



THÈSE

PRÉSENTÉ À

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI (UQAC)

COMME EXIGENCE PARTIELLE

DE LA THÈSE DU DOCTORAT EN SCIENCES ET TECHNOLOGIES DE

L'INFORMATION (3081)

PROGRAM DE L'UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS (UQO)

PAR

AHMAD KARAWASH

SMARTCELLS: A BIO-CLOUD THEORY TOWARDS INTELLIGENT CLOUD

COMPUTING SYSTEM

JUILLET 2015

RÉSUMÉ

Le « *Cloud Computing* » est certes le futur des technologies du web. Il renforce certains vieux concepts de construction d'architectures internet hautement évolutifs, et introduit de nouveaux concepts qui changent complètement la façon dont les applications sont développées et déployées. Au cours des dernières années, certaines entreprises technologiques ont adopté la stratégie du *Cloud Computing*. Cette adoption a eu lieu lorsque ces entreprises ont prédit que le *Cloud Computing* sera les solutions des plusieurs problèmes Web tels que la disponibilité. Toutefois, les organisations pensent qu'il est presque impossible de lancer l'idée du « Cloud » sans adopter les concepts et les normes antérieures comme celle du paradigme orienté service (Service-Oriented Paradigm). En raison de cette dépendance, les problèmes de l'approche orientée service et services web sont transférés au Cloud. En effet, la disponibilité du Cloud actuel s'avère trop chère à cause de la reproduction de services, certains services Cloud sont confrontés à des problèmes de performances, une majorité des services Cloud est faible en matière de sécurité, et ces services sont découverts d'une façon aléatoire, il est difficile de choisir le meilleur d'entre eux ainsi qu'ils sont composés d'un groupe de services web dans un monde de services. Egalement, il est impossible de valider les services Cloud en particulier, avant le temps d'exécution. Finalement, selon les normes du W3C, les services Cloud ne sont pas encore internationalisés. En effet, le web comme prévu, est un modèle de service intelligent bien qu'il manque d'intelligence et d'autonomie. Ainsi, l'adoption d'un modèle axé sur le service n'était pas une décision idéale. Afin de minimiser les conséquences des problèmes du Cloud et réaliser plus de profits, certaines entreprises de Cloud développent leurs propres plateformes de Cloud Computing. Actuellement, les fournisseurs du Cloud font face à un grand problème qui peut se résumer par la « Bataille de la plateforme Cloud ». Le budget de cette bataille coûte des milliards de dollars en l'absence d'un accord pour accéder à une plateforme Cloud standard. Pourquoi une collaboration intelligente n'est pas mise en place entre les nuages distribués pour obtenir de meilleurs résultats ? L'approche appropriée est de restructurer le modèle de cloud afin de couvrir ses problèmes. Des techniques intelligentes multiples peuvent être utilisées pour développer des systèmes Cloud intelligents avancés. Parmi les exemples classiques de systèmes intelligents distribués se trouvent : le corps humain, les colonies d'insectes sociaux, les troupeaux de vertébrés, les systèmes multi-agents, les systèmes de transport, les systèmes multi-robots, et les réseaux de capteurs sans fils. Toutefois, le système intelligent qui pourrait être imité est le système du corps humain dans lequel vivent des milliards de cellules du corps et travaillent ensemble pour atteindre des résultats précis. En s'inspirant de la stratégie Bio-Informatique qui bénéficie de technologies pour résoudre des faits biologiques (comme les gènes). Cette thèse propose une nouvelle stratégie Bio-Cloud qui imite des faits biologiques (comme le cerveau et les gènes) pour résoudre les problèmes du Cloud Computing mentionnés ci-haut. Ainsi, en me basant sur la stratégie Bio-Cloud, j'ai développé au cours de cette thèse la théorie « SmartCells » conçue comme une proposition (approche) cherchant à résoudre les problèmes du Cloud Computing. Cette approche couvre : 1) les problèmes hérités du paradigme services (comme les questions de réutilisation de services, les questions de sécurité, etc.); 2) le problème d'insuffisance d'intelligence dans les systèmes du Cloud Computing. SmartCells se base sur la collaboration entre les composants intelligents (les Cellules) qui profitent de la variété des composants des services web déjà construits afin de produire un système de Cloud intelligent.

