Cyprus University of Technology

Faculty of Engineering and Technology

Doctoral Dissertation

Utilizing Computational Intelligence to Support Decision-making in Distributed Software Systems and Cloud-based Environments

Andreas Christoforou

Limassol, April 2020

CYPRUS UNIVERSITY OF TECHNOLOGY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING, COMPUTER ENGINEERING AND INFORMATICS

Doctoral Dissertation

Utilizing Computational Intelligence to Support Decision-making in Distributed Software Systems and Cloud-based Environments

Andreas Christoforou

Limassol, April 2020

Approval Form

Doctoral Dissertation

Utilizing Computational Intelligence to Support Decision-making in Distributed Software Systems and Cloud-based Environments

Presented by

Andreas Christoforou

Supervisor: Dr. Andreas S. Andreou, Professor

Signature _____

Member of the committee: Dr. Chrysostomos Stylios, Professor

Signature _____

Member of the committee: Dr. Constandinos X. Mavromoustakis, Professor

Signature _____

Cyprus University of Technology

Limassol, April 2020

Copyrights

The approval of the dissertation by the Department of Electrical Engineering, Computer Engineering and Informatics does not imply necessarily the approval by the Department of the views of the writer.

to Maria, Ioanna and Marina

Acknowledgements: I would like to express my special appreciation and say a big thank you to my advisor Professor Andreas S. Andreou. Without his guidance and constant support, this PhD would not have been achievable. I am grateful for giving me the opportunity to work with him and become a member of his team.

I would also like to thank my committee members, professor Chrysostomos Stylios and professor Constandinos X. Mavromoustakis who devoted much of their precious time to read my thesis and provide me valuable comments and suggestions.

Furthermore, I would especially like to thank the members of the Software Engineering and Intelligent Information Systems Research Laboratory for their support and outstanding cooperation we had these years.

An exceptional thanks to my family, my parents Christakis and Koulla, my parents in-law Yiannis and Androulla and my brothers Panayiotis and George for their persistence, love and support.

Last, but most important, I would like to express my deep gratitude to my beloved wife, Maria and my two daughters Ioanna and Marina. Their endless love and support was the driving force behind what I have achieved so far.

ABSTRACT

The design and development of distributed software systems appears nowadays as a prominent way to provide modern, reliable and functional systems. The growth of such systems was greatly supported by the development and rapid expansion of Cloud Computing (CC), the latter offering a variety of on-demand services, thus alleviating the need to purchase and maintain expensive or sophisticated hardware and/or development tools, and supporting efficient data processing and storing.

Cloud Computing has been established as an attractive development environment on which software development companies can deliver and deploy their services in response to the increasing demands. Inevitably this has brought great challenges in the software engineering field, mostly for setting up a new software development environment that simplifies the procedure of building and hosting applications on the Cloud. Despite the great number of solutions proposed in recent years, the continuous evolution and growth of the demands and services offered, the corresponding research challenges in this area remain untackled. These challenges revolve mainly around three axes: delivery of the software system on time, within budget and with a minimum acceptable level of quality.

The persistent incorporation of numerous new technologies, makes Cloud infrastructure management highly complex, with multi-conflicting factors affecting it. Computational intelligence and machine learning techniques appear to have great success when dealing with complex and multifaceted problems. Aiming to investigate a series of research issues and problems in software engineering for the Cloud, this thesis proposes the use of various computational intelligent techniques and approaches which are modified and extended to meet specific challenges of the problem in hand. A special reference is made to techniques based on Artificial Neural Networks, Fuzzy Logic and Evolutionary Computation, which seem to dominate the relevant literature yielding promising results.

The contribution of this thesis may be analyzed into six research steps. The first step introduces a novel, integrated analysis framework based on Multi-Layer Fuzzy Cognitive Maps models, as well as a series of actions to gather useful static and dynamic information. This framework allows the representation of problems described by multiple and intertwined factors, as the ones dealt with in the present thesis. Moreover, the proposed framework provides the means for performing advanced dynamic analysis in the form of what-if scenarios. In the second step, research is focused on the analysis and study of the factors that affect the

adoption of Cloud services. Four different approaches based on Fuzzy Cognitive Maps and Influence Diagrams are proposed in an attempt to support the decision-making process. The third step extends the aforementioned analysis framework with the incorporation of an evolutionary approach based on a novel formulation proposed. Next, the fourth research step involves the construction of a Multi-Layer Fuzzy Cognitive Map to support decision-making towards microservices architecture migration. The fifth step proposed a novel process for the decomposition of existing software components and ultimately their partial or full replacement of their functionality parts with a number of suitable and available microservices. Finally, the sixth step suggests a new resource management approach in a Function-as-a-Service platform. The proposed approach is based on Multi Objective Genetic Algorithms and aims to solve the problem of finding a set of near-optimal solutions that support developers in a Function-as-a-Service environment to select an efficient resource allocation scheme with respect to cost and time.

All of the proposed approaches and models are evaluated both theoretically and practically over real-world case studies. In each case the experimental process is designed, executed and analyzed followed by discussion of the results.

Keywords: Software Engineering, Cloud Computing, Distributed Systems Development, Service Oriented Architecture, Microservices Architecture, Serverless Computing, Computational Intelligence, Decision Support

TABLE OF CONTENTS

ABSTRACT . . . . . **xi**

TABLE OF CONTENTS . . . . . **xiii**

LIST OF TABLES . . . . . **xvii**

LIST OF FIGURES . . . . . **xix**

LIST OF ABBREVIATIONS . . . . . **xxi**

LIST OF PUBLICATIONS . . . . . **xxiii**

1 Introduction **1**

1.1 Problem Statement 2

1.2 Research Challenges 4

1.2.1 Cloud Adoption 4

1.2.2 Microservices Architecture Migration 5

1.2.3 Component-Based to Microservices Architecture Migration 5

1.2.4 Resource Management on a Function as a Service Environment . . . 6

1.3 Thesis Structure 7

2 Theoretical and Technical Background **9**

2.1 Software Engineering for Distributed Systems Development 9

2.1.1 Software Engineering 9

2.1.2 Cloud Computing 10

2.1.3 Software Development and Operation on the Cloud 12

2.1.3.1 Microservice Architecture 12

2.1.3.2 Serverless Computing - Function as a Service 13

2.2 Computational Intelligent Techniques 13

2.2.1 Fuzzy Cognitive Maps 13

2.2.2 Multi-Layer Fuzzy Cognitive Maps . . . 16
2.2.3 Influence Diagrams . . . 17
2.2.3.1 Generic Influence Diagrams . . . 17
2.2.3.2 Fuzzy Influence Diagrams . . . 19
2.2.4 Genetic Algorithms . . . 20
2.2.4.1 Multi-Objective Genetic Algorithms . . . 21

3 A Framework for Analyzing Multi-Layer Fuzzy Cognitive Maps . . . 23
3.1 Introduction . . . 23
3.2 Literature Overview . . . 25
3.3 Static Analysis . . . 26
3.4 Dynamic Analysis . . . 29
3.5 Stepwise Analysis and Inference Process . . . 31
3.6 Summary . . . 32

4 Modeling the Cloud Adoption Decision . . . 35
4.1 Introduction . . . 35
4.2 Literature Overview . . . 36
4.3 ID Modeling . . . 38
4.3.1 Generic ID . . . 40
4.3.1.1 Experimental Results . . . 41
4.3.2 Fuzzy ID . . . 43
4.3.2.1 Experimental Results . . . 44
4.3.3 Comparison of the ID Models . . . 45
4.3.4 What-if Scenario Simulations . . . 47
4.4 FCM Modeling . . . 48
4.4.1 Single Layer FCM . . . 48
4.4.1.1 Experimental Results . . . 50
4.4.1.1.1 Extreme Scenarios . . . 51
4.4.1.1.2 Real-World Scenarios . . . 52
4.4.1.2 Discussion . . . 55
4.4.2 Multi Layer FCM . . . 55
4.4.2.1 Experimental Results . . . 56
4.4.2.1.1 Extreme Scenarios . . . 59
4.4.2.1.2 Real-World Scenarios . . . 59

4.4.2.2 Discussion of Results . . . 60
4.4.2.3 MLFCM Model Analysis . . . 61
4.4.2.3.1 Static Analysis . . . 61
4.4.2.3.2 Dynamic Analysis . . . 64
4.5 Summary . . . 67

5 A Novel Computational Approach for MLFCM . . . 69
5.1 Introduction . . . 69
5.2 Literature Overview . . . 70
5.3 Computational Approach . . . 70
5.3.1 Activation Levels Genetically Evolved MLFCM . . . 72
5.4 Summary . . . 74

6 Supporting the Decision of Migrating to Microservices Architecture . . . 75
6.1 Introduction . . . 75
6.2 Literature Overview . . . 77
6.3 A Novel MLFCM Model . . . 78
6.3.1 Model Construction . . . 78
6.3.2 Model Validation and Calibration . . . 82
6.4 Static and Dynamic Model Analyses . . . 85
6.4.1 Static Analysis . . . 86
6.4.2 Dynamic Analysis . . . 89
6.5 Model Application Over a Real-World Case Study . . . 92
6.6 Summary . . . 97

7 Migration of Software Components to Microservices: Matching and Synthesis 99
7.1 Introduction . . . 99
7.2 Literature Overview . . . 101
7.3 Automatic Specification and Matching of Microservices . . . 102
7.3.1 Specification and Matching framework . . . 102
7.3.2 Profiling . . . 103
7.3.3 Components Decomposition and Microservices Matching . . . 108
7.3.3.1 Component and Microservices Ontology . . . 108
7.3.3.2 Matching Process . . . 109
7.4 Experimental Process . . . 113
7.4.1 Proof of Concept . . . 114

7.4.2 Composition Assessment . . . 115
7.5 Results and Discussion . . . 115
7.6 Summary . . . 118

8 An Effective Resource Management Approach in a FaaS Environment . . . 119
8.1 Introduction . . . 119
8.2 Literature Overview . . . 120
8.3 Multi-objective Optimization Approach . . . 121
8.4 Experimental Process . . . 122
8.4.1 Experimental Environment . . . 122
8.4.2 Exhaustive Algorithm . . . 123
8.4.3 Multi-objective Genetic Algorithms . . . 124
8.5 Results and Discussion . . . 125
8.5.1 Assessing MOGAs' Performance Through Quality Indicators . . . 125
8.6 Summary . . . 128

9 Conclusions and Future Research Steps . . . 129
9.1 Overview . . . 129
9.2 A Framework for Analyzing Multi-Layer Fuzzy Cognitive Maps . . . 130
9.3 Modeling the Cloud Adoption Decision . . . 131
9.4 A Novel Computational Approach for MLFCM . . . 132
9.5 Supporting the Decision of Migrating to Microservices Architecture . . . 133
9.6 Migration of Software Components to Microservices: Matching and Synthesis 133
9.7 An Effective Resource Management Approach in a FaaS Environment . . . 134

REFERENCES . . . 147

LIST OF TABLES

4.1 Factors influencing Cloud adoption 39
4.2 Brief description of real-world cases. 40
4.3 Input values for the five scenarios tested 42
4.4 Calculating probabilities on *Trust* node 43
4.5 Fuzzy values 43
4.6 Input values of the nodes participating in the FID model for the five scenarios tested 44
4.7 Evaluating Trust node in FID 44
4.8 Model's decisions compared with real decisions 45
4.9 Conceptual nodes of the proposed model 49
4.10 Causal relationships and weight values between conceptual nodes on a Likert scale from 1 (very low) to 5 (very high) positive and negative - Row influences column. 50
4.11 FCM initial activation level values of the concepts for the positive scenario. . 52
4.12 FCM initial activation level values of the concepts for the negative scenario. . 53
4.13 Brief description of real-world cases. 54
4.14 Initial activation level values for real world cases. 54
4.15 Model's decisions compared with real decisions. 55
4.16 Conceptual nodes of the proposed MLFCM model. 57
4.17 Sub-FCM Groupings 58
4.18 Initial activation levels for each real-world case 60
4.19 Model's output decisions compared with real decisions 61
4.20 Complexity static measurements for the MLFCM modeling the Cloud Adoption problem 62
4.21 Strength and tendency indicators for the sub-FCMs of the MLFCM model for the Cloud Adoption problem 63
4.22 Correlation coefficient values between each concept and the central node of interest in each sub-FCM (in parentheses) of the MLFCM model 65

6.1 Concepts related to the decision of adopting microservice architectures, and their groupings (FCMs) derived from literature review and evaluated by experts (central concept of each FCM in bold). 80
6.2 Normalized Weight Matrix for causal relationships in FCM1 82
6.3 Industrial Case Studies 86
6.4 Complexity static measurements. 87
6.5 Strength indicators for the sub-FCMs of the MLFCM model for the Microservices Adoption problem. 88
6.6 Indicative simulations part of the dynamic analysis of the model. 91
6.7 Industry case study: Final activation levels 93
6.8 Industry case study: How to improve the final decision 94
6.9 Five ALGE-MLFCM runs targeting a "Very High" value for *Microservices Adoption* and a constant "High" value for *Security* against the nearest solution (NS) reference. 96

7.1 Scoring results of components' functional parts without dependencies. 116
7.2 Scoring results of components' functional parts with dependencies. 116

8.1 Hypervolume(HV) values 126
8.2 Inverted Generational Distance(IGD) values 127
8.3 Pairwise comparison for HV indicator 127
8.4 Pairwise comparison for IGD indicator 127

LIST OF FIGURES

2.1 An example FCM with its weight matrix and initial activation levels (AL) vector. 14
2.2 An example Multi-Layer FCM with two layers. 16
2.3 A simple Influence Diagram 18
2.4 Scalable Monotonic Chaining example combining F-Type and G-Type fuzzy sets 20

4.1 ”Adopt Cloud or Not” Influence diagram 40
4.2 An alternative inductive structure of our FID model 46
4.3 The Cloud adoption CNFCM model. 51
4.4 Extreme scenarios: (a) Positive Case, (b) Negative Case. 53
4.5 The Cloud adoption MLFCM model. 58

6.1 Fuzzification of linguistic variables according to triangular membership functions: (a) seven values, (b) five values 81
6.2 Visual representation of the MLFCM model for the Microservices Architecture adoption 83
6.3 Model validation over two extreme scenarios 84
6.4 The profile of the survey participants 85
6.5 The convergence of the ALGE-MLFCM algorithm for different target values . 92
6.6 The attempt of the ALGE-MLFCM algorithm to find a solution that leads the final decision to “Very High” value with the value of “Security” to start at and remain “High” 95

7.1 The proposed process for component decomposition and microservices substitution. 104
7.2 Component profile in EBNF 106
7.3 Microservice profile in EBNF 107
7.4 Component ontology 108
7.5 Microservice ontology 109
7.6 Near-optimal pareto fronts. 118

8.1 Proposed multi-objective optimization approach 121
8.2 Experimental environment 123
8.3 Pareto front for 1000 fitness evaluations (FE) 126

LIST OF ABBREVIATIONS

DSS:	Decision Support System
CI:	Computational Intelligence
AI:	Artificial Intelligence
ANN:	Artificial Neural Networks
FS:	Fuzzy Systems
EC:	Evolutionary Computation
CC:	Cloud Computing
FCM:	Fuzzy Cognitive Map
ID:	Influence Diagram
QoS:	Quality of Service
FaaS:	Function as a Service
SLA:	Service-Level Agreement
GA:	Genetic Algorithm
SE:	Software Engineering
IT:	Information Technology
SME:	Small-Medium Enterprise
PaaS:	Platform as a Service
IaaS:	Infrastructure as a Service
SaaS:	Software as a Service
SOC:	Service Oriented Computing (SOC)
MDE:	Model Driven Engineering
MCC:	Mobile Cloud Computing
SOA:	Service Oriented Architecture
MLFCM:	Multi-layered Fuzzy Cognitive Map
GID:	Generic Influence Diagram
FID:	Fuzzy Influence Diagram
ALGE-MLFCM:	Activation Levels Genetically Evolved MLFCM
MOP:	Multi-objective Optimization Problems
IGD:	Inverted Generational Distance
CNFCM:	Certainty Neuron Fuzzy Cognitive Maps
CBSE:	Component-based Software Engineering
CBD:	Component-based Development

LIST OF PUBLICATIONS

1. Christoforou A, Andreou AS. A cloud adoption decision support model based on fuzzy cognitive maps. In International Conference on Product Focused Software Process Improvement (PROFES) 2013 Jun 12 (pp. 240-252).

2. Christoforou A, Andreou AS. A cloud adoption decision support model using influence diagrams. In IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI) 2013 Sep 30 (pp. 151-160).

3. Christoforou A, Andreou AS. A multilayer fuzzy cognitive maps approach to the cloud adoption decision support problem. In2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) 2015 Aug 2 (pp. 1-8).

4. Christoforou A, Andreou AS. Investigating cloud adoption using influence diagrams as a decision support model. International Journal on Artificial Intelligence Tools. 2015 Dec 21;24(06):1560005

5. Christoforou A, Andreou AS. A framework for static and dynamic analysis of multi-layer fuzzy cognitive maps. Neurocomputing. 2017 Apr 5;232:133-45.

6. Christoforou A, Garriga M, Andreou AS, Baresi L. Supporting the decision of migrating to microservices through multi-layer fuzzy cognitive maps. In International Conference on Service-Oriented Computing (ICSOC) 2017 Nov 13 (pp. 471-480). Springer, Cham.

7. Christoforou A, Andreou AS, An effective resource management approach in a FaaS environment. European Symposium on Serverless Computing and Applications (ESSCA) 2018 Dec 21.

8. Christoforou A, Odysseos L, Andreou, A. Migration of Software Components to Microservices: Matching and Synthesis.In Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE) 2019 (pp. 134-146).

9. Christoforou A, Andreou AS, Garriga M, Baresi L. Adopting Microservices Architecture: A Decision and Analysis Support Model Based on Activation Levels Genetically Evolved Multi-layer Fuzzy Cognitive Maps (ALGE-MLFCM). Applied Soft Computing. (Submitted March 2020)

Chapter 1

Introduction

The design and development of distributed software systems has seen tremendous growth in recent years and appears to be a prominent way to provide reliable and functional systems. The rapid expansion of distributed software systems was enabled by Cloud Computing (CC), which offers an alternative form of services without the need to purchase and maintain expensive or sophisticated hardware in order to support data processing and storing. Cloud computing systems have benefited by minimizing the initial cost without, at the same time, falling short of a high-level performance, availability, scalability, and fault tolerance.

Software developers may utilize one or more of the offered types of services on the Cloud depending on the kind of application they are building for their prospective clients. Nevertheless, no matter the service offered in a distributed Cloud environment, there are universal software development and delivery issues to tackle. These issues can be considered as highly complex problems with competing and conflicting factors. For example, cost, security, availability, and elasticity are often the factors juxtaposed with performance, and, while addressing these factors, the compliance with the Service Level Agreements should also be taken into account.

In recent years, the rapid development of technology, along with the simultaneous increase in the offered computing power, enable the development of significant approaches from the field of Computational and Artificial intelligence (CI/AI). The utilization of such kind of techniques and approaches in decision-making support is now the dominant trend that is applicable to many fields such as economics, industrial production lines, pharmacology and medicine. Computational Intelligence (CI) is a sub-branch of Artificial Intelligence (AI), which attempts to apply methods and techniques that enable or facilitate intelligent behavior in complex and uncertain environments [1, 2]. These techniques mainly consist of Artificial Neural Networks

(ANN), Fuzzy Systems (FS) and Evolutionary Computation (EC) techniques. CI applications may be found in various engineering and scientific fields, like Software Engineering, Robotics, Economics, Bioinformatics, Telecommunications, etc.

The main goal of a Decision Support System (DSS) is to support the decision-making process; however, over time, this definition and the application scope change. In the early 1970s, DSS was described as a computer-assisted DSS. Towards the end of the 1970s, the DSS began to focus on supporting users in making a more interactive decision to solve unstructured problems. In the 1980s, a DSS should be able to use the available technology to improve the efficiency of administrative and professional activities. From the late 1980s onwards, the challenge was to use smart methods to solve problems. It is evident that at least three parameters form the aim, the application, and the definition of a DSS: The first parameter refers to the evolution and increased complexity of the problems that the DSS undertakes to solve, the second parameter has to do with the available involved technologies, and, finally, the third reflects the emergence of new application fields.

Nowadays, DSS are called upon to address problems of exceptional complexity, having at the same time a vast amount of data to process. An essential parameter for the successful application of a DSS in a particular problem is the utilization of intelligent methods and algorithms that will be able to deliver support by providing reliable information within a reasonable time, but also be able to provide appropriate explanations for the support outcomes they deliver. In addition, what is equally important, it the description of the interaction capabilities mainly to support the execution of simulations so that the user is able to study alternative scenarios before reaching a final decision.

This thesis is involved with a particular area of research that deals with several aspects of software engineering for distributed software systems and Cloud environments. The research activities introduced are targeting to provide methods, algorithms, or frameworks from the area of CI, aiming to meet significant challenges in each sub-area and provide efficient solutions. The ultimate goal is to provide efficient decision support for a number of related problems in the corresponding scientific area, enabling domain experts and practitioners analyze the pertaining factors and take more informative actions.

1.1 Problem Statement

Delivery of a software system on time, within budget and with an acceptable level of quality, remains a challenging issue and an unsolved problem in the area of Software Engineering

through the years, despite the great and sometimes drastic changes proposed both from the software process perspective and the tools used. In recent years the emergence and spread of CC has reset and redefined the way that notions like quality, time and budget are approached. In this context, CC appears to be a beneficial IT paradigm that promises reduced costs, increased efficiency, and better performance. Although the research community has already addressed many issues related to CC, its rapid growth, as well as the persistent incorporation of new technologies, many issues still need further and deeper investigation. Security, resource management, Cloud adoption, pricing, and energy efficiency are only a few of a wide range of issues.

As an attractive development environment, CC motivates software development companies to join a market with increasing demands and turning their focus on Cloud services delivery. Inevitably, this has brought great challenges in the software engineering field, mostly for setting up a new software development environment that simplifies the procedure of building and hosting applications on the Cloud. Building Cloud aware applications needs a full review of available software processes considering the new Cloud architecture and associated capabilities. The transition from a conventional software development environment to the Cloud environment brings about several research problems that need to be addressed.

The target of this thesis is the introduction of models, approaches, and activities in the form of an integrated DSS to support the decision making in migration to the Cloud environment, as well as to tackle a series of challenges in the Cloud-based software development process. The proposed solutions are taking into account the management of the complex and multi-conflicting Cloud infrastructure and aim to shed light on the problem's insights with the utilization of computational intelligence techniques.

Computational intelligence approaches appear to have profound success when dealing with complex and multifaceted problems. In this work, various such techniques and models are utilized and their application in addressing the aforementioned research issues on the Cloud is investigated, while in some cases modifications and enhancements of the models used are proposed to improve their effectiveness and performance. A special reference is made to techniques based on Fuzzy Cognitive Maps (FCMs), Evolutionary Computation (EC) and Influence Diagrams (IDs), which seem to dominate the relevant literature yielding promising results.

1.2 Research Challenges

Despite the tremendous and drastic solutions proposed in recent years in the area of software engineering for a distributed environment, many related aspects remain challenging and unsolved problems. The research challenges, the unsolved problems as well the corresponding implications on the problem under study were determined through comprehensive literature reviews and surveys. Particular reference to the sources is given in the chapters where each challenge is addressed, while a brief overview of the challenges investigated in this thesis is given below.

1.2.1 Cloud Adoption

Despite all benefits and advantages that Cloud computing offers, many potential customers are still unwilling to proceed and adopt Cloud services. The Cloud environment is highly complex, comprising multiple and conflicting factors with in-commensurable units of measurements. These factors often impede adoption of Cloud services and each one separately constitutes a great challenge. It is quite important to properly identify and analyze them aiming to assist decision makers to take the correct decision regarding their transition to a Cloud environment. Proper analysis and assessment of current factors in a Cloud environment is vital not only for Cloud clients, but also for Cloud vendors since the results can turn their focus to those factors that may need to change in order to satisfy clients' concerns.

Nowadays CC is an established technology with a great level of maturity. Nevertheless, the adoption of CC from companies-customers is still a major challenge. Prospective customers need to consider the benefits and risks before proceeding with use and adoption of Cloud services, while on the other side CC vendors are trying to identify customers' concerns and barriers in order to adjust their offered services accordingly. A primary open research issue is the lack of efficient tools and approaches that are able to support decision making with respect to Cloud services adoption from both the customer's and the vendor's perspective. Another equally important open problem is the increase of Cloud reliability by enhancing the various key factors that influence Cloud adoption, such as security, performance, availability etc.

Aiming to address this challenge, the development and application of an integrated model based on FCMs and IDs are examined. The model construction process involves an extensive literature review, as well as the utilization of a group of experts. The suitability and effectiveness of these approaches are assessed on real-world problems. Both approaches appear to

tackle the problem successfully by delivering the correct estimation of the final decision. The multi-layered form of the MLFCM appears to be the best alternative, allowing a more detailed analysis of the problem being studied.

1.2.2 Microservices Architecture Migration

Microservices architecture are gaining more and more momentum as means for the development of applications as suites of small, autonomous and conversational services, which are then easy to understand, deploy and scale [3]. Migrating to microservices enables optimizing the autonomy, replaceability, decentralized governance and traceability of software architectures [4]. Despite the hype for microservices, both industry and academia still lack consensus on the definition of microservices and particularly in the identification of adequate conditions to embrace and benefit from this new paradigm. Most organizations and their on-premise application architectures are not ready to fully exploit the benefits of microservices, and adapting them to this environment is a non-trivial task [5].

Microservices technology is not a silver bullet, as it introduces new complexities to the system, while many factors should be considered to support the decision of adopting this architectural style [6]. Therefore, the study of the parameters and drivers forming the environment behind the decision of migrating to microservices is of paramount importance for different stakeholders.

A specially designed MLFCM model is introduced to support the decision of adopting the microservices architecture, which is constructed following a dedicated process. The model is formed through an extensive literature review followed by experts' feedback aiming at identifying the concepts and drivers related to the decision of adopting microservices architecture. A series of analyses and executions are then performed utlizing this model. With considerable success, the model proves its readiness to overcome bias, increase its performance and improve its explainability.

1.2.3 Component-Based to Microservices Architecture Migration

Once software development organizations decide to adopt the microservices architecture, their attention turns to migration-related challenges [7]. These challenges mainly concern the process of migrating a monolithic software system to one based on microservices. While monolithic applications constitute a large part of the software systems that are currently in

operation, an equally significant number of software systems are based on systematic reuse and more specifically on the component-based architecture.

The migration process from a component-based software system to microservices involves two critical steps, the decomposition of a software component into a set of independent services and the synthesis of selected microservices to substitute their functionality. The decomposition step requires that the component characteristics be identified at the finer level, including those of its individual functional parts. The microservice synthesis relies on locating and combining small functional service components in which their characteristics match those of the decomposed system and put them in the proper order to meet the characteristics and the requirements of the initial component.

This challenge is addressed with a series of tasks proposed to achieve the desired result. Both software components and available microservices are expressed in a semi-formal notation that helps the similarity identification with the use of a particular ontology scheme. A matching algorithm is then employed to introduce a list of candidate microservices ranked based on a specific score. Additionally, the proposed process is integrated with search-based techniques and recommends the optimal synthesis of microservices yielded by Multi-Objective Genetic Algorithms. The proposed process is evaluated through a two-stage experimental process and yields a successful performance in delivering proper solutions.

1.2.4 Resource Management on a Function as a Service Environment

Serverless computing has emerged as new Cloud processing paradigm that appears to be the ideal environment for the deployment of Cloud applications and services. Serverless platforms provide developers a sophisticated environment for the design and implementation of their applications, eliminating at the same time various operational concerns. Serverless model adoption has great impact on several software engineering aspects, such as the development process, pricing model and Quality of Service (QoS). Function as a Service (FaaS) is the leading representative of this new service architecture and is considered as the ideal environment to host and deploy microservices.

A developer can gain various benefits and advantages from the serverless computing adoption, like zero server management, no up-front provisioning, high availability, auto-scalability and an attractive pricing scheme. However, a number of weaknesses still exist and should be taken into account. QoS assurance and the Service Licence Agreement (SLA) satisfaction is essential

for the software development process and relies on an efficient resource management of the whole environment. The identification of the optimum scenario for resource allocation to serve adequately a specific workload is a tedious, computationally complex and time consuming process, since multiple objectives need to be satisfied.

Specific multi-objective genetic algorithms are employed to deal with this problem since it moves in a multi-objective environment. The applicability and the effectiveness of the selected MOGAs are assessed through an experimental process that enables the comparison between the solutions provided with the optimal solutions. The results show the success of the algorithms for approaching optimal solutions with high accuracy and thus demonstrate their ability to serve adequately the problem under study.

1.3 Thesis Structure

This thesis is organised as follows: Chapter 2 provides the theoretical and technical background around the topics of the thesis. Chapter 3 introduces an integrated analysis framework and a series of steps to extract useful static and dynamic information regarding Multi-Layer Fuzzy Cognitive Maps models. The investigation of applying integrated models in different forms to support the decision making process on the Cloud adoption research challenge is presented in Chapter 4. Following this, Chapter 5 proposes a novel Multi-Layer Fuzzy Cognitive Map computational process, which involves a new formulation and a new evolutionary algorithm which is integrated with the model. Chapter 6 introduces a specially designed Multi-layer Fuzzy Cognitive Map model to support the decision of adopting the microservices architecture. Next, Chapter 7 introduces a dedicated automatic process that supports the migration of software components to microservices. Chapter 8 investigates the application of Multi-objective Genetic Algorithms to tackle the resource management challenge in a serverless platform. Finally, the last chapter concludes the thesis by providing a summary for the work carried out, as well as by giving a number of directions for future work.

Chapter 2

Theoretical and Technical Background

To study the research issues addressed by this thesis, specific techniques and models have been selected where they will be used to provide solutions. The selection of these models is based on the success they have shown in dealing with similar problems. The problems under study, in addition to the increased complexity, are distinguished by uncertainty and conflicting intentions. This imposes the use of approaches with high-performance, high adaptability, and scalability as well as the ability to interact with the decision-makers.

This chapter introduces both the theoretical and technical background of the proposed research activities. Initially, a reference is made to all theoretical aspects related to the research challenges, followed by a reference to selected Computational Intelligence (CI) techniques. Specifically, a particular reference is made to Fuzzy cognitive Maps (FCMs), Influence Diagrams (IDs), and Genetic Algorithms (GAs).

2.1 Software Engineering for Distributed Systems Development

2.1.1 Software Engineering

The importance of Software Engineering (SE) is highlighted by the fact that the economies of all developed nations are dependent on software. SE is an engineering discipline that is concerned with the development and maintenance of software systems [8]. There are no universal solutions, methods and techniques for SE since different types of software require different approaches. Software engineers should be able to apply methods, theories and tools

where these are appropriate and always try to discover solutions to problems even when there are no applicable theories and methods. At the same time engineers should follow and work within organizational and financial constraints.

Software process is the systematic approach that is used in SE and consists of a number of activities which lead to the production of a software system. There are four fundamental software activities, namely specification, development, validation and evolution. Software specification activity is where customers and engineers define the software to be produced and the constraints to be met by its operations. Software development activity is where the software is designed and programmed. Software validation activity is where the software is checked to ensure that it is what the customers require. Finally, software evolution is the activity where the software is modified and adjusted to reflect changing customer and market requirements. As previously mentioned, there is no universal SE method or technique that is applicable for all types of software; however, there are fundamental ideas that affect all types. These ideas include managed software process, software dependability and security, requirements engineering and software reuse.

Since 1968 when the term "software engineering" was first proposed [9], a variety of new software engineering techniques and methods were developed and extensively used. During the years the software process was adjusted to comply with the available hardware infrastructure and software platforms. The development and evolution of the World Wide Web (WWW), as well the possibility of developing web enabled software, radically changed the software process approach.

2.1.2 Cloud Computing

Cloud Computing (CC) has been considered as the biggest evolution in the Information Technology (IT) industry that has already reshaped, redefined and redesigned Information Society. Offering powerful processing and storage resources with reduced cost and increased efficiency and performance, CC seems nowadays as a very attractive solution to a large group of cases, ranging from single users, to Small-Medium Enterprises (SMEs) and large organizations [10].

Among the many formal definitions that have been proposed [11], the US National Institute of Standards and Technology (NIST) [12] captured the most common agreed aspect and provided the most widely used definition: "Cloud computing is a model for enabling convenient, on

demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". The Cloud framework is composed of five essential characteristics, three service models and four deployment models. The five characteristics are on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The four deployment models involve private Clouds, community Clouds, public Clouds, and hybrid Clouds. The three service models offer Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

Back in 1969, Leonard Kleinrock [13], an American engineer and computer scientist who played an influential role in the development of the Advanced Research Projects Agency Network (ARPANET), the precursor to the internet, said: "As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of 'computer utilities' which, like present electric and telephone utilities, will service individual homes and offices across the country". This vision became a reality during the 21st century, while computing services are being offered and are available on demand similarly to other utility services.

Any resource can be hosted in CC [7], such as database services, virtual servers, service workflows, complex configurations etc. All these resources, regardless of their nature, are provided to clients via services, e.g. those offered by Amazon, Google, and Microsoft. In addition to services and resources, CC has two forms of providers, service and Cloud providers. A Cloud provider is the entity that offers and maintains hardware in the Cloud (Infrastructure as a Service) and at the same time may offer internally developed software services (Platform as a Service). A service provider is an entity that creates and maintains software services that are published in and run on Clouds (Platform as a Service and Software as a Service). Following the same concept, CC clients can be defined both as service providers and as end-users.

CC forced pre-existing models, like Service Oriented Computing (SOC) [14] and Model-Driven Engineering (MDE) [15], to be redesigned and reshaped, as well as boosted the emergence of new paradigms like Mobile Cloud Computing (MCC) [16, 17]. While CC has positive effects providing the IT community with many benefits and options, a number of risks, threats and vulnerabilities have to be considered and addressed properly [18].

2.1.3 Software Development and Operation on the Cloud

CC is an attractive business model where hardware, software, tools and applications can be leased out as a service over the Internet. This beneficial model offers many advantages like no capital expenditure, speed of application deployment, shorter time to market, lower cost of operation and easier maintenance of resources for the customers [19]. Due to these advantages CC forms an ideal platform for software development and deployment. Consequently, and under these increasing demands, software engineering has to be adapted with relevant activities adjusted on a distributed, shared and self-provisioning environment.

Radical changes brought about by the adoption of CC on software development may be summarized to: (a) New development environment which directly affects methods and techniques for software design, development, testing, deployment and evolution. (b) New architectural and execution infrastructure on which software systems are operating and interacting with a completely new user profile. (c) Users that utilize Cloud services can be defined as always-on users with increasing demands and a wide range of options for accessing services such as laptops, smartphones, tablets etc.

2.1.3.1 Microservice Architecture

Microservice Architecture is a relatively new software development approach that focuses on the creation and synthesis of small autonomous service modules. The most widely adopted definition of microservices architecture is "an approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API" [20]. As an architectural style, microservices overcome the problems of centralized, monolithic architectures [21], in which the application logic is encapsulated in big deployable chunks. In contrast, microservices foster the identification of small components built around business capabilities [22], that are easy to understand, deploy, and scale independently, even using different technological stacks [5].

Microservices share a similar definition with SOAP and RESTful services, that highlights the relationship between microservices and Service-oriented Architectures (SOA). Although microservices can be seen as an evolution of SOA, they are inherently different regarding sharing and reuse. SOA is built on the concept of fostering reuse, a share-as-much-as-possible architecture style, whereas microservices architecture is built on the concept of a share-as-little-as-possible architecture style [23]. Given that service reuse has often been less than

expected [24], instead of reusing existing microservices for new tasks or use cases, they should be "micro" enough to allow for rapidly developing a new one that can coexist, evolve or replace the previous one according to the business needs [4].

2.1.3.2 Serverless Computing - Function as a Service

Serverless computing introduces a new Cloud service which consists of an increasingly popular architecture for building distributed applications. This relatively new service constitutes a new processing paradigm or model, that emerged through the continuous and vast development of the Cloud; it provides a service in which developers can write and deploy code without provisioning or managing servers or containers. The adoption of this paradigm has great impact on several software engineering aspects, such as the development process, pricing model and Quality of Service (QoS) assurance.

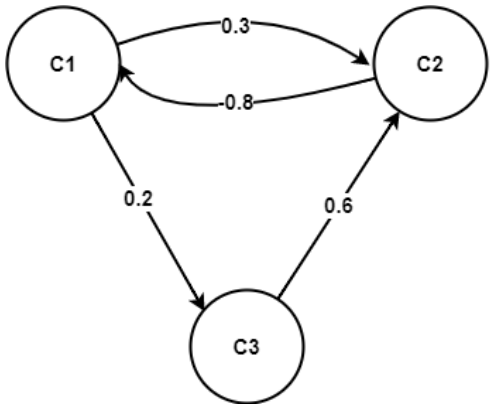
Serverless computing appear to excel other Cloud-based infrastructures by offering a number of advantages and benefits. Serverless architecture offers zero server management, no up-front provisioning, high availability, auto-scalability and pay only for the resources used which means reduced operation cost.

The main representative of this new service architecture is Function as a Service (FaaS) or event-based programming [25], where a function may be triggered through an API call, or by an event. Since Amazon introduced the Lambda serverless platform in late 2014 [26], many other Cloud providers adopted, offer and currently support this architecture. AWS Lambda [27], IBM Cloud Functions [28], Google Cloud Functions [29] and Microsoft Azure Functions [30] are currently the major serverless providers.

2.2 Computational Intelligent Techniques

2.2.1 Fuzzy Cognitive Maps

Fuzzy Cognitive Maps (FCMs) are computationally intelligent, soft computing tools that combine elements of fuzzy logic and neural networks [31–33]. In essence, a FCM is a directed graph with nodes that represent concepts in a domain and weighted edges that describe the various causal relationships that exist among these concepts, either positive or negative. The potential of FCMs is enhanced by the contribution of fuzzy logic which indicates both the type of representation of the causal relationships between the concepts and the strength of



	C1	C2	C3
C1	0	0.3	0.2
C2	-0.8	0	0
C3	0	0.6	0

	AL
C1	0.2
C2	0.6
C3	0.8

Figure 2.1: An example FCM with its weight matrix and initial activation levels (AL) vector.

presence of each concept within the modeled environment.

