To do so, we replace each linguistic variable $\psi \in \Psi$ with the associated TFN $\tilde{d}[\psi] \in \tilde{d}$ (see Table 5.4). Tables 5.5 and 5.6 show *fuzzy importance vectors* for *Alice*, *Bob*, and *Carol* and *fuzzy criteria matrix* $\widetilde{\mathbb{L}}$, respectively.

In the following, we present our approach for, first, measuring the crisp importance of each application in $A$, and then, computing the aggregated weight of each criterion $c_k \in C$, using WTA. Finally, we choose, through a cost-benefit analysis process, the plan that is the best fit for applications in $A$, considering the obtained aggregated weight of each criterion $c_k \in C$ and budget $b$ of users in $U$ for selecting a Cloud plan.

### 5.3.1 MEASURING IMPORTANCE FOR EACH APPLICATION

To be able to consider the importance of applications when computing a crisp weight for each criterion $c_k \in C$ (see Section 5.3.2), the first step of our solution aims at measuring a numerical importance for each application $a_i \in A$ which reflects the aggregated opinions of users in $U$ about the importance of $a_i$, considering their weights, maintained in the user weight vector $W^u$. In the remainder of this section, we will present our solution for computing the importance of applications in $A$ in detail.

**Aggregated fuzzy importance vector.** To determine the importance of each application $a_i \in A$, we propose to adopt WTA technique to aggregate the fuzzy importance values $\widetilde{W^a_1}[i], \ldots, \widetilde{W^a_h}[i]$ of $a_i$ (see Table 5.5), respectively expressed by users $u_1, \ldots, u_h$, considering their weights maintained in the user weight vector $W^u$. To do so, for each application $a_i \in A$, we obtain an aggregated fuzzy importance $\widetilde{W^a_{agg}}[i] = (W^a_{\mathscr{L}}[i], W^a_{\mathscr{M}}[i], W^a_{\mathscr{U}}[i])$ as $\sum_{v=1}^{|U|} W^u[v] \cdot \widetilde{W^a_v}[i] / \sum_{v=1}^{|U|} W^u[v]$, maintained in an *aggregated fuzzy importance vector* $\widetilde{W^a_{agg}}[1, \ldots, n]$. We note that $W^a_{\mathscr{M}}[i]$ is the middle value, and $W^a_{\mathscr{L}}[i]$ and $W^a_{\mathscr{U}}[i]$ respectively are the lower and the upper bound values for the importance of application $a_i$. For example, as depicted in Table 5.5, the aggregated

| | *Alice* | *Bob* | *Carol* | | |
|---|---|---|---|---|---|
| $\mathbf{W^u}$ | 0.7 | 0.4 | 0.2 | | |
| | $\widetilde{\mathbf{W}}^{\mathbf{a}}_{Alice}$ | $\widetilde{\mathbf{W}}^{\mathbf{a}}_{Bob}$ | $\widetilde{\mathbf{W}}^{\mathbf{a}}_{Carol}$ | $\widetilde{\mathbf{W}}^{\mathbf{a}}_{agg}$ | $W^a$ |
| $\mathbf{a_1}$ | $(0.3, 0.75, 1)$ | $(0, 0.1, 0.3)$ | $(0.3, 0.75, 1)$ | $(0.207, 0.550, 0.784)$ | 0.514 |
| $\mathbf{a_2}$ | $(0, 0, 0.1)$ | $(0.1, 0.3, 0.75)$ | $(0.1, 0.3, 0.75)$ | $(0.046, 0.138, 0.400)$ | 0.194 |
| $\mathbf{a_3}$ | $(0.1, 0.3, 0.75)$ | $(0.3, 0.75, 1)$ | $(0.1, 0.3, 0.75)$ | $(0.161, 0.438, 0.826)$ | 0.475 |
| $\mathbf{a_4}$ | $(0.3, 0.75, 1)$ | $(0.1, 0.3, 0.75)$ | $(0.75, 1, 1)$ | $(0.307, 0.650, 0.923)$ | 0.626 |

**Table 5.5:** Application fuzzy importance $\widetilde{W}^a_{Alice}, \widetilde{W}^a_{Bob}, \widetilde{W}^a_{Carol}$ vectors respectively for users *Alice*, *Bob*, *Carol*, application aggregated fuzzy importance $\widetilde{W}^a_{agg}$ vector, and application importance vector $W^a$

fuzzy importance $\widetilde{W}^a_{agg}[a_1]$ of application $a_1$ is $\sum_{v=1}^{3} W^u[v] \cdot \widetilde{W}^a_v[a_1] / \sum_{v=1}^{3} W^u[v] = (0.7 \cdot (0.3, 0.75, 1) + 0.4 \cdot (0, 0.1, 0.3) + 0.2 \cdot (0.3, 0.75, 1))/(0.7 + 0.4 + 0.2) = (0.207, 0.550, 0.784)$.

**Application importance.** To obtain a crisp numerical value for the importance of each application $a_i$, the aggregated fuzzy importance $\widetilde{W}^a_{agg}[i] = (W^a_{\mathscr{L}}[i], W^a_{\mathscr{M}}[i], W^a_{\mathscr{U}}[i])$ of $a_i$ is defuzzified into its best non-fuzzy performance (BNP) value $W^a[i]$ as $((W^a_{\mathscr{U}}[i] - W^a_{\mathscr{L}}[i]) + (W^a_{\mathscr{M}}[i] - W^a_{\mathscr{L}}[i]))/3 + W^a_{\mathscr{L}}[i]$. We use the center of area (COA) method to measure the BNP value of $\widetilde{W}^a_{agg}[i]$, which is one of the most popular and widely used defuzzification methods due to its simplicity and practicality [119] [120]. The importance $W^a[i]$ of each application $a_i$ is maintained in an *importance vector* $W^a$. For example, as depicted in Table 5.5, the importance $W^a[a_1]$ of application $a_1$, measured as $((W^a_{\mathscr{U}}[a_1] - W^a_{\mathscr{L}}[a_1]) + (W^a_{\mathscr{M}}[a_1] - W^a_{\mathscr{L}}[i]))/3 + W^a_{\mathscr{L}}[a_1] = ((0.784 - 0.207) + (0.550 - 0.207))/3 + 0.207 = 0.514$, is higher than the importance $W^a[a_2]$ of application $a_2$ (0.194).

### 5.3.2 MEASURING CRITERIA WEIGHTS

To be able to compare the preferences of applications in $A$, over criteria in $C$, as a single global ecosystem with the characteristics of plans in $P$ (see Section 5.3.3), the second step of our solution aims at measuring a numerical weight for each criterion $c_k \in C$, where higher weights for $c_k$ represent higher importance of $c_k$ for applications. In the remainder of this section, we will present our solution for measuring the weight of each criterion $c_k \in C$ in detail.

**Fuzzy criteria vector.** We adopt WTA technique to properly consider the importance $W^a[i] \in W^a$ of each application $a_i \in A$ when aggregating the fuzzy weights of each criterion $C_k \in C$ for applications, expressed in fuzzy criteria matrix $\widetilde{\mathbb{L}}$ (see Table 5.6). We then obtain, for each criterion $c_k \in C$, a fuzzy weight $\widetilde{W}^c[k] = (W^c_{\mathscr{L}}[k], W^c_{\mathscr{M}}[k], W^c_{\mathscr{U}}[k])$ as $\sum_{i=1}^{|A|} W^a[i] \cdot \widetilde{\mathbb{L}}[i][k] / \sum_{i=1}^{|A|} W^a[i]$, maintained in a *fuzzy criteria vector* $\widetilde{W}^c[1, \ldots, l]$. We note that $W^c_{\mathscr{M}}[k]$ is the middle value, and $W^c_{\mathscr{L}}[k]$ and $W^c_{\mathscr{U}}[k]$ are respectively the lower and upper bound values for the weight of criterion $c_k \in C$. For example, as depicted

| | | $\widetilde{\mathbb{L}}$ | | | |
|---|---|---|---|---|---|
| | $\mathbf{W^a}$ | *Availability* | *Performance* | *Security* | *Elasticity* |
| $\mathbf{a_1}$ | 0.514 | $(0.1, 0.3, 0.75)$ | $(0.3, 0.75, 1)$ | $(0, 0, 0.1)$ | $(0.3, 0.75, 1)$ |
| $\mathbf{a_2}$ | 0.194 | $(0.3, 0.75, 1)$ | $(0.3, 0.75, 1)$ | $(0.75, 1, 1)$ | $(0, 0.1, 0.3)$ |
| $\mathbf{a_3}$ | 0.475 | $(0.3, 0.75, 1)$ | $(0.1, 0.3, 0.75)$ | $(0.3, 0.75, 1)$ | $(0.1, 0.3, 0.75)$ |
| $\mathbf{a_4}$ | 0.626 | $(0.75, 1, 1)$ | $(0.3, 0.75, 1)$ | $(0.1, 0.3, 0.75)$ | $(0.75, 1, 1)$ |
| | $\widetilde{\mathbf{W^c}}$ | $(0.398, 0.708, 0.929)$ | $(0.247, 0.631, 0.934)$ | $(0.194, 0.408, 0.658)$ | $(0.370, 0.648, 0.859)$ |
| | $\mathbf{W^c}$ | 0.678 | 0.604 | 0.420 | 0.626 |

**Table 5.6:** Fuzzy criteria matrix $\widetilde{\mathbb{L}}$, fuzzy criteria vector $\widetilde{W^c}$, and criteria vector $W^c$

in Table 5.6, the fuzzy weight $\widetilde{W^c}[Availability]$ of *Availability* criterion is $\sum_{i=1}^{4} W^a[i] \cdot \widetilde{\mathbb{L}}[i][Availability] / \sum_{i=1}^{4} W^a[i] = (0.514 \cdot (0.1, 0.3, 0.75) + 0.194 \cdot (0.3, 0.75, 1) + 0.475 \cdot (0.3, 0.75, 1) + 0.626 \cdot (0.75, 1, 1))/(0.514 + 0.194 + 0.475 + 0.626) = (0.398, 0.708, 0.929)$.
**Criterion weight.** In order to measure a numerical value for the weight of each criterion $c_k$, adopting COA method, the fuzzy weight $\widetilde{W^c}[k] = (W^c_{\mathscr{L}}[k], W^c_{\mathscr{M}}[k], W^c_{\mathscr{U}}[k])$ of criterion $c_k$ is defuzzfized into its BNP value as $((W^c_{\mathscr{U}}[k] - W^c_{\mathscr{L}}[k]) + (W^c_{\mathscr{M}}[k] - W^c_{\mathscr{L}}[k]))/3 + W^c_{\mathscr{L}}[k]$, maintained in a *criteria vector* $W^c[1, \ldots, l]$. For example, as depicted in Table 5.6, the weight $W^c[Availability]$ of *Availability* criterion, measured as $((W^c_{\mathscr{U}}[Availability] - W^c_{\mathscr{L}}[Availability]) + (W^c_{\mathscr{M}}[Availability] - W^c_{\mathscr{L}}[Availability]))/3 + W^c_{\mathscr{L}}[Availability] = ((0.929 - 0.398) + (0.708 - 0.398))/3 + 0.398 = 0.678$, is higher than the weight $W^c[performance]$ of *performance* criterion (0.604).

## 5.3.3 CHOOSING THE OPTIMAL PLAN

The third step of our approach aims at choosing an affordable plan that best satisfies the aggregated weights of criteria, considering criteria vector $W^c$ and budget $b$, adopting a cost-benefit analysis process. In doing so, to measure to what extent plans in $P$ satisfy criteria in $C$, w.r.t. criteria vector $W^c$, for each plan $p_j \in P$, we obtain a distance $\mathbb{D}[j]$ between $p_j$ and criteria vector $W^c$ as $\sum_{k=1}^{|C|} R_j[k] - W^c[k]$, where higher distance values indicate the better satisfaction of criteria in $C$ by plan $p_j$. Such distance values are maintained in a *distance vector* $\mathbb{D}[1, \ldots, m]$. For example, as depicted in Table 5.7, plan $p_1$ better satisfies criteria in $C$, w.r.t. criteria vector $W^c$, compared to plan $p_2$ because distance $\mathbb{D}[1]$ between $p_1$ and $W^c$ (i.e., 0.016) is higher than distance $\mathbb{D}[2]$ between $p_2$ and $W^c$ (i.e., 0.009).

To choose a plan, we first consider plans in $P$ that are dominant, maintained in a *dominant plan set* $P_{dom} = \{p_1, \ldots, p_d\}$. A dominant Cloud plan $p_j$ in our scenario, is one that, considering each criterion $c_k \in C$, the associated rating $R_j[k]$ is higher than (or equal to) the respective criterion weight $W^c[k]$ in criteria vector $W^c$. Therefore, dominant plans in $P_{dom}$ can better satisfy criteria in $C$, and as a result, more desirable to be selected applications in $A$. Back to our running example, considering the set of dominant plans $P_{dom} = \{p_1, p_2, p_3\}$, for instance, the rating $R_1[k]$ of plan $p_1$, for each criterion $c_k \in C$, is higher than the associated criterion weight $W^c[k]$. However, plan $p_4$

| | *Availability* | *Performance* | *Security* | *Elasticity* | $\mathbb{D}$ | $\mathbb{P}$ | $\mathbb{R}$ | $\Delta^-$ |
|---|---|---|---|---|---|---|---|---|
| **$R_1$** | 0.900 | 0.800 | 0.850 | 0.950 | 1.172 | 70 | 0.016 | ▨ |
| **$R_2$** | 0.720 | 0.780 | 0.620 | 0.750 | 0.542 | 55 | 0.009 | ▨ |
| **$R_3$** | 0.740 | 0.650 | 0.450 | 0.640 | 0.152 | 45 | 0.003 | ▨ |
| **$R_4$** | 0.910 | 0.570 | 0.400 | 0.580 | 0.132 | 45 | ▨ | −0.1 |
| **$R_5$** | 0.240 | 0.650 | 0.520 | 0.700 | −0.218 | 35 | ▨ | −0.438 |
| **$R_6$** | 0.200 | 0.300 | 0.350 | 0.400 | −1.078 | 30 | ▨ | −1.078 |
| **$W^c$** | 0.678 | 0.604 | 0.420 | 0.626 | | | | |

**Table 5.7:** Distance vector $\mathbb{D}$, price vector $\mathbb{P}$, ratio vector $\mathbb{R}$, and negative difference vector $\Delta^-$

is not a dominant one because, considering *Performance* and *Security* criteria, the rating of $p_4$ (i.e., $R_4[\text{Performance}] = 0.570$ and $R_4[\text{Security}] = 0.400$) are respectively lower than the associated weights in criteria vector $W^c$ (i.e., $W^c[\text{Performance}] = 0.604$ and $W^c[\text{Security}] = 0.420$). To measure to what extent the satisfaction of criteria in C by each dominant plan $p_j \in P_{dom}$, w.r.t. criteria vector, prevails the associated price $\mathbb{P}[j]$, we measure the distance-to-price ratio of plan $p_j$ as $\frac{\mathbb{D}[j]}{\mathbb{P}[j]}$, maintained in an ordered *ratio vector* $\mathbb{R}[1, \dots, d]$ satisfying $\mathbb{R}[1] > \dots > \mathbb{R}[d]$. Then, considering budget $b$ and price vector $\mathbb{P}$, we choose an affordable plan $p_j \in P_{dom}$ (i.e., $b >= \mathbb{P}[j]$) with maximum distance-to-price ratio. Back to our running example, as depicted in Table 5.7, considering budget $b = 60\$$ and price vector $\mathbb{P}$, plan $p_2 \in P_{dom}$ is the dominant one that is chosen for applications in A as it is an affordable plan ($b > \mathbb{P}[2]$) with maximum distance-to-price ratio ($\mathbb{R}[2] = 0.009$).