RESUME

Cloud computing is the future of web technologies and the goal for all web companies as well. It reinforces some old concepts of building highly scalable Internet architectures and introduces some new concepts that entirely change the way applications are built and deployed. In the recent years, some technology companies adopted the cloud computing strategy. This adoption took place when these companies have predicted that cloud computing will be the solutions of Web problems such as availability. However, organizations find it almost impossible to launch the cloud idea without adopting previous approaches like that of Service-Oriented approach. As a result of this dependency, web service problems are transferred into the cloud. Indeed, the current cloud's availability is too expensive due to service replication, some cloud services face performance problem, a majority of these services is weak regarding security, and cloud services are randomly discovered while it is difficult to precisely select the best ones in addition to being spontaneously fabricated in an ocean of services. Moreover, it is impossible to validate cloud services especially before runtime. Finally, according to the W3C standards, cloud services are not yet internationalized. Indeed, the predicted web is a smart service model while it lacks intelligence and autonomy. This is why the adoption of service-oriented model was not an ideal decision. In order to minimize the consequences of cloud problems and achieve more benefits, each cloud company builds its own cloud platform. Currently, cloud vendors are facing a big problem that can be summarized by the "Cloud Platform Battle". The budget of this battle will cost about billions of dollars due to the absence of an agreement to reach a standard cloud platform. Why intelligent collaboration is not applied between distributed clouds to achieve better Cloud Computing results? The appropriate approach is to restructure the cloud model basis to recover its issues. Multiple intelligent techniques may be used to develop advanced intelligent Cloud systems. Classical examples of distributed intelligent systems include: human body, social insect colonies, flocks of vertebrates, multi-agent systems, transportation systems, multi-robot systems, and wireless sensor networks. However, the intelligent system that could be imitated is the human body system, in which billions of body cells work together to achieve accurate results. Inspired by Bio-Informatics strategy that benefits from technologies to solve biological facts (like our genes), this thesis research proposes a novel Bio-Cloud strategy which imitates biological facts (like brain and genes) in solving the Cloud Computing issues. Based on Bio-Cloud strategy, I have developed through this thesis project the "SmartCells" framework as a smart solution for Cloud problems. SmartCells framework covers: 1) Cloud problems which are inherited from the service paradigm (like issues of service reusability, security, etc.); 2) The intelligence insufficiency problem in Cloud Computing systems. SmartCells depends on collaborations between smart components (Cells) that take advantage of the variety of already built web service components to produce an intelligent Cloud system.

ACKNOWLEDGEMENTS

I would never have been able to finish my dissertation without the guidance of my advisors, Dr. Hamid Mcheick and Dr. Mohamed Dbouk, and the support of my family.

I would like to acknowledge that this PhD thesis is supported by the “Département d'informatique et de mathématique (DIM)” at the University of Quebec in Chicoutimi, the “Ecole Doctorale des Sciences et de Technologie (EDST)” at the Lebanese University and the “AZM” association.

I would like to express my deepest gratitude to my first advisor, Dr. Mcheick, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. I would like to thank my second advisor, Dr. Dbouk, who let me experience the research of software engineering. I would also like to thank Dr. Mohammad Khalil who represented AL-AZM association in supporting me by money during my PhD study. Special thanks go to Professors, who participated in my final defense committee.