Causal relationships are defined by taking numerical values in the interval $[-1,+1]$. When a positive (negative) correlation exists between two concepts a positively (negatively) weighted arrow is drawn from the causing to the influenced concept. Two nodes without a direct link represent two concepts which are independent from each other.

The interrelations between concepts can be also expressed by a square matrix, the *weight matrix w*. A value $w_{ij} > 0$ means a positive interrelation between concepts C_i and C_j, that is, an increase or decrease of C_i value causes an increase or decrease of C_j value respectively. Inversely, when $w_{ij} < 0$ there is a negative interrelation between concepts C_i and C_j. Finally, if $w_{ij} = 0$ then there is no interrelation between concepts C_i and C_j. Figure 2.1 shows a sample FCM with three nodes and four edges. Naturally, the higher the number of nodes and relationships, the higher the complexity of the resulting map.

A numeric *activation level* AL (or *activation value*) per concept denotes the strength of its presence in the problem domain. Activation levels are defined as a vector that takes values in the interval $[[-1,1]]$ or $[[0,1]]$, depending on the modelling scheme followed. For the FCM in Figure 2.1 the activation levels for a sample scenario could be $AL = [[1, 0.6, 0]]$, where concept $C1 = 1$ is fully active for that scenario, $C2 = 0.6$ is somewhat active and $C3 = 0$ is not active.

The map is initialized with a set of activation levels which represent a particular situation or problem in hand, and then it is executed on a series of discrete steps. Equation 2.1 describes the update rule that calculates the total causal input for a node A_i at a given iteration $(t+1)$. This is how the activation level of node A_i is updated on that step, based on its value in the previous iteration A_i^t, and the influence it receives from all nodes A_j^t that are connected with it

(also known as feeders or sources).

$$A_i^{t+1} = f\left(\sum_{j=1, i\neq j}^{n} w_{ji} A_j^t + A_i^t\right) \tag{2.1}$$

After calculating the total causal input for a node, the updated activation value is decided according to a *transfer function* $f : R \rightarrow I$ which monotonically maps the total causal input R into the normalized range $I = (0, 1)$.

Different types of update functions have been proposed by the research community [34]. However, no rule exists that recommends the use of a specific function; this depends on the requirements of the decision-maker (in our case, as we will see later on, the person(s) in charge of deciding for microservices adoption) and the characteristics of the domain under analysis.

Similarly to ANN, four transfer functions are widely used in FCMs [35]: (a) sigmoid, (b) hyperbolic tangent, (c) step, and, (d) threshold linear. Generally, most of the studies that use FCMs for decision making use the unipolar sigmoid function, which exhibits the highest predictive capacity among all [36]. FCMs that use sigmoid functions are also called sigmoid FCMs.

$$f(x) = \frac{1}{1 + e^{-\lambda x}} \tag{2.2}$$

The slope of the sigmoid function determines how "sensitive" the model is to changes in the activation levels. It is defined by parameter λ in Equation 2.2, adjusted by the designer of the model through simulations and testing with example scenarios. For higher values of λ, the sigmoid function becomes more sensitive to changes in activation levels, and approaches output values close to the upper or lower bounds of the range.

The execution of the map (i.e., the iterative application of the transfer function over the concepts) concludes in one of three ways, as the model: (1) reaches an equilibrium state, (2) exhibits cyclic behavior, or (3) exhibits chaotic behavior. The former two cases are considered "stable", and allow one to make inference. The third case implies that the model is not suitable for analysis and should be revisited. The main outcome of the execution is the final activation value of the central concept of the model, which is then interpreted in the context of the problem.

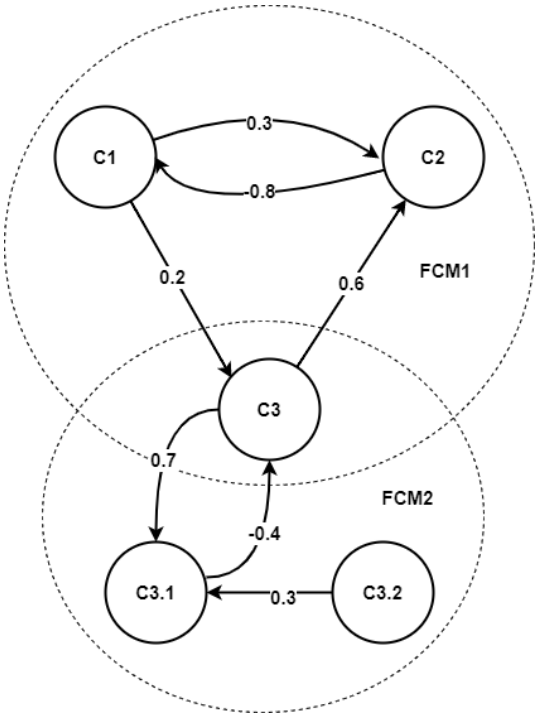


Figure 2.2: An example Multi-Layer FCM with two layers.

2.2.2 Multi-Layer Fuzzy Cognitive Maps

When the problem under study is highly complicated, it is often observed that some of the concepts in a FCM model are composed of other (sub)concepts that influence their activation levels. This issue may be tackled by means of Multi-Layer Fuzzy Cognitive Maps (MLFCM) which are able to handle the complexity introduced by multifaceted concepts [37].

A MLFCM (Figure 2.2) is basically a hierarchical tree structure, with the top-most level (root) including the most abstract or composite concepts, while the lower branches add more details by decomposing concepts of the immediate higher layer. In this way, several ”local” sub-FCMs are formed. The advantage of this structure is the creation of small and more easily manageable models (sub-FCMs), which co-work supporting the main, high level FCM, and at the same time it provides the level of detail required for fully capturing the dynamics of the problem under study.

In Figure 2.1, let us assume that concept C3 is too complex and that it can be further decomposed in C3.1 and C3.2 as shown in Figure 2.2. This generates a two-layer MLFCM with FCM1 at the root level and FCM2 at the leaf level.

Two different approaches have been proposed for traversing and executing MLFCMs. The

first approach follows the Depth-First Search (DFS) recursive algorithm that traverses a graph in a depth-wise motion. When the lower sub-FCM of a branch is visited (leaf), then this sub-FCM is executed and transfers its updated activation level to the upper FCM. The second approach [38] works similarly with the first approach with the difference being that each sub-FCM is executed for one iteration only. The multi-layered map is being traversed as many times as the number of iterations, and each connected node feeds its updated activation level back to the parent FCM during each iteration. The second approach is computationally more simple than the first one in maps where the connected nodes have many interactions in all layers involved. Nevertheless, the first approach is more sensitive to changes observed at lower levels which are immediately propagated upwards affecting all parent-FCMs in the chain to the root. This is the reason why this thesis adopted the first approach.

2.2.3 Influence Diagrams

In general, techniques that use Influence Diagrams (IDs) in modeling the decision process seem to improve the way the problem is approached by offering various benefits: IDs offer the flexibility of representing many dependencies among factors and manage to represent a highly complex problem in a human understandable way [39]. Also, they allow interaction with the experts through execution of the model using various input combinations thus enabling calibration of the model so as to achieve reasonable and helpful answers.

An ID may be conceived as a general, abstract, intuitive modeling tool that is nonetheless mathematically precise [40]. IDs are essentially directed graph networks, with different types of nodes representing uncertain quantities, decision variables, deterministic functions and value models. They were first developed in mid 1970s as a decision analysis tool to offer an intuitive way to identify and display the essential elements, including decisions, uncertainties, and objectives, and how these elements influence each other.

2.2.3.1 Generic Influence Diagrams

In general, an ID is a directed acyclic graph with three types of nodes and three types of arcs between nodes. The first is called *Decision* node, it is drawn as a rectangle and corresponds to some decision to be made. *Chance* or *Uncertainty* node is the second type, which is drawn as an oval and represents an uncertainty to be modeled. The third one is the *Value* node, which is drawn as a hexagon (or octagon) or diamond, and calculates all possible combinations

received from factors in the modeling environment that act as parent nodes. A *Functional* arc ends at a value node and represents the contribution of the node at its tail to the calculated value. The second type of arc is the *Conditional*, which ends at a chance node and indicates that the uncertainty at its head is probabilistically related to the node (oval) at its tail. Finally, an *Informational* arc ends at a decision node and indicates that the decision at its head is made according to the outcome of the node at its tail, which is known beforehand.

A simple example of an ID is presented in Figure 2.3, which represents a decision situation where a venture capitalist wants to know the gain on a prospective investment. The diagram includes a decision node, *Investment Decision*, two chance nodes, *Success of the Venture* and *Expert Forecast* and one value node, *Financial Gain*. It also includes two functional arcs, which end at *Financial Gain* indicating that the calculated value of the latter depends on *Success of the Venture* and *Investment Decision* nodes, and one conditional arc which ends at *Expert Forecast* node indicating that *Expert Forecast* and *Success of the Venture* nodes present a form of dependability.

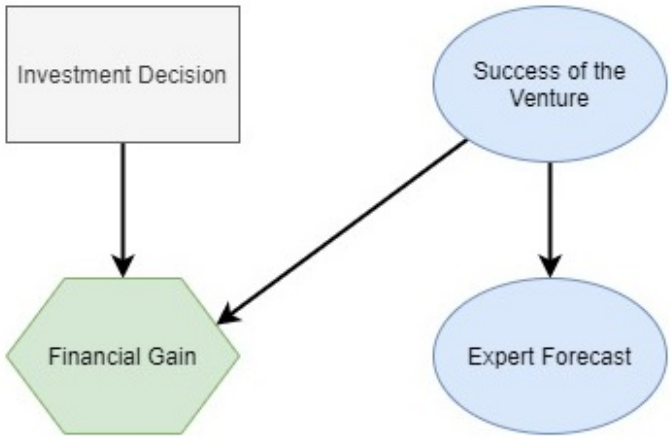


Figure 2.3: A simple Influence Diagram

The nodes of a system under modeling and the weighted arrows connecting these nodes are set to specific values based on the knowledge and beliefs of experts. Some nodes are more important than others, influencing more the value of their direct children nodes and less their indirect ones. The methods for evaluating and solving IDs are based on probabilities: The input and output values of a node in an ID are represented by probabilities. A common technique for evaluating and solving an ID is based on the Bayesian Theorem [41]. Initially, the possible values of each frontier node are defined, as well as the probability of occurrence

of any value. The same procedure is followed iteratively for the direct descendant nodes considering all possible combinations of values of their direct ancestors that yield their own value and then defining the probability of occurrence for each such value. This procedure is terminated when the value node is reached, the latter calculating all possible combinations received.

2.2.3.2 Fuzzy Influence Diagrams

Fuzzy Influence Diagrams (FID) were firstly proposed by [42] in an attempt to combine the features of IDs and the flexibility of Fuzzy Logic. The FID architecture is the same as that of a generic ID in terms of structure, except that it employs fuzzy reasoning instead of probabilities.

Let us assume that we have constructed an FID with l nodes, some of which have children and all these children are summed-up to n. Each of the $(l-n)$ nodes, which are considered the outer elements of the FID, that is, they receive directly the values from the environment, is associated with a fuzzificator F which converts the node's input values to fuzzy values via its membership function [42]. Then, for each of the children nodes there is a fuzzy set G_i which represents the way the current node influences the given child node and its membership function is selected from a set of sigmoid and bell-shaped (Gaussian) functions. In cases of multiple parents m, the influence of each parent node on a certain child is weighted by the set of the m scalars (weights) using a technique called *hedges*. The output to the ith child node is produced by the combination of F and G_i using Scalable Monotonic Chaining [43]. An example of scalable monotonic chaining methodology is shown in Figure 2.4 . This method is effective in cases where two fuzzy areas are correlated with a single analog or inductive inference rule. It can form an expected value, avoiding to fuzzify and defuzzify a fuzzy set, by using the transformation $G_i(F(input))$, where $input$ is the value received from the environment. Therefore, the input received by a children node is the average of the weighted sum of the set of transformations of its parents.

Initialization of the values of each frontier node and the determination of the weights for all nodes are the first steps that should be performed prior to executing the FID model. After initialization, the model is run and the values of the direct children nodes are calculated sequentially using the scalable monotonic chaining approach mentioned above.

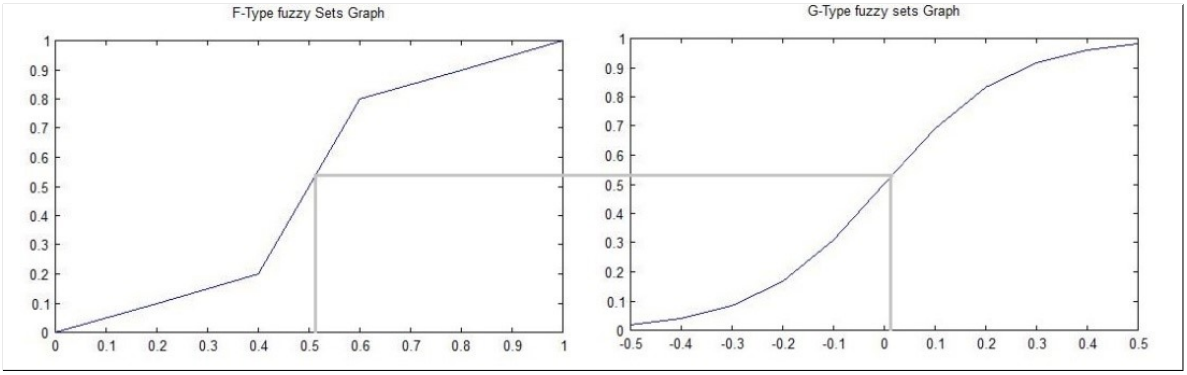


Figure 2.4: Scalable Monotonic Chaining example combining F-Type and G-Type fuzzy sets

2.2.4 Genetic Algorithms

In 1950s and 1960s the need for developing optimization tools to solve engineering problems guided scientists to study and develop evolutionary systems [44]. The main idea in all these techniques was based on evolutionary principles [45], i.e. to evolve a population of candidate solutions to a given problem, using operators inspired by genetic variation and natural selection. Various models of evolutionary computation have been proposed, usually referred to as Evolutionary Algorithms (EA). For many people evolutionary algorithms are synonymous with Genetic Algorithms (GAs), which are the dominant practice.

Algorithm 1 Genetic algorithm

1: Set $x_{min}, x_{max}, t = 0, p_c, p_m$
2: **for** each individual $i \in POP(t)$ **do**
3: $x_i = \text{rand}(x_{min}, x_{max})$
4: **while** maximum iterations reached or stopping criteria satisfied **do**
5: **for** each individual $i \in POP(t)$ **do**
6: Evaluate the position, x_i, of the individual using objective function $F(x_i)$
7: $POP'(t) = \text{select}(POP(t))$
8: $POP'(t) = \text{crossover}(POP'(t), p_c)$
9: $POP'(t) = \text{mutate}(POP'(t), p_m)$
10: Set $POP(t+1) = POP'(t)$
11: Set $t = t+1$

Genetic Algorithms (GA) were proposed as a new type of evolutionary algorithms by John Holland in 1975 [46]. In its simplest form a GA, as proposed by Holland (see Algorithm 1), is based on the notion of survival of the best (fittest) and has the following elements [44]:

populations of chromosomes, selection according to fitness, crossover to produce new offspring and a random mutation of new offspring. Chromosomes in a GA population are usually encoded into bit-strings and each position in the chromosome can take two possible values 0 and 1. A fitness function is required to assign a score (fitness) to each chromosome in the population reflecting how well each chromosome solves the problem under study. Based on their fitness, chromosomes are selected (selection process) for a subsequent genetic manipulation process which consists of two steps: Crossover operation is the first step that recombines parts taken from two selected chromosomes. During the second step, called mutation, parts at one or more (randomly selected) positions of chromosomes are altered. A new population of chromosomes is thus produced by this genetic manipulation and represents the new set of solutions to be evaluated under a repeatable generational process. The latter step is repeated until a termination condition has been reached.

2.2.4.1 Multi-Objective Genetic Algorithms

In the real world, a wide range of problems involves the simultaneous optimization of multiple objectives. These problems are called Multi-objective Optimization Problems (MOP). One of the most popular approaches that tackle such kind of optimization problems is the Multi-objective GAs (MOGAs) [47] which mainly constitute a modification type of the standard GAs. While single-objective GAs deliver a single optimal solution, the MOGAs end up with a set of optimal solutions, known as Pareto-optimal solutions. This set is also called the non-dominated solution set in the sense that no other solution in the search space is superior to them, considering all objectives simultaneously. Another important element is the Pareto-optimal front which is a boundary defined by the set of all points mapped from the Pareto-optimal set. The ultimate goal for a MOGA is to find a set of solutions as close to the Pareto-optimal front as possible. Equally important is the level of diversity of the resulting solutions, that is, the higher the diversity the better the solution set.

$$\begin{gathered} min/max \quad f(x) = [(f_1(x), f_2(x), ..., f_k(x)]^T, \quad k = 1, 2, ..., K \\ subject \quad to \quad x \in X \end{gathered} \tag{2.3}$$

In its simplest form the formulation of the MOP is represented by Equation 2.3. Integer k is the number of objectives and set X, is the feasible set of decision variables that is typically defined by some constraint functions.

Multi-objective optimization has been applied in many fields of science, engineering, eco-

nomics and logistics, where optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. Usually, in case of conflicting and competing objectives, no single solution exists that optimizes each objective simultaneously. The goal for a decision-maker is to use the Pareto set and then to select one or more single solutions that satisfy a particular objective.

In recent years along with the development of new algorithms for multi-objective optimization, a large number of performance indicators have been introduced to measure the quality of Pareto fronts approximations produced by these algorithms [48]. Two of the most widely used performance indicators were selected to be used to evaluate and compare the applied MOGAs during this thesis. The *Hypervolume (HV)* [49] and the *Inverted Generational Distance (IGD)* [50] quality indicators were employed to assist in comparing the MOGAs with respect to performance and scalability, given their ability to assess both convergence and diversity (uniformity and spread) of the algorithms. The HV indicator assesses the volume covered by the non-dominated solutions of a Pareto front in the objective space and therefore, the larger the volume covered by the solutions generated in a run, the higher the HV value, which indicates a better performance. The IGD indicator assesses how far the elements of the true Pareto front are from the non-dominated points of an approximation Pareto front and therefore, the greater the extent of the true Pareto front that is covered by the non-dominated points generated by a run in the objective space, the lower the IGD value, which denotes a better performance.

Chapter 3

A Framework for Analyzing Multi-Layer Fuzzy Cognitive Maps

3.1 Introduction

The fast growth of technology innovation, as well as the continuous technological developments and achievements, have resulted in an increase in complexity for systems and processes that aim to support them. No matter the scientific area, experts, analysts and decision makers in general, often face the inability to effectively and efficiently describe a given problem and thus study its parameters so that they take the proper decisions at the right time. This problem is usually associated with the fact that there exists a high number of intertwined parameters describing the underlying environment, which hinder an in depth description of the behavior under study and, therefore, make it very hard to study the circumstances behind a proper decision. Problems exhibiting a high variety of interacting factors, each with unknown strength of contribution to the formation of the general observed behavior, like for example the tendency of certain groups or cycles in social networks to grow or shrink, or the virality of videos in YouTube, or the forecasting of the outcome of peace negotiations in a war zone, necessitate the use of intelligent, as well as flexible models, that are capable of capturing the underlying dynamics of the problem and describe the complex interactions of the participating factors. One such category of models is the one that introduces FCMs.

As described in Chapter 2, MLFCMs are an extended form of FCMs that introduce the concept of sub-FCMs, that is, smaller structures (maps) of nodes organized in layers, with concepts being grouped together in neighborhoods so as to focus on specific aspects of the same

environment under modeling. This grouping offers a way for analyzing parameters at finer levels of granularity [37] [38] [51]. Therefore, a modeler or decision maker may represent all possible parameters present in a given problem using a parent FCM, then decompose and analyze them into finer details through the creation of child FCMs, and finally study their behavior in conjunction with the general outcome of the parent map. This enables tracking the evolution of the outcome down to the last detailed parameter and studying the associated causes and effects.

Models like MLFCMs work in discrete time steps during which numerical calculations are carried out at different layers. The problem faced with such models is the lack of a systematic and disciplined way of analyzing the model structure and its effect on the determination of its outcome. Due to the high level of complexity introduced by the various layers on one hand, and the number and type of interconnections between nodes in each layer on the other, the analysis of properties like stability, node influence and centrality, or convergence to specific ranges of values, is often a very tedious task. Such kind of analysis, though, becomes very useful when seeking for advanced decision support information, like the discovery of the leading determinants of the model's output. This may lead to strategy formulation or modification so as to lower their impact in case this leads to undesirable effects, or promote the influence of other parameters that may be considered beneficiary. In such cases a what-if analysis is facilitated, offering the ability to simulate hypothetical scenarios and examine their outcome. For example, if we take a MLFCM model representing a cloud service environment, and assume that the node representing the pricing scheme has been characterized through the specialized analysis as one of the stronger parameters that affect the decision of a customer to buy this service or not, then the service provider may decide to change its pricing policy in general so that they attract more customers in the future.

Even though MLFCM modeling offers a visual representation of a problem exhibiting highly complexity, the "observation" of the map itself usually offers very little to the human eye, meaning that its analysis is a quite tedious task. Static and dynamic analysis of the map can help modelers to sense the system better and assess its modeling representation from different perspectives in an attempt to reveal various findings regarding its shape and behavior. This will ultimately lead to a better understanding of the constructed map by bringing to light "hidden" properties and features, as well as by highlighting points that require particular attention. This section is devoted to elaborating on the aforementioned types of FCM analysis and to proposing specific steps that will enable collection of useful information about the model.

In the above context, the present chapter introduces a framework for conducting static and

dynamic analysis of MLFCMs. The former addresses issues like model balance and stability, and calculates different indices for individual nodes and sub-FCMs. The latter analysis investigates how the different layers work together in an intra- and inter-connectivity manner, and suggests ways to study the influence that different nodes exercise on the central node of interest at run-time (dynamically).

3.2 Literature Overview

Despite the big volume of research studies that discuss applications of FCMs, extensions and methodologies, very little research has been devoted to the analysis of FCMs, both static and dynamic [52], the latter being applied individually to the problem under study. Very few and isolated research studies focus on some types of dynamic and static analyses, but these are confined to revealing certain factors of interest: In [53], a form of static analysis is applied which includes the identification of cycles to uncover nontrivial relationships between concepts, the calculation of the model density to obtain an indication of its complexity, and the analysis of importance of individual nodes. In [54] dynamic analysis was performed with simulations using different initial conditions on concepts, and offered a description of the behavior of the model that can be used to support decision making. In [55], the authors performed an experimental evaluation in the form of dynamic analysis to assess the effectiveness of an intuitionistic FCM model for medical decision making.

Based on the above, it becomes evident that currently there is no framework for analyzing the structure and behavior of FCMs. The constantly growing rate of application of FCM models, as well as the increase in complexity of the problems studied, highlight the need for such a framework to help and guide modelers in their effort for better understanding the behavior of their systems and performing the necessary reconfiguration for devising more appropriate models. This is exactly the subject of the present work, that is, to offer a simple and effective framework for performing such type of FCM analyses. More to that, this work focuses on a new construction process of MLFCMs, which works exactly the same way as in simple FCMs but offer the ability to decompose multifaceted concepts and study the factors that determine the model's output in more details. Therefore, the proposed model may be applied on all types of FCMs, and, additionally, it will be explained how to take into consideration the layered form of MLFCMs so as to offer information that will explain the internal dynamics of the interacting nodes at each layer.

3.3 Static Analysis

Static analysis examines a model's properties irrespectively of its behavior over time, that is, the model is examined without execution. Among the different advantages that FCMs offer in modeling complex problems, the capability of analyzing such models, which stems from its graph-based representation, is of utmost importance. Approaches, methods and metrics from the area of Graph Theory [56] can be applied and the results may be used for a quantitative static analysis of the map.

Graph Theory is a branch of Mathematics that deals with the formal description and analysis of graphs, which are defined as representations of a set of nodes linked by edges. Graphs are being used for graphical representation of real-world systems providing a description of a system's elements and their interactions. Some basic concepts and definitions from Graph Theory are quoted below, with their mathematical description as originally defined in literature [56], as well as the corresponding notion or information extracted that is useful for studying our model. All concepts and definitions thereafter are based on the broad consideration of a FCM as a directed weighted graph.

A *directed weighted graph* is defined as a graph $G = (V, E)$ where V is a set of vertices and E is e set of directed edges between the vertices $E = \{(u, v)|u, v \in V\}$ associated with a weight function $w : E \rightarrow R$. Edge weight w_{ij} between node i and j represents the value of the connection from i to j.

Graph density is an indication of the complexity of the map and is defined as the ratio between the number of edges in the map and the maximum number of edges the map can have.

Several *centrality* indicators may be used to identify the level of significance of a node in a graph. *Degree centrality* for directed networks shows that an important node is involved in a large number of interactions. Each node has two different degrees, the *in-degree*, $deg_{in}(i)$ and the *out-degree*, $deg_{out}(i)$, which correspond to the number of incoming to and outgoing edges from node i respectively. Since a FCM is a directed weighted graph, in addition to node degrees and following the same rationale, node value indicators will also be used in this work.

Having introduced the basic notions we will now proceed with describing the parameters and notations that are used in the static analysis of MLFCM models by dividing them into three major categories:

A. Complexity

This category deals with the characterization of the model in terms of complexity of its

structure. The metrics used assess the density, depth and breadth of the MLFCM graph. Specifically, density is given by the following formula:

$$D = \frac{|E|}{|V|(|V|-1)} \tag{3.1}$$

Therefore, density defines how rich a MLFCM model is in terms of nodes and interactions. Furthermore, due to the multi-layer nature and the decomposition property of MLFCMs, complexity is also investigated from the perspective of the sub-FCMs. More specifically, the structure of a MLFCM model is analyzed by measuring its depth, that is, the maximum number of layers from the root to the leaves, as well as its breadth (see Eq 3.5), the latter assessing the level of decomposition that takes place in the model. Decomposition is measured with dec_deg, which shows the number of sub-FCMs present and br_deg, which averages the number of nodes in these sub-models.

Let us assume that C denotes the set of decomposed nodes and FC the set of the sub-FCMs produced, both amounting the same number of elements:

$$C = \{decomposed\ nodes\} = \{c_1, c_2, ..., c_n\}, where\ C \subseteq V \tag{3.2}$$

$$FC = \{sub-FCMs\} = \{fc_1, fc_2, ..., fc_n\} \tag{3.3}$$

$$dec\_deg = |C| = |FC| \tag{3.4}$$

Then, the breadth degree and the depth of the MLFCM are given as follows:

$$br\_deg = \frac{1}{n}\sum_{i=1}^{n} |fc_i| \tag{3.5}$$

$$depth = \sum_{i,j=1, i\neq j}^{deg\_deg} l_{i,j},\ l_{i,j} = \begin{cases} 1,\ if\ \exists\ e_{j,i} \in E,\ where\ C_i \in fc_i\ AND \in fc_j \\ 0, otherwise \end{cases} \tag{3.6}$$

B. Strength

In this category the aim is to gather information that will enable the characterization of the significance of each node in the MLFCM. Therefore, all metrics and indicators target to reveal how strong the presence of each concept in the MLFCM is at all layers and sub-models.

In-value is the sum of the weights of all incoming edges to node i and is denoted by

$$val_{in}(i) = \sum_{j=1}^{|V|} |w_{ji}| \tag{3.7}$$

Out-value is the sum weight of all outgoing edges from node *i* and is given by

$$val_{out}(i) = \sum_{j=1}^{|V|} |w_{ij}| \tag{3.8}$$

The total value of a node is defined by

$$val_{tot}(i) = val_{in}(i) + val_{out}(i) \tag{3.9}$$

Using the above we may calculate the node in-value over maximum in-value with $\frac{val_{in}(i)}{\sum_{w \in E} w}$, the node $out - value$ over maximum $out - value$ with $\frac{val_{out}(i)}{\sum_{w \in E} w}$ and the node $sum - value$ over *maximum sum-value* with $\frac{val_{tot}(i)}{\sum_{w \in E} w}$.

Similarly, the *degree* of a node denotes its significance based on the number of concepts it interacts with (is affected by and it affects). To account for this, node *in-degree*, node *out-degree* and node *total-degree* metrics are calculated:

$$deg_{in}(i) = \sum \delta_j,\ \delta_j = \frac{1,\ w_{j,i} \neq 0}{0,\ w_{j,i} = 0} \tag{3.10}$$

$$deg_{out}(i) = \sum \delta_j,\ \delta_j = \frac{1,\ w_{i,j} \neq 0}{0,\ w_{i,j} = 0} \tag{3.11}$$

$$deg_{tot}(i) = deg_{in}(i) + deg_{out}(i) \tag{3.12}$$

Continuing, we may calculate the node $in - degree$ over *maximum in-degree* $\frac{deg_{in}(i)}{|V|-1}$, the node $out - degree$ over *maximum out-degree* with $\frac{deg_{out}(i)}{|V|-1}$ and the node $total - degree$ over *maximum total-degree* with $\frac{deg_{tot}(i)}{2(|V|-1)}$.

C. Tendency

According to graph theory, a cycled or closed walk is a sequence of vertices that are traversed through connecting edges consistently with their direction; a cycle starts and ends at the same vertex.

The number of feedback cycles in a graph is considered as a strong indicator for the tendency of the map. A positive cycle acts as an amplifier of any initial change, leading to a constant increase of the activation at the end of that cycle. Conversely, a negative cycle reduces any initial change leading to a constant decrease of the activation at the end of the cycle. Therefore, it is important to calculate the number of positive and negative cycles present in the map so that we can infer the tendency of the model towards increasing or decreasing the initial stimuli. Following the same reasoning, it

is equally important to investigate the number of cycles in which a node takes part, negative and positive, so that we can characterize the importance of that particular node in the definition of the tendency of the model.

All tendency indicators are calculated at every layer of the MLFCM, where the discrete cycles present in a certain sub-model are studied independently. Also, the effect of their grouping is assessed in terms of cycles gradually as we move from leaf-FCMs to the upper layers until we reach the root FCM.

Summarizing the above, static analysis of MLFCMs may be performed using the metrics described previously, aiming at revealing certain characteristics that belong to one of the three categories, complexity, strength and tendency. The sequence of their application is not important as there is no formal dependence between them. What we should note, though, is that the MLFCM structure is intrinsically more complex than simple graphs or FCMs, and most of the aforementioned parameters should be applied and measured on each sub-FCM separately. The significance of the nodes on each sub-FCM individually, as well as the different characteristics of the sub-FCMs, provide guidance for further analysis of the model in conjunction with the map's topology. Particular attention should be given to nodes which transfer their activation levels from $Layer_n(child)$ to $Layer_{n-1}(parent)$.

By the end of the static analysis that is applied across the model, modelers shall be able to (i) identify the strong concepts of each sub-FCM and how each of them influences the central concept of the map and to what degree, (ii) reap a strong indication of how each sub-FCM influences the sub-FCM of the upper layer, and, (iii) use indications from static analysis towards setting dynamic analysis simulations.

3.4 Dynamic Analysis

Additional observations regarding the model's behavior could be extracted by applying dynamic analysis. More specifically, in this type of analysis the model is executed and its behavior is investigated. The main target here is the study of the activation levels of the participating concepts (nodes) and how these values change over time. A number of simulations are required under specific and targeted rules that can lead to infer some interesting behavioral properties and to reach to some conclusions.

Before moving, though, to presenting the approach suggested for conducting dynamic analysis, it is imperative to describe a constraint which should be satisfied in order for this type

of analysis to have meaning. This constraint essentially reflects the result of preliminary assessment of the model in terms of correctness and accuracy. More specifically, when starting to investigating a ML-FCM model, what we initially have is a representation of the problem under study, which was derived usually with the aid of domain experts who defined the concepts and their causal relationships. In the absence of any observed behavior, that is, of historical data describing the actual behavior of the corresponding parameters influencing each other similarly to what that model attempts to describe, we must first assess whether this model behaves as expected to. Therefore, we construct and execute two scenarios we call "extreme"; one with extremely favorable parameters which should drive the model to the extreme positive outcome (i.e. the concept of interest to a value close or equal to +1), and one to the extreme negative (i.e. to a value close or equal to 0). Both scenarios should present the anticipated output in equilibrium conditions. If either of the two scenarios yields an outcome different from what is expected, or the model does not present equilibrium (fixed point) conditions, then the suitability of the structure of that particular model is questioned and further dynamic analysis cannot be performed. In such a case the modelers should return back to the "drawing" table and their experts, and try to revise the model so that it passes successfully the extreme scenarios evaluation. Once it does, the various steps of dynamic analysis that are described in the rest of this section may be taken.

The present work suggests two approaches as regards the dynamic analysis of MLFCM models:

Firstly, a number of simulations is performed with randomized initial activation levels, followed by a study of the correlation between the initial activation level of each concept and the final activation level of the central concept of interest. This type of performance investigation is expected to build upon the findings derived from the static analysis with respect to the evaluation and ranking of the concepts' significance.

Secondly, the performance of the model is assessed on specific what-if scenarios, which are built based on the findings of the preceding static analysis, as well as on the characteristics of the problem under modelling. One such scenario is to investigate whether the n strongest nodes suggested by the corresponding indicators of the static analysis may determine the model's decision irrespective of, or with very limited contribution by, the values of the rest of the concepts. In the same context, simulations may be used to identify the contribution of each sub-FCM to the final decision, with focus on the decomposed (transfer) node. If the latter is again characterized as strong by the findings of the dynamic analysis, and hence it is regarded as a leading determinant of the model's output, then the contribution of the effect

the decomposed parts exercise on this node should be further analyzed and assessed. The suggested way to do this is to perform a bottom-up analysis where the values for all initial activation levels of the map are kept constant and the corresponding activation level values of "child" nodes on a specific sub-FCM at the lower layer are varied. The process should start from the leaves and working all the way up the modeler will be able to trace the degree of influence to the upper layers and identify if this declines or strengthens.

It should be noted that for each what-if scenario the definition of the initial values of the concepts should reflect the settings of the problem at the time of execution. In our demonstration example for the Cloud Adoption problem, which will be described later on, such a case would be for instance the current situation of a client that wishes to move its transactions to a Cloud service provider describing notions of concern like security, cost and performance. Similarly, each scenario will reflect the given situation under study, including hypothetical settings which will enable us to study the future behavior of various cases.

3.5 Stepwise Analysis and Inference Process

The stepwise process to be followed in the proposed framework is as follows:

Phase A' - Static Analysis:

Apply the metrics defined in the three categories of parameters, complexity, strength and tendency, in any order of sequence. Collect the corresponding measurements and analyze the model as follows: First identify the complexity of the sub-FCMs and the root-FCM. High magnitude of complexity makes a sub-FCM at a certain layer a good candidate for revision so that this magnitude is lowered. This will be addressed by simulation during dynamic analysis and discovery of unnecessary relationships and/or concepts. Second, rank the concepts in ascending order of significance/strength using the *val* and *deg* indicators. Specific simulations may then be performed focusing on the effect of the strong and significant concepts (nodes) of interest at different layers of the MLFCM. Third, calculate the number of possible cycles, both negative and positive, that exist in a specific sub-FCM. Assess the tendency of each sub-FCM and relate this tendency with the existence of strong and decisive nodes. Finally, evaluate the contribution of the type of cycles present, in conjunction with strong nodes, to the formation of the model's final output. The latter will enable understanding the internal computational dynamics of the MLFCM structure and will lead to indicating further simulations that will identify how the output of the model is actually formed, thus contributing to eliminating the effect of the black-box type of execution that FCMs are often criticized for.

Phase B' - Dynamic Analysis:

First set-up and execute the two extreme scenarios. Examine their outcome and analyze the conditions of stabilization (equilibrium). If the results of the model agree with each scenario's expected value at equilibrium, then proceed to the execution of various simulation scenarios. If not, revise the initial model accordingly and repeat the process.

In case the behavior of the model is consistent with the extreme scenarios, take the significant nodes for each sub-FCM and execute simulations to derive how its central node that transfers its activation to the upper layer is affected by its neighborhood. Repeat this working from bottom to top until you reach the root FCM where this process is repeated for the node that yields the final output of the model. Conduct as many what-if scenarios as required to understand fully which node or combinations of nodes primarily define the value of the node of interest, as well as perform an investigation of how the weights of the causalities contribute to this definition. Based on the above, there will be two different types of inference/actions associated with the corresponding findings:

Inference/action set A': Check whether the original model as defined by the domain experts is actually behaving as expected to or not, and if not, identify the concepts and/or causal relationships that disturb this intended behavior. If yes, define what the internal dynamics that determine the model's behavior are. Therefore, the modeler may thus calibrate or modify the model to reflect better the dynamics of the problem under investigation.

Inference/action set B': Some nodes, or groups of nodes, may have negligible effect on the behavior of the map (e.g. on the central node of interest) so the question posed to the modeler is whether to keep these nodes or not. In the latter case, removal of redundant nodes and/or causal relationships is highly desirable as it decreases the overall complexity of the model, as well as the associated computational burden (e.g. initializations, nodes' updating, discrete sub-FCM executions, etc.) without compromising accuracy and performance.

3.6 Summary

The simplicity and effectiveness of Fuzzy Cognitive Maps have turned them nowadays to a useful tool for modeling real-world problems and supporting the decision making process. Multi-layer FCM structures offer the ability to approach a problem from every relevant

perspective by decomposing multi-faceted parameters and forming neighborhoods of concepts at any level of detail or granularity. One of the main issues when constructing and executing such complex models is to understand the internal dynamics behind their execution so that optimization of their structure can be performed, but most importantly, inferences regarding which node or groups of nodes determine the final outcome can be made. Currently there is lack of methodologies for understanding how FCM-based models work. In this respect the work in this chapter addressed the issue of the analysis, both static and dynamic, of MLFCM models by proposing a framework for conducting a series of steps that aim to reveal the hidden properties of their execution.