If users in U can not afford a dominant plan, among those in $P_{dom}$, then to select a plan, we inevitably need to consider plans that are not dominant. For example, if budget $b = 50\$$, then plans in the set $P_{dom}$ of dominant plans are not affordable for users, and as a result, we need to select a plan among non-dominant ones (e.g., with reference to our running example, $p_4$, $p_5$, and $p_6$). Different ways can be suggested to select an affordable non-dominant plan $p_j \in (P \backslash P_{dom})$ (e.g., selecting the one with minimum number of negative distance values w.r.t. each criterion $c_k$ (i.e., $R_j[k] - W^c[k]$), selecting the one with maximum distance value, w.r.t. either of criteria in C). In this work, to select a non-dominant plan, among those in $P \backslash P_{dom}$, we choose the affordable one $p_j \in (P \backslash P_{dom})$, with the maximum sum of negative differences between the rating $R_j[k]$ of $p_j$ and criteria vector $W^c$ (i.e., $\sum_{\forall c_k \in C : R_j[k] < W^c[k]} R_j[k] - W^c[k]$), where such sum values are maintained in an ordered *negative difference vector* $\Delta^-[1, \dots, m-d]$ which satisfies that $\Delta^-[1] >= \dots >= \Delta^-[m-d]$. In this fashion, it can be assured that, by selecting non-dominant plan $p_j$, while criterion each $c_k \in C$ is adequately fulfilled when $R_j[k] >= W^c[k]$, fulfilling criteria that are not suitably satisfied (i.e., $R_j[k] < W^c[k]$, w.r.t. criterion $c_k \in C$) is maximized. Back to our running example, as depicted in Table 5.7, considering budget $b = 50\$$ and negative difference vector $\Delta^- = [-0.1, -0.438, -1.078]$, plan $p_4$ is the one that is chosen for applications in $A$ as it is an affordable plan ($\mathbb{P}[4] = 45\$$) with the maximum sum of negative differences between the rating of $p_4$ and criteria vector $W^c$ (i.e., $\Delta^-[1] = (R_4[\textit{Performance}] - W^c[\textit{Performance}]) + (R_4[\textit{Security}] - W^c[\textit{Security}]) + (R_4[\textit{Elasticity}] - W^c[\textit{Elasticity}]) = -0.034 - 0.02 - 0.046 = -0.1$), compared to other non-dominant plans (i.e., $p_5$ and $p_6$). We note that, if we need to select a plan, among those affordable non-dominant ones, when the sum of negative differences between their ratings and criteria vector $W^c$ is equal, we will choose the one with the highest distance from $W^c$, and in this manner, it is guaranteed that the selected plan is the best fit for applications in $A$, considering $W^c$ and the budget $b$ of users in $U$ for selecting a plan.

## 5.4 ALGORITHM FOR THE PROPOSED UNCERTAINTY-BASED CLOUD PLAN SELECTION APPROACH

Given the problem of choosing a Cloud plan, among those available ones, we provide a pseudo-code algorithm for our proposed solution. The algorithm, reported in Figure 5.3, takes as input a set $U$ of users, a set $A$ of applications, a set $P$ of plans, a set $C$ of criteria, a set $\Psi$ of linguistic variables, a linguistic criteria matrix $\mathbb{L}$, a set of linguistic importance vectors $W^a_1, \dots, W^a_h$, a user weight vector $W^u$, and a set of rating vectors $R_1, \dots, R_m$, a plan price vector $\mathbb{P}$, and a budget $b$. Also, it returns plan $p \in P$ as an optimal plan based the uncertain requirements of applications, provided by the users in $U$.

First, it is checked if there is no affordable plan, then there is no feasible solution for the problem (lines 1–2). Then, if there is only one affordable plan, then it is returned as only feasible solution (lines 3–4). Next, to handle the impreciseness of linguistic criteria matrix $\mathbb{L}$ (see Table 5.3) and linguistic importance vectors $W^a_1, \dots, W^a_h$ (see Table 5.2), *fuzzy criteria matrix* $\widetilde{\mathbb{L}}$ (lines 5–6) and *fuzzy importance vectors* $\widetilde{W}^a_1, \dots, \widetilde{W}^a_h$ (lines 7–9) are obtained, respectively. In addition, for each application $a_i \in A$, the aggregated fuzzy importance $\widetilde{W}^a_{agg}[i]$ is calculated. Next, the aggregated fuzzy importance $\widetilde{W}^a_{agg}[i]$ for each application is defuzzified to obtain a crisp importance $W^a[i]$ of each application (lines 10–14). Further, for each criterion $c_k \in C$, its fuzzy weight $\widetilde{W}^c[k]$ is determined. Then, a crisp weight $W^c[k]$ is calculated for criterion $c_k \in C$, through a defuzzification process (lines 15–19). Next, we choose the affordable plan, through a cost-benefit analysis described in Section 5.3.3, that best fits criteria vector $W^c$. (lines 20–40).

```
INPUT
$U = \{u_1, \ldots, u_h\}$ /* set of users */
$W^u$ /* user weight vector */
$A = \{a_1, \ldots, a_n\}$ /* set of applications */
$W_1^a, \ldots, W_h^a$ /* application linguistic importance vectors associated with
users */
$C = \{c_1, \ldots, c_l\}$ /* set of criteria */
$P = \{p_1, \ldots, p_m\}$ /* set of plans */
$R_1, \ldots, R_m$ /* rating vectors */
$\mathbb{P}$ /* plan price vector */
$b$ /* budget for selecting a Cloud plan */
$\Psi = \{\psi_i, \ldots, \psi_t\}$ /* set of linguistic variables */
$\mathbb{L}$ /* linguistic criteria matrix */
$\tilde{d}$ /* TFN vector */
OUTPUT
$p \in P$ /* optimal plan */
MAIN
1:  if ($\nexists p_j \in P : \mathbb{P}[j] < b$) /* if there is no affordable plan */
2:    return "There is no feasible solution!"
3:  if ($\exists! \, p_j : \mathbb{P}[j] < b$) /* if there is only one affordable plan */
4:    return $p_j$
5:  let $\widetilde{\mathbb{L}}$ be the fuzzy criteria matrix of size $|A| \times |C|$
    /* fill $\widetilde{\mathbb{L}}$ with TFNs in $\tilde{d}$ */
6:    $\widetilde{\mathbb{L}}$ :=Fill_Fuzzy_Criteria_Matrix($\mathbb{L}, \tilde{d}, \Psi$)
7:  let $\widetilde{W}_1^a, \ldots, \widetilde{W}_h^a$ be the fuzzy importance vectors of size $|A|$
8:  for each $v = 1, \ldots, |U|$ do
    /* fill $\widetilde{W}_v^a$ with TFNs in $\tilde{d}$ */
9:    $\widetilde{W}_v^a$ :=Fill_Fuzzy_Importance_Vector ($W_v^a, \tilde{d}, \Psi$)
    /* Step 1: measure the importance of each application $a_i \in A$ */
10: let $\widetilde{W}_{agg}^a$ be the aggregated fuzzy importance vector of size $|A|$
11: let $W^a$ be the importance vector of size $|A|$
12: for each $i = 1, \ldots, |A|$ do
13:   $\widetilde{W}_{agg}^a[i]$ := Fuzzy_Importance($W^u, \widetilde{W}_1^a, \ldots, \widetilde{W}_h^a$) /* fuzzy importance of $a_i$ */
14:   $W^a[i]$ := Defuzzify_Importance($\widetilde{W}_{agg}^a[i]$) /* importance of $a_i$ */
    /* Step 2: measure the weight of each criterion $c_k \in C$ */
15: let $\widetilde{W}^c$ be the fuzzy criteria vector of size $|C|$
16: let $W^c$ be the criteria vector of size $|C|$
17: for each $k = 1, \ldots, |C|$ do
18:   $\widetilde{W}^c[k]$ := Fuzzy_Weight($W^a, \widetilde{\mathbb{L}}$) /* fuzzy weight of $c_k$ */
19:   $W^c[k]$ := Defuzzify_Weight($\widetilde{W}^c[k]$) /* weight of $c_k$ */
    /* Step 3: Select optimal plan */
20: let $\mathbb{D}$ be the distance vector of size $|P|$
21: for each $j = 1, \ldots, |P|$ do
22:   $\sum_{k=1}^{|C|} R_j[k] - W^c[k]$ /* distance between $p_j$ and $W^c$ */
23: for each $j = 1, \ldots, |P|$ do
24:   if ($\nexists c_k \in C : R_j[k] < W^c[k]$) /* $p_j$ is dominant * /
25:     $P_{dom} := P_{dom} \cup p_j$
27: let $\mathbb{R}$ be the ratio vector of size $|P_{dom}|$
28: for each $j = 1, \ldots, |P_{dom}|$ do
29:   insert $\frac{\mathbb{D}[j]}{\mathbb{P}[j]}$ in $\mathbb{R}$ in decreasing order /* distance-to-price ratio for $p_j \in P_{dom}$ */
30: for each $j = 1, \ldots, |P_{dom}|$ do
31:   let $p_x$ be the plan s.t. $\mathbb{R}[j] = \frac{\mathbb{D}[x]}{\mathbb{P}[x]}$
32:   if $b > \mathbb{P}[x]$ then /* $p_x$ is affordable */
33:     return $p_k$
34: let $\Delta^-$ be the negative difference vector of size $P \setminus P_{dom}$
35: for each $j = 1, \ldots, |P \setminus P_{dom}|$ do
    /* negative differences between $R_j[k]$ and $W^c[k]$ for non-dominant plan $p_j$*/
36:   insert $\sum_{\substack{p_j \in (P \setminus P_{dom}) \\ \forall c_k \in C : R_j[k] < W^c[k]}} R_j[k] - W^c[k]$ in $\Delta^-$ in decreasing order
37: for each $j = 1, \ldots, |P \setminus P_{dom}|$ do
38:   let $p_x$ be the plan s.t. $\Delta^-[j] = \sum_{\substack{p_x \in (P \setminus P_{dom}) \\ \forall c_k \in C : R_x[k] < W^c[k]}} R_x[k] - W^c[k]$
39:   if $b > \mathbb{P}[x]$ then /* $p_x$ is affordable */
40:     return $p_x$
```

**Figure 5.3:** Algorithm for selecting the optimal plan

## 5.5 CHAPTER SUMMARY

In this chapter, we proposed to adopt an approach aimed at choosing a Cloud plan for a set of applications, where the preferences of each application, over a set of criteria (e.g., availability, performance), expressed by multiple users, with a limited budget for selecting a Cloud plan, in an imprecise (and possibly a linguistic) way. In our approach, first, the crisp importance of each application, and then, the aggregated preferences of applications over each criterion (e.g., availability, performance) are calculated, using fuzzy techniques. Finally, through a cost-benefit analysis process, the affordable plan that is the best fit for the set of applications is selected, considering the obtained aggregated criteria preferences. The proposed approach provides a flexible tool which efficiently manages Cloud plan selection scenarios that include dealing with imprecise information, provided by a set of unskilled IT users with a limited budget for selecting Cloud plans.

# 6 RISK-AWARE APPLICATION SCHEDULING IN CLOUD COMPUTING SCENARIOS

An essential challenge in outsourcing scenarios is supporting the business objectives of users when they move their applications to the Cloud. In particular, when outsourcing applications are business-critical (e.g., e-commerce applications), it could cause significant financial loss if they are not available, even for a very short time. Therefore, the presence of the risk of financial loss can make users skeptical about moving their business-critical applications to the Cloud. Therefore, it is important to satisfy the business objectives of users w.r.t the financial profit of applications when they are moved to the Cloud. This chapter introduces a solution in a multiple-application context aimed at maximizing financial profit estimated for each application by selecting a virtual machine (VM), among those available, for each application.

## 6.1 INTRODUCTION

Virtualization is known as the central key technology to deliver on-demand resources (e.g., CPU, memory, network) to users in a cost-efficient and flexible fashion [121]. Cloud providers offer their services to their users by encapsulating their Internet-scale content storage, processing, and delivery capabilities in the form of VMs [122] deployed on multiple physical machines (PMs). For example, Amazon EC2 [123] provides a diverse spectrum of VM instances which shifts away from general purpose VMs to those optimized for specific tasks (e.g., memory or computation intensive tasks).

Also, the economic advantages of Cloud services are considered as one of the main motivations of users for moving their applications to the Cloud. In particular, Cloud users with business-critical applications are concerned with not only avoiding a financial loss but also supporting the financial profit of their applications when they are executed on the Cloud. Since the quality of services (QoSs), guaranteed by Cloud providers, has clear consequences on the financial profit of outsourcing applications,

| Monthly Uptime | Service Credit |
|---|---|
| Less than 99.95% but equal to or greater than 99.0% | 10% |
| Less than 99.0% | 30% |

**Table 6.1:** Service credit schedule for VMs provided by Amazon EC2

it is vital to carefully consider such QoSs when VMs are selected for applications. In particular, one of the principal concerns of Cloud users, when they outsource their applications to the Cloud, is the *availability* of Cloud services [124] because service outage, even for a very short time, can raise serious consequences [125]. Therefore, Cloud providers should guarantee that their offered services support "plug and play", just as on-premise applications [124] which such assurances are provided in SLAs, according to negotiations between Cloud users and providers. Today, major Cloud providers (Microsoft, Amazon, Rackspace) have made huge investments to make their provided services highly available [126]. However, due to several service outage or corruption [127] [128], it is vital to carefully address such issues in scheduling applications because they would result in losing unsatisfied customers to other Cloud providers and even hindering the further adoption of Cloud computing, if the objectives of users cannot be suitably met when they move their applications to the Cloud.

To alleviate these issues, Cloud providers consider some compensation mechanisms when SLA is violated. In particular, if a Cloud provider fails to provide its promised service availability, the provider should compensate users based on some service credit [129]. Service credits play a major role in SLA violation compensation. For example, Rackspace paid $2.5-3.5$ million dollars to its users following a power outage in its Dallas data center in late June 2009 [130]. Moreover, they can restrict the behaviors of Cloud providers to prevent SLA violation, and as a result, avoid penalty costs [131]. Service credits can be measured in different ways. However, they are usually calculated by how long a service was unavailable within a specific billing period [129]. For example, the current service credit of Amazon EC2 service is defined as a percentage of total charges paid by a user for a running VM instance, according to a schedule presented in Table 6.1 [132]. For instance, Amazon compensates each user with 10% of her total monthly payment for her adopted VM, if the uptime percentage of VM is between 99.95% and 99%. Also, Amazon does not refund or any other payment, if the uptime percentage of VMs is more than 99.95% in any monthly billing cycle.