TABLE OF CONTENTS

RÉSUMÉ	2
RESUME	3
ACKNOWLEDGEMENTS	4
TABLE OF CONTENTS	5
LIST OF TABLES	7
LIST OF FIGURES	8
<i>INTRODUCTION</i>	11
Chapter 1 COMPUTING METHODS AND CLOUD PROBLEM STATEMENTS	23
1.1 COMPUTING METHODS OVERVIEW	23
1.2 CLOUD COMPUTING VS. SERVICE-ORIENTED COMPUTING	30
1.3 PROBLEM SYNOPSIS	31
1.3.1 CLOUD SERVICE PROBLEMS OVERVIEW	32
1.3.2 CLOUD COMPUTING PLATFORMS CHALLENGES	40
1.3.3 DEFICIENCY OF CLOUD INTELLIGENCE	42
1.4 CONCLUSION	44
Chapter 2 RELATED WORKS	46
2.1 INTELLIGENT DISTRIBUTED COMPUTING	46
2.2 CLOUD SERVICES ISSUES	49
2.3 ENCOURAGING PROJECTS	55
2.4 CONCLUSION	56
Chapter 3 SMARTCELLS; A CELL-ORIENTED SMART CLOUD APPROACH	58
3.1 TOWARDS SMART CLOUDS COMPUTING	58
3.2 BIO-CELL IMITATED KEY-FEATURES	61
3.3 BIO-CLOUD VS. BIO-CELL: MAPPING MODALITIES	69
3.4 SMARTCELLS APPROACH	79
3.4.1 CELL BASIS AND FOUNDATIONS	80
3.4.2 SOFTWARE ARCHITECTURE AND REQUIRED INFRASTRUCTURES	89
3.5 CONCLUSION	94
Chapter 4 CELL OPERATIONAL MODE	95
4.1 INTRODUCTION	95
4.2 STRUCTURE OF SMARTCELLS COMPONENTS	97
4.2.1 COMMANDER CELL STRUCTURE	97
4.2.2 CLOUD BRAIN STRUCTURE	100
4.2.3 CELL SOURCE	108
4.3 DEFINITIONS AND NOTATIONS	109
4.4 COMPONENTS OF EXECUTIVE CELL	113
4.4.1 DECISION SYSTEM (DS)	114
4.4.2 DEFENSE SYSTEM (DFS)	115
4.4.3 GENE STORE SYSTEM (GSS)	116
4.4.4 PROCESS ANALYSER SYSTEM (PAS)	117
4.4.5 PROCESS VALIDATION SYSTEM (PVS)	128
4.4.6 TRAITS MAINTENANCE SYSTEM (TMS)	138
4.4.7 OUTPUT BUILDER SYSTEM (OBS)	155
4.5 CONCLUSION	157

Chapter 5	VALIDATION; A CASE STUDY	158
5.1	CASE STUDY: AN IDENTITY DETECTOR CELL OUTLINES.....	158
5.2	IMPLEMENTATION: TOOLS AND PLATFORMS	160
5.3	SERVICE-ORIENTED SIMULATION	162
5.4	SMART CELLS SIMULATION	170
5.5	OBSERVATION AND CRITICISM.....	200
	CONCLUSIONS AND PERSPECTIVES	201
	REFERENCES	207
	APPENDIXES.....	227
	INDEX.....	252

LIST OF TABLES

TABLE 3.1 COMPARISON BETWEEN CLOUD AND BODY STRATEGIES	72
TABLE 4.1 REPRESENTATION OF LEVEL 0 OF THE CUBE	126
TABLE 4.2 REPRESENTATION OF LEVEL 1 OF THE CUBE	126
TABLE 4.3 REPRESENTATION OF LEVEL 2 OF THE CUBE	127
TABLE 4.4 THE OUTPUT RESULT OF PARSING THE LOAN BPEL CODE	136