The proposed framework was divided into two approaches: The first studied the MLFCM as a graph model extracting information about its complexity, the significance of the discrete nodes at every layer and its tendency to promote or inhibit an initial activation as a result of the presence of a number of positive and negative cycles. The second approach analyzed the run-time behavior of the model and provides insights regarding its behavior in terms of correctness, correlations between nodes and the effect of the latter to the final outcome yielded. Executions of different simulations and what-if scenarios enabled the revision and restructuring of the model taking into account issues like complexity and computational burden.

Chapter 4

Modeling the Cloud Adoption Decision

4.1 Introduction

CC has been considered as the next big thing in the IT field that is transforming the whole perspective with which we understand computing today. Offering powerful processing and storage resources with reduced cost and increased efficiency and performance, CC seems nowadays as a very attractive solution to a large group of cases, ranging from single users, to SMEs and large organizations. Especially for business organizations that produce and process considerable volumes of information on a daily basis during their working activities, Cloud adoption is still a major challenge. The ever-growing trend from the industry towards adopting Cloud computing has led many of the major software developers or service providers to turn their strategy towards Cloud services, mostly targeting at increasing their market share. On one hand, companies-customers need to consider the benefits, risks and effects of Cloud computing on their organization in order to proceed with adopting and using such services, and on the other, Cloud Computing providers need to be fully aware of customers' concerns and to understand their needs so that they can adjust and fit their services accordingly.

During the last decade the research community has focused on the field of Cloud computing with increasing interest. Nevertheless, a quick review of the relevant literature suggests that there are yet no mature techniques or toolkits to support decision making as regards the adoption of Cloud services. Cloud adoption in general depends on multiple, conflicting factors which introduce high levels of ambiguity and uncertainty in the decision process, making it a highly complex task that cannot be tackled with classical and linear methods. The rapid changes in the Cloud computing environment, both at the supply and the demand level, reveal

how difficult it may be for any model to assist the relevant decision making process timely and correctly. This statement imposes that a framework or model to support the study of Cloud computing adoption should be flexible enough and dynamically adaptable to accommodate these continuous shifts.

The work in this chapter aims to deliver integrated models in different forms to support the decision making process on the Cloud adoption research challenge. More specifically, the application of two methodologies are being examined based on FCMs [57] and IDs [58]. The development of all proposed models is based on the analysis of information that was collected and analyzed in a systematic manner: Firstly, a study was performed of the most recent and relevant literature on Cloud computing and particularly on Cloud adoption, through which all possible factors that influence the final Cloud adoption decision were identified. The results of this study led to the next steps which include the categorization of factors, as well as the building and distribution of a questionnaire to a group of experts. This questionnaire targeted to utilize their knowledge and expertise for approving the list of factors already identified. Furthermore, the experts were asked to define the relation of each factor to Cloud adoption and assign a corresponding weight following a Likert scale. Although the development of all models followed the same rationale as described above, in each case the process was repeated and adapted to the characteristics of each model. In addition, the different timing of the implementation of the two models is also reflected in the outputs, such as the number of nodes used, their categorization, etc.

The constructed models were used to answer the question "Adopt Cloud Services or Not?" under the current state of the offered service and the associated factors describing each customer's particular situation at the moment of decision. Moreover, the benefits and capabilities provided by the proposed approaches are fully exploited during the simulation analysis which is based on targeted what-if scenarios. Particular and extended reference is made to the application of the step-wise analysis and inference process in the Multi-layer approach of the FCM models.

4.2 Literature Overview

An investigation of the current literature revealed a relatively small number of papers discussing Cloud adoption from the perspective of decision making and also current feasibility approaches fall short in terms of decision making to determine the right decision. We introduce a summary of these studies, examining the contribution of each work to the decision making problem.

In [59] a Cloud adoption toolkit is presented which provides a framework to support decision makers in identifying their concerns and match them with the appropriate techniques that can be used to address them. In [60] various issues are examined that impede rapid adoption of Cloud computing such as cost, compliance and performance. The authors in [61] attempted to contribute to the development of an explorative model that extends the practical applications of combining Technology Acceptance Model (TAM) related theories, with additional essential constructs such as marketing effort, security and trust, in order to provide a useful framework for decision makers to assess the issue of SaaS adoption and for SaaS providers to become sensitive to the needs of users. Wu [62] explores the significant factors affecting the adoption of SaaS by proposing an analytical framework containing two approaches: TAM related theories and the Rough Set Theory (RST) data mining. In [63], a solution framework is proposed that employs a modified approach named DEMATEL [64] to cluster a number of criteria (perceived benefits and perceived risks) into a cause group and an effect group, respectively, presenting also a successful case study. Even though all of the above techniques contribute a significant piece to this new open research field, they may be classified as "traditional", single layer approaches, which examine only a specific part of the problem.

A framework called CloudGenius is proposed in [65], which automates the decision-making process based on a model that includes factors specifically for Web server migration to the Cloud. This framework stands over a well-known multi-criteria technique, namely the Analytic Hierarchy Process, to automate the selection process based on a model, factors and QoS parameters related to an application. Aiming to help companies to analyze several characteristics regarding IT resources and identify their favor-ability in the migration to the Cloud, a general ROI model is proposed in [66]. Two decision support tools for Cloud migration in the enterprise are described in [67]. The first is a modeling tool that produces cost estimates of using public IaaS and can be used to compare the cost of different Cloud providers, deployment options and usage scenarios. The second tool is a spreadsheet which outlines the benefits and risks of using IaaS from an enterprise perspective and provides a starting point for risk assessment. In [68] a Cloud adoption toolkit is presented, which provides a framework to support decision makers in identifying their concerns and matching them with the appropriate techniques that can be used to address them. Five techniques have been incorporated to support the process: Technology Suitability Analysis, Energy Consumption Analysis, Stakeholder Impact Analysis, Responsibility Modeling and Cost Modeling. CloudDSF, which was proposed and presented in [69], is a framework in which knowledge about the problem domain (i.e. migration to the Cloud) is gathered, organized, visualized and offered as a publicly available

Web application [70]. Finally, in [71] CloudStep describes a step-by-step decision process that consists of nine activities including enterprise, legacy application and Cloud provider profiling, constraint identification analysis and alternative migration scenarios evaluation and ranking.

4.3 ID Modeling

The work in this section proposes two approaches based on IDs, generic and fuzzy-based, which are able to provide a successful model to support decision making for Cloud adoption. Two methods were employed to gather the necessary information for modeling the Cloud adoption decision-making process: (i) Literature study and (ii) Collection of expert opinion through specially prepared questionnaires followed by interviews. More specifically, a small-scale literature review on the subject was conducted in order to identify a number of factors that potentially influence such a decision and therefore be considered as nodes in our models. A group of three experts with strongly related background to the subject was identified in the next step (i.e. key personnel in Cloud providers). An initial list of factors was then prepared and the experts were asked to evaluate the list and prompted to add or remove factors based on their expertise and working experience. A last round of discussion with the experts was conducted, in order to finalize the list of factors which were used to form the nodes of the ID model. The factors identified are listed in Table 1.

The relationships between the nodes were defined using again the two-stage approach: Firstly, by reviewing the relevant literature and identifying dependencies an initial model was developed. Then, this model was verified, corrected and modified after consultation with the experts. Considering the influencing factors that were extracted, as well as the limitations posed by the combinatorial form of the generic model, the ID model was formed as shown in Figure 4.1. The corresponding factors were grouped in such a way so as to represent better the problem under study. The proposed representation may be characterized as relatively simple with a two-level structure. All factors extracted are formed as frontier nodes in the diagram. Groups of those factors form (and influence) intermediary nodes, which, in turn, influence the unique *Value* node.

After the model form was finalised, both ID approaches were then validated in terms of expected performance. More specifically, two synthetic (hypothetical but realistic) scenarios were created representing the so called “extreme cases”, that is, a certain situation where everything would be in favor of Cloud adoption (positive scenario) and another one where the opposite would hold (negative scenario). The target was to assess the verdict of the evaluation

Table 4.1: Factors influencing Cloud adoption

Name	**Definition**
Legal Issues	Cloud adoption compliance with all legislative issues. Ability to adjust when legal requirements grow.
Availability	The amount of time that Cloud Service(s) is operating as the percentage of total time it should be operating.
Security	Security of service: data transfer, data stores, web servers, web browsers.
Cost / Pricing	Operational - running costs, migration costs etc. Cost benefits from Cloud adoption.
ROI	Return on Investment.
Compliance	Business and Regulatory compliance.
Performance/Processing	Does Cloud adoption perform the process to the desired quality?
Scalability	Ability to meet an increasing workload requirement by incrementally adding a proportional amount of resources capacity.
Privacy/ Confidentiality	Privacy and confidentiality coverage.
Elasticity	Ability to commission or decommission resource capacity on the fly.
Data Access / Import-Export	Access to data in various ways.
Technology Suitability	Does Cloud technology exhibit the appropriate technological characteristics to support the proposed SaaS?
Hardware Access	Degree of Cloud Service accessibility, on local hardware.
Audit ability	Ability of Cloud service to provide access and ways for audit.

node of the models under known situations and check whether their outputs indicated that the model behaves correctly. Finally, the models were tested on the three real-world scenarios that are listed in Table 4.2, that is, cases collected from three international Cloud services providers and were related to real customers. A series of interviews were conducted with the Cloud providers and the customers so as to be able to retrieve the leaf node values for each case separately. The values recorded and adjusted essentially reflect the state of the offered service and the associated factors describing each customer's particular situation at the moment of decision. The first case involved an academic institution of a medium to large size, which requested a comprehensive solution for email services. The second case described an industrial organization which requested a complete email Cloud package and also a Cloud based document management system. Finally, the third case was about a medium insurance broker organization, which requested a complete email Cloud package and also a Cloud infrastructure to fit a heavy tailored-made owned system.

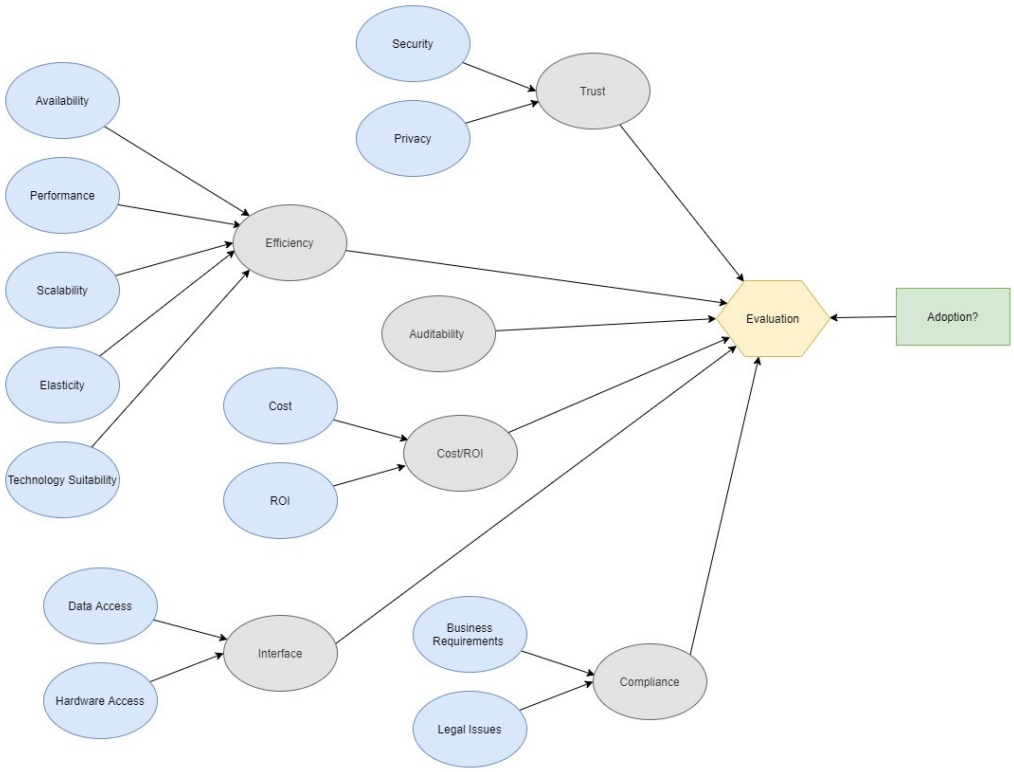


Figure 4.1: ”Adopt Cloud or Not” Influence diagram

Table 4.2: Brief description of real-world cases.

Case	Line of Business	Users	Cloud Services
A	Academic Institution	4500	Mail Server / Mailbox / Mail client
B	Supplies Industry	85	Mail Server / Mailbox / Mail Client / Document Management
C	Insurance Brokers	125	Mail Server / Mailbox / Mail Client / Custom Business System

4.3.1 Generic ID

Initially, all possible values of each frontier node were defined. These values were determined by first considering that all nodes, including the Value node, can be set to three different linguistic value levels, *Low*, *Medium* and *High*. Next, the probability of realization of each level value for each node-factor was defined. Table 4.3 presents the input values that describe the current situation at the point of time when the decision was about to be made for each of the five scenarios tested. Following the procedure described above, we defined the values for each direct ancestor node considering all possible combinations of values of all direct

descendant nodes, including the values of the current node. We should mention here that for the construction of this model, a total of 3024 combinations needed to be calculated, something which proves the computational burden associated with this form of ID modeling.

At this point a brief presentation will be made of how the ID calculations are performed using an indicative part of the diagram containing *Privacy*, *Security* and *Trust* nodes. In order to evaluate probabilities on the Trust node using the Bayesian Theorem [41], a number of calculations (combinations) must be performed as shown in Table 4.4. In these calculations the numerical values of 0.9 for High, 0.5 for Medium and 0.1 for Low were used , as well as the specific weight of contribution of each leaf node (*Security*, *Privacy*) to its parent (*Trust*). Each combination yields the numerical value of the probability P that the value of the *Trust* node is $High$ (and consequently $1 - P$ for Low).

4.3.1.1 Experimental Results

After the execution of the positive scenario, the evaluation node yielded the value of 0.77 for "Yes" and 0.23 for "No". This means that the model indeed recognized correctly the positive environment and suggested that, based on the values "read" in the nodes and the current influences between them, a decision in favor of Cloud adoption should be taken. Executing the model using the values for the negative scenario, the value of 0.75 for "No" and 0.25 for "Yes" were calculated at the evaluation node, which perfectly matched the expected behavior once again. Therefore, the ID model successfully passed the validation test: By using the above "extreme" scenarios it became evident that the proposed model behaves successfully as it recognized correctly the conditions of the environment and predicted the right decision.

Close inspection of the input values of the first real-world scenario indicates that the condition described are in favor of a positive decision. On the contrary, the input values of the second and third real-world scenarios leave practically no room for a safe prediction of the final decision. The execution of the proposed ID model for the first real scenario returned the value of 0.61 for "Yes" and 0.39 for "No". This answer may easily be translated to a clear "Yes", which coincides with the actual decision taken. The second real case yielded a "strong" 0.74 for "No" and 0.26 for "Yes", with this suggestion again coinciding with the actual decision. In the third case our model returned 0.57 for "Yes" and 0.43 for "No". This result is slightly in favor of Cloud adoption and although the decision was not so strong the model once again succeeded in projecting the real decision as there was indeed a debate regarding the cost and benefits of moving to the Cloud but the main argument was that the latter were quite serious to

Table 4.3: Input values for the five scenarios tested

Factor	Term	Positive	Negative	Real1	Real 2	Real 3
Legal	High	0.8	0	0.6	0.2	0.6
	Medium	0.2	0.2	0.3	0.6	0.3
	Low	0	0.8	0.1	0.2	0.1
Availability	High	0.8	0	0.6	0.2	0.6
	Medium	0.2	0.2	0.3	0.6	0.3
	Low	0	0.8	0.1	0.2	0.1
Security	High	0.8	0	0.6	0.2	0.6
	Medium	0.2	0.2	0.3	0.6	0.3
	Low	0	0.8	0.1	0.2	0.1
Cost / Pricing	High	0	0.8	0.3	0.7	0.1
	Medium	0.2	0.2	0.6	0.3	0.6
	Low	0.8	0	0.1	0	0.3
ROI	High	0.8	0	0.3	0	0.1
	Medium	0.2	0.2	0.6	0.2	0.6
	Low	0	0.8	0.1	0.8	0.3
Compliance	High	0.8	0	0.6	0	0.6
	Medium	0.2	0.2	0.3	0.2	0.3
	Low	0	0.8	0.1	0.8	0.1
Performance	High	0.8	0	0.3	0	0.6
	Medium	0.2	0.2	0.6	0.2	0.3
	Low	0	0.8	0.1	0.8	0.1
Scaleability	High	0.8	0	0.8	0	0.8
	Medium	0.2	0.2	0.2	0.6	0.2
	Low	0	0.8	0	0.4	0
Privacy / Confidentiality	High	0.8	0	0.3	0	0.0
	Medium	0.2	0.2	0.6	0.2	0.2
	Low	0	0.8	0.1	0.8	0.8
Elasticity	High	0.8	0	0.8	0	0.8
	Medium	0.8	0	0.2	0.6	0.2
	Low	0.2	0.2	0	0.4	0.0
Data Access / Import-Export	High	0	0.8	0.6	0	0.6
	Medium	0.8	0	0.3	0.6	0.3
	Low	0.2	0.2	0.1	0.4	0.1
Technology Suitability	High	0	0.8	0.6	0	0.6
	Medium	0.8	0	0.3	0.6	0.3
	Low	0.2	0.2	0.1	0.4	0.1
Hardware Access	High	0	0.8	0.3	0.2	0.3
	Medium	0	0.8	0.6	0.8	0.6
	Low	0.2	0.2	0.1	0	0.1
Auditability	High	0.8	0	0.3	0	0.6
	Medium	0.8	0	0.6	0.6	0.3
	Low	0.2	0.2	0.1	0.4	0.1

ignore and therefore determined the final positive decision. Therefore, the model correctly recognized this situation as well and matched correctly the decision.

Table 4.4: Calculating probabilities on *Trust* node

Security	**Privacy**	**Trust**	
$w = 0.65$	$w = 0.35$	$P(Trust = High \mid Security, Privacy)$	$P(Trust = Low \mid Security, Privacy)$
High	High	0.9	0.1
High	Medium	0.76	0.24
High	Low	0.62	0.38
Medium	High	0.64	0.36
Medium	Medium	0.5	0.5
Medium	Low	0.36	0.64
Low	High	0.38	0.62
Low	Medium	0.24	0.76
Low	Low	0.1	0.9

4.3.2 Fuzzy ID

To achieve the best representation of the factors describing the problem under modeling, the combined knowledge of the experts and literature was used to set and construct F- and G-type fuzzy sets. For all nodes in the diagram a common F-type graph shape was selected which represents best the fuzzy set shown in Table 4.5.

Table 4.5: Fuzzy values

Linguistic value	**Numerical value**	**Fuzzy value**
negatively very high	-5	0
negatively high	-4	0.1
negatively medium	-3	0.2
negatively small	-2	0.3
negatively very Small	-1	0.4
positively very Small	1	0.6
positively small	2	0.7
positively medium	3	0.8
positively high	4	0.9
positively very high	5	1

It is obvious that the influence of each node to its child can be described as absolutely linear. Inversely, but in the same spirit, the sigmoid equation (2.2), has been selected to represent the G-Type fuzzy set for all nodes in the model. The range of the G-type fuzzy set is between (-0.5) and $(+0.5)$, with $l = 5$. The FID model was executed using the fuzzy values depicted

in Table 4.6, which describe each of the five scenarios tested.

An example of how the calculations described in Section 2.2.3.2 are performed for evaluating FID nodes is shown in Table 4.7 focusing on the Trust node. Numerical values x, which reflect the initial state of a node, are transformed to $F(x)$ values using the F-type fuzzy set shown in Table 4.5. Next, the combination of the F-type and G-type Fuzzy sets using Scalable Monotonic Chaining computes the $G(F(x))$ values. Finally, the weighted sum of the $G(F(x))$ values produces the fuzzy input value for the *Trust* node.

Table 4.6: Input values of the nodes participating in the FID model for the five scenarios tested

Factor	**Positive**	**Negative**	**Real 1**	**Real 2**	**Real 3**
Legal	0.9	0.1	0.8	0.5	0.8
Availability	0.9	0.1	0.8	0.5	0.8
Security	0.9	0.1	0.8	0.5	0.8
Cost / Pricing	0.1	0.9	0.7	0.8	0.3
ROI	0.9	0.1	0.7	0.1	0.3
Compliance	0.9	0.1	0.8	0.1	0.8
Performance/Processing	0.9	0.1	0.7	0.1	0.8
Scalability	0.9	0.1	0.9	0.3	0.9
Privacy/ Confidentiality	0.9	0.1	0.7	0.1	0.1
Elasticity	0.9	0.1	0.9	0.3	0.9
Data Access / Import-Export	0.9	0.1	0.8	0.3	0.8
Technology Suitability	0.9	0.1	0.8	0.3	0.8
Hardware Access	0.9	0.1	0.8	0.6	0.7
Auditability	0.9	0.1	0.7	0.3	0.8

Table 4.7: Evaluating Trust node in FID

Nodes	x	$F(x)$	$G(F(x))$	**Weight**	$F(Trust)$
Security	3	0.8	0.78	0.65	0.73
Privacy	2	0.6	0.58	0.35	

4.3.2.1 Experimental Results

Our model was led to the value of 0.8529 for the positive scenario, which is translated as "positively high", and, as expected, to the value of 0.1471, that corresponds to "negatively high", for the negative scenario. Like the generic ID model, the performance of the FID

was also validated as successful through these two “extreme scenarios”. The first real-world scenario returned the value 0.7585, which can be translated as “positively medium” and succeeded to suggest the real decision. The second real-world scenario returned 0.2946, which corresponds to “negatively small”, succeeding also to match the actual decision. Following the previous scenarios, the third case again succeeded to match the real decision and the debate scenery, returning 0.5786 as “positively very small”.

4.3.3 Comparison of the ID Models

Summarizing the results listed in Table 4.9, one may infer that both models succeeded to correctly predict the real decisions. Another significant aspect is the fact that both models reached their decision with similar “weights” and this gives an extra value to their reliability and credibility. Considering that the modeling process can be divided into three parts, that is, data collection, model design/construction and model execution (results), then an argument can be made that both models are identical as regards the first and third part, and differ only in the construction and design part. The success of both approaches in the modeling process allows to focus on some differences that are important for the future selection among the two and further study of the process.

Table 4.8: Model’s decisions compared with real decisions

Real-world scenarios	**ID**	**FID**	**Real decision**
1	Yes 60%	0.7585 (positively medium)	Yes
2	No 74%	0.2946 (negatively small)	No
3	Yes 57%	0.2786 (positively very small)	Yes

A generic ID, comparatively to a FID, carries a large volume of literature study and this is an essential fact for its reliability, unlike a FID which is a relatively new model and its relative area may be considered virgin, not so matured and surely not so much studied. However, the FID offers some key benefits which make it a promising approach with valuable contribution in decision making problems. An FID model seems to overcome problems and difficulties associated with probabilities, avoiding increased combinatorial cost in the corresponding calculations. Fuzzy sets values can be estimated in an easier way using a linguistic Likert scale compared to the usual n-level scale of generic IDs. In addition, by using FIDs the interaction of the experts through the execution of the model becomes even easier because each individual

node can be studied separately and different weights and values may be set independently of the other nodes, thus avoiding the recalculation of all ancestors' probabilities.

As mentioned before, the effort and time spent to assign probabilities to all possible outcomes of each node in a large ID is huge and this is prohibitive to construct ID models including nodes with a large number of outgoing arcs. Since this problem is practically overcome by FIDs, for inductive and experimental purposes, an alternative extreme structure of FID model was constructed as shown in Figure 4.2. Exactly the same data was used as before, but without grouping the factors and having the corresponding weight values influencing directly the evaluation node. If this structure was attempted based on the probabilistic approach of the generic ID then it would require the huge number of $14,348,907$ combinations.

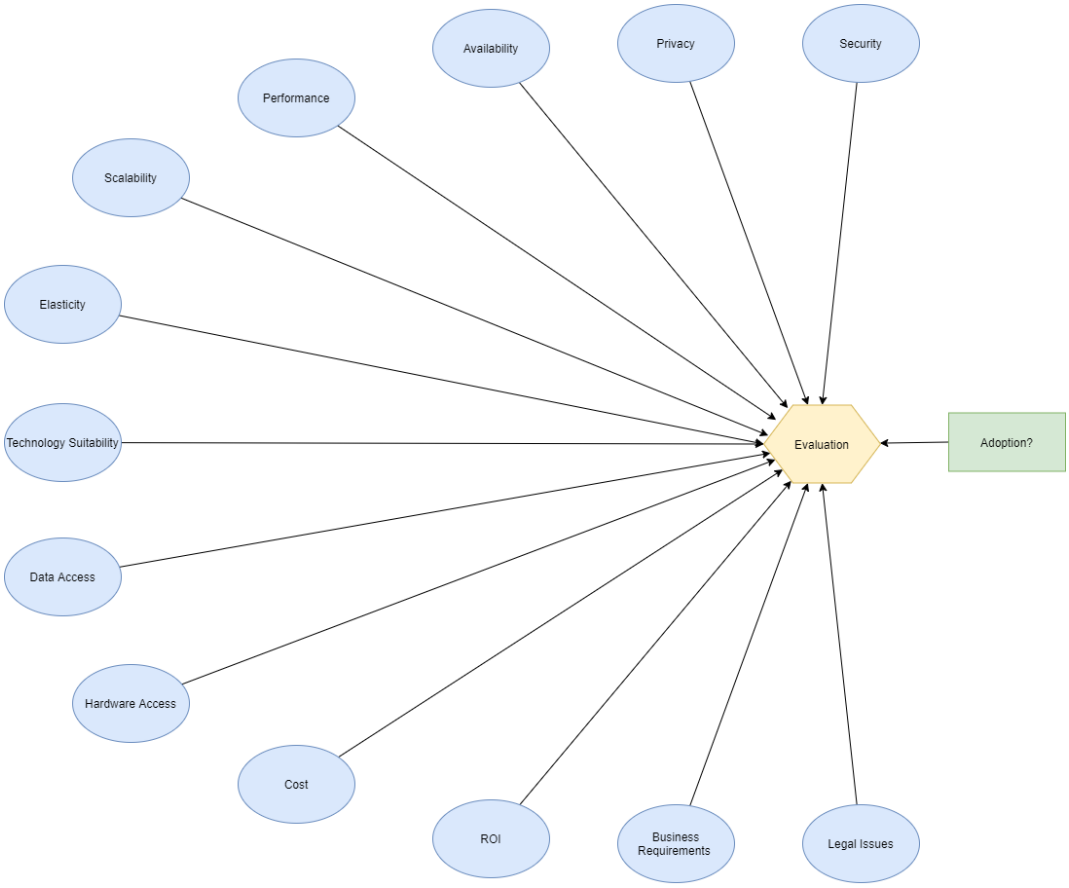


Figure 4.2: An alternative inductive structure of our FID model

The alternative FID model was tested on the same three real-world scenarios and returned 0.7308, that is, "positively small", 0.2543, that is, "negatively small", and 0.5720, that is "positively very small" respectively. The performance of this model confirms the strong ability of the FID approach to succeed in cases of resource demanding structures.

4.3.4 What-if Scenario Simulations

As mentioned above, one of the most significant benefits of the proposed models is their ability to perform simulation analysis based on what-if scenarios making them an efficient tool in the hands of decision makers. More specifically, influence diagrams provide interactive capabilities and enable the study of the effects of changing various parameters (factors) of the problem under modeling so as to trace the source of undesired situations like in our case a negative prediction of Cloud adoption. In such a case, the vendor will be able to identify which factors are responsible for the negative picture and examine if and how their values may be changed, and to what extent, in order to revert the decision. This simulation process essentially provides critical information to vendors and guides them towards performing possible actions. The latter, of course, is always subject to cost/benefit analysis and feasibility depending on the situation. For example, if cost is the decisive factor that drives a Cloud adoption decision to the negative end and the vendor is able to acknowledge the impact of this factor to the final decision, then the vendor may choose not to lower costs in order to make transition to Cloud more appealing to its customers and possibly revert decision simply because by doing so the margins for profit become too narrow. Presented below is a demonstrative example describing how we can work to revert the decision in the second real-world scenario from negative to positive.

Based on the literature review initially performed and the input received from the experts, which finally led to the determination of the significance of the participating factors, the factors were ranked according to their weight of influence on the final decision and selected to process the top five of them, namely *Cost*, *Return of Investment*, *Security*, *Legal* and *Availability*. The new FID model was considered and used the new input values for the second real scenario shown in Table 4.6, which reflect a low to medium increase in the levels of *Legal*, *Availability*, *Security* and *ROI* factors, and a significant decrease (of 50%) for the *Cost* level. The FID model was executed under these changed values and returned 0.5232 that corresponds to "positively very small". Therefore, the negative decision was reverted, simply by following an interactive process which altered the conditions prevailing in the FID model and tested specific input values under a simulation environment. This interactive simulation process may be repeated by vendors using different input values that investigate the evolution of various hypothetical scenarios and planning certain actions according to various considerations each time. As already mentioned, these hypothetical scenario simulations will be examined under the prism of feasibility; for example, if cost reduction is not feasible at the level described

above, then other means of policy making in the associated determinant factors must be sought for reverting the negative decision. The whole process is, of course, time and effort consuming as it is based on trial and error; therefore, a more "sophisticated" approach is needed in this case supported by an automated solution as regards identification of the appropriate values for the determinants.

4.4 FCM Modeling

4.4.1 Single Layer FCM

The development of a single-layered FCM for modeling the Cloud adoption decision-making process was implemented following the two methods mentioned above, specifically literature study and collection of experts opinion through specially prepared questionnaires followed by interviews. Each identified concept is unique in the sense that no overlaps exist between the interpretation of what each concept represents. For example, concept *Compliance* focuses on functional requirements rather than the issues of security, the latter being addressed by *Privacy / Confidentiality* concept. An initial list of concepts was then prepared and the experts were asked to evaluate the list and prompted to add or remove concepts based on their expertise and working experience. The last step included one more round with the experts discussing their comments and reaching to consensus as regards the final list of concepts. These concepts were used to form the nodes of the map and are described in Table 4.9.

Based on the final concept list, the experts were again asked to complete a questionnaire concerning the causal relationships between the nodes of the map and the weights involved, i.e. the degree to which concepts influence each other. The influences were fuzzified using eleven linguistic variables : "negatively very high", "negatively high," "negatively medium," "negatively small," "negatively very small," "neutral," "positively very small," "positively small," "positively medium," "positively high," "positively very high". For simplicity's sake, these variables were encoded in a Likert scale corresponding to integer values within the range [-5, 5]. At the same time, for each defined relation the experts would have to declare the value of confidence of their answers by using integer values in the range [0, 5], which corresponded to six linguistic variables: "zero", "small", "medium", "high", "very high". The experts' ranking was then combined with their answers in a weighted average scheme and the relationships of the nodes in the model were represented by the normalized weight matrix shown in Table 4.10.

Table 4.9: Conceptual nodes of the proposed model

Id	Name	Definition
C1	Legal	Cloud adoption compliance with all legislative issues. Ability to adjust when legal requirements grow.
C2	Availability	The amount of time that Cloud Services is operating as the percentage of total time it should be operating.
C3	Security	Security of service: data transfer, data stores, web servers, web browsers.
C4	Cost / Pricing	Operational - Running costs, migration costs etc. Cost benefits from Cloud adoption.
C5	Compliance	Business and Regulatory compliance.
C6	Performance / Processing	Does Cloud adoption perform the process to the desire quality?
C7	Scalability	Ability to meet an increasing workload requirement by incrementally adding a proportional amount of resources capacity.
C8	Privacy / Confidentiality	Privacy and confidentiality coverage.
C9	Elasticity	Ability to commission or decommission resource capacity on the fly.
C10	Data Access / Import-Export	Access to data in various ways.
C11	Technology Suitability	Does Cloud technology exhibits the appropriate technological characteristics to support proposed SaaS?
C12	Hardware Access	Degree of Cloud Service accessibility, on local hardware.
C13	Audit ability	Ability of Cloud service to provide access and ways for audit.
C14	Exit Process	Guarantee and ensure the output process from provider.
C15	Disaster Recovery	Ability of Cloud service vendor to provide the required disaster recovery.
C16	Cloud Adoption	Central concept of the model.

Figure 4.3 depicts a graphical representation of the map. It is obvious that the structure of the map is quite complex, with 141 total number of connections between nodes. In this modeling, the Certainty Neuron Fuzzy Cognitive Maps (CNFCM) [53] structure was selected, which introduces additional fuzzification to the traditional neuron, that is, it allows various activation levels of each concept instead of only the two extreme cases, activation or not.

The map is initialized by setting values to the concepts (activation levels) so as to reflect the different scenarios being considered. Each scenario represents a certain situation under

Table 4.10: Causal relationships and weight values between conceptual nodes on a Likert scale from 1 (very low) to 5 (very high) positive and negative - Row influences column.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
C1		0	-2	5	0	3	0	-3	0	4	3	0	2	0	0	5
C2	0		0	5	2	0	0	4	0	0	-3	4	4	0	2	4
C3	-3	0		5	0	-4	0	5	0	4	-4	-3	-4	0	0	5
C4	5	5	5		5	5	5	5	3	5	5	5	5	0	5	-5
C5	5	4	-3	5		0	0	-2	0	2	4	4	4	2	4	5
C6	1	0	-2	5	3		0	-4	3	3	0	0	0	2	0	5
C7	5	0	0	5	2	3		0	2	5	0	0	0	0	0	5
C8	0	2	0	5	4	-3	0		-2	3	-4	-3	-3	0	0	5
C9	5	0	-2	5	3	4	0	0		5	0	0	0	0	0	5
C10	4	3	2	5	2	3	5	3	4		4	4	4	0	4	5
C11	0	4	-5	4	4	0	0	-4	0	4		0	5	0	0	4
C12	0	2	-3	5	4	2	0	0	0	4	1		2	0	0	4
C13	0	5	3	5	4	2	0	2	0	2	0	2		0	0	3
C14	0	4	0	5	0	0	0	0	0	0	0	0	0		0	5
C15	0	2	0	5	0	2	0	0	0	5	3	0	0	0		5
C16	-3	0	-3	5	-3	-3	5	0	0	4	0	0	0	0	0	

modeling as this is described through the activation levels, with the aim being to study the evolution of these levels as follows: After a sufficient number of iterations, if the map succeeds to reach equilibrium at a fixed point, then the final values may be further studied. The most important value is that of the central concept of the map, while at the same time the final values of the rest of the concepts may also provide useful information to decision makers and assist in reaching to important conclusions.

4.4.1.1 Experimental Results

Aiming to test and evaluate the performance of the proposed model, two hypothetical scenarios were first conducted representing the so called “extreme cases”, that is, a situation where everything would be in favor of Cloud adoption (positive scenario) and the opposite case (negative scenario). The target was to reach to equilibrium under known situations and assess the performance of the model proving that the model behaves correctly and as expected to. Next, the map was tested on a number of real-world scenarios, that is, cases collected from real customers of three international Cloud services providers with the aid of the same experts that were utilized to construct the map. The two extreme scenarios and the real-world

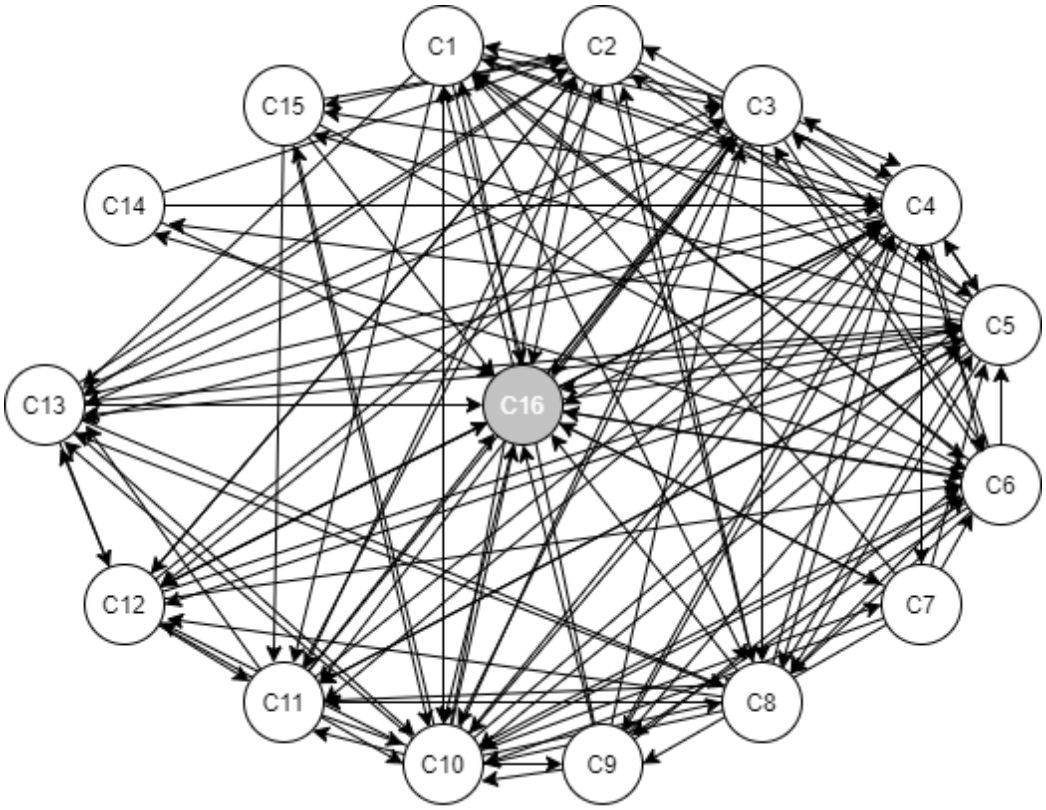


Figure 4.3: The Cloud adoption CNFCM model.

cases experimentation are described below. In all experiments the map was executed for 250 iterations and the results were assessed first to inspect whether they reached equilibrium and then to examine the value of the concept of interest (*Cloud adoption*).