Then, in VM provisioning scenarios, which applications are usually scheduled by mapping each one to a VM, to properly meet the business objectives of users w.r.t. the financial benefit of applications, it is essential to investigate: *1) service credit* considered for each VM, if its uptime percentage is less than the commited one by Cloud provider that offers the VM; *2) off-premise profit* that is the profit of an application when it is mapped to a VM. In fact, the financial profit of each application highly depends on different uptime intervals which is defined in service credit schedules associated with VM that is selected for the application. Two main reasons can be provided for supporting this argument: *1)* if the uptime percentage of a VM, in a defined period (e.g., in an hour), falls in each uptime interval, a different service credit is applied for the VM which should be added to the financial profit of application that is mapped on the VM (see Table 6.1); *2)* since the availability of an application is tied to the availability of selected VM for the application (as it is deployed on the VM), the financial profit of application differs, if the uptime percentage of VM falls in a different uptime interval.

This chapter provides an approach in a multiple-application context aimed at, through selecting an available VM for an application, maximizing the estimated financial profit of each application, considering defined service credit schedules by several Cloud providers.

### 6.1.1 CHAPTER OUTLINE

This chapter is structured as follows. Section 6.2 provides some basic concepts on the problem and its definition. Section 6.3 presents our proposed approach for the defined problem. Section 6.4 provides an algorithm for the proposed solution. We present chapter summary and concluding remarks in Section 6.5.

## 6.2 BASIC CONCEPTS AND PROBLEM DEFINITION

In this section, we first provide an overview of risk analysis to properly clarify the motivation behind its use in our study. Then, we provide basic concepts on the problem together with its definition.

### 6.2.1 AN OVERVIEW OF RISK ANALYSIS IN THE PROPOSED STUDY

Risks to human comes to light from an inherent characteristic to make plans and try to make them fulfilled, while external forces (e.g., failure time, failure recovery time in our scenario) resist and tend to move our attempts away from the objectives of plan [133] (e.g., supporting the business objectives of users in our scenario). The definition of risk involves both uncertainty as well as some consequence might be received which could be symbolically written as [134]:

$$\text{risk}=\text{uncertanity}+\text{damage}$$

Considering our discussion in Section 6.1, the definition of risk in the context of our problem includes responses to the three following questions [134]:

1. What can happen?
    - Answer: falling the uptime percentage of available VMs in different uptime intervals
2. How likely is it that this will happen?
    - Answer: the probability of falling the uptime percentage of available VMs in each uptime interval
3. In case of happening, what are the consequences?

| | | **Consequence** | |
|---|---|---|---|
| **Monthly Uptime Interval** | **Probability** | **Service Credit** | **Monthly Financial Profit** |
| [99.96 100] | 0.997 | 0% | 17280 (usd) |
| [99 99.95] | 0.002 | 10% | 12960 (usd) |
| [98.5 98.99] | 0.001 | 30% | 7560 (usd) |

**Table 6.2:** Example of VM uptime outcomes

- Answer: Changes in *1)* the financial profit of application that is mapped on a VM; *2)* applied service credit percentage for the VM

To answer these questions, we need to make a list of outcomes. For example, as suggested in Table 6.2, which has been inspired from Table I in [134], considering the uptime percentage of a VM that is selected for an application, we can consider three outcomes:

1. if it falls in uptime interval [99.96 100];
2. if it falls in uptime interval [99 99.95];
3. if it falls in uptime interval [98.5 98.99].

For each considered outcome, with a different probability, two possible consequences can be considered. For example, considering Table 6.2, if the uptime percentage of VM falls in interval [99.96 100], with probability 0.997, no service credit is applied and the average financial profit of application is 17280 (usd/month). For simplicity, in the context of our problem, we refer to risk as falling the uptime percentage of available VMs in uptime intervals instead of an outcome list including, uptime intervals, their probabilities, and consequences (applied service credits and the financial profit of applications).

Risk analysis properly deals with decision making in situations that involve uncertainty (i.e., situations with the lack of complete and accurate knowledge about the state of system [135] [136] [137]). Therefore, since we cannot be certain about falling the uptime percentage of an available VM in an uptime interval, we adopt risk analysis to maximize the estimated financial profit of each application by choosing a VM, among those available ones, for each application.

### 6.2.2 PROBLEM DEFINITION

We consider a scenario, as depicted in Figure 6.1, characterized by a user wishing to outsource a set $A = \{a_1, \ldots, a_n\}$ of applications. To this aim, she needs to choose a VM $v_j$, from a set $V = \{v_1, \ldots, v_m\}$ of VMs that are offered by multiple Cloud providers, for each application $a_i \in A$ ($m >= n$). Since in this chapter, we focus on meeting the business objectives of users w.r.t. the financial profit of applications, to enable comparison between VMs in V based on their economic characteristics (e.g., rental cost, service

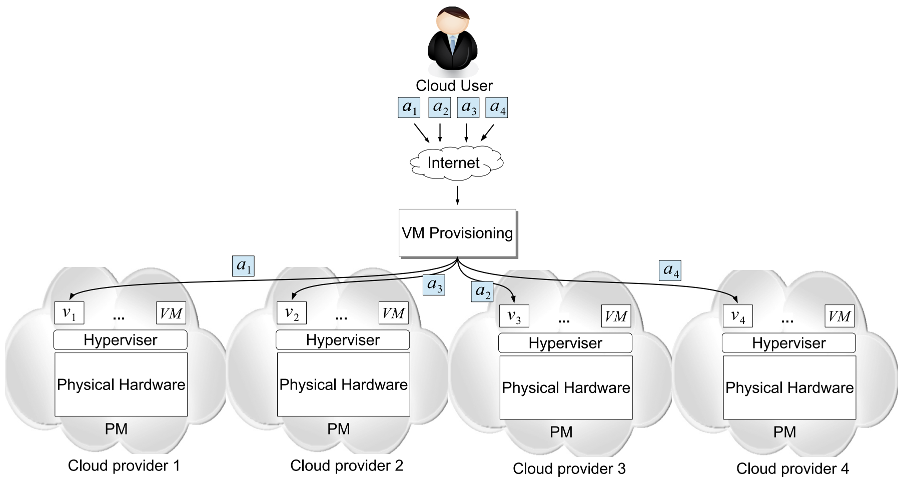

**Figure 6.1:** The reference scenario

credit), we assume that all VMs in V have the same technical characteristics (e.g., CPU rates, memory). In the next two subsections, we present the modeling of available VMs in V and outsourcing applications in A, receptively.

### 6.2.2.1 MODELING AVAILABLE VMS

As we mentioned before in Chapter 2, a Cloud plan can be any type of customized Cloud service (e.g., a (set of) VM(s)). While noting that, in our application scheduling scenario, we do not consider any pre-defined plans, each subset of available VMs that the number of its members is equal to the number $|A|$ of applications in $A$ can be considered as a plan. Figure 6.2, shows an example of three Cloud plans which each plan includes four VMs (e.g., plan $p_1$ includes $v_2, v_3, v_4$, and $v_6$).

Table 6.3 shows available VMs in our running example. Since it does not affect our solution, we assume that the number $m$ of available VMs in our running example is equal to the number $n$ of applications for the sake of simplicity (i.e., $m = n = 4$). Each VM $v_j \in V$ is associated with an hourly rental cost, defined by Cloud providers, which is paid by the user for deploying her application on $v_j$. Such rental cost values are maintained in a *rental cost vector* $R[1, \dots, m]$, where $R[j]$ is the hourly rental cost of VM $v_j \in V$. For example, considering Table 6.3, the rental cost $R[1]$ of VM $v_1 \in V$ is equal to 0.01 (usd/hour). Moreover, we assume that the hourly uptime percentage (HUP) of each VM $v_j \in V$ falls in an interval maintained in an *uptime interval vector* $I[1, \dots, d]$, where $I[k]$ is the $k^{th}$ hourly uptime interval (HUI) of VMs, defined by Cloud providers. For example, considering Table 6.3, $I[2] = [99 \;\; 99.95]$ is the $2^{nd}$ HUI of VMs in V. Also, all VMs in V are associated with a *service credit vector* $\Gamma = [1, \dots, d]$, where $\Gamma[k]$ is service credit percentage (SCP) that is associated with $k^{th}$ hourly uptime interval $I[k]$. For example, considering Table 6.3, $\Gamma[2] = 30\%$ is the SCP that is associated with HUI $I[2] = [99 \;\; 99.95]$.

Also, each VM $v_j \in V$ is associated with a *probability vector* $Pr_j[1, \dots, d]$, where $Pr_j[k]$ is probability that the HUP of VM $v_j \in V$ falls in $k^{th}$ hourly uptime interval $I_k$. For example, considering Table 6.3, $Pr_2[3] = 0.0004$ is the probability that HUP of VM

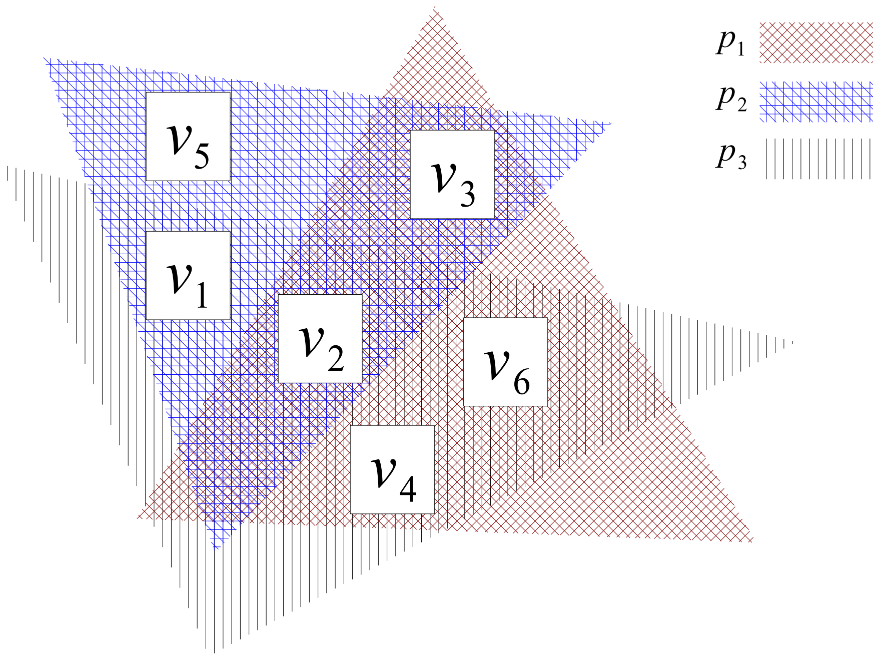

**Figure 6.2:** Example of relations between plans and VMs

| | **R[1]** | **I** | $\Gamma$ | **Pr** |
|---|---|---|---|---|
| **v₁** | 0.01 | [99.96 100] | 0 | 0.997 |
| | | [99 99.95] | 30 | 0.002 |
| | | [95 98.99] | 50 | 0.0009 |
| | | [90 94.99] | 60 | 0.0001 |

| | **R[2]** | **I** | $\Gamma$ | **Pr** |
|---|---|---|---|---|
| **v₂** | 0.007 | [99.96 100] | 0 | 0.96 |
| | | [99 99.95] | 30 | 0.0395 |
| | | [95 98.99] | 50 | 0.0004 |
| | | [90 94.99] | 60 | 0.0001 |

| | **R[3]** | **I** | $\Gamma$ | **Pr** |
|---|---|---|---|---|
| **v₃** | 0.004 | [99.96 100] | 0 | 0.86 |
| | | [99 99.95] | 30 | 0.03 |
| | | [95 98.99] | 50 | 0.095 |
| | | [90 94.99] | 60 | 0.015 |

| | **R[4]** | **I** | $\Gamma$ | **Pr** |
|---|---|---|---|---|
| **v₄** | 0.002 | [99.6 100] | 0 | 0.8 |
| | | [99 99.95] | 30 | 0.1 |
| | | [95 98.99] | 50 | 0.095 |
| | | [90 94.99] | 60 | 0.005 |

**Table 6.3:** Example of available VMs

| | Service type | $\mathbb{P}1$ | I | $\mathcal{P}_1$ |
|---|---|---|---|---|
| $\mathbf{a_1}$ | Social medial marketing | 4.9 | [99.96 100] | 4.8 |
| | | | [99 99.95] | 3.6 |
| | | | [95 98.99] | 2.1 |
| | | | [90 94.99] | 1.2 |
| | **Service type** | $\mathbb{P}2$ | **I** | $\mathcal{P}_2$ |
| $\mathbf{a_2}$ | Average traffic website | 1.9 | [99.96 100] | 4.1 |
| | | | [99 99.95] | 3.1 |
| | | | [95 98.99] | 2.1 |
| | | | [90 94.99] | 0.9 |
| | **Service type** | $\mathbb{P}3$ | **I** | $\mathcal{P}_3$ |
| $\mathbf{a_3}$ | Business logic | 2.11 | [99.96 100] | 4.3 |
| | | | [99 99.95] | 3.3 |
| | | | [95 98.99] | 2.5 |
| | | | [90 94.99] | 1.5 |
| | **Service type** | $\mathbb{P}4$ | **I** | $\mathcal{P}_4$ |
| $\mathbf{a_4}$ | Music streaming | 4.5 | [99.6 100] | 4.6 |
| | | | [99 99.95] | 3.7 |
| | | | [95 98.99] | 2.9 |
| | | | [90 94.99] | 1.9 |

**Table 6.4:** Example of on-promise profit vector $\mathbb{P}$ and off-premise profit vector $\mathcal{P}$

$v_2 \in V$ falls in HUI $I_3 = [95 \;\; 98.99]$. We note that such probabilities either could be defined by Cloud providers, if they are trusted ones or by a third party which is trusted by both the user and Cloud providers, based on the analysis of historical data regarding the availability of VMs in V. Details about the process of obtaining such probabilities are outside of the scope of this chapter.

#### 6.2.2.2 MODELING OUTSOURCING APPLICATIONS

The user defines, for each application $a_i \in A$, an on-premise financial profit which is the average hourly financial profit of $a_i \in A$ before moving to the Cloud. For example, considering Table 6.4, the on-promise financial profit of application $a_i \in A$ is 4.9 (usd/hour). Such on-premise financial profit values are maintained in an *on-promise profit vector* $\mathbb{P}[1, \ldots, n]$, where $\mathbb{P}[i]$ is the on-premise financial profit of application $a_i \in A$. Also, the user estimates, for each application $a_i \in A$, an off-premise financial profit $\mathcal{P}_i[k]$ which is the average hourly financial profit of application $a_i \in A$, if the HUP of VM $v_j \in V$ that is selected for $a_i$ falls in HUI $I[k] \in I$. Such off-premise financial profit values are maintained in an *off-premise profit vector* $\mathcal{P}_i[1, \ldots, d]$, where $\mathcal{P}_i[k]$ is the off-premise financial profit of application $a_i \in A$. For example, considering Table 6.4, the off-premise financial profit of application $a_i \in A$ is equal to 4.8 usd/hour.