LIST OF FIGURES

Figure 1.1 Cloud computing source SOA and web services [source: Service Architecture, Barry, 2013].....	31
Figure 1.2 Security, Availability & Performance Lead Cloud Challenges (Source: IDC Enterprise Panel)	33
Figure 1.3 AWS problem reports number as a function of time between 24 and 25 October 2013	34
Figure 1.4 Availability variation of Amazon, Google & Indonesia Clouds (Source: CloudSleuth).....	36
Figure 1.5 Analysis of cloud performance based on service response tiCCdn (source: CloudSleuth).....	37
Figure 3.1 The human brain is a center of management of body organs	62
Figure 3.2 The brain in the SmartCells architecture.....	63
Figure 3.3 Cells renewal by the brain.....	65
Figure 3.4 Normal vs. cancer cell growth.....	66
Figure 3.5 Genes contain the business process of a cell (source: U.S. energy Department)	68
Figure 3.6 The development of distributed computing methods.....	69
Figure 3.7 The development of body cells	70
Figure 3.8 Service process is the last stone in building cloud service	73
Figure 3.9 A gene is a part of DNA.....	74
Figure 3.10 Example of a cloud service input and output	75
Figure 3.11 Human cell has inputs and outputs.....	75
Figure 3.12 The job application service	76
Figure 3.13 An example of DNA genetic material of a cell	76
Figure 3.14 SmartCells Architecture	90
Figure 3.15 Strategy of Cell Computing.....	83
Figure 4.1 Structure of Client Cell	97
Figure 4.2 Structure of Cell Provider (Cloud Brain).....	100
Figure 4.3 Structure of Cell Source.....	108
Figure 4.4 Components of Executive Cell.....	114
Figure 4.5 Merging multiple network graphs in one Multi-Network graph	119
Figure 4.6 Mapping a multi-network graph into ER diagram	122

Figure 4.7 A structure of a cube with three faces and “k” levels of analysis measures	125
Figure 4.8 Phases to build a compiler	130
Figure 4.9 Example of composite services	131
Figure 4.10 Infinite loop of web service.....	132
Figure 4.11 Loan BPEL example	136
Figure 4.12 The directed graph of the BPEL example	137
Figure 4.13 Main Layers of cloud infrastructure	141
Figure 4.14 <i>QoSDW</i> model components	142
Figure 4.15 Transforming SteamBoat service business process into a tree of sub-services	145
Figure 4.16 Representation of the initial report by <i>QoSDWAnalyser</i>	151
Figure 4.17 The proposed <i>QoSDW</i> schema	154
Figure 5.1: The process of sending a verified anonymous email	159
Figure 5.2 : ProfileInMail service process description	163
Figure 5.3: VerifyMailer service process description.....	165
Figure 5.4: SenderDetector service process description.....	166
Figure 5.5: The variation of ProfileInMail latency as a function of time	167
Figure 5.6: The variation of VerifyMailer latency as a function of time.....	167
Figure 5.7: The variation of SenderDetector latency as a function of time	168
Figure 5.8: Invoking form of VerifyMailer	169
Figure 5.9: Result of sending a verified anonymous email	170
Figure 5.10: SmartCells website.....	172
Figure 5.11: SmartCells selection method.....	173
Figure 5.12: Commander’s page of IdentityMail Cell.....	174
Figure 5.13: Result of Commanding IdentityMail Cell	175
Figure 5.14: Commander’s page of GetIP Cell	176
Figure 5.15: Result of Commanding GetIP Cell	176
Figure 5.16: Commander’s page of GetGeoProfile Cell	177
Figure 5.17: Result of Commanding GetGeoProfile Cell.....	177

Figure 5.18: Commander’s page of SendMail Cell	178
Figure 5.19: Result of Commanding SendMail Cell	179
Figure 5.20: Analysis of SenderDetector service process based on quality of subservices	180
Figure 5.21: Analysis of VerifyMailer service process based on quality of subservices	181
Figure 5.22: Analysis of ProfileInMail service process based on quality of subservices	182
Figure 5.23: Gene map of the GetIP Cell	184
Figure 5.24: Gene map of the GetGeoProfile Cell	185
Figure 5.25: Gene map of the SendMail Cell	186
Figure 5.26: Gene map of the IdentityMail Cell	187
Figure 5.27: initial SQL database by gene map	188
Figure 5.28: Trace the map of IdentityMail Gene	189
Figure 5.29: Trace the possible compositions of IdentityMail Cell	190
Figure 5.30: Validate of possible Gene composition	192
Figure 5.31: Improving the performance based on distance criteria	193
Figure 5.32 : Flexible Gene map analysis	195
Figure 5.33: Minimal composition of Gene map	196
Figure 5.34: Gene analysis based on Degree Centrality measure	197
Figure 5.35: Gene analysis based on Closeness Centrality measure	198
Figure 5.36: Gene analysis based on Betweenness Centrality measure	199