4.4.1.1.1 Extreme Scenarios

The first case assumes an ideal environment where the Cloud services offered perfectly match a customer's needs. Thus, the initial values for each concept were chosen so that they reflect this ideal setting and guide the central concept of interest to a positive value. Following the same logic with linguistic variables as in the case of relation-ships between the concepts, the initial activation levels for this scenario were defined as listed in Table 4.11

The map was executed using the activation level values of Table 4.12 and the normalized form of the weights listed in Table 4.10, transformed in the range [-1, 1]. As shown in Figure 4.4(a), the model reaches an equilibrium state and thus inference is possible. The basic finding here is that the map behaves correctly and leads the central concept of interest to the positive value of 0.795. This means that the model correctly recognized the positive environment and suggested

Table 4.11: FCM initial activation level values of the concepts for the positive scenario.

Concept	Linguistic value	Numerical value	Normalised value
C1	positively high	4	0.8
C2	positively high	4	0.8
C3	positively high	4	0.8
C4	negatively high	-4	-0.8
C5	positively high	4	0.8
C6	positively high	4	0.8
C7	positively high	4	0.8
C8	positively high	4	0.8
C9	positively high	4	0.8
C10	positively high	4	0.8
C11	positively high	4	0.8
C12	positively high	4	0.8
C13	positively high	4	0.8
C14	positively high	4	0.8
C15	positively high	4	0.8
C16	zero	0	0

that a decision in favor of Cloud adoption should be taken based on the values "read" in the concepts and the current influences between the nodes.

Working in the same way as with the positive scenario, appropriate initial values for each concept were chosen this time to guide the central node to a negative value. The initial activation levels for the negative scenario are shown in Table 4.12.

Executing the model using the values for the negative scenario again the map reached equilibrium with its behavior being the one anticipated: the central concept of the map takes the negative value of -0.795 (see Figure 4.4(b)).

From the above "extreme" scenarios it is evident that the proposed model behaves correctly by recognizing the setting fed and therefore we may now proceed with evaluating its performance on real-world cases.

4.4.1.1.2 Real-World Scenarios

As previously mentioned, with the help of Cloud providers four different cases were identified: Two customers who decided to proceed with Cloud adoption and two cases in which they rejected it. These four cases, along with some related information describing the required

Table 4.12: FCM initial activation level values of the concepts for the negative scenario.

Concept	Linguistic value	Numerical value	Normalised value
C1	negatively high	-4	-0.8
C2	negatively high	-4	-0.8
C3	negatively high	-4	-0.8
C4	positively high	4	0.8
C5	negatively high	-4	-0.8
C6	negatively high	-4	-0.8
C7	negatively high	-4	-0.8
C8	negatively high	-4	-0.8
C9	negatively high	-4	-0.8
C10	negatively high	-4	-0.8
C11	negatively high	-4	-0.8
C12	negatively high	-4	-0.8
C13	negatively high	-4	-0.8
C14	negatively high	-4	-0.8
C15	negatively high	-4	-0.8
C16	zero	0	0

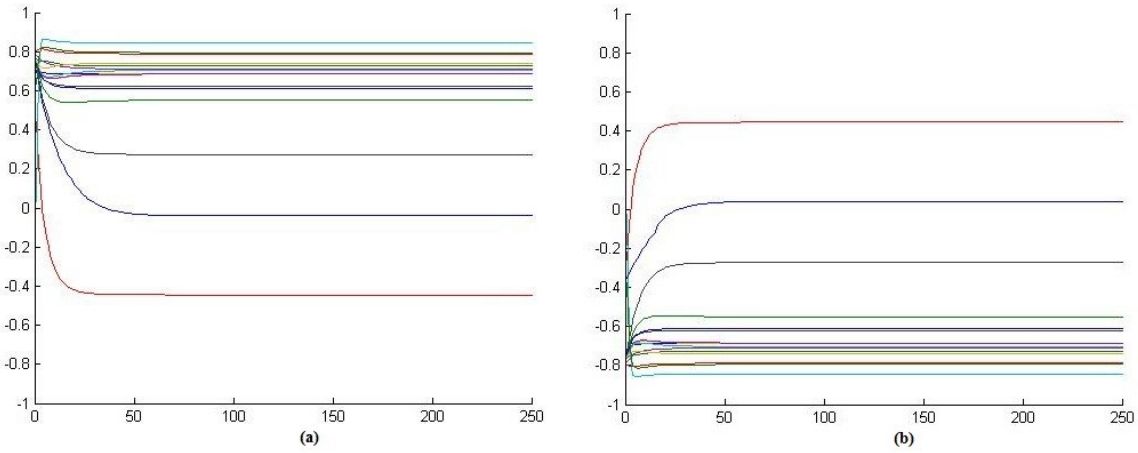


Figure 4.4: Extreme scenarios: (a) Positive Case, (b) Negative Case.

Cloud services, the size of the organization, the line of business and the final decision regarding Cloud adoption, are summarized in Table 4.13.

A series of interviews was conducted, both with the Cloud providers and the customers, so as to identify the initial activation level values for each case separately, which essentially reflected the state of the offered service and the associated factors describing each customer's particular situation at the moment of decision. These initial activation level values for each

Table 4.13: Brief description of real-world cases.

Case	Line of Business	Users	Cloud Services
A	Academic Institution	4500	Mail Server / Mailbox / Mail client
B	Industry	220	Mail Server / Mailbox / Document Management
C	Supplies Industry	85	Mail Server / Mailbox / Mail Client / Document Management
D	Insurance Brokers	125	Mail Server / Mailbox / Mail Client / Custom Business System

case are shown in Table 4.14 in linguistic form, which was found by the people involved in the questionnaires easier to understand and follow.

Table 4.14: Initial activation level values for real world cases.

Concept	Case A	Case B	Case C	Case D
C1	positively very high	positively medium	positively medium	positively medium
C2	positively medium	positively medium	positively medium	positively medium
C3	positively medium	positively medium	positively medium	positively medium
C4	negatively very high	negatively small	positively very small	positively high
C5	positively very high	positively medium	positively medium	negatively very small
C6	positively very high	positively medium	positively medium	negatively very small
C7	negatively high	positively very high	positively very high	negatively medium
C8	positively medium	negatively very high	positively very small	positively very small
C9	positively medium	positively very high	positively very high	negatively medium
C10	positively very high	positively medium	positively medium	negatively small
C11	positively very small	positively medium	positively medium	negatively small
C12	positively very small	positively very small	positively very small	positively very small
C13	positively very small	positively medium	positively very small	negatively medium
C14	negatively very small	negatively very small	negatively very small	negatively high
C15	positively very high	positively very high	positively very high	positively very high
C16	zero	zero	zero	zero

The first case involved an academic institution of a medium to large size, which requested a comprehensive solution for email services. In this case it is easy for someone to discern from the initial activation level values that the conditions were in favor of a positive decision. The second case described a medium industrial organization which requested the provision of some Cloud email services and a Cloud-based document management system. This case can be described also as positive. The third case represented another industrial organization which requested a complete Cloud email package and also a Cloud-based document management system. The decision environment seems again to be positive. Finally, the fourth case involved

a medium insurance broker's organization which requested a complete Cloud email package and also a Cloud infrastructure to fit their own, heavily customised system. This case is hard to detect if it is positive or not based on the initial activation values.

The model was executed again for 250 iterations and reached equilibrium in all cases. The outcome of the model for each of the aforementioned cases is given in Table 4.15 with the actual decisions taken by the customers for comparison purposes.

Table 4.15: Model's decisions compared with real decisions.

Case	FCM model's decision	Real Decision
A	Yes	Yes
B	Yes	No
C	Yes	Yes
D	No	No

4.4.1.2 Discussion

It is evident that the model succeeded in matching its estimation or suggestion with the real decision in three out of four cases. More specifically, in cases A and C our model suggested a positive decision achieving a match with the actual decision. Also, in case D the model estimated a negative decision again in agreement with the real decision. Unlike the previous cases, the model failed to coincide with the actual decision taken for real case B. We attempted to investigate the reason for this and after consulting our experts it became evident that this customer should have decided positively given the context of the particular needs; nevertheless, despite the fact that the offered Cloud package was fulfilling all requirements, the decision was turned negative as a result of the mistrust of the company towards the Cloud environment, something which, on one hand, is definitely beyond the scope of this study, and, on the other, it suggests that the result was correct in the first place.

4.4.2 Multi Layer FCM

To take advantage of the capabilities offered by the MLFCM modeling, and be able to apply a more detailed analysis of the problem under study, an updated process was followed. The literature used in the survey conducted as described in the previous section was revised and enriched with a significant number of research and empirical papers on the subject. The factors

that were identified previously were assessed once again, but, most importantly, through this process new parameters emerged, as well as information regarding groupings of parameters that describe a concept at a higher level of the model's hierarchy. Finally, one more round of discussion with our experts was performed in order to update and finalize the list of parameters, which now comprises the thirty-one concepts listed in Table 4.16.

Following the creation steps of the ML-FCM as described in [38] [37], the identified concepts were grouped in seven sub-FCMs as shown in Table 4.17. Each sub-FCM comprises a different number of concepts that describe one central concept of interest. FCM1 is the main model's FCM and consists of eleven concepts with C31 (*Cloud Adoption*) being the central concept of this map.

Once the MLFCM with all concepts and their causal relationships between them were identified, the fuzzification process followed. The influences between the concepts were fuzzified using eleven linguistic variables : "negatively very high", "negatively high," "negatively medium," "negatively small," "negatively very small," "neutral", "positively very small," "positively small," "positively medium," "positively high," and "positively very high".

A graphical representation of the model is presented in Figure 4.5. One can easily observe the complexity of the problem, with numerous interactions existing between the concepts, the latter being broken down to various layers and neighborhoods according to the MLFCM structure.

4.4.2.1 Experimental Results

Before proceeding with applying the model on a set of real-world cases, two realistic hypothetical scenarios will be used as before ("extreme scenarios"), aiming to test and evaluate its performance. The target is to assess whether a reasonable answer is yielded as this is expressed by the activation level of the central concept of interest, thus validating the performance of the model against the expected behavior. In every experiment, each FCM was executed for 250 iterations.

Two extreme cases are produced, one as a positive scenario where the environment is set to be in favor of Cloud adoption and one as a negative scenario where the factors in the decision environment promote a negative stance towards Cloud adoption.

Table 4.16: Conceptual nodes of the proposed MLFCM model.

Id	Name	Definition
C1	Legal	Cloud adoption compliance with all legislative issues. Ability to adjust when legal requirements grow
C2	Compliance	Business and Regulatory compliance
C3	Compliance	Service compliance ability
C4	ROI	Return of Investment in a midterm period
C5	Pricing	Pricing model of the Cloud service provided
C6	Running/Operational Cost	Operational - running costs
C7	Migration cost	Migration cost i.e training, company's structure changes etc.
C8	Cost	Cloud service general cost
C9	Privacy/Confidentiality	Privacy and Confidentiality coverage
C10	Data Access / Imp-Exp	Access to data in various ways
C11	Backup	Data Backup option
C12	Disaster Recovery	The required disaster recovery provision
C13	Data Management	Managing data and information ability
C14	Exit Process	Guarantee and ensure the output process from provider
C15	Compatibility	Compatibility with existing applications
C16	Hardware Access	Degree of Cloud Service accessibility, on local hardware
C17	Integration	Accessing and sharing information ability
C18	Migration Complexity	How the migration affects to the processes and structures
C19	Availability	The amount of time that Cloud Services is operating as the percentage of total time it should be operating
C20	Processing	Does Cloud adoption perform the process to the desire quality (Speed)?
C21	Scalability	Ability to meet an increasing workload requirement by incrementally adding a proportional amount of resources
C22	Elasticity	Commission or decommission resource capacity on the fly
C23	Support	An interface made available by the Cloud service provider to handle issues and queries raised by customer
C24	Performance	The performance of the Cloud service
C25	Quality of Service Assure	Operational controls to ensure that the results match the desired outcomes
C26	Authentication & Authorization	Verification of the claimed identity of an entity and the process of verifying permissions privileges
C27	Audit ability	Ability of Cloud service to provide access and ways for audit
C28	Reliability	Cloud service to perform its function correctly and withoutfailure
C29	Security	The security level being offered by a Cloud service
C30	Technology Suitability	Does Cloud technology exhibits the appropriate technological characteristics to support proposed SaaS?
C31	Cloud Adoption	Central Concept of the model

Table 4.17: Sub-FCM Groupings

FCM	Concepts	Central Concept	Layer
1	C3, C8, C13, C14, C17, C18 C24, C25, C29, C30, C31	C31	1
2	C15, C16, C17	C17	2
3	C1, C2, C3	C3	2
4	C4, C5, C6, C7, C8	C8	2
5	C19, C20, C21, C22, C23, C24	C24	2
6	C26, C27, C28, C29	C29	2
7	C9, C10, C11, C12, C13	C13	2

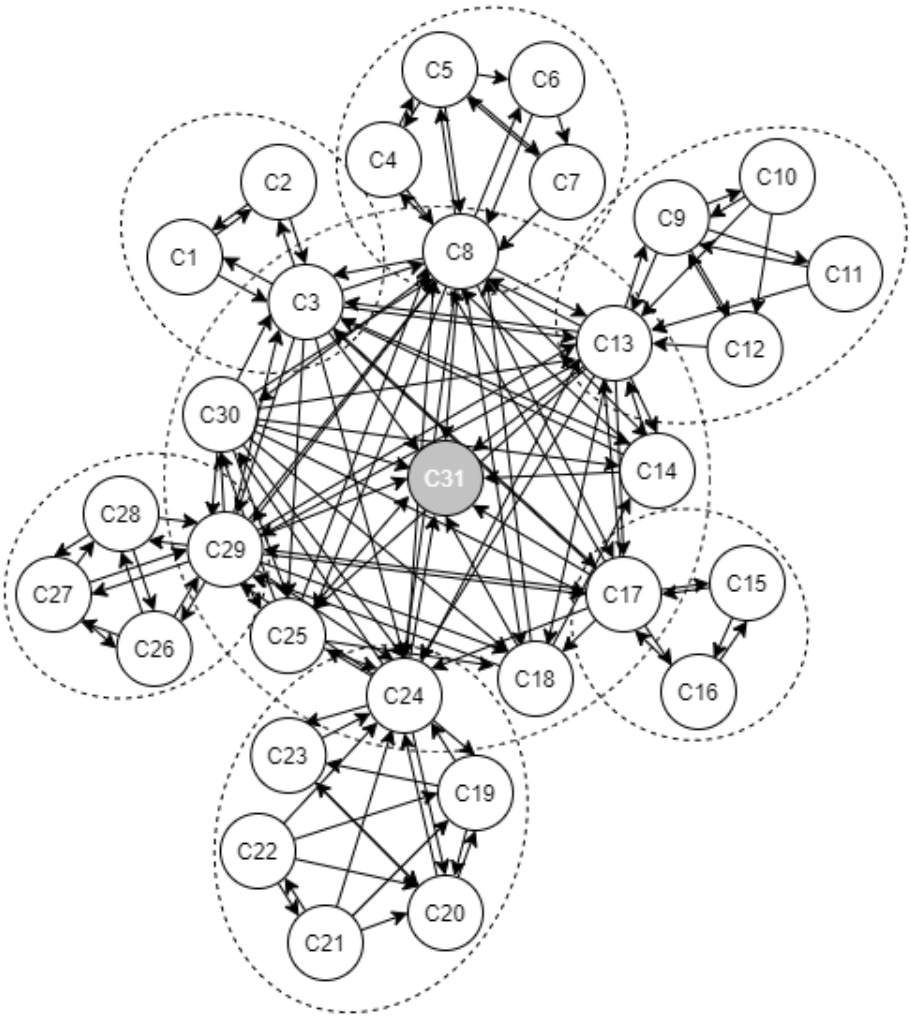


Figure 4.5: The Cloud adoption MLFCM model.

4.4.2.1.1 Extreme Scenarios

An ideal environment was set in which Cloud services offered perfectly match the customer's needs. Therefore, the initial values of each concept on the map were chosen so that they reflect this positive to the Cloud adoption influence. Following the execution process described earlier for MLFCM, the maps of the second layer were firstly executed in order to transfer the final activation levels values of the sub-concepts of interest to their parent nodes, which then use these values as their initial activation level values in the main map. At the end of the process the model reached equilibrium and led the central concept to the positive value of 0.85, meaning that the model correctly recognized the positive environment and suggests a positive decision for Cloud adoption.

Working in the same way as described in the previous paragraph, the initial values were chosen to reflect a negative environment in order to guide the central concept to a negative value. Executing the model again the map reached to an equilibrium state leading the central concept to the negative value of -0.85 as expected.

4.4.2.1.2 Real-World Scenarios

After the successful initial evaluation, we proceeded to test our model under four real-world scenarios as these are described in Section 4.4.1.1.2. Since the MLFCM model has a different structure than the single-layer FCM, a new round of dedicated interviews was conducted, both with the Cloud providers and the customers, to identify the initial activation values for the cases under study. These values essentially reflected the state of the offered service and the associated factors describing each case separately at the time of decision. Table 4.18 lists the initial activation values of the main FCM and sub-FCMs for each real-world case. The final activation levels reached after executing the sub-FCMs are shown in the corresponding row for each concept of interest.

The model reached equilibrium after completing the execution steps in all four cases. The numerical values of the final activation level of the central concept of the map, as well as the linguistic interpretation for each of the corresponding cases, are given in Table 4.19 juxtaposed with the actual decisions that were taken by the customers.

Table 4.18: Initial activation levels for each real-world case

FCM	Concept	Case A	Case B	Case C	Case D
1	C3	0.9	0.5	0.5	-0.1
	C8	-0.9	-0.3	0.1	0.7
	C13	0.7	0.5	0.5	0.3
	C14	-0.1	-0.1	0.1	-0.7
	C17	0.1	0.3	0.3	-0.3
	C18	0.5	0.5	0.3	0.9
	C24	0.9	0.5	0.5	-0.1
	C25	0.9	0.9	0.9	0.7
	C29	0.5	0.5	0.5	0.5
	C30	0.1	0.5	0.5	-0.3
	C31	0	0	0	0
2	C15	0.9	0.5	0.5	-0.1
	C16	0.1	0.1	0.1	0.1
	C17	0.1(0.89)	0.3(0.89)	0.3(0.89)	-0.3(-0.89)
3	C1	0.9	0.5	0.5	0.5
	C2	0.9	0.5	0.5	-0.1
	C3	0.9(0.86)	0.5(0.86)	0.5(0.86)	0.1(0.86)
4	C4	0.7	0.1	-0.1	-0.5
	C5	-0.7	-0.1	0.1	0.7
	C6	-0.5	-0.3	0.1	0.5
	C7	0.5	0.5	0.5	0.5
	C8	-0.1(-1)	0.1(-1)	0.3(0.9)	0.5()0.9
5	C19	0.9	0.5	0.5	-0.1
	C20	-0.7	0.9	0.9	-0.5
	C21	0.5	0.9	0.9	-0.5
	C22	0.7	0.7	0.7	0.3
	C24	0.7(0.91)	0.5(0.91)	0.5(0.91)	-0.3(-1)
6	C26	0.9	0.9	0.9	0.7
	C27	0.1	0.5	0.1	-0.5
	C28	0.9	0.7	0.9	0.1
	C29	0.5(0.79)	0.5(0.79)	0.5(0.79)	0.5(0.79)
7	C9	0.5	-0.9	0.1	0.1
	C10	0.9	0.5	0.5	-0.3
	C11	0.7	0.7	0.7	0.7
	C12	0.9	0.9	0.9	0.9
	C13	0.7(0.90)	0.5(0.90)	0.5(0.90)	0.3(0.90)

4.4.2.2 Discussion of Results

The execution of the proposed model against the four real-world cases presents some interesting findings worth pointing out: First of all, the final decision results coincide with those of the

Table 4.19: Model's output decisions compared with real decisions

Case	Model's final value	Model's decision	Real decision
A	0.8226	Yes	Yes
B	0.8226	Yes	No
C	0.8226	Yes	Yes
D	-0.8226	No	No

previous study with the single-layer FCM, indicating successful behavior of the model with respect to matching its suggestion with the real decision in three out of four cases. More specifically, in cases A and C our model suggested a positive decision while in case D a negative decision, all three being in total agreement with the real decision. In real case B the model's output did not coincide with the actual decision, and as was explained before, when investigating the reason for this and after consulting our experts it became evident that this customer should have decided positively given the context of his particular needs; therefore, the result was correct in the first place.

4.4.2.3 MLFCM Model Analysis

This section describes the application of the stepwise analysis and inference process, as presented in Chapter 3. These analyses aim to gather useful static and dynamic information from the proposed MLFCM model, such as the complexity of the model, the tendency of each sub-FCM, and the strength of the participating concepts (nodes). Also, through dedicated what-if scenarios, the model's behavior is investigated and the influence of specific individual concepts on the final decision is studied.

4.4.2.3.1 Static Analysis

Following the framework's stepwise process, firstly the static analysis was performed by applying metrics and indicators regarding graph complexity. The corresponding measurements are listed in Table 4.20.

It is easily discernible that the density of all sub-FCMs that constitute the MLFCM model is rich and thus the model can be characterized as a highly complex, two-layer structure. Three of the sub-FCMs are complete graphs with density values equal to 1, but this effect is mitigated by their fairly small size which amounts up to 4 nodes.

Table 4.20: Complexity static measurements for the MLFCM modeling the Cloud Adoption problem

FCM	Layer	# nodes	# edges	Density	Cycles(+)	Cycles(-)
1	1	11	75	0.68	20335	20051
2	2	3	6	1	5	0
3	2	3	6	1	5	0
4	2	5	11	0.55	4	0
5	2	6	16	0.53	4	0
6	2	4	12	1	11	9
7	2	5	13	0.65	14	3

The number of feedback cycles have also been calculated for each sub-FCM, divided into positive and negative; the corresponding results are also listed in Table 4.20. As a general outcome it can be inferred that all sub-FCMs appear to have more positive cycles than negative and this is a strong indication of how the model tends to behave: Given a slight positive modification in the activation level of any node, the corresponding level of the central node of interest (*Cloud Adoption*) is promoted and vice-versa.

Continuing with the static analysis, measurements were recorded for all sub-FCMs aiming to identify the significance of each node in the MLFCM model. These measurements define the strength of each node, as well as the tendency of each sub-FCM, and are provided in Table 4.21.

Specifically, the metrics $deg_{in}(i)$, $deg_{out}(i)$, $deg_{tot}(i)$ $\kappa\alpha\iota$ $val_{tot}(i)$ were applied, which show how strong the presence of each node is in the sub-FCM it belongs to. Also, the participation of each concept in cycles was also examined by calculating the number of positive and negative feedback loops containing that concept; these results are also reported in Table 4.21.

A close examination of the results derived from the static analysis reveals a very strong indication regarding node significance, which leaves no doubt as to which are the top three concepts on the main FCM: *Cost*, *Security* and *Data Management*. All three are decomposed at the lower level transferring activation levels from their neighborhood of nodes. This finding calls for further investigation of the behavior of these concepts individually on one hand, and as a group on the other. Further to that, it paves the way to study the behavior of the lower sub-FCMs from which these three important concepts receive their values. Cycles measurements confirm and strengthen the static analysis findings derived thus far.

Table 4.21: Strength and tendency indicators for the sub-FCMs of the MLFCM model for the Cloud Adoption problem

SubFCM	Concept	$deg_{in}(i)$	$deg_{out}(i)$	$deg_{tot}(i)$	$val_{tot}(i)$	Cycles+	Cycles-
FCM1 $dec_{deg}=6$ $br_{deg}=4.33$	Compliance	6	8	14	8.4	17121	16831
	Cost	**9**	**10**	**19**	**15.4**	**18755**	**18503**
	Data (Management)	**7**	**9**	**16**	**11.4**	**17841**	**17551**
	Exit Process	6	5	11	7.4	15306	15129
	Integration	5	7	12	8.2	14845	15284
	Migration Complexity	6	4	10	4.8	13674	13424
	Performance	7	6	13	8.6	16539	16163
	QoS Assurance	6	6	12	9.2	15788	15672
	Security	**9**	**10**	**19**	**12.4**	**18675**	**18583**
	Technology Suitability	4	10	14	11.2	16966	16738
	Cloud Adoption	10	0	10	8.6	0	0
FCM2 $dec_{deg}=0$ $br_{deg}=0$	**Compatibility**	**2**	**2**	**4**	**2.8**	**4**	**0**
	Hardware Access	2	2	4	2	4	0
	Integration	2	2	4	2.8	4	0
FCM3 $dec_{deg}=0$ $br_{deg}=0$	Legal	2	2	4	3.2	4	0
	B&R Compliance	**2**	**2**	**4**	**3.4**	**4**	**0**
	Compliance	2	2	4	3.4	4	0
FCM4 $dec_{deg}=0$ $br_{deg}=0$	**ROI**	**3**	**4**	**7**	**5.8**	**4**	**0**
	Pricing	**1**	**3**	**4**	**3.6**	**2**	**0**
	Running Operational Cost	**2**	**2**	**4**	**3.8**	**2**	**0**
	Migration Cost	1	2	3	2	1	0
	Cost	4	0	4	3.6	0	0
FCM5 $dec_{deg}=0$ $br_{deg}=0$	**Availability**	**3**	**3**	**6**	**5.4**	**2**	**0**
	Processing	**4**	**3**	**7**	**4.4**	**3**	**0**
	Scalability	1	4	5	3.6	1	0
	Elasticity	1	4	5	3.4	1	0
	Support	2	2	4	2.6	2	0
	Performance	5	0	5	4.6	0	0
FCM6 $dec_{deg}=0$ $br_{deg}=0$	**Authentication & Authorization**	**3**	**3**	**6**	**5.2**	**8**	**7**
	Auditability	3	3	6	4.4	8	7
	Reliability	3	3	6	4.6	8	7
	Security	3	3	6	4.2	6	9
FCM7 $dec_{deg}=0$ $br_{deg}=0$	Privacy/ Confidentiality	2	2	4	2.4	5	3
	Data Access / Import-Export	**3**	**4**	**7**	**5**	**12**	**2**
	Backup	2	2	4	3.6	8	1
	Disaster Recovery	2	2	4	3.2	8	1
	Data (Management)	4	3	7	5.8	11	3

Starting the investigation of the strong nodes from the root level to the leaves, the study is focused on the sub-FCMs which are responsible for defining their values. *Cost* is decomposed into the concepts of FCM4, which, according to Table 4.21, is dominated by concepts *ROI*, *Pricing* and *Running Operational Cost*. All three nodes appear to have the strongest position in this sub-FCM and their participation in only positive cycles proves their positive effect on *Cost*. *Data* is broken down to the concepts of FCM7; *Data Access / Import-Export* node seems to have the strongest position in this sub-FCM, which also participates in almost all of the cycles with positive influence. Finally, the value of node *Security* is determined by the concepts of FCM6, which appears to be the most balanced map of all sub-FCMs, with concept *Authentication & Authorization* standing out for its influence on *Security*. Dynamic analysis scenarios will follow, which will be directed by the aforementioned results in an attempt to confirm or question these findings through execution.

4.4.2.3.2 Dynamic Analysis

The successful behavior of the model in the extreme scenarios, as well as in the real-world cases, allows performing simulations towards a dynamic analysis. A number of simulations have been set-up and executed following again the stepwise process described in Chapter 2. In all simulations the CNFCM approach [72] was employed, with the corresponding updating function (see eq. 3).

The first step of the dynamic analysis involved 1000 executions with randomised initial activation levels. Then the correlation coefficient between the initial activation level of each concept and the final activation level of the central concept of interest at each layer and sub-FCM was calculated. The corresponding results are provided in Table 4.22 where the strong nodes are indicated in bold letters. It should be noted that for the root FCM the top three nodes are indicated in bold, while for the rest of the sub-FCMs only the strongest one. The ranking of the concepts based on their correlations perfectly matched the findings of the static analysis indicating once again that the initial activation levels of *Cost*, *Security* and *Data Management* have a significant impact on the final activation level of the central node *Cloud Adoption*. In addition, all three nodes, when decomposed, are influenced by exactly the same nodes at the lower level as those that were indicated by the static analysis. Therefore, so far the results between the two types of analysis are enhanced and cross-checked.

Another important finding derived from the correlation analysis is that under randomized initial activation levels the model reached equilibrium with a positive value in 57% of the cases

Table 4.22: Correlation coefficient values between each concept and the central node of interest in each sub-FCM (in parentheses) of the MLFCM model

Sub-FCM	Concept	Correlation Coef.
FCM1 (Cloud Adoption)	Compliance	0.05
	Cost	**0.44**
	Data Management	**0.31**
	Exit Process	0.28
	Integration	0.12
	Migration Complexity	-0.21
	Performance	0.28
	QoS Assurance	0.17
	Security	**0.44**
	Technology Suitability	0.14
FCM2 (Integration)	**Compatibility**	**0.70**
	Hardware Access	0.58
FCM3 (Compliance)	Legal	0.55
	B&R Compliance	**0.63**
FCM4 (Cost)	ROI	-0.41
	Pricing	**0.62**
	Running Operational Cost	0.36
	Migration Cost	0.17
FCM5 (Performance)	Availability	-0.03
	Processing	**0.74**
	Scalability	0.43
	Elasticity	0.03
	Support	0.05
FCM6 (Security)	**Authentication & Authorization**	**0.56**
	Auditability	0.43
	Reliability	0.50
FCM7 (Data)	Privacy / Confidentiality	-0.01
	Data Access / Import-Export	**0.56**
	Backup	0.26
	Disaster Recovery	0.48

and with a negative value in 43% of them. This proves that the map's tendency is towards positive values, something that was already suggested by the majority of the positive cycles in the static analysis. Therefore, in this case, if the desired modeling outcome should have been a perfectly balanced map where the number of cycles would be (nearly) the same and this tendency should have been neutralized, the modelers should revisit the structure of the map with focus on studying and possibly changing the sign of the cycles present. This, of

course, does not mean that one should arbitrarily change the map to fix possible anomalies introduced by the cycles; on the contrary, this finding can be used as guidance for modelers to seek corrections by introducing new concepts or removing less significant ones so that a balance is reached cycle-wise. In the case under study and aiming to decrease complexity, it was decided to remove concepts, but this will be performed after gathering some more information via experimentation regarding the significance of the current concepts, which will provide more insights as to which nodes have the least contribution to the final outcome and thus may be deleted.

Since *Cost* was confirmed through static and dynamic analysis to be a key concept with strong influence to the final activation level of the central node of the root map, further investigation was deemed necessary to investigate how this concept receives its value from its sub-model FCM4. Several simulations were set up and executed focusing on the concepts that comprise FCM4, based on the findings of the static and dynamic analysis. These scenarios aimed to confirm or question the significant role of *ROI*, *Pricing* and *Running Operational Cost* concepts on *Cost*. A very weak value (0.1 or -0.1) was selected for the initial activation levels of the rest of the nodes in all scenarios, so that by setting a high value to the strong nodes any change in the behavior of the model may be observed. Having established from the simulations thus far that *Cost* positively affects *Cloud Adoption*, it will now be examined exactly how the value of *Cost* is affected in the FCM4 environment. The corresponding static analysis results clearly indicated that both the *ROI* and *Pricing* nodes decisively influence *Cost* but with opposite signs. Several scenarios were run with different values and confirmed that *Cost* is led to positive or negative values, with the strongest influence received from *ROI*. Nevertheless, *Pricing* is also a very decisive factor, which, combined with *ROI*, directs the evolution of the final activation level for *Cost*. To make this clearer, an indicative numerical example is the following: Having set *ROI*=0.9 we varied *Pricing* from -0.1 to -0.2. The value of *Cost* is initially calculated by the model to be equal to -0.78, while that of Cloud Adoption rises to 0.70. The small decrease by 0.1 in the value of *Pricing* leads *Cost* to -0.82, a slight change compared to its precious value, but nevertheless enough to revert *Cloud Adoption's* value to the negative side, calculated to be equal to -0.55. Therefore, it may additionally be inferred that the *Pricing* node can have a serious and decisive contribution to the final value of *Cloud Adoption* through the transfer of the value of concept *Cost* from its decomposed parts. Similar investigation was performed with *Running Operational Cost*, which revealed the exact same contribution to *Cost*. Such sensitivity of the model to certain nodes can easily be revealed by following specific paths for executing the model with different scenarios. This

is exactly where the framework can be extremely useful to modelers and domain experts.

Similar analysis was conducted for FCM6 and FCM7 to investigate *Security* and *Data* (Management) concepts that appeared to have a very strong influence on the final decision of the model. In the case of FCM6, the participating nodes appear to have the same contribution to the final value of *Security* in all scenarios, wherein opposite initial values were mutually neutralized. Setting positive values to all nodes of the FCM6 drives *Security* to a strong positive value which is then transferred to the root map. FCM7 seems to exhibit similar behavior to FCM6 except that on this map the node *Data Access / Import-Export* has a very strong influence on the local central concept *Data (Management)* and it was verified that it prevails and overlaps the influence of the remaining nodes. High initial values for that node determine the value of *Data (Management)* no matter the value or sign of the activation level of the other nodes.

The investigation in relation to the actions that may be taken can be concluded with the following: Since the model behaved consistently with the extreme scenarios there was no need to revise it to achieve correct behavior. Further to that, the simplification of the model's structure was also investigated by removing the three nodes *QoS Assurance*, *Exit Process* and *Migration Complexity* that were characterized as the most "weak". The simplified MLFCM executed the extreme scenarios once again and the behavior did not change. Therefore, this simplification may be applied permanently to the model under analysis without loss of accuracy.

4.5 Summary

This chapter attempted to tackle the complex problem of the adoption of cloud computing with the use of approaches coming from the area of computational intelligence. Specifically, the application of the two methodologies was examined based on FCMs and IDs. Four different models, two of each methodology, were constructed following a systematic process that involved a literature review and expert's opinion through interviews and questionnaires. All four models were successfully evaluated against extreme scenarios and have shown considerable success in real-world cases. The modeling process highlighted the advantages and disadvantages of each method and what a decision-maker should know when choosing an approach. Furthermore, the benefits from the interactive capabilities of both approaches were studied by executing simulations based on dedicated what-if scenarios and locating the effect of each factor to the problem under study. Particular emphasis was given to MLFCMs and

extensive experimentation was conducted through the application of the static and dynamic analyses , as described in Chapter 3 which revealed very interesting and helpful, to the decision making process, findings.

Chapter 5

A Novel Computational Approach for MLFCM

5.1 Introduction

As the complexity of a problem grows, so does the uncertainty inherent in the results that a modeling approach delivers. This fact underlines the importance and need for techniques that can be interpreted by humans, and justifies the growing interest of the research community [73]. As models and techniques delivering predictions, and in general support decision making, become more complex, the task of producing a transparent version becomes more difficult. In many cases, companies or organizations are reluctant to adopt such kind of technologies due to their weaknesses in explaining the results. The composite challenge that arises is the design and development of models that are able to overcome bias, performance and explainability issues in a balanced coexistence.

Towards the delivery of a sufficient model with enhanced features, this chapter introduces a new MLFCM computational process which involves two new approaches. The first approach comprises a new FCM formulation that handles a number of weaknesses and drawbacks of existing methods, and boosts up the abilities for multidimensional and multi-targeted analyses. The second approach introduces the utilization of a genetically evolved algorithm as an extension of the dynamic analysis. This algorithm evolves the initial activation levels of the concepts participating in the MLFCM aiming at finding solutions that satisfy a target final activation value (i.e. after execution of the map) of the concept(s) of interest which correspond to a certain scenario. This work can be considered as an extension of the work introduced in

Chapter 3 and proposes an integrated framework with a novel model able to cope with the challenges imposed by the need to analyze and explain complex decisions.

5.2 Literature Overview

The need for delivering intelligent models that could simultaneously achieve both the goals of accuracy and the ability to explain their decisions, led many research groups to focus on the so-called explainable Artificial Intelligence (XAI). As a result, the number of relevant publications has a significant increase in recent years. A comprehensive background of the field is introduced in [74], with the authors attempting to address a wide range of relevant aspects. A particular reference to a number of definitions, approaches and applications is made, while conclusively, the need for further and deeper research is underlined. The research work in [75] quotes the current state of the research field on machine learning interpretability through an extensive literature review. A special reference is made on the social impact, the developed methods and metrics, and, finally, on the future directions for the work that needs to be done. An overview of the relevant field is conducted in [76] which mostly focuses on concepts related to XAI models and on an XAI literature taxonomy in terms of explainability of different AI models.

5.3 Computational Approach

Despite the promising and successful results yielded by the application of FCM models on various systems and problems, such type of modelling suffers from certain drawbacks and limitations [77]. This section describes an attempt to provide solutions that tackle the following drawbacks: (i) The activation level of a concept may change during the iterative execution of the map, even if the specific concept is not influenced by any other concept. Normally, this should not be allowed; on the contrary, the concept must bear the same activation level at all discrete execution steps. (ii) Due to the implementation of the sigmoid function (Equation 2.2), the final activation values are confined to the interval [0.5, 1] and therefore the lower part of the (0,1) fuzzification scale remains unused. (iii) The ability of the model to converge to reasonable values, even for $\lambda < 0$, is limited. (iv) It is occasionally observed that certain concept activation levels converge to the same output no matter the initial activation value. (v) The representation and calculation of the initial activation level for inactive concepts must be provided.