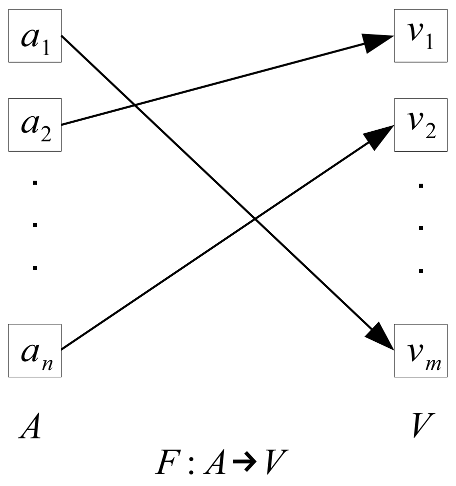

**Figure 6.3:** Example of mapping function F

Let us introduce a one-to-one *mapping function* $F : A \to V$ that takes the set $A$ of applications as input and maps each $a_i \in A$ to a VM $v_j \in V$, according to our proposed application scheduling approach. The notation $F(a) = v$ indicates that VM $v$ is selected for application $a$. Figure 6.3, which has been inspired from Figure 1.b in [68], shows an example for a mapping generated by $F(a_1, \ldots, a_n) \to (v_1, \ldots, v_m)$, where $F(a_1) = v_m, F(a_2) = v_1$, and $F(a_n) = v_2$. Mapping function F should satisfy the following two objectives when maps applications in A to VMs in V:

First, mapping function F has to guarantee the supporting of the financial profit of each application $a_i \in A$ when $a_i$ is assigned to a VM $v_j \in V$. To do so, mapping function F must capture risk associated with the falling of the HUP of each VM $v_j \in V$ in each HUI $I[k] \in I$ as it can affect:

1. *hourly penalty* for a VM $v_j \in V$. For example, suppose that, application $a_1 \in A$ (see Table 6.4) is mapped on VM $v_1 \in V$ (see Table 6.3) with rental cost $R[1] = 0.01$ (usd/hour). If the HUP of VM $v_1 \in V$ falls in HUI $I[3]$, its hourly penalty is $R[1] \cdot \Gamma[3] = 0.01 \cdot 50\% = 0.005$ (usd/hour). However, if the HUP of VM $v_1$ falls in HUI $I[2]$, its hourly penalty is $R[1] \cdot \Gamma[2] = 0.01 \cdot 30\% = 0.003$ (usd/hour).

2. *hourly off-premise financial profit* $\mathcal{P}_i[k]$ of application $a_i \in A$. For example, considering Table 6.4, if $F(a_1) = v_1$ and the HUP of VM $v_1 \in V$ falls in HUI $I[3]$, then $\mathcal{P}_1[3] = 2.1$ (usd/hour), while if the HUP of $v_2$ falls in HUI $I[2]$, then $\mathcal{P}_1[2] = 3.6$ (usd/hour).

Second, mapping function F must satisfy importance assigned to each application $a_i \in A$ when a VM $v_j \in V$ is selected for $a_i$. For example, suppose that application $a_i \in A$ has a higher importance compared to application $a_j \in A$. Also, suppose that, both applications $a_i, a_j \in A$ would make their maximum financial profit if they are mapped on VM $v_j \in V$. Therefore, VM $V_j \in V$ must be assigned to application $a_i \in A$ due to its higher importance compared to application $a_j \in A$. In our scenario, the importance of applications is defined by the user, according to their on-premise financial profit maintained in $\mathbb{P}$ (i.e., applications with higher on-premise financial profit $\mathbb{P}_i$ have higher importance from the view of user). For example, as depicted in Table 6.4, applications $a_1 \in A$ (which is a social media marketing application) and $a_2 \in A$ (which is a website with average traffic) respectively have the highest and the lowest importance from the view of user, considering their associated on-premise financial profit (i.e., $\mathbb{P}[1] = 4.9$ (usd/hour) and $\mathbb{P}[2] = 1.9$ (usd/hour)).

In the light of above discussion, now we can elaborate the problem definition more clearly. Given a set $A = \{a_1, \dots, a_n\}$ of applications and a set $V = \{v_1, \dots, v_m\}$ of VMs, since the HUP of each VM $v_j \in V$ can fall in any of $d$ HUIs in $I = [1, \dots, d]$, it may not be possible to measure a precise hourly financial profit for each application $a_i \in A$ when it is assigned to $v_j$ (i.e., $F(a_i) = v_j$). Therefore, a solution to this problem is the estimation of average hourly financial profit for each application $a_i \in A$, considering risk associated with the falling of the HUP of VM $v_j \in V$ in each HUI $I[k]$ with probability $Pr_j[k]$. In the next section, we will present our risk-aware application scheduling approach aimed at, by selecting a VM $v_j \in V$ for each application $a_i \in A$, maximizing hourly financial profit that is estimated for each application, considering the importance of applications.

## 6.3 PROPOSED APPROACH

Our approach, presented in the following sections, operates in two main steps *i) measure an hourly penalty* for each VM $v_j \in V$ w.r.t. each HUI $I[k] \in I$; *ii) estimate hourly financial profit* for each application to be scheduled w.r.t. each VM $v_j \in V$.

### 6.3.1 MEASURING PENALTY FOR EACH VM

The first step of our proposed approach is measuring an hourly penalty for each VM $v_j \in V$, w.r.t. its HUP, which is paid by Cloud providers. In fact, since the HUP of each $v_j \in V$ can fall in any HUI $I_k \in I$, we should calculate the hourly penalty of $v_j \in V$, considering each $I[k]$. To do so, each VM $v_j \in V$ is associated with a *penalty vector* $\xi_j[1, \dots, d]$, where $\xi_i[k]$ is the hourly penalty of $v_j \in V$ if its HUP falls in HUI $I[k] \in I$ and calculated as:

$$\xi_i[k] = R[j] \cdot \Gamma_k \tag{6.1}$$

For example, as depicted in Table 6.5, the penalty of $\xi_2[3]$ of VM $v_2 \in V$, if its HUP falls in HUI $I[3]$ is calculated as $R[2].\Gamma[3] = 0.007 \cdot 50\% = 0.0035$ (usd/hour). In the next subsection, we will present our approach for estimating the hourly financial profit of each application $a_i \in A$.

### 6.3.2 ESTIMATING FINANCIAL PROFIT FOR AN APPLICATION

The second step of our proposed approach is estimating an average hourly financial profit for each application $a_i \in A$ if it is mapped on each VM $v_j \in V$. As we discussed in Section 6.2.2, in our scenario, the importance of each application $a_i \in A$ is defined according to its hourly on-premise financial profit $\mathbb{P}[i]$. Also, as we discussed before in Section 6.2.2, the second objective of mapping function $F : A \to V$ is to guarantee the satisfaction of the importance of applications in $A$. To meet this objective, we schedule applications in $A$ in the decreasing order of their importance. Back to our running example, the first application to be scheduled is application $a_1 \in A$ with the highest importance among others in $A$. Then, we schedule applications $a_4 \in A$ and $a_3 \in A$, and finally $a_2 \in A$. For simplicity, in the following, we refer our discussion for scheduling the first application to be scheduled (i.e., $a_1 \in A$) with the note that the process described is executed for all applications in $A$.

#### 6.3.2.1 ESTIMATING HUI-WISE FINANCIAL PROFIT FOR AN APPLICATION

As we mentioned in Section 6.2.2.1, we assume that the HUP of each VM $v_j \in V$ falls in a HUI $I[k] \in I$. Also, as we discussed before in Section 6.2.2, risk associated with the falling of the HUP of selected VM $v_j \in V$ for application $a_i \in A$ (i.e., $F(a_i) = v_j$) in each HUI $I_k \in I$ can affect: *1) hourly penalty* for a VM $v_j \in V$ and *2) off-premise financial profit* for application $a_i \in A$. Therefore, to efficiently estimate the financial profit of applications in $A$, we need to consider all possible outcomes of falling the HUP of VMs in $V$ in HUIs in $I$, each with a different probability maintained in $Pr_j$. In other words, to select a VM for application $a_i \in A$, we need to estimate, for each HUI $I[k] \in I$, a financial profit if $a_i \in A$ is assigned to a VM $v_j \in V$ (i.e., $F(a_i) = v_j$). For example, considering Table 6.3, the HUP of each VM $v_j \in V$ can fall in four HUIs in $I$ ($d = 4$), which as a result, we need to estimate four values for the financial benefit of $a_i \in A$, considering each VM $v_j \in V$.

Let $\overline{\mathcal{P}}_{i,j}[k]$ denotes the estimated hourly financial profit of current application $a_i \in A$ to be scheduled when it is mapped on a VM $v_j \in V$ (i.e., $F(a_i) = v_j$) and the HUP of $v_j \in V$, with rental cost $R[j]$, falls in $I[k] \in I$ with penalty $\xi_j[k]$. Then, we propose Formula 6.2 for $\overline{\mathcal{P}}_{i,j}[k]$.

$$\overline{\mathcal{P}}_{i,j}[k] = \mathcal{P}_i[k] + \xi_j[k] - R[j] \tag{6.2}$$

Such estimated values for the HUI-wise financial profit of application $a_i \in A$ are maintained in an *HUI-wise profit vector* $\overline{\mathcal{P}}_{i,j}[1, \ldots, d]$. For example, as depicted in Table 6.5, the estimated hourly financial profit $\overline{\mathcal{P}}_{a_1,v_2}[3]$ of application $a_1 \in A$ when it is mapped on VM $v_2 \in V$ (i.e., $F(a_1) = v_2$) and the HUP of $v_2 \in V$ falls in HUI $I[3]$ is equal to 2.096 (usd/hour). In the next sub-section, through a risk analysis process, we choose a VM $v_j \in V$ for the current application $a_i \in A$ to be scheduled which maximizes the estimated financial profit of application $a_i \in A$.

#### 6.3.2.2 MEASURING EXPECTED MONETARY VALUE FOR AN APPLICATION

To select a VM for the current application $a_i \in A$ to be scheduled, we estimate, for each VM $v_j \in V$, the hourly financial profit $\overline{\mathcal{P}}^{emv}_{i,j}$ of $a_i \in A$, as its expected monetary value (EMV), if it is mapped on $v_j \in V$ as:

| | $\mathbf{R[v_1]}$ | $\boldsymbol{I}$ | $\boldsymbol{\Gamma}$ | $\boldsymbol{Pr_1}$ | $\boldsymbol{\xi_1}$ | $\overline{\mathcal{P}}_{a_1,v_1}$ | $\overline{\mathcal{P}}^{emv}_{a_1,v_1}$ |
|---|---|---|---|---|---|---|---|
| $v_1$ | 0.010 | [99.96 100] | 0 | 0.997 | 0 | 4.790 | 4.784 |
| | | [99 99.95] | 30 | 0.002 | 0.003 | 3.593 | |
| | | [95 98.99] | 50 | 0.0009 | 0.005 | 2.095 | |
| | | [90 94.99] | 60 | 0.0001 | 0.006 | 1.196 | |

| | $\mathbf{R[v_2]}$ | $\boldsymbol{I}$ | $\boldsymbol{\Gamma}$ | $\boldsymbol{Pr_2}$ | $\boldsymbol{\xi_2}$ | $\overline{\mathcal{P}}_{a_1,v_2}$ | $\overline{\mathcal{P}}^{emv}_{a_1,v_2}$ |
|---|---|---|---|---|---|---|---|
| $v_2$ | 0.007 | [99.96 100] | 0 | 0.96 | 0 | 4.793 | 4.744 |
| | | [99 99.95] | 30 | 0.0395 | 0.0021 | 3.595 | |
| | | [95 98.99] | 50 | 0.0004 | 0.0035 | 2.096 | |
| | | [90 94.99] | 60 | 0.0001 | 0.0042 | 1.197 | |

| | $\mathbf{R[v_3]}$ | $\boldsymbol{I}$ | $\boldsymbol{\Gamma}$ | $\boldsymbol{Pr_3}$ | $\boldsymbol{\xi_3}$ | $\overline{\mathcal{P}}_{a_1,v_3}$ | $\overline{\mathcal{P}}^{emv}_{a_1,v_3}$ |
|---|---|---|---|---|---|---|---|
| $v_3$ | 0.004 | [99.96 100] | 0 | 0.86 | 0 | 4.796 | 4.449 |
| | | [99 99.95] | 30 | 0.03 | 0.0012 | 3.597 | |
| | | [95 98.99] | 50 | 0.095 | 0.002 | 2.098 | |
| | | [90 94.99] | 60 | 0.015 | 0.0024 | 1.198 | |

| | $\mathbf{R[v_4]}$ | $\boldsymbol{I}$ | $\boldsymbol{\Gamma}$ | $\boldsymbol{Pr_4}$ | $\boldsymbol{\xi_4}$ | $\overline{\mathcal{P}}_{a_1,v_4}$ | $\overline{\mathcal{P}}^{emv}_{a_1,v_4}$ |
|---|---|---|---|---|---|---|---|
| $v_4$ | 0.002 | [99.6 100] | 0 | 0.8 | 0 | 4.798 | 4.403 |
| | | [99 99.95] | 30 | 0.1 | 0.0006 | 3.598 | |
| | | [95 98.99] | 50 | 0.095 | 0.001 | 2.099 | |
| | | [90 94.99] | 60 | 0.005 | 0.0012 | 1.199 | |

**Table 6.5:** Rental cost vector $R$, HUI vector $I$, service credit vector $\Gamma$, probability vector $Pr_1, \dots, Pr_4$, penalty vectors $\xi_1, \dots, \xi_4$, HUI-wise profit vector $\overline{\mathcal{P}}_{a_1,v_j}$, and estimated financial profit $\overline{\mathcal{P}}^{emv}_{a_1,v_j}$ of application $a_1 \in A$ w.r.t. each VM $v_j \in V$

$$\overline{\mathcal{P}}^{emv}_{i,j} = \sum_{k=1}^{d} \overline{\mathcal{P}}_{i,j}[k] \cdot Pr_j[k] \tag{6.3}$$

We note that $\overline{\mathcal{P}}_{i,j}[k]$ denotes the estimated HUI-wise financial profit of application $a_i \in A$ (see Section 6.3.2.1) and $Pr_j[k]$ denotes the probability that the HUP of VM $v_j \in V$ falls in HUI $I_k \in I$. Back to our running example, Table 6.5 shows the estimated hourly financial profit of application $a_1 \in A$ w.r.t. each VM $v_j \in V$. To meet the first objective of mapping function $F$ (see Section 6.2.2), we need to select a VM $v_j \in V$ for the current application $a_i \in A$ to be scheduled that, compared to other VMs in $V$, application $a_i \in A$ would make the maximum hourly financial profit, if it is mapped on $v_j \in V$. Back to our running example, considering Table 6.5, VM $v_1 \in V$ is selected for the current application $a_1 \in A$ (i.e., $F(a_1) = v_1$) because $a_1 \in A$ would make the maximum hourly financial profit, if it is mapped on $v_1 \in V$ ($\overline{\mathcal{P}}^{emv}_{a_1,v_1} = 4.784$ (usd/hour)), compared to other other VMs in $V$.