INTRODUCTION

Today's revolution of classical Cloud Computing theory along with the competition between Cloud vendors has pushed scientist of technology to think of an intelligent Cloud Computing strategy. This thesis project aims to contribute to the advancement of theoretical foundations, principles, and technologies of intelligent Cloud systems, as well as to tackle more pragmatic issues such as their practical application by developing real and smart Cloud system and solving real-Cloud problems.

CLOUD COMPUTING OVERVIEW

The advancements in information technology require a new computing methodology that supports delivery of smart computing services on minimal charges without installing them at local sites. Cloud computing offers a part of that methodology, in which services are delivered over the internet in an on-demand elastic way for which the charges are paid at release time of resources.

“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. This Cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models” (NIST definition, 2010).

Cloud computing is now going through the development phase of the Cloud life cycle. As a result of the absence of a standard Cloud platform, each Cloud company tries to reach a steady Cloud state, but suddenly fails. For example, in 2009, NASA was the first to enter the Cloud battle by its Cloud platform “Nebula”. But few years later, precisely in 2012, NASA has shut down “Nebula”. This shut down was based on a five month test for its Cloud service quality in comparison to other Cloud vendors. The short history of Cloud shows that random Cloud platforms that compete on quality of Cloud vendors will fail.

There are a lot of good features in Cloud computing platforms, but not all researchers and developers appreciate them since they are still in the development phase of their life cycle; consequently, they are considered not ideal enough. The anticipated web is a smart and semantic web; this is why it is recommended to insert new essential properties to the Cloud paradigm such as: autonomy and intelligence. Clouds collaboration is a very good method, why not to apply this collaboration to achieve better Cloud Computing results?

PROBLEM POSITION

Cloud computing is an extremely new computing paradigm to share processing, storage, networking and applications. Generally, Cloud is a miscellaneous technological paradigm that comprises an extension of many existing technologies such as: parallel and distributed computing, Service-Oriented model, virtualization, networking, etc.

“Service-oriented computing and Cloud computing have a reciprocal relationship - one provides the computing of services, and the other provides the services of computing” (Wei and Blake, 2010). Furthermore, by the support of service-oriented model, Cloud computing has become a more popular paradigm. Even though the service computing paradigm constituted a revolution in World Wide Web, it is still viewed as a non-autonomous pattern. However, most computing procedures are directed towards intelligence and towards a decrease in processing time and cost.

“By 2017, 10 percent of computers will be learning rather than processing...”
Gartner, *Top predictions for IT organizations and IT users for 2014 and beyond*.

In addition to intelligence insufficiency, service paradigm problems are transferred into the Cloud. Consequently, several problems facing the Cloud’s progress can be summarized as follows:

- Intelligence Insufficiency: Current service based control measures do not sufficiently tackle Cloud computing’s third-party data storage and analysis needs (Chow *et al.*, 2009). Cloud Intelligence is required to provide a certain consistent flow of Cloud business.
- Expensive availability: The difficulty of using the replication procedure in Clouds is that it is very expensive to achieve enough stability on a worldwide range (Li *et al.*, 2012).
- Performance issues: Before moving applications to the Cloud environment, organizations should test whether the Cloud infrastructure they are using can

support these applications. Inappropriately, many cloud organizations do not have technology capabilities to perform good service testing. Thus, it is challengeable to prevent performance issues from occurring before end-users are impacted (ManageEngine, 2014). Currently, there is an augmented need to use virtualized systems in enterprise Cloud computing in order to more powerfully consume resources. However, it is difficult to precisely model virtualized systems in order to analyze performance issues (Altamash and Niranjana, 2013).