A new FCM mathematical formulation is introduced here aiming to address the aforementioned

problems and improve the capabilities of the model. This new formulation is described in Equations 5.1 to 5.4.

$$A_i^{t+1} = \begin{cases} f\left(B^t + C^t\right), & B^t \neq 0 \\ \\ A_i^t, & B^t = 0 \end{cases} \tag{5.1}$$

where:

$$B^t = \sum_{j=1, i \neq j}^{n} w_{ji} \Delta A_j^t \tag{5.2}$$

where:

$$\Delta A_j^t = \begin{cases} A_j^t - A_j^{t-1}, & t \geq 1 \\ \\ 0, & t = 0 \end{cases} \tag{5.3}$$

and:

$$C^t = \begin{cases} f^{-1}(x) = \frac{ln(\frac{1-A_i^t}{A_i^t})}{-\lambda}, & A_i^t \neq 0 \\ \\ 0, & A_i^t = 0 \end{cases} \tag{5.4}$$

Equation 5.1 is the general form of the new evolved formulation that is derived from Equation 2.1. The lower part of the new equation handles the case of leaving the activation level value unchanged through the iterative process in case of a concept with zero external influence, $B^t = 0$. The calculation of B^t, is given by Equations 5.2 and 5.3 and suggests the following reasoning: The new value A_i^{t+1} is affected by the change ΔA_j^t and not by A_j^t directly. This approach was proposed by Vergini and Groumpos in [78] and addresses the problem of convergence to the same value irrespective of the initial value. Due to the fact that an activation level can take values near to zero but never exactly zero during either the defuzzification process or the iterative execution, it is considered that activation values fall within the interval [0^+, 1]. This allows the possibility of assigning the zero value to concepts that have unknown initial activation levels. In such a case, $A_i^t = 0$ in the calculation of the new concept activation

value and therefore, only the value of the influence it receives from B^t is taken into account. Otherwise, with $A_i^t \neq 0$, the contribution of C^t in Equation 5.1 represents the value of A_i^t before applying to it the activation function of Equation 2.2. The latter two claims are described by Equation 5.4. This "anti-bounded" method is achieved with the application of the inverse sigmoid function over the A_i^t value and restores the correct sign in the aggregated value $B^t + C^t$. Subsequently, the new calculated activation levels can also take values below 0.5 .

5.3.1 Activation Levels Genetically Evolved MLFCM

Aiming at enhancing and strengthening the dynamic analysis, and, therefore, the explainability of the model, this work proposes the Activation Levels Genetically Evolved MLFCM (ALGE-MLFCM). This new approach integrates a Genetic Algorithm (GA) for the production of a set of near-optimal solutions in the form of initial activation values for achieving particular targeted final activation values throughout the multilayered model's structure. To the best of our knowledge, this is the first time that the evolution of the set of initial activation levels is reported in literature. The use and integration of GAs for the evolution of the activation values, keeping the weight values constant, became feasible with the adoption of the new formulation proposed in Section 5.3 and described by Equations 5.1 to 5.4.

Genetic algorithms are the most popular type of Evolutionary Computation (EC) algorithms and belong to the area of Artificial and Computational Intelligence [79]. The evolution process is based on a probabilistic algorithm that maintains a population P_t, of individuals x_i for each generation t ($t \in [1..G]$) with G the maximum number of generations set aiming to optimize an objective function φ. Each individual x_i represents a potential solution to the problem at hand and it is evaluated using some measure of its "fitness". While the termination conditions are not satisfied, the new population P_{t+1} is formed following the *Selection*, *Crossover* and *Mutation* steps. After some number of generations the algorithm terminates and the best individual is expected to represent a near optimum solution.

The proposed ALGE-MLFCM is targeting an optimal set of initial activation values vector that result in one or more desired final activation values as these are calculated by the MLFCM model. The fitness of each population P is calculated by the following equation:

$$Fitness(P) = \frac{1}{1 - \frac{\sum_{i=1}^{N} fitness(x_i)}{N}} \tag{5.5}$$

where N is the size of the population and $fitness(x_i)$ is calculated by:

$$fitness(x_i) = \frac{\sum_{j=1}^{M}(Y_j - T_j)}{M}, Y_j, T_j \in [0..1] \tag{5.6}$$

where M is the number of objectives, Y_j and T_j are the calculated value and the target value of objective j respectively. Objectives in this case correspond to the desired value of specific concepts.

The structure of the proposed algorithm in a pseudo code form is shown in Algorithm 2.

Algorithm 2 The ALGE-MLFCM algorithm in pseudo-code

```
#The ALGE-MLFCM algorithm
t = 0
InitializeP(t)
EvaluatePopulation(P(t))

while not(termination condition) do
    t = t + 1
    #Evolution of the population
    P_new = Selection() -> Crossover() -> Mutation()
    P_t = P_new
    EvaluatePopulation(P(t))
Return(best individual x_i in P(t))
procedure EVALUATEPOPULATION(P)
    for (each individual x_i in P) do
        FinalActivationValues(x_i) = MLFCM(x_i)
        SumFitness += Fitness(FinalActivationValues(x_i))
    Fitness(P) = Mean(SumFitness)
```

The utilization of the proposed algorithm offers enhanced capabilities to the model's dynamic analysis and also in the analysis and study of real cases. A series of actions, that complement the action sets introduced in Section 3.5, may be designed and executed depending on the desired type of analysis or study. Below, three such indicative actions are described and explained.

Action 1: Identify the tendency of the map through the execution of simulations over random scenarios. Specifically, with random initial activation values the ALGE-MLFCM algorithm is

called to search for solutions that lead the main concept of interest (objective) to a particular value i.e. to ”Very High” or to ”Very Low”. After conducting a relatively large number of simulations, by examining the center of gravity of the convergence speed for each objective value separately, an inference for the map’s tendency becomes possible.

Action 2: Working with the same pattern as in the previous step, useful conclusions can be extracted by analyzing the suggested solutions that have emerged for targeted values of one or more objectives. Targeted simulations can also be executed by keeping constant, and out of the evolution process, initial activation values corresponding to one or more concepts.

Action 3: The comparison of any set of initial activation values that reflect a specific case study with solutions from a reference list, can provide explanation of success or failure of meeting the target value. Specifically, once the unique solutions provided by ALGE-MLFCM algorithm are filtered and cleared, the resulting list can be used as a comparison reference for one or more specific target values. Furthermore, through this exercise, the changes to the initial values that need to be made so as to achieve the desired value(s) for the concept(s) of interest may be defined. Finally, it should be mentioned that in each of the aforementioned actions, the resulting solutions must be processed and filtered before being adopted as potential solutions. This is mainly due to the fact that solutions may be produced that do not describe well the problem under study and should be discarded.

5.4 Summary

This chapter introduced a new MLFCM computational process, which relies on two new approaches and aims to provide enhanced decision support capabilities and increased explainability. The first approach is dealing with a new formulation for the computational steps, which addresses several known FCM weaknesses and drawbacks. The second approach involves an extended dynamic analysis with the utilization of a new genetically evolved algorithm, the ALGE-MLFCM. The algorithm is able to deliver a set of solutions in terms of initial activation levels that drive the model to yield particular targeted final activation values, thus enabling the study of particular scenarios of interest and their outcome. This type of analysis also addresses the challenge of designing models that are able to overcome bias, increase performance, and improve explainability.

Chapter 6

Supporting the Decision of Migrating to Microservices Architecture

6.1 Introduction

Microservice architectures are the new weapon-of-choice for the development of cloud-native applications as suites of small, autonomous, and conversational services, which are then easy to understand, deploy, and scale. Microservice architectures enable optimizing the autonomy, replaceability, and decentralized governance of software systems. Despite the hype for microservices, both industry and academia still lack consensus on the adequate conditions to embrace and benefit from this new paradigm [6]. Most organizations and their on-premise application architectures are not ready to fully exploit the benefits of microservices, and adapting to this environment is a non-trivial task [5], since such architectures are highly complex, and comprise multiple, often conflicting factors. Moreover, projects intending to adopt microservices from scratch risk to pay the *microservices premium* price, entailing extra complexity and cost where it was not actually needed, given the requirements of the system under construction [80]. Therefore, the study and the analysis of the factors forming the environment behind the decision of adopting microservices, either for migrating a legacy, monolithic system, or for developing a new one from scratch, is of paramount importance for different stakeholders. Particularly, an approach to assist in the decision making process is still needed [81].

The importance and the need of human interpretable techniques increasingly concern the research community [73]. As models and techniques delivering predictions, and in general

support for decision making, become more complex, the task of producing a transparent version becomes more difficult. In many cases, companies or organizations are reluctant to adopt such kind of technologies due to their weaknesses in explaining the results. The composite challenge that arises is the design and development of models that are able to overcome bias, performance and explainability issues in a balanced coexistence. The research work described in this chapter picks up this challenge and introduces a fine tuned model to support the decision of adopting microservices architecture in a more sophisticated form with enhanced explainability and interpretability features.

The construction of the proposed model starts with a literature review conducted to identify an initial set of concepts that potentially influences the decision of adopting microservices. When the set of concepts is finalized, a group of experts with related background on the subject, are called to refine and organize the identified concepts, as well as to define their relationships (causalities). The outcome of this process is a model in the form of a Multi-Layer Fuzzy Cognitive Map (MLFCM), a graph-based computational intelligent model that captures the behavior of a given problem in nodes which essentially represent knowledge in the domain, and edges that represent their influence and interrelation. Towards the delivery of a sufficient model with enhanced features, the construction process involves the analysis framework introduced in Chapter 3 and the new computational approach introcuced in 5.

Over the resulting MLFCM a two step validation and calibration process is performed. The first step is to execute two ”extreme” (synthetic) scenarios to drive the model to a positive and negative outcome respectively and provide a first level of assessment. The second step includes the execution of the model over a number of real-world cases targeting at calibrating its accuracy in terms of the actual decision taken.

Once the model is fine-tuned through the validation and calibration process, both a static and a dynamic analysis y performed, with graph analysis and simulations execution respectively. These simulations enable the investigation of whether such intelligent techniques can help reveal insights about the behavior of the problem under study and find answers to hypothetical but very likely situations (scenarios) upfront.

Finally, the model is applied over an industrial case study. In this case the model and its supporting procedures are applied to a completed migration project by assessing the final outcome and running simulations to the benefit of the company’s decision makers with whom extensive discussions took place to assess the results and predictive ability of the model.

6.2 Literature Overview

From the industrial perspective, Netflix was one of the early adopters of microservices, transitioning from a traditional development model with hundreds of engineers maintaining a monolithic video-on-demand application, to many small teams responsible for the end-to-end development of hundreds of microservices to serve millions of users on a daily basis[1]. The main goals for this migration at Netflix were three. First, to speed-up the development process, by providing the corresponding automation tooling. Second, to embrace cloud-based virtualization technologies, such as virtual machines and containerization, that drastically increase the performance and scalability of the architecture. Third, to isolate problems and failures by adopting novel design-for-failure patterns, such as circuit-breakers[2].

At organizational level, Netflix shifted from traditional siloed teams to product-oriented teams following a DevOps methodology [82]: each microservice is owned by a team that handles its whole lifecycle, from conception through deployment to operation. This also fosters continuous delivery, where each microservice can be containerized and updated independently of the others [83].

Similarly, SoundCloud also started as a single, monolithic application[3] and shifted to microservices to support the increasing demand (about 12 hours of music uploaded per minute, and hundreds of millions of users). This was achieved by slicing domain logic into very small components, designed according to the Bounded Context principle [84] and exposing a well-defined API.

The academia is still in an early stage of documenting and analyzing the adoption of microservices that is taking place in industry [85, 86]. Balalaie et al. [87] document the steps for the microservification of an on-premise Software-as-a-Service (SaaS) platform. According to the authors' experience, the main drivers for adopting microservices are the need for reusability (of cohesive functionality in the form of microservices), decentralized data governance (since a shared database blurs the boundaries among different services), automated deployment (to decouple the build lifecycle of each service), and scalability (fine tuned for each individual microservice through load-balancing and service discovery).

Taibi et al. [86] performed a survey among industry practitioners who adopted microservices,

[1] `https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/`

[2] http://microservices.io/patterns/reliability/circuit-breaker.html

[3] https://developers.soundcloud.com/blog/building-products-at-soundcloud-part-1-dealing-with-the-monolith

in order to analyze the motivations, as well as the pros and cons, of migrating from monolithic to microservice architectures. Maintainability problems were rated as a very important driver, whilst scalability, delegation of responsibility to independent teams, and easy support for DevOps also frequently drive adoption. Interestingly, several practitioners reported to be adopting microservice architectures because "a lot of other companies are adopting them". Then, the authors propose a framework for the migration process based on the feedback from practitioners.

To summarize, almost all successful stories about microservices adoption have started with a monolith that grew too much and then had to be decomposed into microservices. However, how to decide whether the conditions are favorable to adopt microservices [88] is still unclear, and "because other companies are adopting them" cannot be the main driver [86]. The suitability of a Decision Support System (DSS) in the context of such a decision is also suggested in [81], based on reference models for enterprise architectures.

6.3 A Novel MLFCM Model

As mentioned earlier, the main contribution of this chapter is to propose and deliver an enhanced MLFCM model to support the decision of adopting microservices architecture with a dedicated structure, new execution formulas and innovative supportive methods of analysis compared to what has been previously proposed in literature. The following paragraphs describe in detail the actions performed towards the development of this model.

6.3.1 Model Construction

The development of the proposed model followed a multistep process, with the first step being a Literature Review (LR) that helped to identify the concepts that are potentially relevant to the decision of migrating to microservices architecture. This process followed the key guidelines proposed in [89]. Although a full, systematic review is outside the scope of this work, the process of finding and classifying relevant works was organized carefully. Search was performed for articles in the broader topic of microservices, indexed in different online databases, namely Scopus [90], Science Direct [91], Wiley Online [92], IEEExplore [93], Springer Link [94] and ACM [95]. The search strings used were "microservices", "microservice architecture(s)", "microservice migration", and "microservice adoption", with publication year up to 2018. Then, snowballing [96] was applied, by looking at the relevant references included in the

works originally found and trying to identify other potentially relevant ones. Both journal and conference/workshop articles were considered, while duplicates were removed.

The initial list of concepts that emerged from the previous step, was shared with a group of experts consisted of researchers, as well as industry practitioners with background related to the subject. The experts evaluated the list and provided suggestions to add, remove, group, or decompose concepts working with the same rationale as described before. This process concluded with the list of concepts and their groupings presented in Table 6.1. The resulting list of concepts was organized and shaped as a MLFCM with a total of six sub-FCMs divided into two layers (see also figure 6.2).

Based on the identified concepts and groups/decomposition, a second round of feedback collection through structured interviews with the experts was conducted. This time the experts were asked to complete a questionnaire concerning the causal relationships between concepts and their weights, i.e., the degree to which concepts influence each other. Such a degree of influence was fuzzified using seven linguistic values (from *negatively high* to *positively high*, see Figure 6.1(a)). For example, if an expert considers that *Security* strongly affects *Reliability* in a positive way (i.e. when *Security* rises *Reliability* rises as well), then in the questionnaire she marks the corresponding cell as ”positively high”. Conversely, if *Governance* does not affect *Cost*, she marks their relationship as ”Neutral”.

After the experts defined the causal relationships between concepts as described above, they declared their degree of confidence (fuzzified from ”low” to ”very high”), which was used to form a weighted average of causality scores. The calculation of the level of causality from the concept i to concept j was performed by the formula in Equation 6.1, where N is the number of experts and C_k and W_k are the levels of confidence and influence respectively for each expert k.

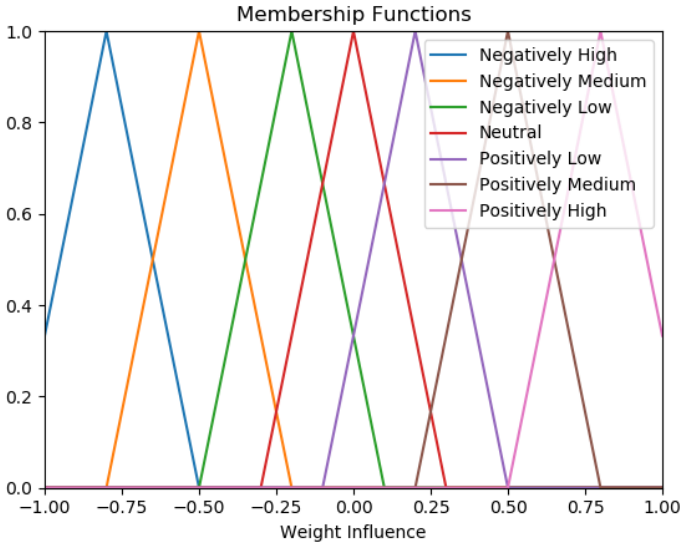
$$w_{i,j} = \frac{\sum_{k=1}^{N} W_k C_k}{\sum_{k=1}^{N} C_k} \tag{6.1}$$

The fuzzification method for both the activation levels and the relationship (weight) values utilized the triangular membership function [97]. The triangular function allows one to map and normalize the linguistic scale to the range $[0,1]$ for the activation values, and $[-1,1]$ for the relationships. The membership functions used to represent the fuzzified linguistic values for the causal relationships between concepts and their activation levels, are depicted in Figure 6.1 ((a) and (b) respectively).

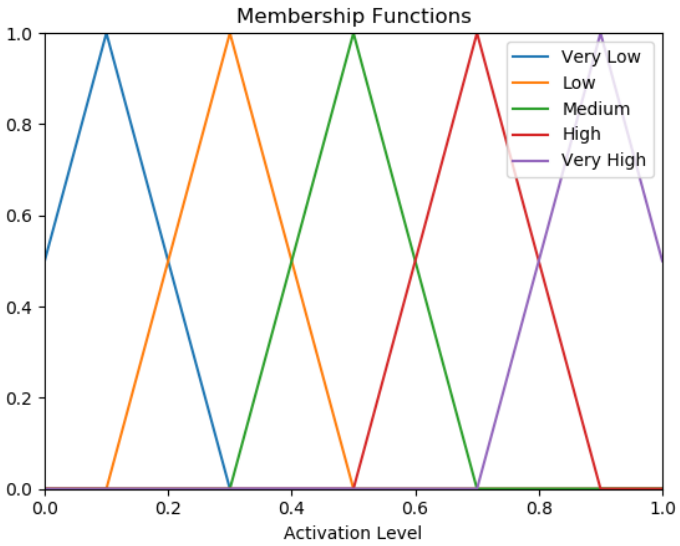
Table 6.1: Concepts related to the decision of adopting microservice architectures, and their groupings (FCMs) derived from literature review and evaluated by experts (central concept of each FCM in bold).

No.	FCM	Concept Name
C1	**1, 2**	**Governance**
C2	**1, 3**	**Infrastructure and Management Services**
C3	1	Maintainability and Evolvability
C4	1	Operational Complexity
C5	1	Business Complexity
C6	1	Reliability
C7	1	Security
C8	**1, 4**	**Cost**
C9	**1, 5**	**Design**
C10	**1, 6**	**DevOps**
C11	1	Data Migration
C12	2	Decentralized Governance
C13	2	Data Governance
C14	3	Containerization
C15	3	Scalability/Elasticity
C16	3	Monitoring
C17	3	Serverless Architecture
C18	4	Migration Cost
C19	4	Operations Cost
C20	5	Design For Failure
C21	5	Granularity and Bounded Context
C22	5	Service Contracts
C23	5	Communication Model
C24	5	Decentralization
C25	6	Organization Culture
C26	6	Skilled and Educated DevOps Teams
C27	6	Tool Support
C28	6	Continues Activities
C29	6	Automated Tasks
C30	6	Information Sharing
C31	**1**	**Microservice Adoption**

All linguistic variables proposed by the experts were aggregated using a weighted average scheme and deffuzzified following the Center of Gravity (COG) method [98]. The result of the process is that of several normalized weight matrices, one for each sub-FCM, which numerically represent the concepts and their relationships as values in the interval [0, 1] and [-1, 1] respectively. Table 6.2 shows the weight matrix for concepts in the main FCM



(a) Positive Scenario



(b) Negative Scenario

Figure 6.1: Fuzzification of linguistic variables according to triangular membership functions: (a) seven values, (b) five values

(FCM1). The last column shows the influence of each concept over the central one (C31 - Microservices Adoption), according to the experts' feedback. For example, *Operational Complexity (C4)* has a negative influence over *C31*, with a weight value of -0.5. This suggests that a high operational complexity in a given situation will go against the decision of adopting microservices.

Table 6.2: Normalized Weight Matrix for causal relationships in FCM1

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C31
C1	0	-0.2	0.5	0.8	0	-0.2	-0.5	-0.5	0	0.5	-0.3	0.8
C2	0.5	0	0.5	0.8	0	0.2	0.2	0.5	-0.5	0.5	0.3	0.5
C3	0.5	0	0	-0.2	0	0.5	0.5	-0.5	0.5	0.5	0.2	0.5
C4	-0.5	0.8	-0.5	0	0	-0.2	-0.2	0.5	-0.2	-0.5	0.3	-0.5
C5	-0.5	0	-0.5	0	0	0	0	0.8	-0.8	-0.5	0.2	0.8
C6	0.5	0.5	0.5	0.5	0	0	0.5	-0.5	-0.5	-0.2	0.1	0.2
C7	-0.2	0.8	-0.5	0.8	0	0.5	0	0.5	-0.2	-0.2	-0.3	-0.5
C8	0	0	0	0	0	0	0	0	0	0	0	-0.8
C9	0.8	-0.2	0.8	-0.5	-0.2	0.8	0.8	-0.5	0	0.8	-0.3	0.8
C10	0.5	0	0.5	-0.5	0	0.5	-0.2	0.5	0	0	0.3	0.8
C11	-0.1	-0.2	-0.3	0.4	0.2	-0.1	-0.4	0.7	-0.4	0	0	-0.5
C31	0	0	0	0	0	0	0	0	0	0	0	0

The visual representation of the final map structure (derived from the weight matrices) consists of FCM1 at the top layer and five sub-FCMs at the lower layer, as depicted in Figure 6.2.

6.3.2 Model Validation and Calibration

The performance of the resulted model was validated over two realistic hypothetical scenarios, which were constructed and executed to assess whether the model yields a reasonable output-answer, as this expressed by the final activation level of the central concept of interest, *Microservice Adoption*. Two “extreme” positive and negative scenarios were produced and executed, which were expected to drive the model to the extreme positive or negative outcome respectively. For the positive scenario the environment was set in favor of the *Microservices Adoption* and, inversely, the initial values in the negative scenario were selected to reflect a negative situation so as to guide the central concept to a negative response (i.e., close to 0 value).

The results of the two scenarios described above are depicted in Figure 6.3. It is easily discernible that in both executions the model behaved as expected and correctly led the concept of interest to the desired outcome. Specifically, as shown in Figure 6.3a, the model successfully recognized the positive environment and yielded the value of 0.83 that corresponds to the ”Very High” linguistic value. Similarly, the model successfully recognized the negative environment that was set against the *Microservices Adoption* and converged to 0.22 (”Low”)

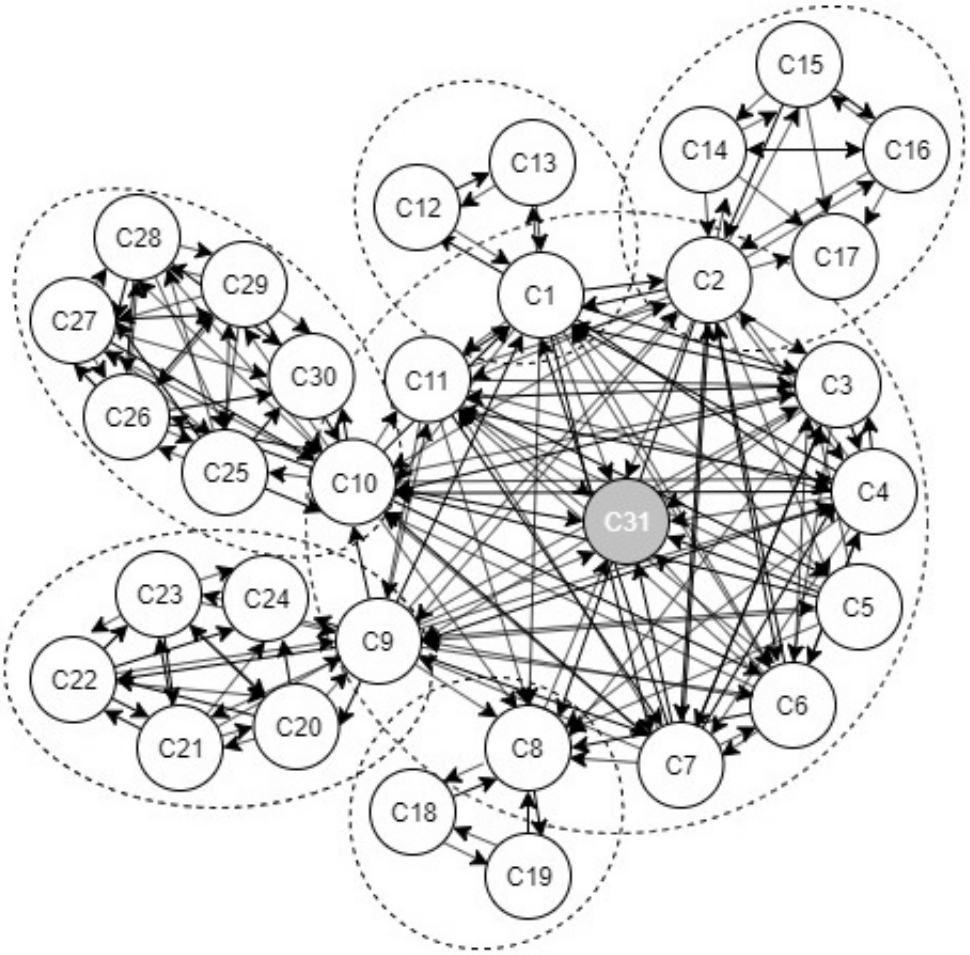
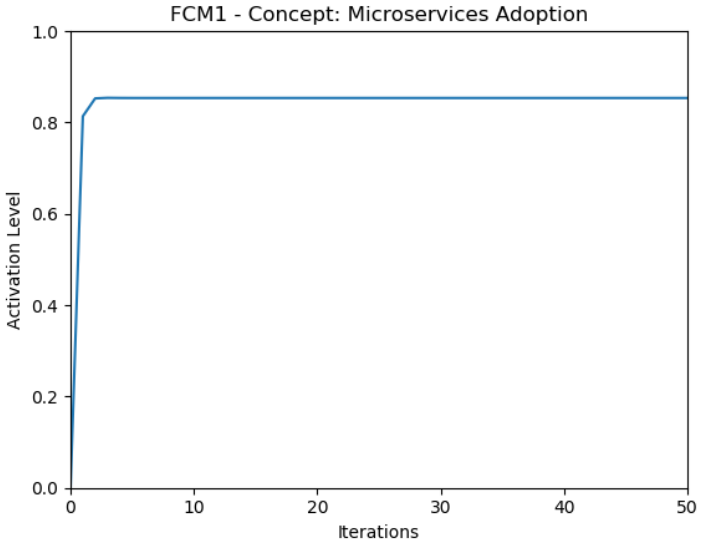


Figure 6.2: Visual representation of the MLFCM model for the Microservices Architecture adoption

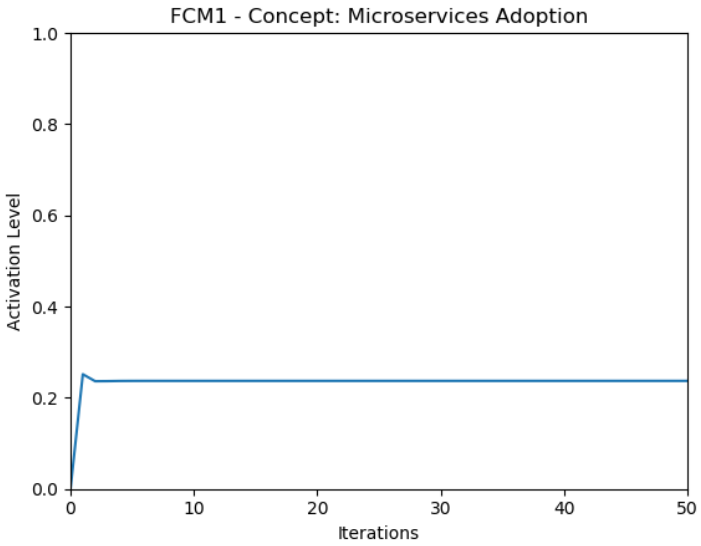
(Figure 6.3b).

The model's calibration process that followed, incorporated a series of experiments over real-world cases, in order to identify the "optimal" λ value with which the model achieves the best performance, i.e. the most correct answers compared to the actual ones. Industrial case studies were gathered through a focused questionnaire[4] distributed among industry practitioners with long experience in software development projects in large companies. The profile of the survey participants is shown in Figure 6.4.

[4]Survey available at: `https://goo.gl/5mbHgN`



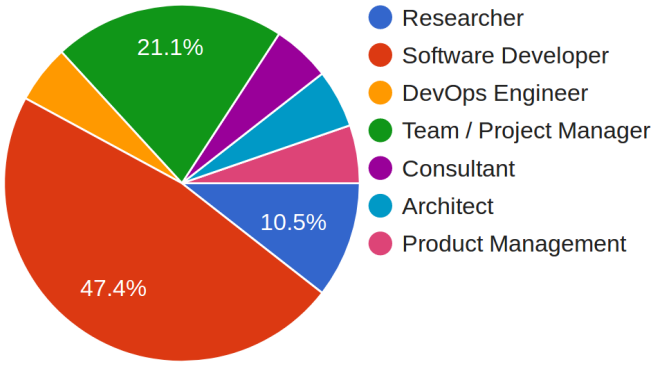
(a) Positive Scenario



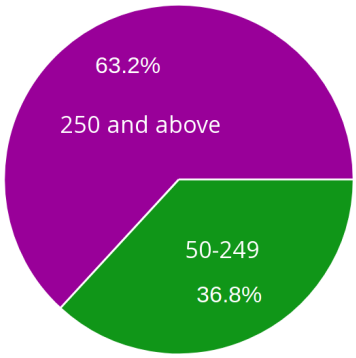
(b) Negative Scenario

Figure 6.3: Model validation over two extreme scenarios

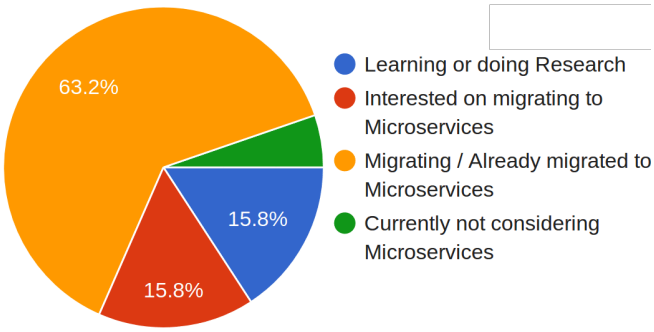
The 16 cases gathered essentially defined the initial activation values of the concepts in the model. These cases were then used to compare the outcome with the actual decision using different λ-values (Table 6.3). Particularly, the bottom row of the table shows a distance value, which denotes the difference between the actual decision taken and the output of the model, where the lower the value the better the result. The best result (i.e. the lowest distance) is 8 with $\lambda = 0.5$, having 56% of the cases matching the expected outcome, and the other 44%



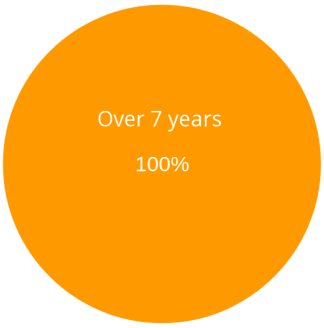
(a) Main Occupation



(b) Size of the Company



(c) Phase of Interest



(d) Age of the Company

Figure 6.4: The profile of the survey participants

with a deviation of only one degree in the fuzzification scale. One may note that there is a slight positive predisposition towards adoption in comparison with the experts' answers; this will be investigated during the static analysis of the model, that is, whether the model tends to behave positively. In such a case, the selection of the λ-value to lay below 1 will restrain the final values at lower levels.

6.4 Static and Dynamic Model Analyses

Aiming to investigate the model's behaviour and form a better understanding of its internal features properties and features, a series of steps were applied that belong to two classes of analysis, static and dynamic. In particular, the step-wise process introduced in [36] was followed, enhanced by the new evolutionary approach proposed in this work which extends and strengthens the dynamic analysis part.

Table 6.3: Industrial Case Studies

number	actual	$\lambda = 0.2$		$\lambda = 0.3$		$\lambda = 0.4$		$\lambda = 0.5$		$\lambda = 0.6$	
		num	ling.	num	ling.	num	ling.	num	ling.	num	ling.
1	Med	0.593	**Med**	0.639	High	0.683	High	0.723	High	0.760	High
2	High	0.589	Med	0.634	**High**	0.678	**High**	0.719	**High**	0.757	**High**
3	High	0.598	Med	0.648	**High**	0.696	**High**	0.740	**High**	0.779	**High**
4	High	0.576	Med	0.618	**High**	0.660	**High**	0.700	**High**	0.738	**High**
5	V.High	0.580	Med	0.622	High	0.662	High	0.701	High	0.737	High
6	High	0.580	Med	0.622	**High**	0.662	**High**	0.700	**High**	0.736	**High**
7	High	0.610	**High**	0.663	**High**	0.711	**High**	0.755	**High**	0.792	**High**
8	Low	0.530	Med	0.550	Med	0.573	Med	0.600	Med	0.630	High
9	Med	0.566	**Med**	0.604	High	0.643	High	0.682	High	0.720	High
10	High	0.564	Med	0.601	**High**	0.641	**High**	0.683	**High**	0.724	**High**
11	High	0.546	Med	0.571	Med	0.597	Med	0.625	**High**	0.655	**High**
12	V.High	0.578	Med	0.618	High	0.657	High	0.694	High	0.727	High
13	High	0.572	Med	0.615	**High**	0.659	**High**	0.703	**High**	0.745	**High**
14	Med	0.504	**Med**	0.511	**Med**	0.521	**Med**	0.537	**Med**	0.558	**Med**
15	V.High	0.578	Med	0.618	High	0.657	High	0.696	High	0.732	High
16	Med	0.572	**Med**	0.610	High	0.648	High	0.684	High	0.718	High
	Distance		**14**		**8**		**8**		**7**		**8**

6.4.1 Static Analysis

Static analysis examines the properties of a model prior to its execution irrespectively of its behavior during run-time, by exploiting the structure of the FCM. The graph-based representation of FCMs and their consideration as directed weighted graphs, allow the application of methods and metrics from the area of graph theory and offer important information about the model's structure and its constituent elements, as well as their interactions.

Three major indicators were used for the static analysis of our FCM model: (i) *Complexity*, which comprises *density* (number of nodes and interactions), *depth* (number of layers), and *breadth* (sub-FCMs and their nodes), deals with the characterization of the model in terms of complexity of its structure. (ii) *Strength*, comprises metrics which indicate the significance of each node in the model and are calculated using the weight and number of its incoming

(*in-value, in-degree*) and outgoing (*out-value, out-degree*) edges. Finally, (iii) *Tendency*, which counts the total number of feedback cycles in the graph (positive and negative), and also the number of cycles in which each node takes part and it is considered as a strong indicator of the tendency of the map.

Table 6.4: Complexity static measurements.

FCM	Layer	Density	Cycles+	Cycles-
1	1	0.72	64642	65900
2	2	1	5	0
3	2	1	68	16
4	2	1	5	0
5	2	1	409	0
6	2	1	2365	0

Table 6.4, presents the applied metrics and measurements regarding the map's complexity and tendency. The density appears to be very high for all sub-FCMs and, thus, one can easily claim that the two-layer structure model is quite complex. As regards the tendency of the model, this is reflected in the number of feedback cycles of each sub-FCM. The main FCM is differentiated by introducing more negative cycles than positive, while for each individual FCM the ratio between positive and negative cycles is greater than one. This result makes the determination of the general tendency of the model unclear and suggests a deeper and further study and analysis.

Aiming to identify the strength and the significance of each concept in the model, a series of measurements were recorded for all sub-FCMs. These measurements show how strong the presence of each node is in the sub-FCM it belongs to, by calculating the total number of incoming and outgoing degrees and values, as well as the number of positive and negative cycles containing that node. The corresponding results are provided in Table 6.5 and the discussion over the main findings follows.

Examining the results derived from the static analysis, various indications regarding the strongest concepts are given in boldface letters. By combining the results of the corresponding measurements, one may argue that the top three concepts of the main FCM are *Operational Complexity (C4)*, *Design (C9)* and *Data Migration (C11)*. This finding calls for further attention to the behavior of these concepts individually on one hand, and as a group on the other. Moreover, *Design* should be further decomposed to understand which factors influence

Table 6.5: Strength indicators for the sub-FCMs of the MLFCM model for the Microservices Adoption problem.

FCM	Node	$deg_{tot}(i)$	$val_{tot}(i)$	Cycles+	Cycles-
1	C1	18	8.4	56703	57479
	C2	16	7.2	53897	54464
	C3	18	8.5	56149	57610
	C4	18	**8.7**	56724	57633
	C5	9	4.5	30705	31338
	C6	18	7	56355	58002
	C7	18	7.8	56445	57912
	C8	11	6.3	0	0
	C9	18	**9.6**	**57279**	**58307**
	C10	16	7.5	54070	55041
	C11	**19**	5.6	**58390**	**59921**
	C31	11	6.7	0	0
2	C1	4	1.3	4	0
	C12	4	**2**	4	0
	C13	4	1.4	4	0
3	**C2**	8	**4.2**	53	11
	C14	8	3.5	53	11
	C15	8	3.8	53	11
	C16	8	2.1	48	16
	C17	8	3.8	48	16
4	**C8**	4	**1.7**	4	0
	C18	4	1.3	4	0
	C19	4	1.2	4	0
5	C9	10	4.3	325	0
	C20	10	3.2	325	0
	C21	10	4.2	325	0
	C22	10	3.5	325	0
	C23	10	2.8	325	0
	C24	10	**4.4**	325	0
6	**C10**	12	**7.3**	1956	0
	C25	12	5.1	1956	0
	C26	12	5.1	1956	0
	C27	12	4.7	1956	0
	C28	12	5.6	1956	0
	C29	12	5.8	1956	0
	C30	12	6.2	1956	0

this concept at a finer level of granularity, since *Design* is already decomposed into the concepts of FCM5. Interestingly, *Business Complexity (C5)* and *Cost (C8)* are the weakest

concepts in the main FCM, which is logical since they correspond to "business" aspects rather than technological ones.