As we discussed before in Section 6.2.2, in our scenario, the function $F : A \to V$ is a one-to-one mapping function. That is, each application $a_i \in A$ is mapped on

| | $\mathbf{R[v_2]}$ | $I$ | $\Gamma$ | $Pr_2$ | $\xi_2$ | $\overline{\mathcal{P}}_{a_4,v_2}$ | $\overline{\mathcal{P}}^{emv}_{a_4,v_2}$ |
|---|---|---|---|---|---|---|---|
| $v_2$ | 0.007 | [99.96 100] | 0 | 0.96 | 0 | 4.593 | 4.556 |
| | | [99 99.95] | 30 | 0.0395 | 0.0021 | 3.695 | |
| | | [95 98.99] | 50 | 0.0004 | 0.0035 | 2.896 | |
| | | [90 94.99] | 60 | 0.0001 | 0.0042 | 1.897 | |
| | $\mathbf{R[v_3]}$ | $I$ | $\Gamma$ | $Pr_3$ | $\xi_3$ | $\overline{\mathcal{P}}_{a_4,v_3}$ | $\overline{\mathcal{P}}^{emv}_{a_4,v_3}$ |
| $v_3$ | 0.004 | [99.96 100] | 0 | 0.86 | 0 | 4.596 | 4.367 |
| | | [99 99.95] | 30 | 0.03 | 0.0012 | 3.697 | |
| | | [95 98.99] | 50 | 0.095 | 0.002 | 2.898 | |
| | | [90 94.99] | 60 | 0.015 | 0.0024 | 1.898 | |
| | $\mathbf{R[v_4]}$ | $I$ | $\Gamma$ | $Pr_4$ | $\xi_4$ | $\overline{\mathcal{P}}_{a_4,v_4}$ | $\overline{\mathcal{P}}^{emv}_{a_4,v_4}$ |
| $v_4$ | 0.002 | [99.6 100] | 0 | 0.8 | 0 | 4.598 | 4.333 |
| | | [99 99.95] | 30 | 0.1 | 0.0006 | 3.698 | |
| | | [95 98.99] | 50 | 0.095 | 0.001 | 2.899 | |
| | | [90 94.99] | 60 | 0.005 | 0.0012 | 1.899 | |

**Table 6.6:** Rental costs $R[v_2]$, $R[v_3]$, $R[v_4]$, HUI vector $\underline{I}$, service credit vector $\Gamma$, probability vectors $Pr_2, \ldots, Pr_4$, HUI-wise profit vectors $\overline{\mathcal{P}}_{a_4,v_j}$, and estimated financial profit $\overline{\mathcal{P}}^{emv}_{a_4,v_j}$ of application $a_4$ w.r.t. each VM $v_j \in V$

one VM $v_j \in V$ and each $v_j$ is selected for one $a_i \in A$. Therefore, to avoid mapping an application to an already selected VM, we remove VM $v_j \in V$ from the set $V$ of available VMs when it is selected for application $a_i \in A$ (i.e., $F(a_i) = v_j$). Back to our running example, since application $a_i \in A$ is mapped on VM $v_1 \in V$ (i.e., $F(a_1) = v_1$), then $v_1$ is removed from the set $V$ of available VMs (i.e., $V = \{v_2, v_3, v_4\}$).

According to our discussion in Section 6.3.2 and considering Table 6.4, the next application to be scheduled is $a_4 \in A$, then $a_3 \in A$, and finally, $a_2 \in A$. As depicted in Table 6.6, application $a_4 \in A$ is mapped on VM $v_2 \in V$ (i.e., $F(a_4) = v_2$) because hourly financial profit $\overline{\mathcal{P}}^{emv}_{a_4,v_2} = 4.556$ (usd/hour) estimated for $a_4 \in A$ is maximum, if it is mapped on $v_2 \in V$, compared to VMs $v_3$ ($\overline{\mathcal{P}}^{emv}_{a_4,v_3} = 4.367$ (usd/hour)) and $v_4 \in V$ ($\overline{\mathcal{P}}^{emv}_{a_4,v_4} = 4.333$ (usd/hour)). Also, VM $v_2 \in V$ is removed from the set $V$ of available VMs (i.e., $V = \{v_3, v_4\}$) to avoid its selection for unscheduled application(s) (i.e., $a_2, a_3 \in A$).

The next application to be scheduled is $a_3 \in A$, which according to Table 6.7, is mapped on VM $v_3 \in V$ (i.e., $F(a_3) = v_3$), because financial profit $\overline{\mathcal{P}}^{emv}_{a_3,v_3} = 4.053$ (usd/hour) that is estimated for $a_3 \in A$ is maximum, if it is mapped on $v_3 \in V$, compared to VM $v_4 \in V$ ($\overline{\mathcal{P}}^{emv}_{a_3,v_4} = 4.013$ (usd/hour)). Also, VM $v_3 \in V$ is removed from the set $V$ of available VMs (i.e., $V = \{v_4\}$) to avoid its selection for unscheduled applications (i.e., $a_2 \in A$). Finally, the last application to be scheduled is $a_2 \in A$ which is mapped on VM $v_4 \in V$ as the only available VM in the set $V$ of available VMs and, as a result, $V = \{\}$.

| | $\mathbf{R[v_3]}$ | $I$ | $\Gamma$ | $Pr_3$ | $\xi_3$ | $\overline{\mathcal{P}}_{a_3,v_3}$ | $\overline{\mathcal{P}}^{emv}_{a_3,v_3}$ |
|---|---|---|---|---|---|---|---|
| $v_3$ | 0.004 | [99.96 100] | 0 | 0.86 | 0 | 4.296 | 4.053 |
| | | [99 99.95] | 30 | 0.03 | 0.0012 | 3.297 | |
| | | [95 98.99] | 50 | 0.095 | 0.002 | 2.498 | |
| | | [90 94.99] | 60 | 0.015 | 0.0024 | 1.498 | |
| | $\mathbf{R[v_4]}$ | $I$ | $\Gamma$ | $Pr_4$ | $\xi_4$ | $\overline{\mathcal{P}}_{a_3,v_4}$ | $\overline{\mathcal{P}}^{emv}_{a_3,v_4}$ |
| $v_4$ | 0.002 | [99.6 100] | 0 | 0.8 | 0 | 4.298 | 4.013 |
| | | [99 99.95] | 30 | 0.1 | 0.0006 | 3.298 | |
| | | [95 98.99] | 50 | 0.095 | 0.001 | 2.499 | |
| | | [90 94.99] | 60 | 0.005 | 0.0012 | 1.499 | |

**Table 6.7:** Rental costs $R[v_3], R[v_4]$, HUI vector I, service credit vector $\Gamma$, probability vectors $Pr_3, Pr_4$, penalty vectors $\xi_3, \xi_4$, HUI-wise profit vectors $\overline{\mathcal{P}}_{a_3,v_j}$, and estimated financial profit $\overline{\mathcal{P}}^{emv}_{a_3,v_j}$ of application $a_3$ w.r.t. each VM $v_j \in V$

Figure 6.4 shows a decision tree which graphically presents the risk (falling the HUP of VM $v_j \in V$ in each HUI $I_k \in I$, if $F(a_1) = v_j$) of mapping $a_1 \in A$ on each available VM in V and the associated consequence (changes in estimated HUI-wise financial profit $\overline{\mathcal{P}}_{i,j}[k]$).

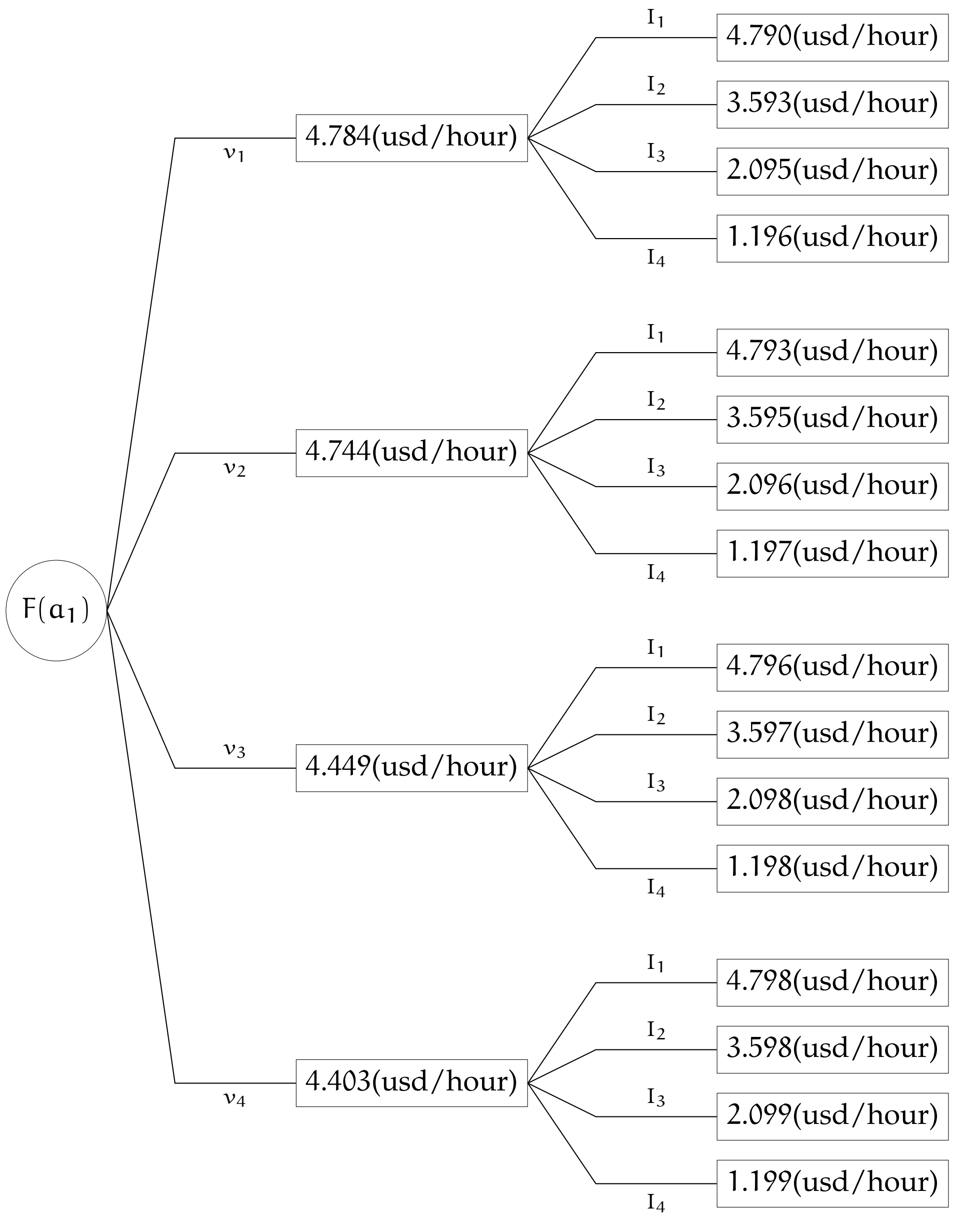

**Figure 6.4:** Decision tree for selecting a VM $v_j \in V$ for application $a_1 \in A$

## 6.4 ALGORITHM FOR THE PROPOSED RISK-AWARE APPLICATION SCHEDULING APPROACH

In this section, given our application scheduling problem, we provide a pseudo-code algorithm for the proposed solution which is reported in Figure 6.5. Our algorithm takes as input the set of applications $A$, the set of VMs $V$, rental cost vector $R$, probability vectors $Pr_1, \dots, Pr_m$, uptime interval $I$, service credit vector $\Gamma$, on-premise profit vector $\mathbb{P}$, and off-premise profit vectors $\mathcal{P}_1, \dots, \mathcal{P}_n$ and returns mapping function $F : A \to V$.

The algorithm, first, for each VM $v_j \in V$, calculates an hourly penalty $\xi_j[k]$, if the HUP of $v_j \in V$ falls in HUI $I_k \in I$ (lines 1–4). Next, applications in $A$ are sorted in decreasing order of their on-premise financial profit, maintained in $\mathbb{P}$ (lines 5–7). Then, for the current application $a_i \in A$ to be scheduled, an HUI-wise financial profit $\overline{\mathcal{P}}_{i,j}[k]$ is estimated (lines 8–13), considering each HUI $I[k] \in I$ defined in the service credit schedule of each VM $v_j \in V$. Next, an hourly financial profit $\overline{\mathcal{P}}_{i,j}^{emv}$ for the current application $a_i \in A$ to be scheduled is estimated as the EMV of $a_i \in A$, if it is mapped on each VM $v_j \in V$ (lines 14–16). Then, a VM $v_j \in V$ is selected for application $a_i \in A$ that, compared to other VMs in $V$, $a_i \in A$ would make the maximum hourly financial profit $\overline{\mathcal{P}}_{i,j}^{emv}$ if it is mapped on $v_j \in V$ (lines 17–21). Finally, VM $v_j \in V$ that is selected for the current application $a_i \in A$ to be scheduled is removed from the set $V$ of available VMs to avoid selecting $v_j$ for other unscheduled applications in $A \backslash \{a_i\}$ (line 22).

```
INPUT
$A = \{a_1, \ldots, a_n\}$ /* set of applications */
$V = \{v_1, \ldots, v_m\}$ /* set of VMs */
$R[1, \ldots, m]$ /* vector of hourly rental costs for VMs $v_1, \ldots, v_m$*/
$I[1, \ldots, d]$ /* vector of uptime interval */
$Pr_1, \ldots, Pr_m$ /* vectors of probability for VMs $v_1, \ldots, v_m$ */
$\Gamma$ /* vector of service credit */
$\mathbb{P}$ /* vector of on-premise profit */
$\mathcal{P}_1, \ldots, \mathcal{P}_n$ /* off-premise profit vectors for applications $a_1, \ldots, a_n$ */
OUTPUT
$F : A \to V$ /* mapping function */
MAIN
/* Step 1: Calculate hourly penalty for VMs */
1:  for each $v_j \in V$ do
2:    let $\xi_j$ be the penalty vector of size $|I|$
3:    for each $k = 1, \ldots, |I|$ do
4:      $\xi_j[k] := R[j] \cdot \Gamma[k]$ /* penalty for $v_j$ w.r.t. $I[k]$ */
5:  let S be a list of size $|A|$ to contain sorted applications
6:  for each $i = 1, \ldots, |A|$ do
7:    insert $a_i \in A$ in S in decreasing order of $\mathbb{P}$
    /* Step 2: estimate a financial profit for the current application to be scheduled */
8:  for each $i = 1, \ldots, |A|$ do
9:    let $a_x$ be the application in the position of $i$ in S
10:   let $\overline{\mathcal{P}}_{x,j}$ be the HUI-wise profit vector of size $|I|$
11:   for each $j = 1, \ldots, |V|$ do
12:     for each $k = 1, \ldots, |I|$ do
13:       $\overline{\mathcal{P}}_{x,j}[k] := \mathcal{P}_x[k] + \xi_j[k] - R[j]$ /* HUI-wise estimated profit for $a_x$, if $F(a_x) = v_j$ w.r.t. $I[k]$ */
14:   for each $j = 1, \ldots, |V|$ do
15:     for each $k = 1, \ldots, |I|$ do
16:       $\overline{\mathcal{P}}^{emv}_{x,j} := \overline{\mathcal{P}}_{x,j}[k] \cdot Pr_j[k]$ /* estimated profit for $a_x$ if $F(a_x) = v_j$ */
17:   let $O_x$ be a list of size $|V|$ to contain estimated profit $\overline{\mathcal{P}}^{emv}_{x,j}$ of $a_x$ w.r.t. $v_j \in V$
18:   for each $j = 1, \ldots, |V|$ do
19:     insert each VM $v_j \in V$ in O in decreasing order of $\overline{\mathcal{P}}^{emv}_{x,j}$
20:   let VM $v_* \in V$ be the VM in the first position of O
21:   $F(a_x) := v_*$ /* map $a_x$ to $v_*$ */
22:   $V := V \backslash \{v_*\}$ /* remove $v_*$ from V */
```