- Security: The rapid growth of Cloud computing also increases severe security apprehensions. Lack of security is the only hurdle in wide adoption of Cloud computing. “Cloud computing is surrounded by many security issues like securing data, and examining the Cloud utilization by the Cloud computing vendors” (Shaikh and Haidar, 2011).
- Reusability: Cloud service composition, which includes several tasks such as discovery, compatibility checking, selection, and deployment, is a complex process. “Service composition in Cloud raises even new challenges caused by diversity of users with different expertise requiring their applications to be deployed across difference geographical locations with distinct legal constraints” (Dastjerdi and Buyya, 2014). Accordingly, the Cloud service composition problem is modeled as a major business development problem.
- Process Validation: Testing a Cloud refers to validation of applications to conform the expectations of the Cloud computing business model (Mehrotra, 2011). As the Cloud system is directed towards autonomic distributed applications, the Cloud service validation issues are attracting great attention. The

standardization of composition languages (as BPEL) led researchers to investigate validation techniques mainly focusing on the sequence of events in the composition (Cesare *et al.*, 2008). In addition, Cloud services are impossible to be validated specially before runtime step since there is no standard service composition compiler (Karawash *et al.*, 2013).

- **Process Analysis:** The main reason behind the need for handling Cloud's big data is to be able to gain value from data analysis. Analytic method requirements focus on developing techniques that can be able to process large and growing data sets (Ahuja and Moore, 2013). Simplification of the analysis process of big data towards an automated approach is a major goal behind big data (Bryant, 2008).

Finally, according to the W3C standards, Cloud services are not yet internationalized; in other terms, they are not compatible to all worlds' languages. Furthermore, with the extensive deployment of Cloud computing, management, interoperability and integration of these systems have become challenging problems. With this in mind, investigators have researched and developed important technologies to cope with these problems.

CONTRIBUTION

One of the results of the continuous evolution of distributed computing in the last decade is the Cloud computing paradigm, which offers an evolution in the processes of architecting, design and implementation, as well as in deploying e-business and integration approaches. In order to reach a quick launch of Cloud computing technology, it was recommended to depend on the strategies and standards of the previous technologies such

as the service oriented paradigm, distributed system, virtualization, clustering, grid computing, and many more.

This thesis project introduces a new style of intelligent distributed computing, known as SmartCells approach. SmartCells approach aims to enhance the Cloud computing model through combining the advantages of other models such as service oriented computing and intelligent computing model. Consequently, SmartCells approach tries to reach the following resolutions:

- Decrease the negative effects of random distribution of Cloud services by classifying them, according to their functions, under major types of components called Cells (Karawash *et al.*, 2015).
- Monitor the changes in the process map of services and provide instantaneous and automatic composition of a Cloud Cell process which serves Cloud clients (Karawash *et al.*, 2015).
- Validate the composition of new formulated component via a distributed compiler of Cell process (Karawash *et al.*, 2013).
- Improve the performance of Cloud services through applying pruning strategy of Cell process after deep levels of analysis and optimization (Karawash *et al.*, 2014a).
- Decrease the replication of Clouds that increases availability costs, based on the spare process methodology (Karawash *et al.*, 2015).

- Develop an accurate Cloud Cell decision system based on Cell quality, different from the traditional service selection procedures (Karawash *et al.*, 2014b).
- Add decision and management center characterized by intelligence and autonomy to follow life cycle of combinations of processes used by Cloud (Karawash *et al.*, 2015).
- Increase the Cloud security by isolating Cloud customer's side from Cloud providers.

Indeed, the service-oriented computing procedures are complex and currently Clouds inherit this complexity to support their clients. In general, Cloud Architects search for optimized and simpler Cloud solutions. "Achieving Cloud perfection and Competitiveness requires from companies to frequently modify their computing systems through adding new features or deleting old ones in a relatively short period of time" (Darekar, 2013).

RESEARCH METHODOLOGY

The research project presented in this thesis was carried out through a research methodology divided into eight key steps.