Moving to the second layer and examining *Design*, which appears through the static analysis to be the strongest concept, further investigation over FCM5 was conducted focusing on measurements of the constituent concepts. The strongest position in this sub-FCM is held by *Decentralization (C24)*, which positively affects the central concept of the map, *Microservices Adoption (C31)*.

A part of the dynamic analysis that follows is directed by the results of the static analysis in an attempt to confirm or question the aforementioned findings through execution.

6.4.2 Dynamic Analysis

As previously mentioned, the successful performance of the model over the two extreme scenarios during the validation process (Section 6.3.2) is a prerequisite for performing dynamic analysis. The dynamic analysis is a powerful and effective decision support tool that serves multiple purposes: Firstly, it can be used to assess whether a given MLFCM model may be simplified, that is, whether certain nodes may be removed without loss of accuracy or explainability. Secondly, it may verify or contradict the results of the static analysis, the latter providing indications about the tendency of the map towards a positive or negative output based on the cycles formed in the model. Thirdly, and most importantly, it may be employed to construct and execute what-if scenarios with different configurations (i.e., initial activation levels of the participating concepts) and study their outcome. This task practically offers the ability to a decision-maker to form hypothetical situations and forecast their result through their simulation via the MLFCM model. In addition, she may then focus on tracing the causes for reaching the value level of the concept(s) of interest, as well as investigating how things should change among the leading determinants (i.e. the initial values of the key factors-concepts) to drive the target value(s) at lower or higher levels as desired. The latter is feasible through the ALGE part of the proposed model, while the solutions yielded by the ALGE algorithm provide enough insights for more informed decisions.

As a first step, a series of executions were conducted targeting at simplifying the model by investigating the behavior of the n strongest or weakest nodes, as these were suggested by static analysis, in terms of defining the final outcome (decision) of the model. The goal was to make the model simpler, easier to execute and understand, as one will have to define and

analyze fewer concepts. More specifically, various experiments were designed and executed on one hand to verify the strongest and weakest nodes in the model, and, on the other, to investigate if keeping only the strongest nodes, or removing part or all of the weakest nodes, causes a significant change in the final decision value compared to that of the full map. Further executions of the model were performed to assess also the contribution of each sub-FCM to the final outcome, by focusing on their central concepts, i.e., those nodes that transfer their values to the upper-layer FCM. This process was applied in a bottom-up manner varying the activation values of the child nodes in each specific sub-FCM. In all executions, the approach described in Section 5.3 was used, with the transfer function being the sigmoid as given in Equation 2.2.

The results of the above series of executions confirmed the results of the static analysis regarding the 3 strongest and 2 weakest concepts. To this end, the results of four indicative scenarios are listed in Table 6.6, which clearly suggest this. The first two scenarios examined the effect of concept *C4*, which was characterized as a strong node, while in the last two, the effect of node *C5*, which was defined as a weak node, was assessed. In both cases, two initial values were considered, 0.1 and 0.9, while the initial activation value of every other concept was kept constant. After execution the difference observed in the corresponding final activation values of the main concept, although marginal (being equal to 0.08 for the *C4* case and 0.04 for the *C5*) is considered significant. This result also fully supports the findings of static analysis in Table 6.5 as the effect of *C4* on the final output is double that of *C5*.

Investigating the possibility of simplifying the model without affecting the outcome, various simulations were executed considering a map formed only with the top three strongest concepts and the corresponding sub-FCMs, or removing the weakest ones. The results produced showed a significant difference between the output of the simplified map and that of the full map in both cases, with the strongest case, though, yielding a larger difference. Although these findings confirmed that the strength and level of influence of the participating concepts in the model vary, essentially suggests that none of the strongwer concepts or even an isolated group of concepts is able to dominate the model's final outcome. In addition, the concepts with the smallest influence in the model may not be considered negligible as they play a small but significant role in forming the final output.

The rest of the steps of the dynamic analysis incorporated additional experiments utilizing the proposed ALGE-MLFCM algorithm. For all scenarios tested the variables involved, as well as the algorithm's configuration, were set as follows: The population size was set equal to 50 and the number of generations to 250. The *Elitistic* method was used for the

Table 6.6: Indicative simulations part of the dynamic analysis of the model.

No.	**Sim1**	**Sim2**	**Sim3**	**Sim4**
Initial activation values				
C1	0.5	0.5	0.5	0.5
C2	0.3	0.3	0.3	0.3
C3	0.9	0.9	0.9	0.9
C4	**0.1**	**0.9**	0.7	0.7
C5	0.7	0.7	**0.1**	**0.9**
C6	0.3	0.3	0.3	0.3
C7	0.3	0.3	0.3	0.3
C8	0.5	0.5	0.5	0.5
C9	0.3	0.3	0.3	0.3
C10	0.1	0.1	0.1	0.1
C11	0.5	0.5	0.5	0.5
Final activation values				
C31	**0.71**	**0.63**	**0.61**	**0.65**
Diff.	**0.08**		**0.04**	

evolution of the generations. Moreover, the *Roulette Wheel* method was used for the selection of individuals that undergo genetic alteration. Finally, *Single-point* crossover and *Random Resetting* operators, both with probability equal to 0.1, were used for the *Crossover* and *Mutation* steps, respectively.

The next step employed the ALGE-MLFCM algorithm to identify the tendency of the model. Initially, the target value for the central concept, *C31*, was set to 0.9 (Very High) and then to 0.1 (Very Low). For each case, a series of executions were performed and the convergence performance was recorded. The corresponding results are shown in Figure 6.5. By observing the results, one can easily discern that when the target value was set to Very High, the algorithm delivered solutions faster than when it was set to Very Low. This indicated that there are more combinations of initial activation values, and, therefore, solutions that can lead the model to higher-level positive values than in the opposite case. Although the static analysis initially suggested a negative tendency, this result indicates that the model's tendency is clearly in favor of microservices adoption. This finding sheds some light on the question posed in Section 6.4.1: A partial positive tendency exists in the sub-FCMs which indeed affects the overall tendency of the model.

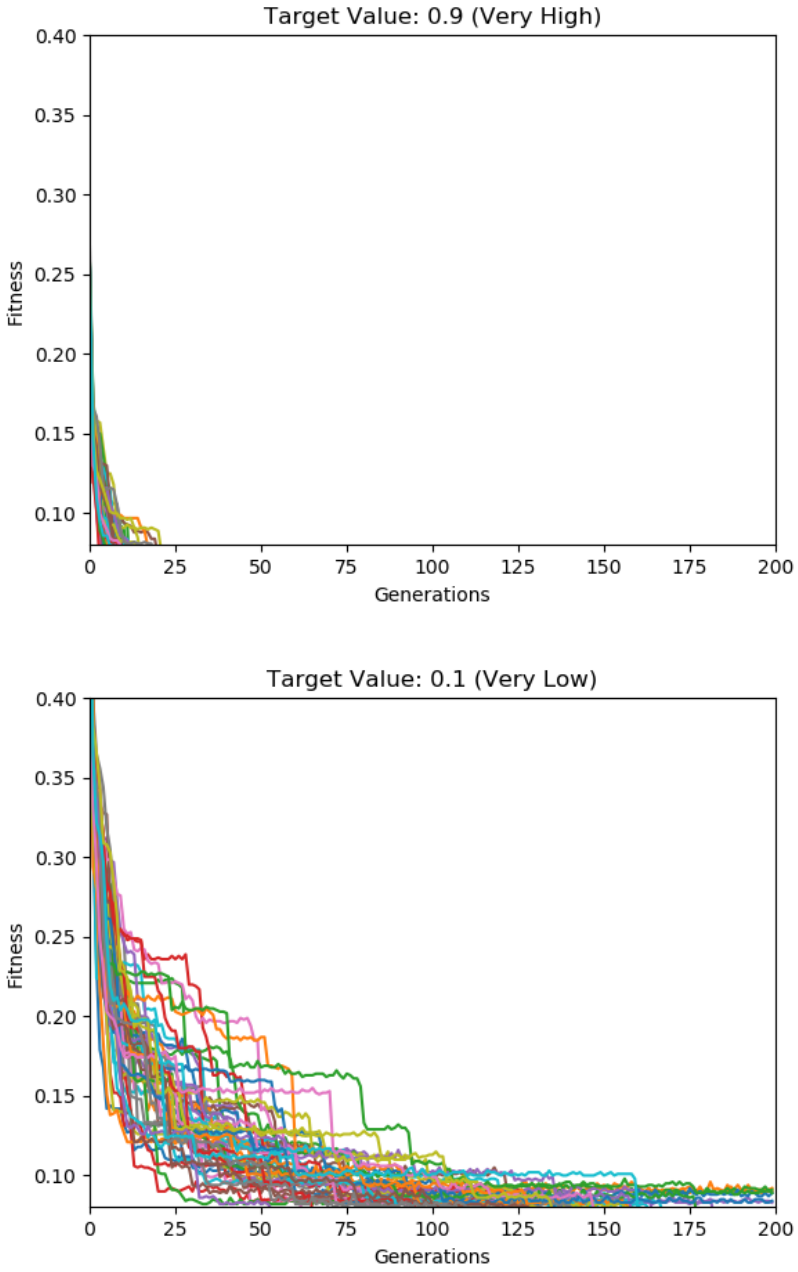


Figure 6.5: The convergence of the ALGE-MLFCM algorithm for different target values

6.5 Model Application Over a Real-World Case Study

A real-world case study is presented in this section to demonstrate the applicability of the model and the analysis process. This case study was modeled by capturing the dynamics of a migration project within an international content delivery company. Specifically, the project was involved with the holistic migration of the monolithic architecture of the company's main

running software services to a SOA. Starting from a dedicated version of questionnaire, two rounds of interviews were conducted with the Chief Technical Officer (CTO) of the company: The first one aimed to describe the situation, both at a project and organizational level, at the time of the decision, expressed in the initial activation levels of the model's concepts prior to execution. The second one was utilized after execution to discuss the output of the model regarding the actual decision and the main concepts that formed such a decision. Indeed, the model successfully matched the real decision, which was in favor of moving to the SOA environment, yielding the numerical value 0.64 which corresponds to the linguistic value "High".

The next step involved the investigation of the model's ability to correctly match the final concepts' values with the level of status of each concept after the project completion. Although the central concept, that is, *Microservices Adoption (C31)*, is only influenced by other concepts and does not feed any other, it was quite interesting to investigate whether the interrelations between the rest of the concepts were able to correctly estimate their future status.

Table 6.7: Industry case study: Final activation levels

No.	**Initial AL**	**Real FAL**	**Model FAL**
C1	Medium	High	**High**
C2	Low	High	**High**
C3	Very High	Very High	**Very High**
C4	High	Very High	**Very High**
C5	High	High	**High**
C6	Low	Low	**Low**
C7	Low	Low	**Low**
C8	Medium	High	**High**
C9	Low	Low	Medium
C10	Very Low	High	Low
C11	Medium	Low	Medium
C31	-	High	**High**

Table 6.7 shows the initial activation levels, in linguistic form, of the concepts which constitute the main FCM. In the first column the values listed are those formed through questionnaires and interviews, and reflect the environment before the migration project started. The values in the second column represent the status of each concept after the completion of the migration, and were also extracted through questionnaires and interviews. The third column presents

the final activation values calculated through the execution of the model. It is clear that the model's output matches the real values in eight out of eleven concepts (bold letters). This result is considered satisfactory, while at the same time underlines the need for further investigation focusing on the structure of the model.

As described in Section 5.3.1, the utilization of the results yielded by the ALGE-MLFCM algorithm may provide answers or explanations for the results of a particular case study. In this case, the ALGE-MLFCM algorithm was employed to deliver solutions targeting the value 0.9 (Very High) for the concept of interest, so as to investigate how the environment should have been shaped before the adoption to reach such a value.

A particular solution was selected that reflects a realistic status for each concept of the model. The corresponding linguistic values, along with those of the industrial case study, are compared in Table 6.8. The operators "+" and "-" are used to indicate the increase or decrease in the initial activation values compared to the original initialization, which led to the desired outcome. The number of operators used indicate the level of increase or decrease respectively.

Table 6.8: Industry case study: How to improve the final decision

No.	**Real AL**	**Proposed AL**	**Indicator**
C1	Medium	Very High	+ +
C2	Low	High	+ +
C3	Very High	Very High	
C4	High	Very Low	- - -
C5	High	Very High	+
C6	Low	High	+ +
C7	Low	Very Low	-
C8	Medium	Very Low	- -
C9	Low	Very High	+ + +
C10	Very Low	High	+ + +
C11	Medium	Very Low	- -

As shown by the comparison in Table 6.8, the three concepts that need to change their values the most are *Operational Complexity (C4)*, which is considered to be high and should be decreased, *Design (C9)* that should be increased and *DevOps (C10)* that should also be increased. For five other concepts a significant change in their values is indicated: *Governance (C1)*, *Infrastructure and Management Services (C2)* and *Reliability (C6)* should be increased, while *Cost (C8)* and *Data Micration (C11)* should be decreased. For concepts *Business*

Complexity (C5) and *Security (C7)* a slight or no change is suggested, while *Evolvability (C3)* is identical.

The results above call for further investigation by focusing on particular scenarios involving concepts of significant interest. Specifically, the decision-makers from the company side expressed their interest in studying the scenario involving the concept of *Security (C7)* and *Microservices Adoption (C31)*. This scenario was formed in a way to address a specific question: How the environment should initially be formed in terms of concept activation values, to lead the model to a definite positive final decision and also preserving a high level of security?

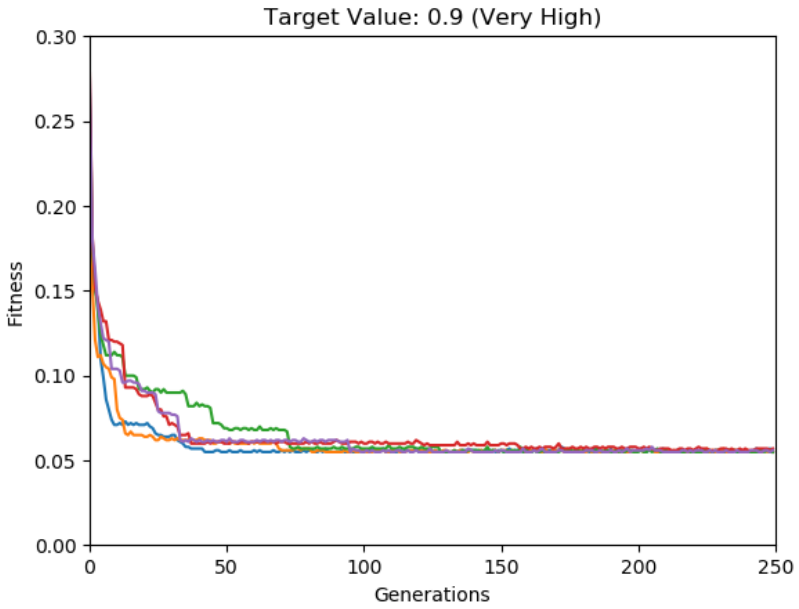


Figure 6.6: The attempt of the ALGE-MLFCM algorithm to find a solution that leads the final decision to "Very High" value with the value of "Security" to start at and remain "High"

To provide an answer to this question, a systematic approach was followed: First, the main objective was defined, with the ALGE-MLFCM algorithm being directed to deliver solutions that would lead the central concept *Microservice Adoption (C31)* to 0.9 (Very High). The algorithm was executed with the maximum number of generations being set to 250 and resulted in a set K of near-optimal solutions that matched perfectly the acceptance fitness criterion (see Figure 6.6).

Further investigation of this result was then conducted to introduce the second objective, which is the level of security. A filtering process was applied according to which identical solutions

Table 6.9: Five ALGE-MLFCM runs targeting a "Very High" value for *Microservices Adoption* and a constant "High" value for *Security* against the nearest solution (NS) reference.

No.	(*NS*)	**Runs**				
C1	0.8	0.9	0.7	0.8	0.9	0.9
C2	0.8	0.9	0.8	0.9	0.8	0.7
C3	0.9	0.9	0.9	0.9	0.9	0.8
C4	0.1	0.1	0.1	0.1	0.2	0.1
C5	0.7	0.9	0.9	0.9	0.9	0.9
C6	0.9	0.9	0.8	0.7	0.8	0.8
C7	0.1	0.7	0.7	0.7	0.7	0.7
C8	0.1	0.1	0.1	0.1	0.1	0.1
C9	0.9	0.9	0.9	0.9	0.9	0.9
C10	0.8	0.8	0.9	0.9	0.9	0.8
C11	0.1	0.1	0.1	0.1	0.1	0.1

and solutions that did not satisfy the objectives set were removed. The remaining M solutions were then processed to calculate the median for each gene, that is, of each initial activation level for each concept in the map. Thus, a reference solution S was formed containing the medians for each gene. Next, the pool of M solutions was used to identify the nearest solution NS to S, i.e. with the smallest difference in the values of the initial activation levels, calculated as distances of a linguistic form. The latter measured the difference between the fuzzy sets that the two initial activation level values (of a solution compared to S) fell into. Finally, solution NS was used as a baseline to conduct experiments and investigate the performance of the algorithm, this time keeping the value of *Security (C7)* to 0.7 (High) apart from leading *C31* to 0.9. The algorithm was executed five times and the results are given in Table 6.9. In all five runs the algorithm managed to reach the same level for microservices adoption ($C31$=0.83) with security stable at the level of 0.7. It is interesting to note that all solutions converge to the same initial activation levels values for all concepts except for *Business Complexity (C5)* which rises by 0.2 compared to the baseline configuration. This finding was cross-checked with the company experts and was confirmed that there are cases in which the required level of security must be high so as to compensate with the rich complexity of the business environment in terms of business rules, processes and requirements. Therefore, the experts verified the significance of the results and greatly appreciated the support provided by the model. Fianlly, it should be noted that the small difference between the target level and the actual solutions for the concept of interest (*C31*) clearly indicates that the expected solution is not that easy to achieve

in full. Interpreting the resulting effect for the problem under study, it can be inferred that the model reflects nicely the big concern about reduced security when adopting a microservices architecture. Moreover, one may assume that the two concepts, *Security* and *Microservices Adoption*, are directly or indirectly conflicting. Therefore, to find the best trade-off solution, the use of Multi-Objective Optimization (MOO) approaches may be required.

6.6 Summary

This chapter introduced a specially designed MLFCM model to support the decision of adopting the microservices architecture. The proposed model provides enhanced decision support capabilities and increased explainability. The contribution of this work is multifaceted: First, a fine-tuned model based on MLFCM was constructed through literature review and experts feedback, which was formed as a set of interacting concepts and drivers related to the decision of adopting microservices architecture. The model captures the dynamics of the problem under study, that is, of the factors that decisively affect the decision of migrating from a monolithic application environment to microservices. Second, the utilization of the proposed framework as described in Chapters 3 and 5 , provided a better understanding of the dynamics of the environment under study, as well as an explanation of the model's results.

The proposed model was initially validated over two "extreme" scenarios and then it was calibrated over a number of real-world cases. After calibration, investigation of the model was performed by means of (static) graph analysis and (dynamic) simulation over various customized scenarios, aiming to reveal the strongest and weakest concepts in terms of influencing the relevant decision, and study their behavior. Further experimentation was conducted next, with the model being applied over an industrial case study. The model successfully managed to match the real decision, as well as to correctly describe the status of each concept after project completion as this was expressed in the final concept activation values. Moving a step further, the outcome of the model over the industrial case was analyzed by studying the solutions resulted by ALGE-MLFCM and suggesting changes and improvements to the company's environment aiming to strengthen the decision in favor of microservices adoption. The practitioners led this investigation and defined concepts of significant interest to study in detail their behavior, interrelation and contribution to the final decision of the model. The whole process was acknowledged as quite supportive and informative, while the flexibility in setting-up and executing hypothetical scenarios was commented very positively by the participating practitioners.

The work in this chapter was mainly focused on the analysis and study of the factors forming the environment behind the decision of migrating to microservices, either from a monolith system or for developing a new one from scratch. Throughout the research carried out, a particular challenge emerged that needs special attention; that is, the migration from the software components architecture to the microservices architecture. A significant number of software systems that are currently operating in many companies are based on a component-based architecture, and in this context, the next chapter described an attempt to address the challenge of full or partial migration from software components to microservices.

Chapter 7

Migration of Software Components to Microservices: Matching and Synthesis

7.1 Introduction

Nowadays more and more software companies, as well as individual software developers, adopt the microservice architecture for their software solutions. Although many software systems are being designed and developed from scratch, a significant number of existing monolithic solutions tend to be transformed to this new architectural style. What is less common, though, is how to migrate component-based software systems to systems composed of microservices and enjoy the benefits of ease of changes, rapid deployment and versatile architecture. This chapter proposes a novel and integrated process for the decomposition of existing software components with the aim being to fully or partially replace their functional parts with by a number of suitable and available microservices. The proposed process is built on semi-formal profiling and utilizes ontologies to match between properties of the decomposed functions of the component and those offered by microservices residing in a repository. Matching concludes with recommended solutions yielded by multi-objective optimization which considers also possible dependencies between the functional parts.

Despite the differences in their approach and the time lag in their introduction to the software engineering community, Component-based Software Engineering (CBSE) [99], or, alternatively Component-based Development (CBD), and Microservices Architecture (MSA) [100] share the same inceptions, motivation and focus towards reuse of software artefacts. Both approaches aim at reducing complexity of the software development process, facilitate easy

maintenance and support the operations for IT support. It may be argued that Service-Oriented Architecture (SOA) [101], as the most recent emerging distributed development architecture, constitutes the common denominator between these two paradigms as it originated from component-based architecture and evolved to microservices architecture.

Following the new software engineering trends, Microservices architecture is tightly connected to the DevOps approach [5], which inherits its basic principles from agile methodologies and describes best practices to support the software development and operation processes. One may also argue that Microservices architecture actually supports the DevOps automation process and affects software engineering in a positive manner. To be more specific, it affects Software Engineering by introducing a different development approach. As regards how the DevOps process is automated, the latter relies primarily on the fact that the adoption of the Microservices architecture comprises a number of critical tasks than may be automated apart from the rest automated tasks like communication, coordination, monitoring, problem solving and deployment.

As has been said in the previous chapter, in recent years, Microservices architecture is gaining popularity in software development and the research community has turned its attention to related challenges [7], such as the decomposition of a monolithic system into a set of independent services, followed by synthesis of selected microservices to substitute their functionality. Microservice synthesis relies on locating and combining small functional service components the characteristics of which match those of the decomposed system and put them in a proper order so as to meet the characteristics and the requirements of the initial monolithic system.

While literature includes a number of research works for decomposition approaches and migration from monolithic to microservice architecture, to the best of our knowledge, no work has been yet published that deals with software component decomposition and replacement of its functional parts with microservices. This work aims to introduce an automatic process that identifies and recommends the full or partial replacement of a software component by a number of available microservices that support specific business operations. The proposed process adopts the basic principles proposed in [102] related to a layered component-based software development architecture, which was adapted and refined to accommodate the differences and peculiarities of the Microservices environment. The framework in [102] supports the process of matching available components against a set of specifications expressed in a formalised syntax and utilizing ontologies. Apart from modifications to this framework, the present work adds new tasks that extend and improve its recommendation layer.

7.2 Literature Overview

To the best of our knowledge this is the first attempt to propose a structured process targeting the decomposition of well described software components and replace the identified functional parts with microservices to the greatest possible degree. However, a brief literature overview has been carried out over the two topics that are strongly related to this research work, the software decomposition and the services synthesis.

Several different approaches have been proposed to deal with the decomposition of monolithic systems or services. Baresi, Garriga and De Renzis in [103] propose a clustering-like approach to support the identification of microservices and the specifications of the extracted artefacts during either the design phase of a new system, or while the re-architecting of an existing system. Service Cutter [104] is a tool framework that is based on a structured repeatable approach to decompose a monolith into microservices. A stepwise technique to identify microservices on monolithic systems is proposed in [105] in which the authors deliver an approach based on a dependency graph among three distinct parts of an application, client, server and database. Balalaie, Heydarnoori and Jamshidi in [106] describe their experiences of an ongoing project on migrating an on-premise application to microservice architecture. Their approach is based on architectural refactoring, considering the characteristics of microservice architecture.

The vast majority of the literature which deals with services composition is concerned with web services. The work in [107] presents a review of existing proposals for services selection by quoting the advantages and disadvantages of each approach. A systematical review of recent research on QoS-aware web service composition using computational intelligence techniques is presented in [108]. A classification was developed for various research approaches along with the analysis of the different algorithms, mechanisms and techniques identified. An analysis and comparison of the latest representative approaches in the area of automated web service composition is the main contribution of the work in [109]. The existing research approaches were grouped into four distinct categories, workflow-based, model-based, mathematics-based and AI planning.

It is evident that the current literature on software services synthesis is limited, and especially in the case where this synthesis targets the migration from component-based development to microservices is rare if not non-existent. This is the gap the current paper aspires to fill.

7.3 Automatic Specification and Matching of Microservices

7.3.1 Specification and Matching framework

As previously mentioned, the present work aims to introduce an automatic process that identifies and recommends the full or partial replacement of a software component by a number of available microservices. The proposed process adopts basic principles proposed by the authors in a previous work [102]. More specifically, the utilization of the description layer in that work leverages the decomposition of a software component into distinct operations and respectively profiles all candidate microservices that may substitute these operations and simultaneously adhere to the same constraints (e.g. performance). Additionally, the translation of software components and microservices textual profiles into ontologies assists the automatic matching process towards the integrated replacement.

The proposed process follows the same 5-layers architecture as in previous work [102]: (i) The Description layer provides a profile structure which includes all relevant information that describes the component(s) under decomposition and the available microservice(s). A developer/vendor of a component or microservice, defines a set of properties (functional and non-functional) that describe the specific artefact: (a) the component description which will serve as the basis for its decomposition and the properties characterizing each decomposing part, and, (b) the properties of a microservice that describe what it has to offer in terms of functionality, performance, availability, reliability, robustness etc. that one may look for when attempting to locate suitable microservices for integration and substitution of the component parts. (ii) The Location layer essentially provides general-purpose actions, like searching, locating and retrieving the microservice(s) of interest that match the profile of the component's decomposed parts. (iii) The Analysis layer evaluates the level of suitability of the candidate microservice(s) and provides matching results that will guide the selection of microservices for integration. (iv) The Recommendation layer uses the information provided by the previous layers and produces suggestions as to which of the candidate microservice(s) may be best integrated and why, based on an assessment made to ensure that certain requirements at the microservice level are preserved also at the integrated level. (vi) Finally, the Build level essentially comprises a set of integration and customization tools for combining component(s) to build larger systems. The present paper focuses on the first four layers and describes a novel way for automatic matching between desired and available microservice(s) based on the directions provided in [6] and extending or revising them where appropriate. The interested

reader may refer to that work for more details on the layered component architecture, whilst every effort has been made to make the current paper self-explanatory.

Components and microservices are first expressed in a semi-structured form of natural language which is then transformed into an ontology. This ontology standardises the description of the properties of the two software artefacts and will constitute the cornerstone of the specifications that will be used to match decomposed parts of components with the available microservice(s), the latter being stored in a repository. Thus, the problem of finding suitable microservice(s) to replace component functionality is reduced to matching (aligning) ontologies. This process is executed by automatically parsing the profile(s) of the two software artefacts (for simplicity we assume one component and N microservices) and their translation into instance values of two dedicated ontologies, one for each artefact, which are built so as to reflect the most critical properties suggested in literature for that artefact. At the same time, as the ontology of the component is being built, certain parts are marked so that a second step may then be executed which isolates these parts, as these are recognised to be directly comparable to parts of the microservices ontology. Hence, the latter step transforms them into a meta-ontology (subset of the component's initial ontology) describing the so-called 'required' or possible functions to be executed by available microservices. Then the matching of properties between the required and offered microservice(s) takes place automatically at the level of ontology items and a suitability ratio is calculated that suggests which microservice(s) to consider for possible integration. The whole process is graphically depicted in Figure 7.1.

7.3.2 Profiling

Based on the previous work mentioned above and an extended literature study, we identified a set of desired properties for components and microservices thus providing their profile. A profile is categorized into functional, non-functional and other properties:

(i) Functional properties: Include those properties that describe what the component or microservice actually does. It involves a general description of the functionality delivered through methods along with their specific descriptions.

(ii) Non-functional properties: Include properties reflecting how the component or microservice behaves, mostly in terms of performance, using indicators such as time for execution, bytes of data processed per second, operations executed per second, number of concurrent users supported, cold start (the transition time from deployment to actual

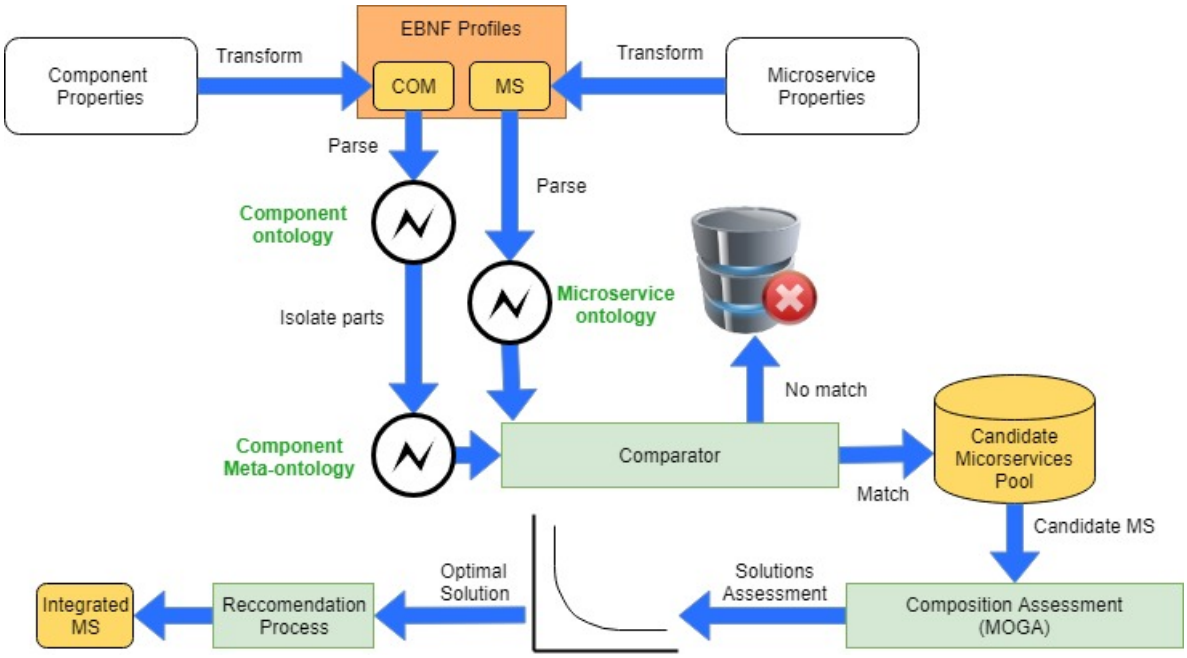


Figure 7.1: The proposed process for component decomposition and microservices substitution.

execution), etc.

(iii) Other properties: Such properties involve other critical attributes of a component or microservice that may not be considered as functional or non-functional. These constitute mostly properties that provide useful general information regarding the artefact and its usage. In this part a profile provides information regarding the programming language it is implemented with, the level of security it provides, its auditability, data exchange, interaction protocol, type (data source, application login, GUI, etc.), data format, load balancing, obligations and constraints, automation and level of binding, verification and validation issues, cost, the data storage which describes how the component stores its data, and, lastly, a service descriptor. Numerical properties included in the categories above may provide minimum or maximum threshold values, which will be used to guide the matching process for selecting suitable microservices for substitution.

The Extended Backus-Naur Form (EBNF) was selected to be used to express the component and microservice descriptions. This form allows the formal proof of key properties, such as formatting and closure, thus helping to validate semantics. The proposed grammar has been developed with the Another Tool for Language Recognition (ANTLR) (http://www.antlr.org/), a parser and translator generator tool that supports language grammars in EBNF syntax. Fig-

ures 7.2 and 7.3 depict the EBNF description of a component and a microservice respectively, which are analysed below.

The profile of a component includes, from top to bottom, the following: First, some definitions of component items are provided, including a name and a list of one or more services it offers. Each service is defined by a primary and a secondary function, the latter being more informative, as well as an optional description. Primary types involve general functionality, like I/O, security, networking, etc.; the secondary types explicitly define the actual function it executes, e.g. printing, authentication, video streaming, audio processing etc. For example, the service could be [Security, Login Authentication].

When decomposition takes place, this is one of the main features that will guide the searching for a microservice and it is considered as a Constraint, something which means that a candidate microservice will be rejected if it does not offer such functionality. Interfacing information comes next that outlines the various methods that implement its logic; a method is further analysed to Preconditions, Postconditions, Invariants and Exceptions, if any. This piece of information is provided upfront by the component developer/vendor. Non-functional requirements or properties are defined next denoting mandatory behaviour in terms of performance. Finally, general information intended to serve reusability purposes (application domain, programming language, OS etc.) is provided. It should also be mentioned that certain features in the profile may be assigned specific values along with a characterization as to whether this feature is minimised (i.e. the value denotes an upper acceptable threshold) or maximised (i.e. the value denotes a lower acceptable threshold) in the component under decomposition. For example, if performance is confined under 15 seconds, then next to the performance indicator the couple (15, minimise) is inserted.

The definition of the attributes included in the microservice profile is defined in a more detailed form compared to the component profile and thus the microservice profile may be considered as a more refined version of the component profile. The microservice profile first describes the data types used to define the property values. These types suggest three categories of microservice attributes. The first category is the ‘functional requirements’ where a specific textual description is provided regarding the function a microservice delivers. Since microservices are smaller and more specific than components, so is their description, which intuitively documents what it does. The second category is the ‘non-functional requirements’ which includes information describing mostly performance, as well as other constraints. These attributes include performance indicators like bytes processed per second and operations executed per second, the level of security it provides, and information regarding its data storage

```
(**** Component EBNF Profile ****)
DIGIT : 0|1|2|3|4|5|6|7|8|9;
INTEGER : DIGIT {DIGIT};
CHAR : A|B|C|... |W|a|b|c|... |W|!|@|#|...;
STRING : CHAR {CHAR};
Variable_type : CHAR|INTEGER|... ;
Variable_name : STRING;
Primary_Type : 'Input'|'Output'|'Security'|'Multimedia'|'Networking'|'GUI'|...;
Secondary_Type : 'Authentication'|'Data processing'|'Video'|'Audio'|'File access'|'Printing'|...;
Details_Description : CHAR {CHAR};
Min_Max_Type : 'Minimize'|'Maximize';
Required_Type : 'CONSTRAINT'|'DESIRED';
Service : 'S' INTEGER Primary_Type, Secondary_Type { Details_Description} Required_Type;
Service_List : Service {Service};
Operator : 'exists'|'implies'|'equals'|'greater than'|'less than'|... ;
Condition : Variable_Name Operator {Value} {Variable};
Precondition : Condition {Condition}; (*IF THESE ARE PROVIDED BY DEVELOPER/VENDOR*)
Postcondition : Condition {Condition}; (*IF THESE ARE PROVIDED BY DEVELOPER/VENDOR*)
Invariants : Condition {Condition}; (*IF THESE ARE PROVIDED BY DEVELOPER/VENDOR*)
Exceptions : Condition {Details_Description} {Exceptions}; (*IF THESE ARE PROVIDED BY DEVELOPER/VENDOR*)
Method : 'M' INTEGER {Variable Variable_Type} {Precondition} {Postcondition} {Invariant} {Excpetion}; (*IF THESE
ARE PROVIDED BY COMPONENT DEVELOPER/VENDOR*)
(*==== INTERFACING ====*)
Service_analysis : 'Service' INTEGER ':' 'Method' INTEGER ':' STRING Method {Method};
(*==== NON_FUNCTIONAL PROPERTIES ====*)
Performance_indicators : [ 'Response time' (INTEGER) Min_Max_Type Required_Type | 'Concurrent users' (INTEGER)
Min_Max_Type Required_Type | 'Records accessed' (INTEGER) Min_Max_Type Required_Type | ... ]
{Performance_indicators};
Resource_requirements : [ 'memory utiliztion' (INTEGER) Min_Max_Type Required_Type | 'CPU reqs' (INTEGER)
Min_Max_Type Required_Type | ... ] {Resource_requirements};
Quality_features : [ 'Availability' (INTEGER) Min_Max_Type Required_Type | 'Reliability' (INTEGER) Min_Max_Type
Required_Type | ... ] {Quality_features;
(*==== END OF NON-FUNCTIONAL PROPERTIES; NEW ITEMS MAY BE ADDED HERE ====*)
(*==== REUSABILITY PROPERTIES ====*)
Application_domain : 'Medical' Required_Type | 'Financial' Required_Type | 'Business' Required_Type | ...
{Application_domain};
Programming_language : 'C' Required_Type | 'C++' Required_Type | 'Java' Required_Type | 'VB' Required_Type | ...
{Application_domain};
Operating_systems : 'Windows' Required_Type | 'Linux' Required_Type | 'Unix' Required_Type | 'IOS' Required_Type |
'Android' Required_Type | ... {Operating_systems};
Openness : 'black' Required_Type | 'glass' Required_Type | 'grey' Required_Type | 'white' Required_Type;
Price : INTEGER;
Development_info : STRING;
Developer: STRING;
Version : STRING; (*IF THESE ARE PROVIDED BY DEVELOPER/VENDOR*)
Protocols_Standards : [ 'JMS/Websphere' Required_Type | 'DDS/NDDS' Required_Type | 'COBRA/ACE TAO'
Required_Type | 'POSIX' Required_Type | 'SNMP' Required_Type | ... ] {Protocols_Standards};
Documentation : [ 'Manuals' Required_Type | 'Test cases' Required_Type | ... ]; (*IF THESE ARE PROVIDED BY
DEVELOPER/VENDOR*)
(*==== END OF REUSABILITY PROPERTIES; NEW ITEMS MAY BE ADDED HERE ====*)
SPECIFICATIONS PROFILE :
        'Specifications Profile :' STRING; 'Descriptive title :' STRING;
        'Functional Properties :' Service_List;
        'Interfacing :' Service_analysis {Service_analysis};
        'Non-functional Properties :' Performance_indicators, Resource_requirements, Quality_features;
    'Reusability Properties :' Application_domain, Programming_language, Operating_systems, Openness, Price,
                               Protocols_Standards, Documentation;
```

Figure 7.2: Component profile in EBNF

```
(**** Microservice EBNF Profile ****)

(*==== General Properties ====*)
BINARY : 'Yes' | 'No';
STRING : (' '..'~')+;
NUMBER: ('0'..'9')+;

WS : [\r\n\t] + -> skip;
NEWLINE : [\r\n]+;

(*==== Functional Requirements ====*)
functional_description : STRING;

(*==== Non-functional Requirements ====*)
securityLevel : NUMBER;
bytesProcessedPerSecond : NUMBER;
operationsExecutedPerSecond : NUMBER;
coldStart : NUMBER '.' NUMBER;

(*==== Other ====*)
programmingLanguage : 'C' | 'C++' | 'Java' | 'Python';
dataStorage : 'None' | 'SQL' | 'Graph' | 'Document' | 'File';
auditability : BINARY;
dataExchange : 'REST' | 'SOAP' | 'RPC';
interactionProtocol : 'Synchronous' | 'Asynchronous';
type : 'Data Source' | 'Application Logic' | 'GUI';
dataFormat : 'JSON' | 'RSS' | 'XML';
loadBalancing : 'N/A' | NUMBER 'threads';
obligationsConstraints : 'Public' | 'Private' | 'Local';
automationLevelOfBinding : 'Manual' | 'Semi-automated' | 'Fully automated';
verificationValidation : 'Yes with test data' | 'Yes without test data' | 'No';
serviceDescriptor : 'N/A' | 'UML' | 'WSDL' | 'OWL-S' | 'BPEL';
cost : NUMBER '.' NUMBER;
```

Figure 7.3: Microservice profile in EBNF

(SQL, GraphDB, Document Store, File). The third and last category is 'other requirements' and provides additional general information regarding the microservice. The properties in this category include the programming language used to implement the microservice, the ability to audit events in logs (auditability), information regarding the data exchange protocol (REST, SOAP, RPC) and interaction protocol (Synchronous, Asynchronous) supported, data format in which data is exchanged (JSON, XML), load balancing, cost etc., as shown in Figure 7.3. After briefly describing both profiles, we will now focus on the process which connects components with microservices and demonstrate how component decomposition is performed and microservices are matched through ontology instances.