**Figure 6.5:** Algorithm for the proposed risk-aware application scheduling approach

## 6.5 CHAPTER SUMMARY

In this chapter, we provided a solution for scheduling a set of applications to support the business objectives of users w.r.t. the financial profit of applications. To do so, through a risk analysis process, we map each application to an available VM, according to the expected monetary value of application when it is mapped to each available VM. In this manner, our solution maximizes the estimated financial profit of each application when it is mapped on an available VM, according to its importance.

# 7

# CONCLUSIONS AND FUTURE WORKS

In this thesis, we provided models and tools to support users in evaluating applications that are moved to the Cloud and in selecting the most suitable Cloud plans, considering their characteristics and the requirements of applications. After a brief introduction and reviewing some related works, we focused mainly on four specific aspects: *1)* evaluating the modularity of applications, *2)* Consensus-based selection of Cloud plans, *3)* business-oriented Cloud plan selection, and *4)* Cloud plan selection under uncertainty. In this chapter, we shortly review the original contributions of this thesis and present some future work.

## 7.1 SUMMARY OF THE CONTRIBUTIONS

The main contributions of this thesis can be summarized as follows.

**Application assessment in outsourcing scenarios.** We presented a software-engineering-based approach for evaluating the modularity of applications as a metric for their change flexibility and/or distributability. The proposed approach could be used as a tool to see to what extent applications can be easily moved to the Cloud, considering the dynamic and/or distributed properties of Cloud environments.

**Consensus-based Cloud plan selection.** We proposed a solution aimed at balancing the satisfaction of applications' requirements by selecting a Cloud plan according to a consensus among them. The proposed solution provides a tool which chooses a plan that is globally considered the most acceptable by all applications.

**Cloud plan selection under uncertainty.** We proposed a method aimed at selecting a Cloud plan for multiple applications when a set of users, with a limited budget for selecting a Cloud plan, do not have precise ideas about the requirements of applications.

Our approach provides a tool which efficiently captures the uncertainty of imprecise information about applications, provided by multiple unskilled IT users, while satisfies the budget constraints of users for selecting a Cloud plan.

**Business-oriented Cloud plan selection.** We proposed a method aimed at, by assigning each application to a virtual machine (VM) offered by multiple Cloud providers, maximizing financial profit that is estimated for applications when they are executed on the Cloud, according to the importance of each application. Our approach provides a tool which suitably supports the business objectives of users w.r.t. the financial profit of applications, considering service level agreement compensation mechanisms, offered by Cloud providers.

## 7.2 FUTURE WORK

The research described in this thesis leaves several opportunities for future work which can be summarized as follows.

**Application assessment in outsourcing scenarios.** Our approach for estimating the modularity of applications properly supports users to see to what extent applications are ready to be moved to the Cloud w.r.t. adaptability and/or distributability. Considering this contribution, we plan to focus on providing an assessment framework to evaluate the suitability of applications for running on the Cloud. To do so, w.r.t. different criteria (e.g., maintainability, reliability) that are considered necessary for applications, their suitability for moving to the Cloud will be evaluated.

**Consensus-based Cloud plan selection.** In our approach for selecting a Cloud plan based on the consensus between applications on the satisfaction of their requirements, we did not consider a consensus degree between applications on the level of the satisfaction of their requirements. Also, we did not consider the satisfaction of the objectives of Cloud provider(s) (e.g., maximizing the profit of Cloud provider by renting Cloud resources) as well as applications' requirements. Therefore, an interesting future line of research could be providing a solution aimed at reaching a partial consensus between applications as well as fulfilling the objectives of Cloud providers.

**Cloud plan selection under uncertainty.** In our solution for supporting Cloud plan selection in the presence of multiple users, possibly without an IT background, we considered Cloud plans, with fixed prices and predefined ratings for each criterion which is almost consistent with the current trends of plan selection in the Cloud market. Also, to select a Cloud plan, we assume that users have the same opinions about the preferences of applications, over considered criteria, which is often the case in typical scenarios. An interesting line of research for future studies would be deciding on a customized plan, considering imprecise information which is provided by a set of unskilled IT users, each with possibly different opinions about the requirements of applications.

**Business-oriented Cloud plan selection.** Our approach for supporting the business objectives of users w.r.t. the financial profit of applications is risk neutral as we did not consider uncertainty degree in the proposed scenario. That is, we considered an equal number of uptime intervals, each with equal service credits for all VMs. Therefore, the proposed approach is indifferent between VMs with equal estimated financial profit, and as a result, a VM with a higher degree of uncertainty in the associated service credit schedule (e.g., a VM with a higher number of uptime intervals, each with a possibly different service credit) could be selected for an application. Then, the potential consequence could be selecting a VM for an application which would not maximize the financial profit of application. Therefore, scheduling applications by selecting a VM, among those available ones, for each application when Cloud providers consider service credit schedules with different degrees of uncertainty could be an alternative to investigate for future research.

# BIBLIOGRAPHY

[1] M. Sagar, S. Bora, A. Gangwal, P. Gupta, A. Kumar, and A. Agarwal, "Factors affecting customer loyalty in cloud computing: A customer defection-centric view to develop a void-in-customer loyalty amplification model," *Global Journal of Flexible Systems Management*, vol. 14, no. 3, pp. 143–156, 2013.

[2] A.-R. Sadeghi, T. Schneider, and M. Winandy, "Token-based cloud computing," in *Proc. of the 3rd International Conference on Trust and Trustworthy Computing (TRUST 2010)*, Berlin, Germany, June 2010, pp. 417–429.

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[4] G. Irazoqui, M. S. Inci, T. Eisenbarth, and B. Sunar, "Wait a minute! a fast, cross-VM attack on AES," in *Proc. of the 17th International Workshop on Recent Advances in Intrusion Detection (RAID 2014)*, Gothenburg, Sweden, September 2014, pp. 299–319.

[5] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the weather tomorrow?: Towards a benchmark for the cloud," in *Proc. of the 2nd International Workshop on Testing Database Systems (DBTest '09)*, New York, NY, USA, June 2009, pp. 9:1–9:6.

[6] G. Rosen, "The business of clouds," *Crossroads*, vol. 16, no. 3, pp. 26–28, 2010.

[7] A. Arman, A. Al-Shishtawy, and V. Vlassov, "Elasticity controller for cloud-based key-value stores," in *Proc. of the 18th International Conference on Parallel and Distributed Systems (ICPADS)*, Singapore, December 2012, pp. 268–275.

[8] Y. Liu and R. K. Tyagi, "Outsourcing to convert fixed costs into variable costs: A competitive analysis," *International Journal of Research in Marketing*, vol. 34, no. 1, pp. 252–264, 2017.

[9] J. S. Rellermeyer and S. Bagchi, "Dependability as a cloud service - a modular approach," in *Proc. of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*, Boston, MA, USA, June 2012, pp. 1–6.

[10] R. S. Pressman, *Software Engineering: A Practitioner's Approach.* McGraw-Hill Higher Education, 2001.

[11] M. Jabalameli, A. Arman, and M. Nematbakhsh, "Improving the efficiency of term weighting in set of dynamic documents," *International Journal of Modern Education and Computer Science*, vol. 7, no. 2, p. 42, 2015.

[12] K. Alhamazani, R. Ranjan, P. P. Jayaraman, K. Mitra, F. Rabhi, D. Georgakopoulos, and L. Wang, "Cross-layer multi-cloud real-time application QoS monitoring and benchmarking as-a-service framework," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[13] M. Rönkkö, C. Frühwirth, and S. Biffl, "Integrating value and utility concepts into a value decomposition model for value-based software engineering," in *Proc. of the International Conference on Product-Focused Software Process Improvement (PROFES 2009)*, Oulu, Finland, June 2009, pp. 362–374.

[14] T. N. Al-Otaiby, M. AlSherif, and W. P. Bond, "Toward software requirements modularization using hierarchical clustering techniques," in *Proc. of the 43rd Annual Southeast Regional Conference - Volume 2 (ACM-SE 43)*, Kennesaw, Georgia, March 2005, pp. 223–228.

[15] S. Nelson and J. Schumann, "What makes a code review trustworthy?" in *Proc. of the 37th Annual Hawaii International Conference on System Sciences*, Big Island, HI, USA, January 2004, pp. 10–pp.

[16] B. Meyer, *Object-oriented software construction*. Prentice hall New York, 1988.

[17] M. Alenezi and M. Zarour, "Modularity measurement and evolution in object-oriented open-source projects," in *Proc. of the International Conference on Engineering & MIS (ICEMIS '15)*, Istanbul, Turkey, September 2015, pp. 16:1–16:7.

[18] R. W. Schwanke, "An intelligent tool for re-engineering software modularity," in *Proc. of the 13th International Conference on Software Engineering*, Austin, TX, USA, May 1991, pp. 83–92.

[19] C. Sant'Anna, E. Figueiredo, A. Garcia, and C. Lucena, "On the modularity assessment of software architectures: Do my architectural concerns count?" in *Proc. the International Workshop on Aspects in Architecture Descriptions (AARCH. 07)*, Vancouver, Canada, March 2007, pp. 183–192.

[20] H. Deng and C. Mercado, "A method for metric-based architecture level quality evaluation," Master's thesis, Blekinge Inistitue of Technology, 2008.

[21] P. Johansson and H. Holmberg, "On the modularity of a system," Master's thesis, Malmö högskola/Centrum för teknikstudier, 2010.

[22] P. Meirelles, C. S. Jr., J. Miranda, F. Kon, A. Terceiro, and C. Chavez, "A study of the relationships between source code metrics and attractiveness in free software projects," in *Proc. of the Brazilian Symposium on Software Engineering*, Salvador, Brazil, September–October 2010, pp. 11–20.

[23] S. R. Chidamber and C. F. Kemerer, *Towards a metrics suite for object oriented design*. ACM, 1991.

[24] ——, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[25] C. Sant'Anna, E. Figueiredo, A. Garcia, and C. J. P. Lucena, "On the modularity of software architectures: A concern-driven measurement framework," in *Proc. of the 1st European Conference on Software Architecture (ECSA 2007)*, Madrid, Spain, Septmeber 2007, pp. 207–224.

[26] M.-H. Tang, M.-H. Kao, and M.-H. Chen, "An empirical study on object-oriented metrics," in *Proc. of the 6th International Software Metrics Symposium (Cat. No.PR00403)*, Boca Raton, FL, USA, November 1999, pp. 242–249.

[27] S. Yeresime, J. Pati, and S. K. Rath, "Review of software quality metrics for object-oriented methodology," in *Proc. of the 6th International Conference on Internet Computing and Information Communications (ICICIC Global)*, Milan, Italy, September 2014, pp. 267–278.

[28] M. Hitz and B. Montazeri, "Measuring coupling and cohesion in object-oriented systems," in *Proc. of the International Symposium on Applied Corporate Computing (ISACC'95)*, Monterrey,Mexico, October 1995, pp. 25–27.

[29] T. J. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, 1976.

[30] F. B. e Abreu and M. Goulao, "Coupling and cohesion as modularization drivers: are we being over-persuaded?" in *Proc. of the 5th European Conference on Software Maintenance and Reengineering*, Lisbon, Portugal, March 2001, pp. 47–57.

[31] M. Hitz and B. Montazeri, "Chidamber and kemerer's metrics suite: a measurement theory perspective," *IEEE Transactions on Software Engineering*, vol. 22, no. 4, pp. 267–271, 1996.

[32] A. S. Nuñez-Varela, H. G. Pérez-Gonzalez, F. E. Martínez-Perez, and C. Soubervielle-Montalvo, "Source code metrics: A systematic mapping study," *Journal of Systems and Software*, vol. 128, no. C, pp. 164–197, 2017.

[33] A. Nicklas and K. Jimmy, "Att utforma och utvärdera ett komponentbaserat programmeringsgränssnitt," Master's thesis, Malmö högskola/Teknik och samhälle, 2013.

[34] J.-F. Zhao and J.-T. Zhou, "Strategies and methods for cloud migration," *International Journal of Automation and Computing*, vol. 11, no. 2, pp. 143–152, 2014.

[35] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

[36] P. Samarati and S. D. C. di Vimercati, *Cloud security: Issues and concerns*. Wiley, New York, 2016.

[37] V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch, "How to adapt applications for the cloud environment," *Computing*, vol. 95, no. 6, pp. 493–535, 2013.

[38] A. Khajeh-Hosseini, D. Greenwood, J. W. Smith, and I. Sommerville, "The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise," *Software: Practice and Experience*, vol. 42, no. 4, pp. 447–465, 2012.

[39] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, and P. Teregowda, "Decision support tools for cloud migration in the enterprise," in *Proc. of the 4th IEEE International Conference on Cloud Computing (CLOUD)*, Washington, DC, USA, July 2011, pp. 541–548.

[40] G. Garrison, S. Kim, and R. L. Wakefield, "Success factors for deploying cloud computing," *Communications of the ACM*, vol. 55, no. 9, pp. 62–68, 2012.

[41] P. V. Beserra, A. Camara, R. Ximenes, A. B. Albuquerque, and N. C. Mendonça, "Cloudstep: A step-by-step decision process to support legacy application migration to the cloud," in *Proc. of the 6th IEEE International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA 2012)*, Trento, Italy, September 2012, pp. 7–16.