In the first step of the research project, Cloud computing model fundamentals and basis were studied deeply. At this stage, it was first aimed to gain a general knowledge of the areas of research, by reviewing the key books (Rountree et Castrillo, 2013; Mahomood et Hill, 2011; Buyya *et al.*, 2011; Wang, 2012) and works (Hayes, 2008; Is, 2010; Azab, 2009) in the field. Later, and to get more advanced in the Cloud computing world, I tried

to accumulate further details through reading more books (Barry et Dick, 2013; Bento et Aggarwal, 2013).

As a second step, we conducted a much more targeted survey by reviewing more works related to the Cloud computing problems such as: expensive availability (Sun *et al.*, 2012; Zaho, 2012; Sun, 2012), Security (Onwubiko, 2010; Karn, 2010), performance issues (Khanghahi et Ravanmehr, 2013; Miet *al.*, 2011), reusability (Zeng, 2009; Ylianttila, 2012; Zeng, 2009), process composition validation (Tsai, 2011; Riungu, 2010; Nguyen, 2011), and big data analysis (Ning, 2012; Chuob, 2011; Hong-Linh, 2011; Sarnovsky *et al.*, 2012).

In the third step, we developed a new model for service process validation stimulated from the advantages and weakness points discovered in the literature review. As a summary, we built a distributed compiler of web processes that notifies deadlocks in the design phase of an application and decreases the number of fixes of Cloud services Runtime errors (Karawash *et al.*, 2013).

As a fourth step, we developed a new model for solving the big data analysis resulting from the massive Cloud network. Analyzing Cloud networks is helpful for organizations that profit from how network nodes (e.g. web users) interact and communicate with each other. Many attempts have been made to develop an analytical approach that works on multiple big data networks simultaneously. Our model proposes to map web multi-network graphs in a data model. The result is a multidimensional database that offers numerous analytical measures of several networks concurrently. It supports

real-time analysis and online analytical processing (OLAP) operations, including data mining and business intelligence analysis (Karawash *et al.*, 2014a).

In the fifth step, we participated in improving the Cloud service selection approach by introducing the quality of sub-service data warehouse model. This model proposes to study the properties of sub-services share in composing a complex service. As a simulation, a data warehouse is built using Microsoft SQL server 2008 and OLAP operations were applied to reach required results (Karawash *et al.*, 2014b).

As an overall research in the seventh step, we propose SmartCells as a novel theory that offers smart approaches for Cloud Computing problems (Karawash *et al.*, 2015). In contrast to the Bio-Informatics strategy that benefits from web technologies to solve and discover biological facts (like Genes), we proposed to imitate the functions of these biological facts in solving the Cloud computing issues.

Finally, the last stage of this project consisted in simulating our model using the some software tools (such as Eclipse, Xampp, etc.), database tools (like Microsoft SQL server 2008/2012) and Apache server. The infrastructure used consists of local machines (such as Lenovo PCs and university server) and Cloud services (like Google Cloud).

Through all the project's steps, contributions to the field were made, which took the forms of scientific book chapters in important books published by Springer International publisher (Azar and Sundarapandian, 2015; Mahmood, 2014; Bessis and Ciprian, 2014; Lee, 2013).

ORGANIZATION OF THE THESIS

The thesis document is composed of five chapters. However, cross-references throughout the document make a reading thread inviting to follow the sequence of chapters from the introduction to the conclusions. The document is organized as follows:

Chapter 1 - Computing Methods and Cloud Problems Statement:

The web history is full of several computing models which were developed to satisfy the client needs. Web companies build new computing approaches in order to keep on better services and replace the weak points of the old computing models. Cloud computing has received a lot of popularity in the last few years and market observers believe it to be the future. Experts declare that Cloud computing is at its nascent stage and providers will have to address issues related security, availability, performance and more to expand in the future. This chapter discusses the growth of distributed computing approaches till reaching Cloud computing then it shows some Cloud computing problems and their effect on the web.