7.3.3 Components Decomposition and Microservices Matching

7.3.3.1 Component and Microservices Ontology

A special form of ontology is devised to facilitate the subsequent steps of decomposing a component into individual functional parts, and then locating and assessing the suitability of available microservices for integration using a self-contained description. The ontology is built around the property axes of the components and microservices profiles described above, the latter conforming to the same semantic rules as the former, so as to facilitate their automatic transformation to instances of the ontology. Figures 7.4 and 7.5 depict the largest parts of these ontologies, while some details have been intentionally omitted due to size limitations.

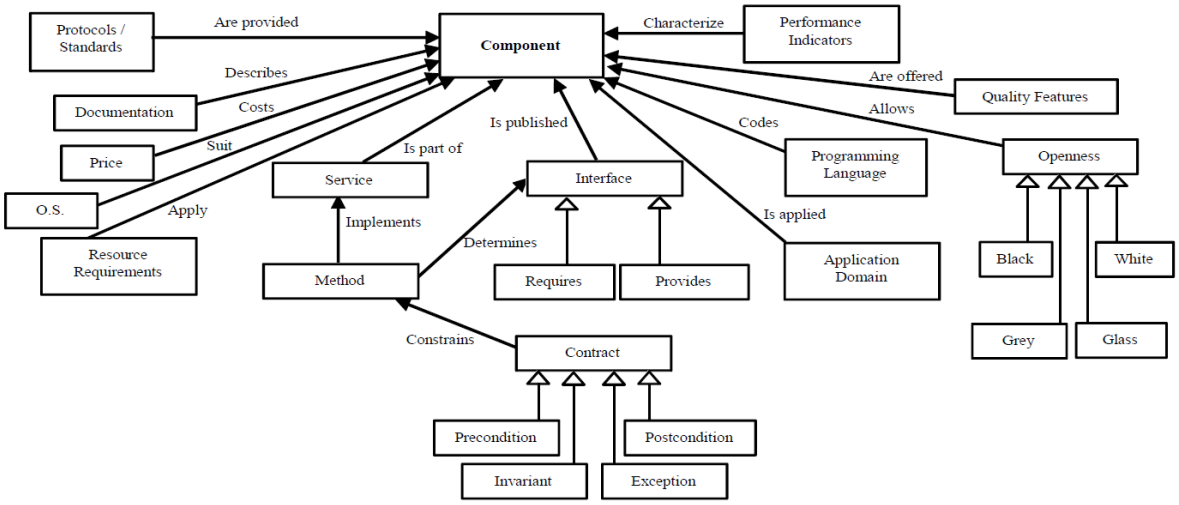


Figure 7.4: Component ontology

The matching process works at the level of the ontology tree and not the textual descriptions of the profile, something that makes comparisons more easy and quick, both computationally and graphically (visually). This is due to an ontology alignment algorithm used to compare (align) two same-structured ontology instances aiming at locating attribute similarities and attribute values distances. The ontology alignment algorithm works by parsing both ontologies as ontology tree instances and investigates their structure level by level in a tree structure hierarchy. After the schema similarity is compared, the algorithm calculates the value distance for each attribute depending on their data type since there is a form of heterogeneity between attribute values. For example, some attributes may be of binary type, while some others may be of numerical. The solution is to handle distance calculation differently depending on the data type compared in each attribute. In this case the ontology alignment algorithm used is Graph

Matching for Ontologies (GMO) [110]. GMO initially parses two ontologies and transforms them to RDF bipartite graphs following some matrix operations to determine the structural similarity. This is the first and most crucial step of the proposed methodology as it initially discards non-matching microservices from the pool of available candidate microservices. The matching process is described in detail in the next section.

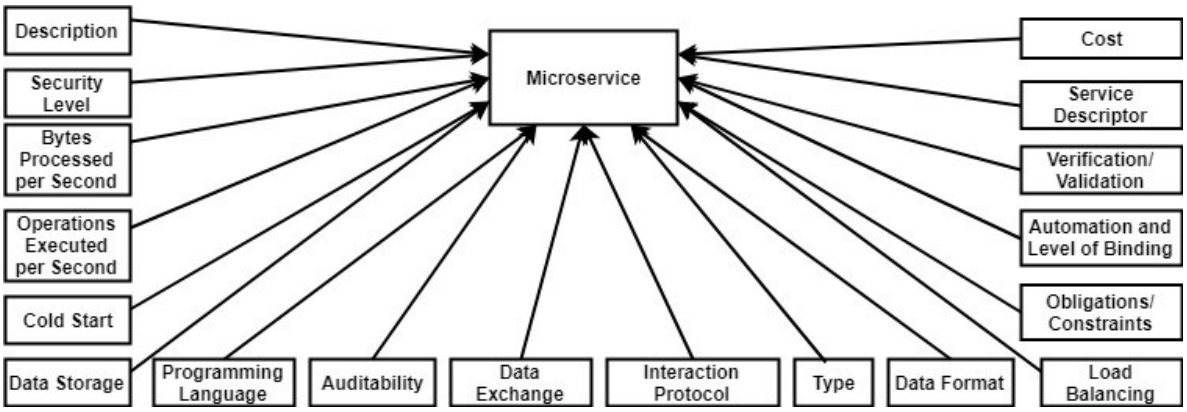


Figure 7.5: Microservice ontology

7.3.3.2 Matching Process

Different methods are proposed in literature for description processing, such as simple string, [111], signature matching [112] and behavioural matching [113]. The approach followed in this chapter is slightly different; it employs a hybrid form combining string and behavioural matching. More specifically, a dedicated parser is implemented that recognises certain parts in a profile (functional, non-functional and other properties as previously described) which is translated into an ontology instance (either of a component or a microservice).

The parser first verifies that the profile is expressed in the proper context and semantics of the structures presented earlier (see Figures 7.2 and 7.3) using the ANTLR framework and then proceeds with building the ontology tree of instances according to the the recognized parts. Parsing and transformation essentially build the ontology tree instances that describe the software components under decomposition and the available microservices. The next step is to match properties between ontology items. The tree instance of the component under migration is projected on top of any other candidate microservice assessing the level of requirements fulfilment in two phases: The first phase checks that all required functions (the component's part to be replaced) are satisfied by the available microservices; therefore, we treat these as functional constraints. In this case the list of services sought (decomposed part) must be at

least a subset of the services offered (candidate microservices). The second phase is executed once all functional constraints are satisfied and calculates the level of suitability of each candidate microservice. A demonstration example for this phased approach is given in the experimental section, while a more detailed description of the matching process is provided below.

Firstly, the functionality offered by a software component is decomposed into one or more functions (methods) and the associated non-functional aspects (performance indicators). This is performed by traversing the ontology tree in a depth-first-search manner until we reach the leafs, that is, the details of the methods (e.g. interfaces, arguments, conditions, etc.) the component is made of (see component profile in figure 2). Then the algorithm climbs up the ontology structure until it reaches the definition of the method to which this detailed information refers. This way the functional parts of interest in the component's ontology instance are isolated creating a form of meta-ontology as depicted in Figure 7.1 and described earlier. The non-functional properties are then visited on the ontology tree top-down using a string-matching approach, where we differentiate between two cases: (i) The overall performance indicator(s), which describe how the component behaves as one entity of integrated functions. This will be used during the synthesis part of the matching algorithm to guide the process of recommending microservices for integration taking into consideration how their combination should behave as a whole, any incompatibilities in terms of interfacing, timing (synchronous/asynchronous), its type (SOAP, REST), etc. (see experimental part in Section 7.4); (ii) Method-specific indicators, that constrain the way a certain function (method) delivers its functionality as a single unit. This piece of information will be used by the matching algorithm when assessing the suitability of a microservice as it is considered a mandatory requirement. As soon as all meta-ontology parts (i.e. methods) are isolated, the proposed matching algorithm is invoked. Considering a single function (method) from the derived decomposition, we aim to match it with a candidate microservice that resides in the pool of available microservices. For simplicity, let the instance of the source (component) function for microservice substitution be denoted as M_{sk} ($k=1..M$, where M is the number of decomposed component functions), which is considered as the profiled microservice sought after decomposition (from now on we will refer to this as the 'source microservice'). At the other end, the profile of all of the available microservices is also parsed and ontology instances are created, let these be M_{ti} ($i=1..N$, where N is the number of available microservices). Due to the fact that there is a form of heterogeneity between microservice attributes that concern their data types, a combination of metrics is used in order to assess the matching score of each

target microservice instance T_i while taking into account the aforementioned heterogeneity.

The ontology profile, as described through EBNF, has three distinct data types, namely binary, numerical and string. Therefore, a different metric function is used for each data type. For the binary data type, the similarity function score is given by the following formula:

$$S_{bin} = \frac{1}{N} \sum_{i=1}^{N} b_{st,i} \tag{7.1}$$

where,

$$b_{st,i} = \begin{cases} 0, \ if \ binary \ attribute \ i \ required \ in \ M_s \ is \ not \ satisfied \ in \ M_t \\ 1, \ if \ binary \ attribute \ i \ required \ in \ M_s \ is \ satisfied \ in \ M_t \end{cases}$$

and M_s and M_t are the source and target microservice ontology instances respectively.

Respectively, the score between any two sets of numerical attributes is given by:

$$S_{num} = \frac{1}{N} \sum_{i=1}^{N} \{max_{st,i}, min_{st,1}\} \tag{7.2}$$

where, $max_{st,i}$ is the formula for attribute i to be maximized between source and target ontology instances given by:

$$max_{st,i} = 1 - \frac{n_{s,i} - n_{t,i}}{max(n_{s,i}, n_{t,i})} \tag{7.3}$$

and $min_{st,i}$ is the formula for attribute i to be minimized between source and target ontology instances given by:

$$min_{st,i} = 1 + \frac{n_{s,i} - n_{t,i}}{max(n_{s,i}, n_{t,i})} \tag{7.4}$$

Since some attribute values can be maximized or minimized, we use the correct formula for attribute value similarity calculation each time. For example, the attribute bytes processed per second is maximized because it has to score higher if the value offered is higher than the desired one. On the contrary, the attribute cost has to be minimized due to the exact opposite reason. Cost similarity value has to score higher if the offered value is less than the desired one.

Lastly, the score between any two sets of string attributes s_s and s_t is given by the mean of the Jaccard similarity coefficient:

$$S_{str} = \overline{(J(S_s, S_t))} \tag{7.5}$$

where (S_s, S_t) is the Jaccard similarity coefficient between source and target string sets respectively, and is calculated as:

$$J(S_s, S_t) = \frac{|S_s \bigcap S_t|}{|S_s \bigcup S_t|} = \frac{|S_s \bigcap S_t|}{|S_s| + |S_t| - |S_s \bigcap S_t|} \tag{7.6}$$

where $|S_s|$ is the number of terms contained in string S_s, $|S_t|$ is the number of terms contained in string S_t, and $|S_s \bigcap S_t|$ is the number of shared terms between strings S_s and S_t respectively.

Using the equations above we can now describe the procedural flow of the matching algorithm. The algorithm consists of two sequential phases:

Phase 1: All attributes of the source microservice, which are considered as mandatory, must map one-on-one to the attributes of the target microservice. This means that, by traversing all of the available target microservices, each attribute of the source microservice is verified to exist in the target microservice. Otherwise, the target microservice is discarded and it is removed from the pool of candidate microservices. Therefore, after Phase 1 concludes, the pool of candidate target microservices has been reformed to include only those target microservices that in general match the mandatory requirements of the source microservice; the level of suitability of the microservices in this pool may vary depending on secondary, desired features or properties they may possess, the respective values of which are subsequently assessed in Phase 2 by the score functions previously described. As previously mentioned, Phase 1 is supported by a variation of the GMO algorithm which was developed to parse every pair of the compared microservice ontologies (source and target) and defines their structural similarity.

Phase 2: The similarity between a source microservice and a specific target microservice in the pool of candidate microservices formed by Phase 1 is assessed through the relevant score functions depending on their data type. The algorithm calculates the mean of binary, numerical and string score of the pair and produces a similarity value. This is repeated for every pair of source and target microservice in the pool, and the final outcome is a ranked matching score:

$$S_{tot} = \overline{(S_{bin} + S_{num} + S_{str})} \tag{7.7}$$

The matching algorithm is shown in Algorithm 3.

Algorithm 3 Microservice ontology matching algorithm

```
#Decompose software component
source_microservices = decompose(component)

#Parse Ontologies
for s ∈ source_microservices do
    M_s = parseOntology(s)
    for M_t ∈ target_ontologies do
        #Phase 1
        candidates = []
        #Structurally similar microservices cause the target microservice to be included
        #in the microservice candidate pool
        if GMO(M_s, M_t) == 1 then
            #Phase 2
            candidates.append(M_t : score(M_s, M_t))
```

```
#The score function is the algorithm's phase 2 which is implemented below according
#to the similarity functions as defined above
function SCORE(M_s, M_t)
    S_bin = scoreBinary(M_s.getBinaryAttributes(), M_t.getBinaryAttributes())
    S_num = scoreNumerical(M_s.getNumericalAttributes(), M_t.getNumericalAttributes())
    S_str = scoreString(M_s.getStringAttributes(), M_t.getStringAttributes())
    score = (S_bin + S_num + S_str)/3
    return score
```

7.4 Experimental Process

A two-stage experimental process was designed and executed aiming to assess the efficiency of the proposed framework. Specifically, in the first stage (proof of concept) we examined the ability of the framework to deliver and recommend a list of microservices that are suitable to replace specific functions of a component, ranked based on the suitability score of equation 7.7. In the second stage (composition assessment) two MOGAS were employed to deliver near-optimal synthesis of candidate microservices taking into account the required dependencies as these were defined in the software component design. All scripts that support the aforementioned experimental environment were implemented in Python 3.7 and the full sets of results are available in this link . The two stages are described in detail below:

7.4.1 Proof of Concept

During the first stage of the experimental evaluation, we have tested the proposed framework using two specific cases. In the first case we consider having the functional parts (profiles) of a decomposed CRUD component, which provides the simple functions of create, read, update and delete for a business artefact (e.g. a customer or invoice). We assume in this case that there are no dependencies across the functional parts of the decomposed component, that is, every operation is individual and does not depend on any other operation. This means that execution of one of the above operations does not require the prior execution of another. In the second case we focus on seeking to replace the functional parts of a component that are part of an inventory system and are dedicated for invoice updating. This component consists of five different functions as follows: *Update Invoice Items*, *Update Invoice Headers*, *Update Corresponded Posting*, *Update Debtor's Balance* and *Print Invoice*. These functions are all sequentially dependent (in the order listed), that is, every function depends on its previous one and starts as soon as its predecessor has concluded.

For the execution of the experiments, two EBNF profiles, aligned with the proposed framework, were created so as to fulfil the description of the two software components in hand for the stages described above. A pool of 5000 synthetic microservices profiles were randomly constructed ensuring that a minimum number of 200 microservices match the requirements of each functional part for both software components. This intuitively means that we make sure that every decomposed part has at least 200 candidate microservices in the pool that are matched and satisfy the mandatory requirements but with unknown suitability score. This will enable examining the correctness of the matching algorithm.

The scores computed by the matching algorithm were validated by varying certain attribute values of the functional parts that derive from the decomposed component and repeating the matching process. We observed that by varying the attribute values in a series of repetitions and experiments, the matching algorithm correctly yields different scores and proper rankings among the candidate microservices as expected. This verified that changing the requirements of the functional parts triggers different scores and microservices that previously matched a specific functional part with a relatively high score tend to score lower when the requirements shift and vice versa.

7.4.2 Composition Assessment

As explained above, this experimental stage aims to examine the suitability of the utilization of heuristic approaches to deliver near-optimal microservices synthesis considering the satisfaction of two or more objectives related to non-functional characteristics of the software component. The vast solution space of the problem under study prohibits the utilization of increased time required and the complexity of the computational process. We resorted to using heuristic approaches and, more specifically, genetic algorithms, as the problem under study was rich in candidate solutions with conflicting objectives. Therefore, multi-objective genetic optimization was selected, as it has been proven to be quite efficient in such cases.

Two MOGAS were selected to solve the multi-objective optimization problem, which will also be used to compare their performance and effectiveness: The Non-dominated Sorting Genetic Algorithm II (NSGA-II) and the Strength Pareto Evolutionary Algorithm 2 (SPEA2). The selection of these two specific algorithms was made due to their wide acceptance and use, but most importantly, their good performance in such kind of applications, which was proven in this case too, after a quick verification with preliminary runs. The multi-objective optimization environment was accordingly adjusted and configured based on the problem under study. The minimization of the microservice cost and the execution time (performance) are formed the two objectives. We assume that the two are competing in the sense that the higher the performance, the more expensive the microservice. The set of decision variables was constructed by five vectors, each corresponding to a decomposed function and yielding values related to the selected candidate microservice that delivers the same functionality. Two constraints were also set, one for each objective, both denoting an upper value for the objectives (cost, time) that cannot be tampered. The experimental implementation of the algorithms was performed using Platypus[1], a Python-based multi-objective optimization algorithms library.

7.5 Results and Discussion

The results generated by the execution of the proposed process over the two experimental cases are provided in Tables 7.1 and 7.2 respectively (sample of the best five ranked microservices). The results consist of the id of the best five microservices for each functional part along with their matched score in descending order.

For the first case, a total number of 955 unique microservices have been positively assessed

[1] https://platypus.readthedocs.io

Table 7.1: Scoring results of components' functional parts without dependencies.

Create	**Read**	**Update**	**Delete**
184 (0.48)	1316 (0.62)	445 (0.89)	1317 (0.75)
37 (0.48)	227 (0.56)	406 (0.89)	814 (0.63)
91 (0.47)	353 (0.54)	524 (0.87)	747 (0.55)
53 (0.44)	236 (0. 53)	409 (0.87)	659 (0.53)
73 (0.44)	379 (0.51)	563 (0.86)	728 (0.52)

Table 7.2: Scoring results of components' functional parts with dependencies.

Print invoice	**Update invoice items**	**Update invoice headers**	**Update debtors balance**	**Update posting**
800 (0.65)	104 (0.50)	329 (0.63)	740 (0.61)	421 (0.90)
1455 (0.60)	1506 (0.49)	1612 (0.60)	692 (0.59)	545 (0.89)
1995 (0.58)	65 (0.48)	312 (0.55)	706 (0.58)	499 (0.89)
2079 (0.57)	101 (0.47)	273 (0.53)	611 (0.55)	524 (0.88)
2137 (0.57)	82 (0.47)	231 (0.52)	1506 (0.54)	1480 (0.87)

and included in the candidate microservices pool in descending order based on the calculated suitability score. Specifically, *Create* function included 249 candidates, *Read* function 225 candidates, *Update* function 233 candidates and finally *Delete* function 248 candidates. As regards the second case, a total number of 1709 unique microservices have fulfilled the mandatory requirements and were selected to be included in the candidate microservices pool as follows: 652 microservices were included in *Print Invoice* function's list, 283 in *Update Invoice* function's list, 236 microservices in *Update Invoice Headers* function's list, 280 microservices in *Update Debtors Balance* function's list, and, finally, 258 microservices are included in *Update Posting* function's list.

Firstly, we observed that the proposed algorithm performed successfully discarding all candidate microservices that failed to satisfy even one mandatory requirement. Secondly, by choosing and comparing arbitrarily microservices from the same list of candidates we confirmed the correct assessment of the microservices by the matching algorithm reflected in the calculated suitability scores, as well as the correctness of their prioritization.

As described in the experimental process design, the results extracted from the second case (software component decomposed into a series of dependent actions), were then used for the

assessment of the microservices synthesis. The number of possible solutions (PS) in this case is calculated by Equation 7.8 to be over 3 trillions.

$$|PS| = \prod_{i=1}^{N} x_i \tag{7.8}$$

N in Equation 7.8 corresponds to the number of decomposed functions and X_i is the number of recommended microservices for function i.

Each MOGA was run 100 times for 500000 fitness evaluations (FE) resulting in the generation of 100 Pareto fronts. By combining these Pareto fronts, a near-optimal Pareto front was produced for each algorithm. The two near-optimal Pareto fronts are depicted in Figure 7.6. The first observation one can make when inspecting the Pareto fronts is that both MOGAs delivered similar solutions. Going a step further and by studying the microservices combinations which corresponded to the optimal solutions, we observed that microservices with high individual suitability scores were missing from the proposed optimal solutions and respectively microservices belonging to the optimal solutions sets had relatively low suitability scores compared to others. This finding is perfectly reasonable as the specific experiment was focused on optimising cost and performance, while the suitability score is the collection of other parameters as well. Therefore, the recommended solutions that will drive the synthesis of microservices will always depend on the aspects designers need to optimise each time.

The performance of the two MOGAs was assessed and compared with the use of the Hypervolume (HV) [49] and the Inverted Generational Distance (IGD) [50] quality indicators. Each algorithm was run 10 times and both HV and IGD values were calculated for each algorithm. In order to compare the performance of the two algorithms the median HV and IGD were calculated. The HV value for both algorithms was identical and equal to 0.0257. The IGD value for the NSGA-II was 0.6113 and for SPEA2 0.6150.

Considering that the results of the two indicators suggest a balanced performance with no clear distinction being observed between the two algorithms used, we may safely conclude that none of the two overcomes the other. Two statistical tests were used to determine if there is any statistical difference between the two algorithms. Both the Wilcoxon signed-rank test and the Mann-Whitney U test suggested that there is no statistical difference (p¡0.05) between the HV and IDG results of the two algorithms. Therefore, the two MOGAs are equally suitable to offer a sound basis for automatically guided microservices synthesis.

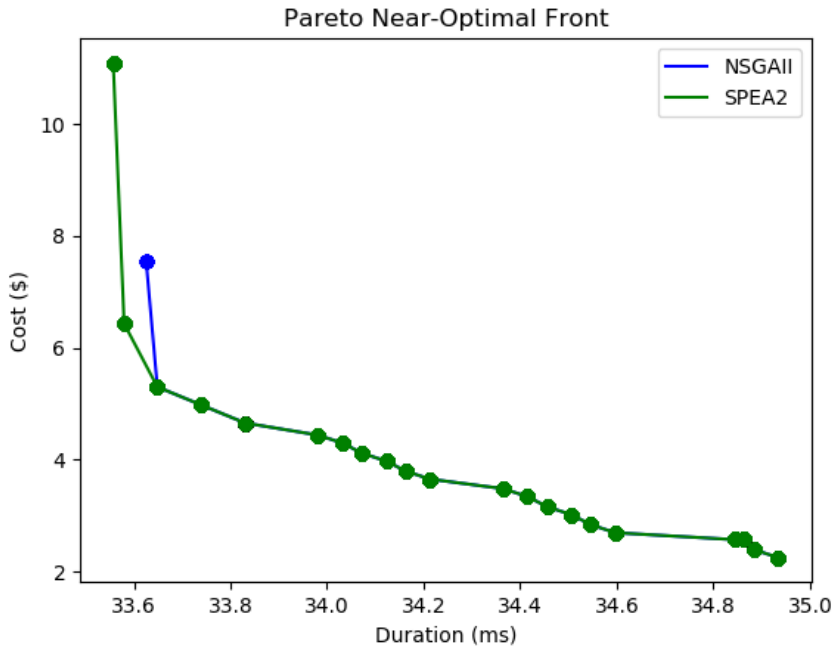


Figure 7.6: Near-optimal pareto fronts.

7.6 Summary

The contribution of the research work presented in this chapter is to support software developers to migrate from software components to microservices. This work aimed to provide a well-described automatic process that identifies and recommends the full or partial replacement of a software component's functionality from a list of available microservices. The proposed process comprises a series of tasks that a developer may follow to receive a recommended solution. The component is expressed in a semi-formal notation in EBNF, which is parsed to identify its functional parts. This identification takes place using an ontology scheme. The decomposed functions are then matched against available microservices. First, the microservices are screened based on the required functionality, and the successful candidates are scored using a matching algorithm. Additionally, the proposed process is integrated with search-based techniques and recommends the optimal synthesis of microservices yielded by Multi-Objective Genetic Algorithms. The proposed process was evaluated through a two-stage experimental process and presented a successful performance in delivering proper solutions.

Chapter 8

An Effective Resource Management Approach in a FaaS Environment

8.1 Introduction

The work in this chapter investigates and proposes a new resource management approach in a FaaS platform, based on intelligent techniques. A number of experiments were performed through a dedicated framework consisting of a client application and a Lambda function. Three GAs were employed to deliver optimal solutions in a multi-objective environment with the results appearing quite promising.

Despite the various benefits and advantages of serverless computing, like for example zero server management, no up-front provisioning, high availability, auto-scalability and pay only for the resources used, there are also several weaknesses that should also be taken into account: As currently offered from providers, it is not suitable for long term tasks because of the limited time a service can run; additionally, there is increasing complexity of the underlying architecture, which is intensified by the lack of appropriate operational tools.

This new cloud paradigm is becoming increasingly popular and is gaining great attention from the software industry and research community. This is not irrelevant with the fact that FaaS appears to be an ideal platform to host microservices since provides everything required for their development and deployment. A number of new technical challenges and open problems [114] have emerged, while a number of questions have been posed about the importance and future of serverless computing.

Resource management support in such a FaaS environment is essential for the software

development process itself, which is directed towards satisfying the SLA and providing QoS assurance. The identification of the optimum scenario for resource allocation to serve adequately a specific workload is a tedious, computationally complex and time consuming process since multiple objectives need to be satisfied. This research work addresses this challenge by investigating the implementation and application of intelligent techniques for the delivery of efficient resource management support in such an environment.

The Amazon Lambda platform is considered a complete platform as it offers the most features compared to the rest, while it presents the greatest market share; for these reasons it was selected as the experimental environment of this chapter.

8.2 Literature Overview

Very few relevant research works were identified and as a result, a short literature review has been carried out that is not limited to resource management approaches only.

Two research works refer to efficient resource management: In [115] a solution using a well-known resource allocation strategy for a Lambda platform was presented based on the model predictive controller (MPC). This solution designs a resource allocation policy through the understanding of the run-time attributes of the workload. The authors in [116] performed a dedicated test to identify if cost optimization is feasible when utilizing increased resources for lowering processing times.

Evaluation of main FaaS providers was performed in [117] and relevant results were presented in terms of throughput, network bandwidth, file I/O and performance computed according to concurrent invocations. In [118], the authors report results from a comprehensive investigation on the performance of microservices hosted by a serveless platform. The investigation dealt with implications of infrastructure elasticity, load balancing, provisioning variation, infrastructure retention, and memory reservations. A micro-benchmark introduced in [119] was used to evaluate the performance and cost model of popular FaaS providers.

An open-source FaaS tool called Snafu was introduced in [120] which is employed for managing, executing and testing functions across provider-specific interfaces.

Finally, the work of Hong et al. [121] describes six serverless design patterns that can be used to build serverless applications and services.

8.3 Multi-objective Optimization Approach

The general purpose here is to allocate a sufficient amount of resources in a FaaS environment that should be able to serve a specific workload. By utilizing an exhaustive algorithm the aim is to identify the optimal solutions for both objectives, cost and performance. Exhaustive approaches have great demands on resources and, subsequently, on cost, and thus they cannot be the answer to the first research question. In this research work an exhaustive algorithm will be applied on a low demand environment and a small scale workload, with the results obtained being considered as the reference data for the proposed intelligent approaches, the latter being able to reach to solutions faster.

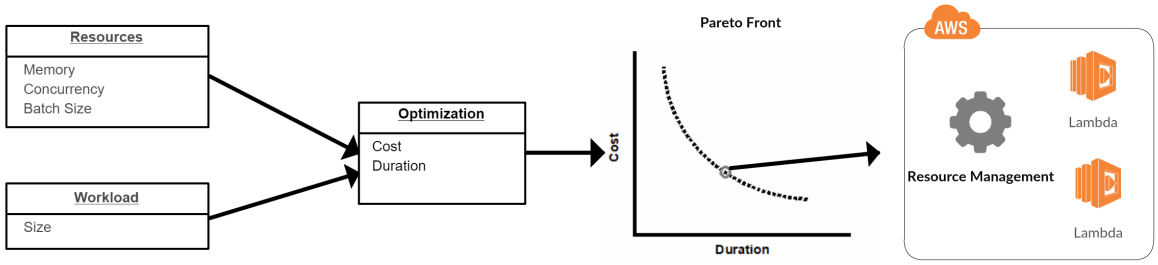


Figure 8.1: Proposed multi-objective optimization approach

This work proposes the employment of multi-objective genetic algorithms (MOGAs) [122] as the optimization method that will generate near-optimal solutions, with their performance being assessed by comparing their outputs with the reference optimal solutions extracted by the exhaustive algorithm.

As described in Chapter 2, genetic algorithms are a type of evolutionary algorithm, which are widely used to solve search-based optimization problems by simulating the theory of natural evolution on a population of individuals (candidate solutions). Problems like the one this study is dealing with, require the optimization of multiple criteria at the same time. In such a case multi-objective genetic algorithms can be adopted, where the goal is to find the best solution by otimizing a set of objective functions. In case of conflicting or competing objectives, a multi-objective genetic algorithm normally delivers the Pareto optimal set (or Pareto front) that contains those solutions that are not dominated by any other solution yielded during evolution. Since each optimal solution constitutes a specific balance between the objectives under optimization, where any improvement in one of them leads to worsening the other (conflicting targets, e.g. cost vs time in this case), a decision maker is supported to take decisions as to which values of the decision variables are most suited based on the targets and

the requirements of his/her application.

In the context of the application, the candidate solutions, or alternatively the decision variables, consist of two of the available configuration options offered by AWS Lambda platform, namely memory allocation and number of maximum concurrency functions, as well as a third variable, the batch size, that represents the number of inputs that each individual function is required to process. The proposed approach is graphically demosntrated in Figure 8.1 where the three decision variables, in conjunction with the characteristics of the workload, define the level of Cost and Duration.

8.4 Experimental Process

8.4.1 Experimental Environment

A software application that is used to perform the experimental process has been implemented by utilizing services offered by Amazon AWS platform. More specifically, this application is based on the idea introduced in Amazon Big Data Blog [1] where in a map-reduce style it counts the words in files stored in an S3 [2] bucket. This application streams the total number of words each function has calculated, returns it back to the user in real-time and demonstrates how a Lambda function can efficiently process large amounts of data in short time and provide immediate results to users.

In the context of the experimentation study, the application was implemented as follows: On the client side, besides the main function that triggers the whole process, a cascade function was also implemented which is responsible to sense the workload size and accordingly distribute the data in a synchronous way over a number of Lambda functions. This function is also responsible to collect and aggregate the results taken from the lambda functions responses; when all functions are completed it returns the final result to the user. On the AWS platform, a Lambda function was created that counts the words of a given batch of files which are stored in a AWS S3 bucket. Data stored in the S3 bucket represents the workload that will be processed with specific features. Both the client and lambda sides were implemented in Python 3.7 and the integration with the AWS services, S3 and Lambda was achieved through the Boto3 [3], the AWS SDK for Python. The synchronous invocations of Lambda functions is executed

[1] https://aws.amazon.com/blogs/big-data/building-scalable-and-responsive-big-data-interfaces-with-aws-lambda/

[2] https://aws.amazon.com/s3/

[3] https://aws.amazon.com/sdk-for-python/

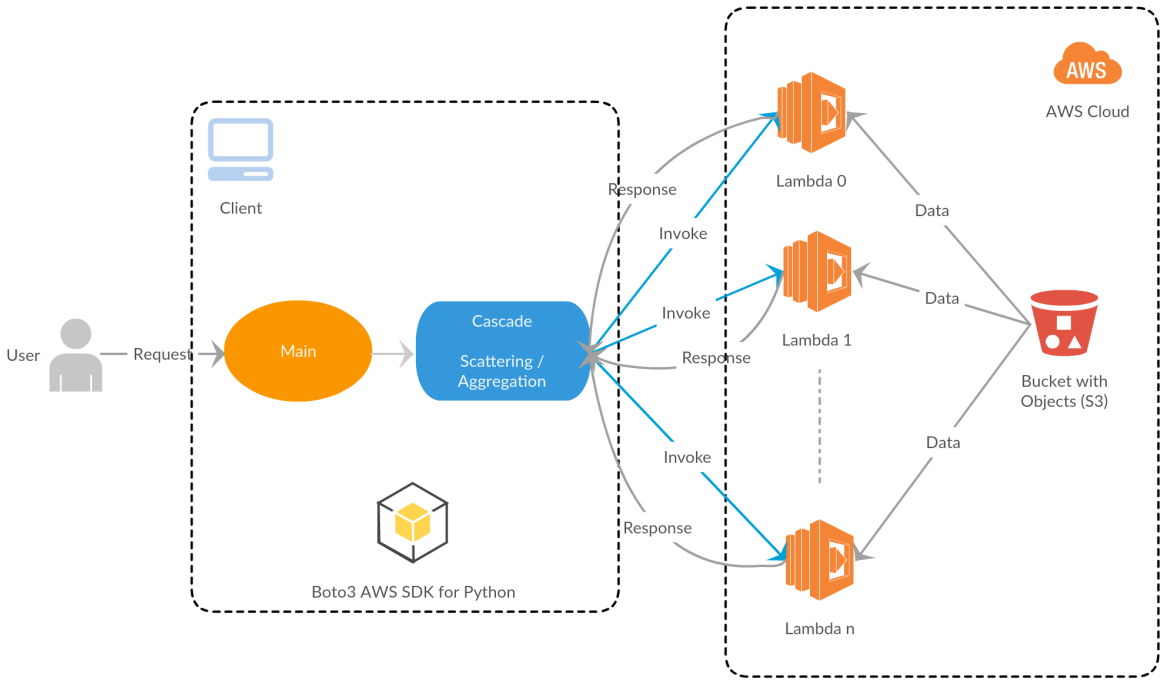


Figure 8.2: Experimental environment

and controlled by utilizing Python's multi-threading module. The integrated experimental environment is depicted in Figure 8.2.

8.4.2 Exhaustive Algorithm

As mentioned before, the proposed multi-objective optimization approach was adjusted and configured based on the AWS Lambda platform and taking into account the available options offered. The two objectives are cost and performance. The cost objective is the minimization of the total cost required for the completion of the process of the input workload and is calculated using the formulas and rules as these are given by Amazon [4]. The calculation of the cost depends on the number of Lambda functions executions, the total duration of all executed functions and the allocated memory. The performance objective is also the minimization of the total duration needed for the completion of the workload process and is calculated as the time from the moment the user sends the start request until the application delivers back to the user the total count of words. The set of decision variables consists of the memory allocation size, the number of maximum concurrent functions and the batch size. Memory allocation denotes the amount of memory you want to allocate for your lambda function. The values

[4] https://aws.amazon.com/lambda/pricing/

of memory ranged from 128MB to 3008 MB, with 64MB increment step. The concurrent execution limit varies from 1 to 1000. Finally, the batch size represents the number of files that each function will process.In our experiments the corresponding values are relative to a percentage of the workload size and fall into the following set: [1, 2, 5, 10, 20, 25, 50, 100]. The S3 bucket used for the workload, contains 100 text files, with each file containing 638 words.