[42] V. Tran, J. Keung, A. Liu, and A. Fekete, "Application migration to cloud: A taxonomy of critical factors," in *Proc. of the 2nd International Workshop on Software Engineering for Cloud Computing (SECLOUD '11)*, New York, NY, USA, May 2011, pp. 22–28.

[43] P. Saripalli and G. Pingali, "MADMAC: Multiple attribute decision methodology for adoption of clouds," in *Proc. of the 4th IEEE International Conference on Cloud Computing (CLOUD)*, Washington DC, USA, July 2011, pp. 316–323.

[44] "Migrating your existing applications to the AWS cloud," White Paper, Amazon, October 2010.

[45] W. Zhang, A. J. Berre, D. Roman, and H. A. Huru, "Migrating legacy applications to the service cloud," in *Proc. of the 14th Conference companion on Object Oriented Programming Systems Languages and Applications (OOPSLA 2009)*, Orlando, Florida, USA, October 2009, pp. 59–68.

[46] Q. H. Vu and R. Asal, "Legacy application migration to the cloud: Practicability and methodology," in *Proc. of the 8th IEEE World Congress on Services Services (SERVICES)*, Honolulu, HI, USA, June 2012, pp. 270–277.

[47] V. T. K. Tran, K. Lee, A. Fekete, A. Liu, and J. Keung, "Size estimation of cloud migration projects with cloud migration point (CMP)," in *Proc. of the International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Banff, AB, Canada, September 2011, pp. 265–274.

[48] M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 243–254, 2010.

[49] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez, "An analysis of security issues for cloud computing," *Journal of Internet Services and Applications*, vol. 4, no. 1, p. 5, 2013.

[50] S. C. Misra and A. Mondal, "Identification of a company's suitability for the adoption of cloud computing and modelling its corresponding return on investment," *Mathematical and Computer Modelling*, vol. 53, no. 3, pp. 504–521, 2011.

[51] "Planning the migration of enterprise applications to the cloud," White Paper, Cisco, August 2010.

[52] M. A. Babar and M. A. Chauhan, "A tale of migration to cloud computing for sharing experiences and observations," in *Proc. of the 2nd International Workshop on Software Engineering for Cloud Computing (SECLOUD '11)*, Waikiki, Honolulu, HI, USA, May 2011, pp. 50–56.

[53] D. Durkee, "Why cloud computing will never be free," *Queue*, vol. 8, no. 4, pp. 20:20–20:29, 2010.

[54] P. Costa, J. P. Santos, and M. M. d. Silva, "Evaluation criteria for cloud services," in *Proc. of the 6th IEEE International Conference on Cloud Computing (CLOUD)*, Santa Clara, CA, USA, June–July 2013, pp. 598–605.

[55] W. Liu, "Research on cloud computing security problem and strategy," in *Proc. of the 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Yichang, China, April 2012, pp. 1216–1219.

[56] A. Goscinski and M. Brock, "Toward dynamic and attribute based publication, discovery and selection for cloud computing," *Future Generation Computer Systems*, vol. 26, no. 7, pp. 947–970, 2010.

[57] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, 2013.

[58] Z. u. Rehman, F. K. Hussain, and O. K. Hussain, "Towards multi-criteria cloud service selection," in *Proc. of the 5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, Seoul, South Korea, 30 June–02 July 2011, pp. 44–48.

[59] E. Cavalcante, T. Batista, F. Lopes, F. C. Delicato, P. F. Pires, N. Rodriguez, A. L. de Moura, and R. Mendes, "Optimizing services selection in a cloud multiplatform scenario," in *Proc. of the IEEE Latin America Conference on Cloud Computing and Communications (LatinCloud)*, Porto Alegre, Brazil, November 2012, pp. 31–36.

[60] M. Sun, T. Zang, X. Xu, and R. Wang, "Consumer-centered cloud services selection using AHP," in *Proc. of the International Conference on Service Sciences (ICSS)*, Shenzhen, China, April 2013, pp. 1–6.

[61] M. Brock and A. Goscinski, "Publishing dynamic state changes of resources through state aware WSDL," in *Proc of the IEEE International Conference on Web Services (ICWS '08)*, Beijing, China, September 2008, pp. 449–456.

[62] Y. Zhu, H. Hu, G. J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[63] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in *Proc.of the 16th ACM conference on Computer and communications security (CCS '09)*, Chicago, IL, USA, November 2009, pp. 187–198.

[64] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati, "Integrity for distributed queries," in *Proc. of the IEEE Conference on Communications and Network Security (CNS)*, San Francisco, CA, USA, October 2014, pp. 364–372.

[65] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Efficient integrity checks for join queries in the cloud," *JCS*, vol. 24, no. 3, pp. 347–378, 2016.

[66] E. Stefanov and E. Shi, "Multi-cloud oblivious storage," in *Proc. of the ACM SIGSAC conference on Computer & communications security (CCS'13)*, Berlin, Germany, November 2013, pp. 247–258.

[67] S. D. C. di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, and P. Samarati, "Three-server swapping for access confidentiality," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[68] R. Jhawar, V. Piuri, and P. Samarati, "Supporting security requirements for resource management in cloud computing," in *Proc. of the 15th IEEE International Conference on Computational Science and Engineering (CSE)*, Paphos, Cyprus, December 2012, pp. 170–177.

[69] Z. Wang and X. Xu, "A sharing-oriented service selection and scheduling approach for the optimization of resource utilization," *Service Oriented Computing and Applications*, vol. 6, no. 1, pp. 15–32, 2012.

[70] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[71] A. V. Dastjerdi and R. Buyya, "Compatibility-aware cloud service composition under fuzzy preferences of users," *IEEE TCC*, vol. 2, no. 1, pp. 1–13, 2014.

[72] L. Zhang, F. Ding, Y.-d. Fang, and J.-y. Wu, *Service Scheduling Optimization in the Next Generation Networked Manufacturing Systems*. Springer Berlin Heidelberg, 2013, pp. 251–261.

[73] C. Qu and R. Buyya, "A cloud trust evaluation system using hierarchical fuzzy inference system for service selection," in *Proc. of the 28th IEEE International Conference on Advanced Information Networking and Applications (AINA 2014)*, Victoria, BC, Canada, May 2014, pp. 850–857.

[74] L. Sun, J. Ma, Y. Zhang, H. Dong, and F. K. Hussain, "Cloud-FuSeR: Fuzzy ontology and MCDM based cloud service selection," *Future Generation Computer Systems*, vol. 57, no. C, pp. 42–55, 2016.

[75] S. Liu, F. T. Chan, and W. Ran, "Decision making for the selection of cloud vendor: An improved approach under group decision-making with integrated weights and objective/subjective attributes," *Expert Systems with Applications*, vol. 55, no. C, pp. 37–47, 2016.

[76] L. Zhang, Y. d. Fang, and J. y. Wu, "Multi granularity resource encapsulation for p2p semantic manufacturing grid (ie&em)," in *Proc. of the 18th IEEE International Conference on Industrial Engineering and Engineering Management (IE&EM 2011)*, Changchun, China, September 2011, pp. 458–462.

[77] Z. Wu and Y. Chen, "The maximizing deviation method for group multiple attribute decision making under linguistic environment," *Fuzzy Sets and Systems*, vol. 158, no. 14, pp. 1608–1617, 2007.

[78] Z. Yue, "A method for group decision-making based on determining weights of decision makers using TOPSIS," *Applied Mathematical Modelling*, vol. 35, no. 4, pp. 1926–1936, 2011.

[79] S.-I. Chang, D. C. Yen, C. S.-P. Ng, and W.-T. Chang, "An analysis of IT/IS outsourcing provider selection for small- and medium-sized enterprises in Taiwan," *Information & Management*, vol. 49, no. 5, pp. 199–209, 2012.

[80] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya, "SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in *Proc. of the 11th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I (ICA3PP 2011)*, Melbourne, Australia, October 2011, pp. 371–384.

[81] Y. Choi and Y. Lim, "Resource management mechanism for SLA provisioning on cloud computing for IoT," in *Proc. of the International Conference on Information and Communication Technology Convergence (ICTC 2015)*, Jeju, South Korea, October 2015, pp. 500–502.

[82] L. Wu, S. K. Garg, and R. Buyya, "SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments," in *Proc. of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011)*, Newport Beach, CA, USA, May 2011, pp. 195–204.

[83] D. Dib, N. Parlavantzas, and C. Morin, "SLA-based profit optimization in cloud bursting PaaS," in *Proc. of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Chicago, IL, USA, May 2014, pp. 141–150.

[84] X. Chen, H. Chen, Q. Zheng, W. Wang, and G. Liu, "Characterizing web application performance for maximizing service providers' profits in clouds," in *Proc. of the International Conference on Cloud and Service Computing (CSC)*, Hong Kong, China, December 2011, pp. 191–198.

[85] L. Gkatzikis and I. Koutsopoulos, "Mobiles on cloud nine: Efficient task migration policies for cloud computing systems," in *Proc. of the 3rd IEEE International Conference on Cloud Networking (CloudNet)*, Luxembourg, October 2014, pp. 204–210.

[86] P. D. Sanzo, D. Rughetti, B. Ciciani, and F. Quaglia, "Auto-tuning of cloud-based in-memory transactional data grids via machine learning," in *Proc. of the 2nd Symposium on Network Cloud Computing and Applications (NCCA)*, London, UK, December 2012, pp. 9–16.

[87] A. N. Zhirabok, Ü. Kotta, and A. E. Shumsky, "Accommodation to defects in the discrete dynamic systems," *Automation and Remote Control*, vol. 75, no. 6, pp. 997–1009, 2014.

[88] M. Azaiez and W. Chainbi, "A multi-agent system architecture for self-healing cloud infrastructure," in *Proc. of the International Conference on Internet of Things and Cloud Computing (ICC '16)*, New York, NY, USA, March 2016, pp. 7:1–7:6.

[89] F.-L. Lian, J. Moyne, and D. Tilbury, "Network design consideration for distributed control systems," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 2, pp. 297–307, 2002.

[90] J.-Y. Huang, "Patent portfolio analysis of the cloud computing industry," *Journal of Engineering and Technology Management*, vol. 39, no. C, pp. 45–64, 2016.

[91] W. Tian, C. S. Yeo, R. Xue, and Y. Zhong, "Power-aware scheduling of real-time virtual machines in cloud data centers considering fixed processing intervals," in *Prof. of the 2nd IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, Hangzhou, China, October–November 2012, pp. 269–273.

[92] M. Zhang, R. Ranjan, M. Menzel, S. Nepal, P. Strazdins, W. Jie, and L. Wang, "An infrastructure service recommendation system for cloud applications with real-time QoS requirement constraints," *IEEE Systems Journal*, no. 99, pp. 1–11, June 2017.

[93] H. Yuan, J. Bi, and B. Li, "Workload-aware request routing in cloud data center using software-defined networking," *Journal of Systems Engineering and Electronics*, vol. 26, no. 1, pp. 151–160, 2015.

[94] D. Trihinas, G. Pallis, and M. Dikaiakos, "Monitoring elastically adaptive multicloud services," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[95] M. Lecznar and S. Patig, "Cloud computing providers: Characteristics and recommendations," in *Proc. of the 5th International Conference on E-Technologies (MCETECH 2011)*, Les Diablerets, Switzerland, January 2011, pp. 32–45.

[96] A. Wirotyakun and P. Netisopakul, "Improving software maintenance size metrics a case study: Automated report generation system for particle monitoring in hard disk drive industry," in *Proc. of the 9th International Conference on Computer Science and Software Engineering (JCSSE)*, Bangkok, Thailand, May–June 2012, pp. 334–339.

[97] A. Mitchell and J. F. Power, "Using object-level run-time metrics to study coupling between objects," in *Proc. of the 20th Annual ACM Symposium on Applied Computing (SAC '05)*, Santa Fe, New Mexico, USA, March 2005, pp. 1456–1462.

[98] A. Garcia, C. Sant'Anna, E. Figueiredo, U. Kulesza, C. Lucena, and A. von Staa, *Modularizing Design Patterns with Aspects: A Quantitative Study.* Springer Berlin Heidelberg, 2006, pp. 36–74.

[99] U. Kumari and S. Bhasin, "Application of object-oriented metrics to C++ and Java: A comparative study," *SIGSOFT Softw. Eng. Notes*, vol. 36, no. 2, pp. 1–10, 2011.

[100] M. Choi, J. Lee, and J. Ha, "A component cohesion metric applying the properties of linear increment by dynamic dependency relationships between classes," in *Proc. of the International Conference on Computational Science and Its Applications (ICCSA 2006), Part II*, Glasgow, UK, May 2006, pp. 49–58.

[101] J. M. Bieman and B.-K. Kang, "Cohesion and reuse in an object-oriented system," in *Proc. of the Symposium on Software Reusability (SSR '95)*, Seattle, Washington, USA, April 1995, pp. 259–262.

[102] C. Bonja and E. Kidanmariam, "Metrics for class cohesion and similarity between methods," in *Proc. of the 44th Annual Southeast Regional Conference (ACM-SE 44)*, Melbourne, Florida, March 2006, pp. 91–95.

[103] C. Sant'Anna, A. Garcia, C. Chavez, C. Lucena, and A. Von Staa, "On the reuse and maintenance of aspect-oriented software: An assessment framework," in *Proc. of the Brazilian symposium on software engineering*, Manaus, Brazil, October 2003, pp. 19–34.

[104] K. El Guemhioui, "A framework for distributing object-oriented designs," *International Journal on Software Tools for Technology Transfer*, vol. 4, no. 3, pp. 381–396, 2003.

[105] I. Chowdhury and M. Zulkernine, "Can complexity, coupling, and cohesion metrics be used as early indicators of vulnerabilities?" in *Proc. of the ACM Symposium on Applied Computing (SAC '10)*, Sierre, Switzerland, March 2010, pp. 1963–1969.

[106] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," *Information and Software Technology*, vol. 59, no. C, pp. 170–190, 2015.

[107] J. A. Dallal, "Improving the applicability of object-oriented class cohesion metrics," *Information and Software Technology*, vol. 53, no. 9, pp. 914–928, 2011.

[108] H. Ching-Lai and K. Yoon, *Multiple attribute decision making: methods and applications*. Springer-Verlag, 1981.

[109] J. C. de Borda, *Memoire sur les Elections au Scrutin*. Histoire de l'Academie Royale des Sciences de Paris, 1781.

[110] M. Galster and E. Bucherer, "A taxonomy for identifying and specifying nonfunctional requirements in service-oriented development," in *Proc. of the IEEE Congress on Services - Part I (SERVICES-1)*, Honolulu, HI, USA, July 2008, pp. 345–352.

[111] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[112] Z. Rehman, O. Hussain, and F. Hussain, "IaaS cloud selection using MCDM methods," in *Proc. of the 9th International Conference on e-Business Engineering (ICEBE)*, Hangzhou, China, September 2012, pp. 246–251.

[113] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proc. of the 10th international conference on World Wide Web (WWW '01)*, Hong Kong, China, May 2001, pp. 613–622.

[114] W. D. Cook and L. M. Seiford, "On the Borda-Kendall consensus method for priority ranking problems," *Management Science*, vol. 28, no. 6, pp. 621–637, 1982.

[115] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Integrity for approximate joins on untrusted computational servers," in *Proc. of the 30th International Information Security and Privacy Conference (SEC 2015)*, Hamburg, Germany, May 2015, pp. 446–459.

[116] P. Samarati and S. De Capitani di Vimercati, "Cloud security: Issues and concerns," in *Encyclopedia on Cloud Computing*, S. Murugesan and I. Bojanova, Eds. Wiley, 2016.

[117] G. Bojadziev and M. Bojadziev, *Fuzzy logic for business, finance, and management*. World Scientific, 1997.

[118] İ. Ertuğrul and N. Karakaşoğlu, "Comparison of fuzzy AHP and fuzzy TOPSIS methods for facility location selection," *The International Journal of Advanced Manufacturing Technology*, vol. 39, no. 7, pp. 783–79, 2008.

[119] M.-F. Chen and G.-H. Tzeng, "Combining grey relation and TOPSIS concepts for selecting an expatriate host country," *Mathematical and Computer Modelling*, vol. 40, no. 13, pp. 1473–1490, 2004.

[120] A. Jamshidi, A. Yazdani-Chamzini, S. H. Yakhchali, and S. Khaleghi, "Developing a new fuzzy inference system for pipeline risk assessment," *Journal of Loss Prevention in the Process Industries*, vol. 26, no. 1, pp. 197 – 208, 2013.

[121] R. Nasim and A. J. Kassler, "Deploying Openstack: Virtual infrastructure or dedicated hardware," in *Proc. of the 38th IEEE International Computer Software and Applications Conference Workshops (COMPSACW)*, Vasteras, Sweden, July 2014, pp. 84–89.

[122] H. Hu, Y. Wen, T. S. Chua, J. Huang, W. Zhu, and X. Li, "Joint content replication and request routing for social video distribution over cloud CDN: A community clustering method," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 7, pp. 1320–1333, 2016.

[123] "Amazon EC2 instance types." [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[124] W.-J. Fan, S.-L. Yang, H. Perros, and J. Pei, "A multi-dimensional trust-aware cloud service selection mechanism based on evidential reasoning approach," *International Journal of Automation and Computing*, vol. 1, no. 2, pp. 208–219, 2015.

[125] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud security and privacy: an enterprise perspective on risks and compliance*. O'Reilly Media, Inc., 2009.

[126] B. P. Rimal, E. Choi, and I. Lumb, *A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems*. Springer London, 2010, pp. 21–46.

[127] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[128] P. T. Jaeger, J. Lin, and J. M. Grimes, "Cloud computing and information policy: Computing in a policy cloud?" *Journal of Information Technology & Politics*, vol. 5, no. 3, pp. 269–283, 2008.

[129] L. Badger, T. Grance, R. Patt-Corner, J. Voas *et al.*, "Cloud computing synopsis and recommendations," 2012. [Online]. Available: http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=909505

[130] K. Dahbur, B. Mohammad, and A. B. Tarakji, "A survey of risks, threats and vulnerabilities in cloud computing," in *Proc. of the International Conference on Intelligent Semantic Web-Services and Applications (ISWSA '11)*, New York, NY, USA, April 2011, pp. 12:1–12:6.

[131] Y. Gu, W. Zhang, and J. Tao, "A study of SLA violation compensation mechanismin complex cloud computing environment," in *Proc. of the 2nd International*

*Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC)*, Harbin, China, December 2012, pp. 1448–1451.

[132] "Amazon EC2 service level agreement." [Online]. Available: https://aws.amazon.com/ec2/sla/

[133] C. Kirchsteiger, "On the use of probabilistic and deterministic methods in risk analysis," *Journal of Loss Prevention in the Process Industries*, vol. 12, no. 5, pp. 399–419, 1999.

[134] S. Kaplan and B. J. Garrick, "On the quantitative definition of risk," *Risk Analysis*, vol. 1, no. 1, pp. 11–27, 1981.

[135] S. Ferson, L. Ginzburg, V. Kreinovich, H. T. Nguyen, and S. A. Starks, "Uncertainty in risk analysis: towards a general second-order approach combining interval, probabilistic, and fuzzy techniques," in *Proc. of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'02)*, Honolulu, HI, USA, May 2002, pp. 1342–1347.

[136] G. E. Gürcanli and U. Müngen, "An occupational safety risk analysis method at construction sites using fuzzy sets," *International Journal of Industrial Ergonomics*, vol. 39, no. 2, pp. 371–387, 2009.

[137] V. Kreinovich and S. Ferson, "A new cauchy-based black-box technique for uncertainty in risk analysis," *Reliability Engineering & System Safety*, vol. 85, no. 1, pp. 267–279, 2004.

[138] A. Karami and Z. Guo, "A fuzzy logic multi-criteria decision framework for selecting it service providers," in *Proc. of the 45th Hawaii International Conference on System Sciences (HICSS)*, Maui, HI, USA, January 2012, pp. 1118–1127.

[139] M. Tajvidi, R. Ranjan, J. Kolodziej, and L. Wang, "Fuzzy cloud service selection framework," in *Proc. of the 3rd IEEE International Conference on Cloud Networking (IEEE CloudNet 2014)*, Luxembourg, October 2014, pp. 443–448.

[140] K. Toczé, M. Vasilevskaya, P. Sandahl, and S. Nadjm-Tehrani, "Maintainability of functional reactive programs in a telecom server software," in *Proc. of the 31st Annual ACM Symposium on Applied Computing (SAC '16)*, Pisa, Italy, April 2016, pp. 2001–2003.

[141] M. Wimmer, A. Schauerhuber, G. Kappel, W. Retschitzegger, W. Schwinger, and E. Kapsammer, "A survey on UML-based aspect-oriented design modeling," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 28:1–28:33, 2011.

[142] C. Atkinson and T. Kuhne, "Aspect-oriented development with stratified frameworks," *IEEE Software*, vol. 20, no. 1, pp. 81–89, 2003.

[143] S. Sehestedt, C.-H. Cheng, and E. Bouwers, "Towards quantitative metrics for architecture models," in *Proc. of the 11th Working IEEE/IFIP Conference on Software Architecture (WICSA 2014)*, Sydney, Australia, April 2014, pp. 5:1–5:4.

[144] A. F. M. Hani, I. V. Paputungan, and M. F. Hassan, "Renegotiation in service level agreement management for a cloud-based system," *ACM Comput. Surv.*, vol. 47, no. 3, pp. 51:1–51:21, 2015.

[145] Y. Yao and H. Chen, "QoS-aware service composition using NSGA-II1," in *Proc. of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS '09)*, Seoul, Korea, November 2009, pp. 358–363.

[146] T. Lin, B. Park, H. Bannazadeh, and A. Leon-Garcia, *SAVI Testbed Architecture and Federation.* Springer International Publishing, 2015, pp. 3–10.

[147] H.-J. Shyu and R. Hillson, "A software workbench for estimating the effects of cumulative sound exposure in marine mammals," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 1, pp. 8–21, 2006.

[148] Y. Cai, S. Huynh, and T. Xie, "A framework and tool supports for testing modularity of software design," in *Proc. of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE '07)*, New York, NY, USA, 05–09 2007, pp. 441–444.

[149] P. Inverardi and M. Tivoli, *Software Architecture for Correct Components Assembly.* Springer Berlin Heidelberg, 2003, pp. 92–121.

[150] C.-C. Chiang and C. W. Ford, "Maintainability and reusability issues in corba-based systems," in *Proc. of the 43rd Annual Southeast Regional Conference - Volume 2 (ACM-SE 43)*, Kennesaw, Georgia, March 2005, pp. 275–280.

[151] J.-H. Lo, S.-Y. Kuo, M. R. Lyu, and C.-Y. Huang, "Optimal resource allocation and reliability analysis for component-based software applications," in *Proc. of the 26th Annual International Computer Software and Applications (COMPSAC 2002)*, Oxford, UK, UK, August 2002, pp. 7–12.

[152] J. Idziorek, M. Tannian, and D. Jacobson, "Modeling web usage profiles of cloud services for utility cost analysis," in *Proc. of the Winter Simulation Conference (WSC)*, Phoenix, AZ, USA, December 2011, pp. 3318–3329.

[153] X. Xu, "From cloud computing to cloud manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 1, pp. 75 – 86, 2012.

[154] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, "NIST cloud computing reference architecture," *NIST special publication*, 2011.

[155] R.-C. Tsaur, "Decision risk analysis for an interval topsis method," *Applied Mathematics and Computation*, vol. 218, no. 8, pp. 4295–4304, 2011.

[156] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, "A framework for native multi-tenancy application development and management," in *Proc. of the 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, Tokyo, Japan, July 2007, pp. 551–558.

[157] S. Kolb and G. Wirtz, "Towards application portability in platform as a service," in *Proc. of the 8th IEEE International Symposium on Service Oriented System Engineering (SOSE)*, Oxford, UK, April 2014, pp. 218–229.

[158] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid clouds," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 861–870, 2012.

[159] D. Talia, "Cloud computing and software agents: Towards cloud intelligent services," in *Proc. of the 12th Workshop on Objects and Agents*, Rende (CS), Italy, July 2011, pp. 2–6.

[160] S. A. Baset, "Cloud SLAs: Present and future," *SIGOPS Oper. Syst. Rev.*, vol. 46, no. 2, pp. 57–66, 2012.

[161] A. Undheim, A. Chilwan, and P. Heegaard, "Differentiated availability in cloud computing SLAs," in *Proc. of the 12th IEEE/ACM International Conference on Grid Computing (Grid 2011)*, Lyon, France, September 2011, pp. 129–136.

[162] Y. B. Ma, S. H. Jang, and J. S. Lee, "Ontology-based resource management for cloud computing," in *Proc. of the 3rd Asian Conference on Intelligent Information and Database Systems (ACIIDS 2011), Part II*, Daegu, Korea, April 2011, pp. 343–352.

[163] L. Wu, R. Buyya *et al.*, "Service level agreement SLA in utility computing systems," 2012. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1010/1010.2881.pdf

[164] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.

[165] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs Containerization to support PaaS," in *Proc. of the IEEE International Conference on Cloud Engineering (IC2E)*, Boston, MA, USA, March 2014, pp. 610–614.

[166] A. Arman, S. Foresti, G. Livraga, and P. Samarati, "A consensus-based approach for selecting cloud plans," in *Proc. of the 2nd IEEE International Forum on Research and Technologies for Society and Industry (RTSI 2016)*, Bologna, Italy, September 2016, pp. 1–6.

# A

# PUBLICATIONS

Some ideas and significant results present in this thesis were published in:

1. **“A consensus-based Approach for Selecting Cloud Plans”**

   A. Arman, S. Foresti, G. Livraga, and P. Samarati

   2nd International Forum on Research and Technologies for Society and Industry (RTSI 2016), Bologna, Italy, September 2016

   **Abstract:** An important problem when moving an application to the cloud consists in selecting the most suitable cloud plan (among those available from cloud providers) for the application deployment, with the goal of finding the best match between application requirements and plan characteristics. If a user wishes to move multiple applications at the same time, this task can be complicated by the fact that different applications might have different (and possibly contrasting) requirements. In this paper, we propose an approach enabling users to select a cloud plan that best balances the satisfaction of the requirements of multiple applications. Our solution operates by first ranking the available plans for each application (matching plan characteristics and application requirements) and then by selecting, through a consensus-based process, the one that is considered more acceptable by all applications.

2. **“A Risk-aware Application Scheduling Model in Cloud Computing Scenarios”**

   A. Arman

International Journal of Intelligent Systems and Applications(IJISA), vol. 8, no. 10, pp. $11-20$, 2016

**Abstract:** Cloud providers usually apply some compensation mechanisms when their promised qualities of services are not met. Therefore, to support the business objectives of users, such compensation mechanisms, which can have high impacts on the financial profit of applications when they are executed on the cloud, should be carefully considered in application scheduling scenarios. We propose an approach aimed at, by mapping each application to an available VM offered by multiple Cloud providers, maximizing financial profit that is estimated for each application, according to its importance. It mainly works in three phases. Considering each possible VM availability scenario, we first measure a penalty which is paid by Cloud provider if the promised uptime of VM is not met, and then, estimate a financial profit for the current application to be scheduled if it is assigned to each available VM. Finally, through a risk analysis process, we assign each application to an available VM, according to the expected monetary value of application when it is mapped to each available VM.

3. **"Towards an Analytical Approach to Measure Modularity in Software Architecture Design"**

M. Ghasemi, S. M. Sharafi, and A. Arman

Journal of Software, vol. 10, no. 4, pp. $465-479$, 2015

**Abstract:** We propose a software engineering approach to analytically evaluate the modularity of applications to estimate to what extent they can be easily moved to the Cloud w.r.t. change flexibility and/or distributability. Our proposed solution operates in three main steps: *i)* defining an evaluation classification which includes different modularity attributes (e.g., coupling, cohesion) and their associated metrics (e.g., coupling between classes for coupling attribute), according to the context of problem (e.g., application type, execution context of application on the Cloud); *ii)* estimating modularity at micro (component) level, considering attributes and metrics studied in the first step; *iii)* evaluating modularity at macro (system) level, considering the obtained modularity at micro level.

4. **"Improving the Efficiency of Term-weighting in Dynamic Sets of Documents"**

M. Jabalameli, A. Arman, and M. Nematbakhsh

International Journal of Modern Education and Computer Science, vol. 7, no. 2, pp. 42-47, 2015

**Abstract:** The performance refinement of outsourcing applications is considered as an essential issue because it can considerably avoid the wasting of Cloud resources and unnecessary extra costs, paid by Cloud users. In this paper, a method is proposed to improve the efficiency of computation-intensive applications in term-weighting scenarios by considering a special part of dynamic documents' revisions instead of their whole revision history. The evaluation of proposed method shows its ability to keep the quality of retrieved information at an acceptable rate, while notably decreases the analysis time.

5. **"Considering Application Importance in Cloud Plan Selection"** (Manuscript in preparation)

   **Abstract:** Selecting the right cloud plan is a key issue when outsourcing applications to the cloud. When multiple applications need to be deployed at the same time on the same plan, it is necessary to combine their requirements to determine a plan that is suitable for all applications. In this paper, we address the problem of selecting a cloud plan when the outsourced applications have different importance, differentiating the impact that their preferences should have in the selection process. Our approach permits different stakeholders to express applications importance, to simplify definition and capture the imprecision of human judgements, through linguistic variables. Our solution then aggregates the importance of applications, considering the opinions of different stakeholders with different relevance in the decision process. Applications importance is then taken into consideration for cloud plan selection.