The exhaustive algorithm calculated and delivered all possible candidate solutions. To reduce execution time and cost, the solutions for concurrency over 100 were discarded since the size of the workload is 100. The number of Possible Solutions (PS) was calculated using equation 8.1 to be equal to 36800.

$$|PS| = \sum_{1}^{N}\sum_{1}^{M}\sum_{1}^{K} \tag{8.1}$$

where, N=1..100, M=1..46, K=1..8

8.4.3 Multi-objective Genetic Algorithms

Three well-known and widely used MOGAs were selected to assess their ability to solve the problem in hand and compare their results: The Non-dominated Sorting Genetic Algorithm II (NSGA-II) [122], the Non-dominated Sorting Genetic Algorithm III (NSGA-III) [123] and the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [124]. The selection of these specific algorithms was made after performing some preliminary experimentation which indicated that these three present a consistently good performance.

The relative configuration of the required parameters, as well as the overall implementation of the algorithms, were performed using Platypus [5], a Python-based multi-objective optimization algorithms library. The aim was to minimize a vector consisting of the two objective functions, Cost and Duration, for values that belong to the PS set (see equation 8.2).

$$minimize f(x) = (f_{duration}(x), f_{cost}(x)), x \in PS \tag{8.2}$$

Since all three decision variables are real-valued, we accordingly use best practices for setting the MOGAs configuration. For crossover and mutation operators, the Simulated

[5] https://platypus.readthedocs.io/en/latest/index.html

Binary Crossover (SBX) and Polynomial Mutation (PM) were selected respectively. The same parameters and settings were used for all executions (see next section). As one can easily discern from Figure 8.3, all algorithms yielded very similar solutions which are almost identical to the reference optimal. A more detailed analysis of the results follows.

8.5 Results and Discussion

The execution of the exhaustive algorithm on the experimental application delivered a complete list of PS. The extracted values constitute the aggregation of five different executions in order to minimize or even eliminate possible variations between the results under exactly the same configuration conditions. As described above, the results from the exhaustive algorithm were considered as the reference data and were used for the assessment of the three MOGAs employed. Each MOGA was run 100 times for different values of fitness evaluations (FE) ranging from 500 to 4500 with increment step 500. The Pareto optimal front that emerged from the reference optimal solutions in contrast to the Pareto near-optimal solutions yielded by each MOGA for 1000 fitness evaluations, is depicted in Figure 8.3. By observing the Pareto fronts, one can easily conclude that all three MOGAs approached the optimal solutions to a high degree; in fact, in some cases the dominant MOGA solutions are exactly the same as those of the reference set. At this point some metrics will be utilised in an attempt to assess further and compare the performance [125] of the three MOGAs were employed.

8.5.1 Assessing MOGAs' Performance Through Quality Indicators

The hypervolume (HV) [49] and the inverted generational distance (IGD) [50] quality indicators were employed to assist in comparing the three MOGAs with respect to performance and scalability. Each algorithm was run 100 times for each number of FE and the median values of the HV and IGD were calculated for each algorithm. These values are presented in Tables 8.1 and 8.2 respectively.

The differences observed in the indicators values between the compared algorithms are too small, and this most probably is the result of the small complexity of the workload used; however, in cases of application workloads with increasingly greater complexity and/or scale, these differences will become more profound. As regards the HV indicator, SPEA2 presents the best performance, that is, the highest hypervolume value, and this is consistent along all fitness evaluation numbers used. Second best for this indicator is the NSGAII. It is important

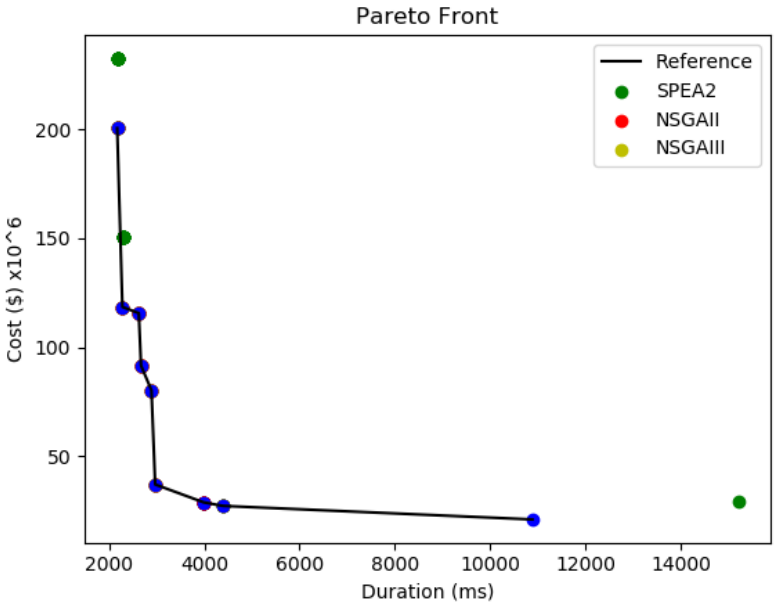


Figure 8.3: Pareto front for 1000 fitness evaluations (FE)

Table 8.1: Hypervolume(HV) values

FE	$HV\,(x10^{-6})$		
	NSGAII	**NSGAIII**	**SPEA2**
500	999667.063	999658.576	999755.043
1000	999755.043	999656.438	999960.329
1500	999960.329	999655.593	999960.329
2000	999960.329	999654.920	999960.329
2500	999960.329	999655.384	999960.329
3000	999960.329	999655.777	999960.329
3500	999960.329	999655.230	999960.329
4000	999960.329	999655.687	999960.329
4500	999960.329	999657.071	999960.329

to observe that in the case of SPEA2 with 1000 fitness evaluations, the HV value stabilizes at a constant value from the second measurement onwards. The same behaviour is also observed for NSGII but from the third measurement onwards. On the contrary, NSGIII presents more fluctuations in its measurements.

In the case of the IGD indicator, things appear more complicated since none of the algorithms seems to prevail. A notable point is that for the lowest measure of the fitness evaluations numbers, SPEA2 clearly outperforms the others. The values for NSGII and SPEA2 starting

Table 8.2: Inverted Generational Distance(IGD) values

FE	IGD ($x10^{-6}$)		
	NSGAII	**NSGAIII**	**SPEA2**
500	60727.255	61344.367	57375.237
1000	60250.148	57791.945	57792.130
1500	57792.130	57792.189	57792.130
2000	57792.130	57787.907	57792.130
2500	57792.130	57792.125	57792.130
3000	57792.130	57791.964	57792.130
3500	57792.130	57790.733	57792.130
4000	57792.130	57792.097	57792.130
4500	57792.130	57792.179	57792.130

from the third measurement onwards are fully identical, while NSGAIII fluctuates more, while in three cases it seems better than the other two.

Table 8.3: Pairwise comparison for HV indicator

	NSGAII	**NSGAIII**	**SPEA2**
NSGAII		**0.074**	**0.18**
NSGAIII			0.005
SPEA2			

Table 8.4: Pairwise comparison for IGD indicator

	NSGAII	**NSGAIII**	**SPEA2**
NSGAII		**0.241**	**0.18**
NSGAIII			**0.285**
SPEA2			

The Wilcoxon signed-rank test was applied on both quality indicators to detect whether or not a statistically significant difference exists among the three algorithms. To handle the family-wise error rate accumulated, *p-values* were adjusted using a post-hoc Holm procedure. The *p-values* resulting from the pairwise comparison for the HV and IGD indicators are shown in Tables 8.3 and 8.4 respectively, where pairs of algorithms with a statistically significant difference *($p < 0.05$)* are shown in boldface. According to the pairwise comparisons, no

significant difference is observed between NSGA-II and NSGA-III, NSGA-II and SPEA2, and NSGA-III and SPEA2 in either indicator except only for the pair NSGA-III and SPEA2 in the case of HV.

8.6 Summary

This chapter presented a preliminary investigation to assess whether heuristic approaches for multi-objective optimization, are able to deliver near-optimal solutions that support developers in a FaaS environment to select an efficient resource allocation scheme with respect to cost and time. A dedicated experimental process was designed over a specific framework consisting of a client application and a Lambda function. Firstly, an exhaustive algorithm was utilised with the aim to identify the optimal solutions for both objectives, cost and performance. Subsequently, three multi-objective genetic algorithms (MOGAs) were employed to generate near-optimal solutions and their performance was compared with the reference optimal solutions extracted by the exhaustive algorithm. The results were particularly successful and confirmed the ability of the proposed approaches to accurately approximate the optimal solutions.

Chapter 9

Conclusions and Future Research Steps

9.1 Overview

The present thesis was motivated by a series of challenges from a particular scientific area that deals with several aspects of software engineering for distributed systems and Cloud environments. In addition, the management of the Cloud infrastructure is characterised as highly complex, with multi-conflicting factors due to the continuous development and integration of various new technologies.

The main target of this thesis was to support decision making in different problems of distributed software systems development and servicing on the Cloud environment, through the introduction of models, techniques, and methods belonging to the broader area of Computational Intelligence. Such approaches appear to have profound success when dealing with complex and multifaceted problems. Moreover, an investigation over a series of challenges in the Cloud-based software development process was performed.

The contribution of this thesis may be summarized in six major research steps and outcomes. The first step introduced a novel, integrated analysis framework based on Multi-Layer Fuzzy Cognitive Maps models which were used as the spearhead of all modeling cases thereafter. The framework included also a series of actions to gather useful static and dynamic information of the model developed. The second step of our research focused on the analysis and study of the factors that affect the adoption of Cloud services. The third step extended the aforementioned analysis framework with the incorporation of an evolutionary approach based on a new formulation and computational execution of the nodes in the model. Next, the fourth research step involved the construction of a dedicated Multi-Layer Fuzzy Cognitive Map to support

decision-making towards microservices architecture migration. The fifth step proposed a novel process guiding the decomposition of existing software components and their partial or full replacement with a number of suitable and available microservices. Finally, in the sixth step, a new Cloud resource management approach was introduced in a Function-as-a-Service platform.

Future research steps involve many potential topics that can be studied and explored further. These steps are aimed at two directions: The first direction targets the improvement and extension of the current approaches and models, while the second involves the application of these models to more real-world cases. The analysis of future work is described in each of the topics addressed in this thesis in the following sub-sections.

9.2 A Framework for Analyzing Multi-Layer Fuzzy Cognitive Maps

The introduction of a novel framework for the analysis of MLFCM models was motivated by the lack of methodologies for understanding how FCM models work. In this respect, the corresponding research work addressed the issue of the analysis, both static and dynamic, of MLFCM models by proposing a framework for conducting a series of steps that aim to reveal hidden properties in their execution. The proposed framework was divided into two approaches: The first studied the MLFCM as a graph model extracting information about its complexity, the significance of the discrete nodes at every layer and its tendency to promote or inhibit an initial activation as a result of the presence of a number of positive and negative cycles. The second approach analyzed the run-time behavior of the model and provided insights regarding its behavior in terms of correctness, correlations between nodes and the effect of the latter to the final outcome yielded. Executions of different simulations and what-if scenarios enabled the revision and restructuring of the model taking into account issues like complexity and computational burden. The demonstration of the framework over real world problems proved the efficiency and efficacy of the proposed approach, as the results obtained described successfully and fully the relationships between the factors that affected such a decision. Moreover, the findings of the two types of analysis showed that the framework is able to work equally well in different domains and sizes of models, with the results being perfectly aligned with the real circumstances faced in the corresponding application domain, as these were described by the corresponding literature and the domain experts.

The future research steps in this topic will involve the application of the framework on several

other real-world problems and will investigate further the issues of computational complexity and ease of use. A software tool is currently under development, which will automate the different steps of the framework, and especially support the dynamic analysis by offering an interactive way for defining and executing what-if scenarios in conjunction with the results of the static analysis.

9.3 Modeling the Cloud Adoption Decision

Although Cloud Computing has gone from infancy to a more mature state, customers are still facing many challenges with respect to its adoption. The study and understanding of various parameters, such as benefits and problems that are involved in this transition, is far from an easy and straightforward procedure. This challenge was addressed by investigating the applications of integrated models in different forms to support the decision making process on the Cloud adoption. The application of two methodologies based on FCMs and IDs were examined. All of the proposed models were formed using on one hand parameters that are addressed in the relevant literature, and expert knowledge on the other. In general, all models followed the same rationale as described above, but in each case separately the process was repeated and adapted to the characteristics of each model. In the ID modeling it was demonstrated how two new models based on Influence Diagrams can be constructed and applied to face decision making issues of the Cloud adoption problem. Both ID models succeeded in matching their estimation with the corresponding real decisions in three real-world cases. Moreover, the two models successfully demonstrated their ability to execute hypothetical scenarios in an interactive environment, something which suggests their efficiency in actively supporting the decision-making process. While both models performed equally well, the fuzzy version clearly outperformed the generic one in flexibility and the ability to handle larger structures.

Modeling and analysis of this problem continued with FCMs. In this context, two FCM models were constructed and applied to face the decision making on the Cloud adoption problem, exploiting the advantages offered by using techniques that combine fuzzy logic and neural networks. Both models were based on the CNFCM type, with the former being single-layered and the latter multi-layered. The resulted MLFCM was a significantly enhanced model compared to the single-layer CNFCM, both in terms of the number and type of the participating parameters, as well as of the way these were organized in layers of interacting concepts thus requiring a totally new computational approach. The models were evaluated by applying them over four real-world scenarios collected from experts/developers in the local

Cloud software industry. Although both models exhibited high-performance succeeding in matching their estimation with the corresponding real decisions, the multi-layered form of the MLFCM enabled a more detailed investigation so as to trace the causes for its behavior in each case.

Although the results from the application of both FCM and ID approaches in the Cloud adoption problem may be considered quite encouraging, there are quite a few steps that may be executed in the future to enrich and optimize these models. The speed with which Cloud Computing and its corresponding technology evolves necessitates the continuous study of the parameters (factors) taking part in the models. Also, more real-world case scenarios could give helpful feedback for better calibration of the models used. In addition, possible expansion and re-identification of the maps and/or the diagrams, respectively, will be investigated to include more nodes representing better the real Cloud environment and reflecting more accurately the decision scenery.

9.4 A Novel Computational Approach for MLFCM

Beyond the very encouraging results yielded by the application and analysis of the MLFCM thus far, the need for improvement also emerged. Towards the delivery of a sufficient and enhanced model, a new MLFCM computational process was introduced. The proposed model involved two new approaches. The first approach comprised a new FCM formulation that handles a number of weaknesses and drawbacks of existing methods, and boosts up the abilities for multidimensional and multi-targeted analyses. The second approach introduced the utilization of a genetically evolved algorithm as an extension of the dynamic analysis, the ALGE-MLFCM. This algorithm evolves the initial activation levels of the concepts participating in the MLFCM aiming at finding solutions that satisfy a target final activation value (i.e. after execution of the map) of the concept(s) of interest corresponding to a certain scenario. This work can be considered as an extension of the previous steps as it proposed an integrated framework with a novel model able to cope with the challenges imposed by the need to analyze and explain complex decisions.

More experimental applications to real-world problems are included in the future research steps of this topic. Through these applications, the assessment of the proposed approach in terms of applicability and efficiency will become feasible, as well as the demonstration of its generalizability.

9.5 Supporting the Decision of Migrating to Microservices Architecture

The number of factors involved in the decision, as well as the complexities presented in the connections between them, make the decision to adopt the microservices architecture extremely tough. A specially designed MLFCM model was introduced aiming to provide enhanced decision support capabilities and increased explainability. The model construction and analysis process utilized the enhanced framework for MLFCM introduced in this thesis. The model captured the dynamics of the problem under study and the hidden properties of the model were highlighted through the static and dynamic analysis. The model was evaluated over an industrial case study and successfully managed to match the real decision, as well as to correctly describe the status of each concept after execution. Moving a step further, the outcome of the model over the industrial case was analyzed by studying the solutions resulted by ALGE-MLFCM and suggesting changes and improvements to the company's environment aiming to strengthen the decision in favor of microservices adoption. The practitioners led this investigation and defined concepts of significant interest to study in detail their behavior, interrelation and contribution to the final decision of the model. The whole process was acknowledged as quite supportive and informative, while the flexibility in setting-up and executing hypothetical scenarios was commented very positively by the participating practitioners.

The future research steps in this topic include, at domain level, further improvement of the model by considering other concepts at finer levels of detail, i.e. decomposing more nodes to sub-FCMs, and then performing simulations with new what-if scenarios engaging yet more experts with different backgrounds. At MLFCM level, effort will be devoted to test different activation functions, e.g. hyperbolic tangent, instead of the sigmoid and assess whether there is improvement of performance. Finally, future work will involve integrating multi-objective optimization approaches with ALGE-MLFCM and extending the pool of scenarios, as well as automating the reasoning process applied over the produced results.

9.6 Migration of Software Components to Microservices: Matching and Synthesis

Although, a significant number of software systems which are based on the component-based architecture are currently in operation, the current literature which targets the migration from

component-based development to microservices is rare, if not non-existent. Aiming to fill this gap, a well-described automatic process was introduced that identifies and recommends the full or partial replacement of a software component's functionality by a number of available microservices. The proposed process comprises a series of tasks which a developer may follow to receive a recommended solution. The component is expressed in a semi-formal notation in EBNF which is parsed to identify its functional parts. This identification takes place using an ontology scheme. The decomposed functions are then matched against available microservices. First, the microservices are screened based on the required functionality and the successful candidates are scored using a matching algorithm. Additionally, the proposed process is integrated with search-based techniques and recommends the optimal synthesis of microservices yielded by Multi-Objective Genetic Algorithms. The proposed process was evaluated through a two stage experimental process and presented successful performance in delivering proper solutions.

Quite a few challenges and open issues still exist in this area, some of which may constitute future work. Specifically, the constant increase in the availability of microservices with business orientation will require the design and execution of more advanced and extended experiments. Furthermore, an investigation will be performed for improving the profiling tasks by adopting different description models and assess whether this may improve also the automation level of the proposed process. Finally, more real-world cases will be employed to assess further the practical benefits of the proposed approach.

9.7 An Effective Resource Management Approach in a FaaS Environment

The final research step involved a preliminary investigation to assess whether heuristic approaches for multi-objective optimization are able to solve the problem of finding a set of near-optimal solutions that support developers to select an efficient resource allocation scheme with respect to cost and time. The general aim was to allocate a sufficient amount of resources in a FaaS environment that should be able to serve a specific workload. This target was verified through a dedicated experimental process which involved an application that utilized services offered by Amazon AWS platform. An exhaustive algorithm was executed over the experimental application that yielded a set of optimal solutions which considered as the reference data. Then, the execution of three well-known MOGAs followed, with each of them resulting in with a set of near-optimal solutions. The Pareto optimal front that emerged from

the reference optimal solutions in contrast to the Pareto near-optimal solutions yielded by each MOGA was used to assess the MOGAs performance. The results demonstrated the success of the algorithms to approximate optimal solutions very accurately and thus demonstrated their ability to serve the problem under study adequately.

Future research steps involve, primarily, examining the scalability of the proposed approach in terms of checking whether it performs equally well on larger workloads. Furthermore, it is worth investigating further the performance of the MOGAs in real-time response environments, ”execution on the fly as workloads come in” and the level to which this support can be generalized to cover also workloads of unknown characteristics.

REFERENCES

[1] Engelbrecht AP. Computational intelligence: an introduction. John Wiley & Sons; 2007.

[2] Konar A. Computational intelligence: principles, techniques and applications. Springer Science & Business Media; 2006.

[3] Lewis J, Fowler M. Microservices: a definition of this new architectural term; 2014. Retrieved from: `http://martinfowler.com/articles/microservices.html`.

[4] Hassan S, Bahsoon R. Microservices and their design trade-offs: A self-adaptive roadmap. In: 2016 IEEE International Conference on Services Computing (SCC). IEEE; 2016. p. 813–818.

[5] Balalaie A, Heydarnoori A, Jamshidi P. Microservices architecture enables devops: Migration to a cloud-native architecture. Ieee Software. 2016;33(3):42–52.

[6] Wootton B. Microservices: a definition of this new architectural term; 2014. Retrieved from: `http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html`.

[7] Esposito C, Castiglione A, Choo KKR. Challenges in delivering software in the cloud as microservices. IEEE Cloud Computing. 2016;3(5):10–14.

[8] Sommerville I. Software engineering 9th Edition. ISBN-10. 2011;137035152.

[9] Bauer FL. Software engineering: report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968. Scientific Affairs Division, NATO; 1969.

[10] A Vouk M. Cloud computing–issues, research and implementations. Journal of computing and information technology. 2008;16(4):235–246.

[11] Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, et al. Above the clouds: A berkeley view of cloud computing. Dept Electrical Eng and Comput Sciences, University of California, Berkeley, Rep UCB/EECS. 2009;28(13):2009.

[12] Mell P, Grance T, et al. The NIST definition of cloud computing. 2011;.

[13] Kleinrock L. A vision for the Internet. ST Journal of Research. 2005;2(1):4–5.

[14] Wei Y, Blake MB. Service-oriented computing and cloud computing: Challenges and opportunities. IEEE Internet Computing. 2010;14(6):72–75.

[15] Bruneliere H, Cabot J, Jouault F. Combining Model-Driven Engineering and Cloud Computing. In: Modeling, Design, and Analysis for the Service Cloud-MDA4ServiceCloud'10: Workshop's 4th edition (co-located with the 6th European Conference on Modelling Foundations and Applications-ECMFA 2010); 2010. .

[16] Klein A, Mannweiler C, Schneider J, Schotten HD. Access schemes for mobile cloud computing. In: 2010 eleventh international conference on mobile data management. IEEE; 2010. p. 387–392.

[17] Dinh HT, Lee C, Niyato D, Wang P. A survey of mobile cloud computing: architecture, applications, and approaches. Wireless communications and mobile computing. 2013;13(18):1587–1611.

[18] Grobauer B, Walloschek T, Stocker E. Understanding cloud computing vulnerabilities. IEEE Security & privacy. 2010;9(2):50–57.

[19] Guha R. Impact of semantic web and cloud computing platform on software engineering. In: Software Engineering Frameworks for the Cloud Computing Paradigm. Springer; 2013. p. 3–24.

[20] Fowler M, Lewis J. Microservices a definition of this new architectural term. URL: http://martinfowler com/articles/microservices html. 2014;p. 22.

[21] Fowler M. Monolith first (2015); 2017.

[22] Newman S. Building microservices: designing fine-grained systems. " O'Reilly Media, Inc."; 2015.

[23] Richards M. Microservices vs. service-oriented architecture. O'Reilly Media; 2015.

[24] Wilde N, Gonen B, El-Sheikh E, Zimmermann A. Approaches to the evolution of SOA systems. In: Emerging Trends in the Evolution of Service-Oriented and Enterprise Architectures. Springer; 2016. p. 5–21.

[25] Fox GC, Ishakian V, Muthusamy V, Slominski A. Status of serverless computing and function-as-a-service (faas) in industry and research. arXiv preprint arXiv:170808028. 2017;.

[26] Adzic G, Chatley R. Serverless computing: economic and architectural impact. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM; 2017. p. 884–889.

[27] AWS Lambda;. Accessed: 2018-09-30. https://aws.amazon.com/lambda/.

[28] IBM Cloud Functions;. Accessed: 2018-09-30. https://www.ibm.com/cloud/functions.

[29] Google Cloud Functions;. Accessed: 2018-09-30. https://cloud.google.com/functions/.

[30] Microsoft Azure Functions;. Accessed: 2018-09-30. https://azure.microsoft.com/en-us/services/functions/.

[31] Jang JSR, Sun CT, Mizutani E. Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. 1997;.

[32] Earl C. The Fuzzy systems handbook. A practitioner's guide to building, using, and maintaining fuzzy systems. AP Professional.; 1994.

[33] Kosko B. Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence/book and disk. Vol 1Prentice hall. 1992;.

[34] Felix G, Nápoles G, Falcon R, Froelich W, Vanhoof K, Bello R. A review on methods and software for fuzzy cognitive maps. Artificial Intelligence Review. 2017;p. 1–31.

[35] Bueno S, Salmeron JL. Benchmarking main activation functions in fuzzy cognitive maps. Expert Systems with Applications. 2009;36(3):5221–5229.

[36] Christoforou A, Andreou AS. A framework for static and dynamic analysis of multi-layer fuzzy cognitive maps. Neurocomputing. 2017;232:133–145.

[37] Mateou NH, Andreou AS. Tree-structured multi-layer fuzzy cognitive maps for modelling large scale, complex problems. In: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). vol. 2. IEEE; 2005. p. 131–139.

[38] Mateou N, Andreou AS, Stylianou C. A new traversing and execution algorithm for multilayered fuzzy cognitive maps. In: 2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence). IEEE; 2008. p. 2216–2223.

[39] Papatheocharous E, Trikomitou D, Yiasemis PS, Andreou AS. Cost Modeling and Estimation in Agile Software Development Environments using Influence Diagrams. In: ICEIS (3); 2011. p. 117–127.

[40] Howard RA. Readings on the principles and applications of decision analysis. vol. 1. Strategic Decisions Group; 1983.

[41] Shachter RD. Evaluating influence diagrams. Operations research. 1986;34(6):871–882.

[42] Mateou NH, Hadjiprokopis A, Andreou AS. Fuzzy influence diagrams: an alternative approach to decision making under uncertainty. In: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). vol. 1. IEEE; 2005. p. 58–64.

[43] Howard RA, Matheson JE. The principles and applications of decision analysis. Strategic Decisions Group, Palo Alto, CA. 1984;p. 719–762.

[44] Mitchell M. An introduction to genetic algorithms. MIT press; 1998.

[45] Dumitrescu D, Lazzerini B, Jain LC, Dumitrescu A. Evolutionary computation. CRC press; 2000.

[46] Holland JH, et al. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press; 1992.

[47] Murata T, Ishibuchi H. MOGA: multi-objective genetic algorithms. In: IEEE international conference on evolutionary computation. vol. 1; 1995. p. 289–294.

[48] Audet C, Bigeon J, Cartier D, Le Digabel S, Salomon L. Performance indicators in multiobjective optimization. Optimization Online. 2018;.

[49] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE transactions on Evolutionary Computation. 1999;3(4):257–271.

[50] Van Veldhuizen DA, Lamont GB. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. Evolutionary computation. 2000;8(2):125–147.

[51] Mateou N, Andreou A, Stylianou C. Evolutionary multilayered fuzzy cognitive maps: a hybrid system design to handle large-scale, complex, real-world problems. In: 2006 2nd International Conference on Information & Communication Technologies. vol. 1. IEEE; 2006. p. 1663–1668.

[52] Papageorgiou EI, Salmeron JL. A review of fuzzy cognitive maps research during the last decade. IEEE Transactions on Fuzzy Systems. 2012;21(1):66–79.

[53] Tsadiras AK, Kouskouvelis I, Margaritis KG. Using fuzzy cognitive maps as a decision support system for political decisions. In: Panhellenic Conference on Informatics. Springer; 2001. p. 172–182.

[54] Stylios CD, Georgopoulos VC, Malandraki GA, Chouliara S. Fuzzy cognitive map architectures for medical decision support systems. Applied Soft Computing. 2008;8(3):1243–1251.

[55] Iakovidis DK, Papageorgiou E. Intuitionistic fuzzy cognitive maps for medical decision making. IEEE Transactions on Information Technology in Biomedicine. 2010;15(1):100–107.

[56] Wilson RJ. Introduction to Graph Theory. Longman; 1996.

[57] Kosko B. Fuzzy cognitive maps. International journal of man-machine studies. 1986;24(1):65–75.

[58] Howard RA, Matheson JE. Influence diagrams. Decision Analysis. 2005;2(3):127–143.

[59] Khajeh-Hosseini A, Greenwood D, Smith JW, Sommerville I. The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. Software: Practice and Experience. 2012;42(4):447–465.

[60] Kim W, Kim SD, Lee E, Lee S. Adoption issues for cloud computing. In: Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia. ACM; 2009. p. 2–5.

[61] Wu WW. Developing an explorative model for SaaS adoption. Expert systems with applications. 2011;38(12):15057–15064.

[62] Wu WW. Mining significant factors affecting the adoption of SaaS using the rough set approach. Journal of Systems and Software. 2011;84(3):435–441.

[63] Wu WW, Lan LW, Lee YT. Exploring decisive factors affecting an organization's SaaS adoption: A case study. International Journal of Information Management. 2011;31(6):556–563.

[64] Gabus A, Fontela E. World problems, an invitation to further thought within the framework of DEMATEL. Battelle Geneva Research Center, Geneva, Switzerland. 1972;p. 1–8.

[65] Menzel M, Ranjan R. CloudGenius: decision support for web server cloud migration. In: Proceedings of the 21st international conference on World Wide Web. ACM; 2012. p. 979–988.

[66] Misra SC, Mondal A. Identification of a company's suitability for the adoption of cloud computing and modelling its corresponding Return on Investment. Mathematical and Computer Modelling. 2011;53(3-4):504–521.

[67] Khajeh-Hosseini A, Sommerville I, Bogaerts J, Teregowda P. Decision support tools for cloud migration in the enterprise. In: 2011 IEEE 4th International Conference on Cloud Computing. IEEE; 2011. p. 541–548.

[68] Greenwood D, Khajeh-Hosseini A, Smith J, Sommerville I. The cloud adoption toolkit: Addressing the challenges of cloud adoption in enterprise. Arxiv preprint; 2010.

[69] Andrikopoulos V, Darsow A, Karastoyanova D, Leymann F. CloudDSF–the cloud decision support framework for application migration. In: European Conference on Service-Oriented and Cloud Computing. Springer; 2014. p. 1–16.

[70] Andrikopoulos V, Strauch S, Leymann F. Decision support for application migration to the cloud. Proceedings of CLOSER. 2013;13:149–155.

[71] Beserra PV, Camara A, Ximenes R, Albuquerque AB, Mendonça NC. Cloudstep: A step-by-step decision process to support legacy application migration to the cloud. In: 2012 IEEE 6th international workshop on the maintenance and evolution of service-oriented and cloud-based systems (MESOCA). IEEE; 2012. p. 7–16.

[72] Tsadiras AK, Margaritis KG. Cognitive mapping and certainty neuron fuzzy cognitive maps. Information Sciences. 1997;101(1-2):109–130.

[73] Došilović FK, Brčić M, Hlupić N. Explainable artificial intelligence: A survey. In: 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO). IEEE; 2018. p. 0210–0215.

[74] Adadi A, Berrada M. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). IEEE Access. 2018;6:52138–52160.

[75] Carvalho DV, Pereira EM, Cardoso JS. Machine Learning Interpretability: A Survey on Methods and Metrics. Electronics. 2019;8(8):832.

[76] Arrieta AB, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion. 2020;58:82–115.

[77] Groumpos PP. Overcoming Intelligently Some of the Drawbacks of Fuzzy Cognitive Maps. In: 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA). IEEE; 2018. p. 1–6.

[78] Eleni V, Petros G. New concerns on fuzzy cognitive maps equation and sigmoid function. In: 2017 25th Mediterranean Conference on Control and Automation (MED). IEEE; 2017. p. 1113–1118.

[79] Goldberg DE, Holland JH. Genetic algorithms and machine learning. Machine learning. 1988;3(2):95–99.

[80] Fowler M. Microservices Premium; 2015. Retrieved from: `https://martinfowler.com/bliki/MicroservicePremium.html`.

[81] Zimmermann A, Schmidt R, Sandkuhl K, Jugel D, Bogner J, Möhring M. Decision-Controlled Digitization Architecture for Internet of Things and Microservices. In: International Conference on Intelligent Decision Technologies. Springer; 2017. p. 82–92.

[82] Bass L, Weber I, Zhu L. DevOps: A Software Architect's Perspective. Addison-Wesley Professional; 2015.

[83] Baresi L, Guinea S, Leva A, Quattrocchi G. A Discrete-time Feedback Controller for Containerized Cloud Applications. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. FSE 2016. ACM; 2016. p. 217–228.

[84] Evans E. Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional; 2004.

[85] Richardson C. Microservices Architectures: Who is using microservices?; 2014. Retrieved from: http://microservices.io/articles/whoisusingmicroservices.html.

[86] Taibi D, Lenarduzzi V, Pahl C. Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. IEEE Cloud Computing. 2017 September;4(5):22–32.

[87] Balalaie A, Heydarnoori A, Jamshidi P. Migrating to cloud-native architectures using microservices: An experience report. In: European Conference on Service-Oriented and Cloud Computing. Springer; 2015. p. 201–215.

[88] Fowler M. Monolith First; 2015. Retrieved from: `http://martinfowler.com/bliki/MonolithFirst.html`.

[89] Kitchenham B. Guidelines for performing systematic literature reviews in software engineering. In: Technical report, Ver. 2.3 EBSE Technical Report. EBSE. sn; 2007. .

[90] 2017. Scopus; 2017. `https://www.scopus.com/search/`.

[91] 2017. ScienceDirect; 2017. `https://www.sciencedirect.com/`.

[92] 2017. Wiley Online Library; 2017. `https://onlinelibrary.wiley.com/`.

[93] 2017. IEEE Xplore Digital Library; 2017. `https://ieeexplore.ieee.org/Xplore/`.

[94] 2017. Springer Link; 2017. https://link.springer.com/.

[95] 2017. ACM Digital Library; 2017. https://dl.acm.org/.

[96] Wohlin C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering. ACM; 2014. p. 38.

[97] Pedrycz W. Why triangular membership functions? Fuzzy Sets and Systems. 1994;64(1):21 – 30. Available from: http://www.sciencedirect.com/science/article/pii/0165011494900035.

[98] Huang Z, Shen Q. A new fuzzy interpolative reasoning method based on center of gravity. In: The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ'03.. vol. 1. IEEE; 2003. p. 25–30.

[99] Cai X, Lyu MR, Wong KF, Ko R. Component-based software engineering: technologies, development frameworks, and quality assurance schemes. In: Proceedings Seventh Asia-Pacific Software Engeering Conference. APSEC 2000. IEEE; 2000. p. 372–379.

[100] Dragoni N, Giallorenzo S, Lafuente AL, Mazzara M, Montesi F, Mustafin R, et al. Microservices: yesterday, today, and tomorrow. In: Present and ulterior software engineering. Springer; 2017. p. 195–216.

[101] Rosen M, Lublinsky B, Smith KT, Balcer MJ. Applied SOA: service-oriented architecture and design strategies. John Wiley & Sons; 2012.

[102] Andreou AS, Papatheocharous E. Automatic matching of software component requirements using semi-formal specifications and a CBSE ontology. In: 2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE). IEEE; 2015. p. 118–128.

[103] Baresi L, Garriga M, De Renzis A. Microservices identification through interface analysis. In: European Conference on Service-Oriented and Cloud Computing. Springer; 2017. p. 19–33.

[104] Gysel M, Kölbener L, Giersche W, Zimmermann O. Service cutter: A systematic approach to service decomposition. In: European Conference on Service-Oriented and Cloud Computing. Springer; 2016. p. 185–200.

[105] Levcovitz A, Terra R, Valente MT. Towards a technique for extracting microservices from monolithic enterprise systems. arXiv preprint arXiv:160503175. 2016;.

[106] Barba L. Computing high-Reynolds number vortical flows: a highly accurate method with a fully meshless formulation. In: Parallel Computational Fluid Dynamics 2004. Elsevier; 1996. p. 305–312.

[107] Moghaddam M, Davis JG. Service selection in web service composition: A comparative review of existing approaches. In: Web Services Foundations. Springer; 2014. p. 321–346.

[108] Jatoth C, Gangadharan G, Buyya R. Computational intelligence based QoS-aware web service composition: A systematic literature review. IEEE Transactions on Services Computing. 2015;10(3):475–492.

[109] Zeginis C, Plexousakis D. Web service adaptation: State of the art and research challenges. Institute of Computer Science, FORTH-ICS, Heraklion, Crete, Greece, Tech Rep Technical Report. 2010;410.

[110] Hu W, Jian N, Qu Y, Wang Y. Gmo: A graph matching for ontologies. In: Proceedings of K-CAP Workshop on Integrating Ontologies; 2005. p. 41–48.

[111] Frappier M, Matwin S, Mili A. Software metrics for predicting maintainability. Software Metrics Study: Tech Memo. 1994;2.

[112] Zaremski AM, Wing JM. Signature matching: a tool for using software libraries. ACM Transactions on Software Engineering and Methodology (TOSEM). 1995;4(2):146–170.

[113] Zaremski AM, Wing JM. Specification matching of software components. ACM Transactions on Software Engineering and Methodology (TOSEM). 1997;6(4):333–369.

[114] Baldini I, Castro P, Chang K, Cheng P, Fink S, Ishakian V, et al. Serverless computing: Current trends and open problems. In: Research Advances in Cloud Computing. Springer; 2017. p. 1–20.

[115] HoseinyFarahabady M, Taheri J, Tari Z, Zomaya AY. A dynamic resource controller for a lambda architecture. In: Parallel Processing (ICPP), 2017 46th International Conference on. IEEE; 2017. p. 332–341.

[116] Hoang A. Analysis of microservices and serverless architecture for mobile application enablement. California State University, Northridge; 2017.

[117] Lee H, Satyam K, Fox G. Evaluation of production serverless computing environments. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). IEEE; 2018. p. 442–450.

[118] Lloyd W, Ramesh S, Chinthalapati S, Ly L, Pallickara S. Serverless computing: An investigation of factors influencing microservice performance. In: Cloud Engineering (IC2E), 2018 IEEE International Conference on. IEEE; 2018. p. 159–169.

[119] Back T, Andrikopoulos V. Using a Microbenchmark to Compare Function as a Service Solutions. In: European Conference on Service-Oriented and Cloud Computing. Springer; 2018. p. 146–160.

[120] Spillner J. Snafu: Function-as-a-service (faas) runtime design and implementation. arXiv preprint arXiv:170307562. 2017;.

[121] Hong S, Srivastava A, Shambrook W, Dumitras T. Go serverless: securing cloud via serverless design patterns. In: 10th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 18). USENIX; 2018. .

[122] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation. 2002;6(2):182–197.

[123] Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based non dominated sorting approach, part I: Solving problems with box constraints. IEEE Trans Evolutionary Computation. 2014;18(4):577–601.

[124] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-report. 2001;103.

[125] Riquelme N, Von Lücken C, Baran B. Performance metrics in multi-objective optimization. In: Computing Conference (CLEI), 2015 Latin American. IEEE; 2015. p. 1–11.