Chapter 2 - Related Works:

Cloud Computing presents several technology and engineering challenges, many of which relate to the traditional requirements of distributed systems. The recent distributed systems models must be restructured in the context of virtualized environments. This chapter discusses the previous approaches that proposed to the Cloud service problem in general.

Chapter 3 - SmartCells: A Cell-Oriented Smart Cloud Computing:

Cloud computing systems are important in the era of recently established and future tasks in computer science. As computing jobs become gradually more directed towards intelligence and autonomy, thus intelligent computations techniques will be the key for all future applications. The predicted cloud and web is a smart and semantic web while the service-based model lack of intelligence and autonomy. This chapter discusses the components and mechanisms of proposed SmartCells theory, which applies new computing concepts to reach smart Cloud.

Chapter 4 - Cell Operational Mode:

Nowadays, research centres require the development of architectures of intelligent and collaborated systems; these systems must be capable of solving computing problems by themselves with less processing time and reduced costs. Building an intelligent style of distribution that controls the whole distributed system requires communications that must be based on a completely consistent system. One of the known systems to be adopted in building an intelligent distributed computing structure is the human body system, specifically the body's cells. As an artificial and virtual simulation of the high degree of intelligence that controls the body's cells, this chapter shows the Cell-Oriented computing paradigm, as a new approach to achieving the desired intelligent distributed computing system. The details about Cell paradigm were presented in four Springer chapters as follows: (Karawash *et al.*, 2013), (Karawash *et al.*, 2014a), (Karawash *et al.*, 2014b) and (Karawash *et al.*, 2015).

Chapter 5 - Validation; a case study:

This Chapter discusses a case study and simulation about the proposed SmartCells approach. This chapter shows the importance of SmartCells approach by discussing the Identity Cell scenario as a technique that contributes in solving the anonymous email problem.

Chapter 1

COMPUTING METHODS AND CLOUD

PROBLEM STATEMENTS

The web history is full of computing models which were developed to satisfy the client needs. Web companies build new computing approaches in order to keep on better services and replace the weak points of the old computing models. Recently, Cloud computing has reached a degree of reputation and web market experts believe that it will be the future. Cloud computing still not yet exceeds its development stage and providers will have to address issues related security, availability, performance and more to expand in the future. This chapter discusses the main computing approaches and the recent problems that are inherited by the today's Cloud computing approach.

1.1 COMPUTING METHODS OVERVIEW

The initial approaches of computing have started with closed, monolithic mainframe systems. Monolithic applications were the result of the evolution of single-processor systems in which the processing and management of data is totally centralized. Gradually, with the time, new types of computing system were developed to reach today's computing

system; the Cloud computing. This section shows a short survey about how the computing paradigms developed till reaching the current Cloud.

Procedural computing: It involves the process executed on a single machine and handles the data through direct access operations. A procedure program consists of one or more procedures or functions. Every program has a main function which is its starting point. This type of computing has many possible dependencies between program algorithms and does not approve their alteration easily.

Client-server computing: It is a term used to describe a computing model for the development of computerized systems. *Client-server* computing is the logical porch of modular programming with the fundamental assumption that separation of a huge program into its ingredient parts ("modules") can create the possibility for further adjustment, easier development and better maintainability. This model is based on the distribution of functions between two types of independent and autonomous processors: servers and clients. A client is any process that requires specific services from server processes. A server is a process that provides solutions for clients. Client and server processes can be located in in the same machine or in different networks. A Client-Server system is one in which the server executes some kind of service that is consumed by many clients. The basic Client-Server architecture has two tiers (Client and Server). But the necessity to support clean separation of data and application logic layer from the presentation layer caused to replace client-server technologies by three tiers, then N tiers.

Object Oriented Computing: It supports the development of software with encapsulating both data and behavior into abstract data types, called classes. Instances of classes are formed into small modules, called objects. An object oriented programming may be viewed as a group of interacting objects in which a program is seen as a list of tasks (subroutines) to perform. Any changes in data representation only affect the immediate object that encapsulates the data. Classes can live everlastingly; however, objects have a limited lifetime. The main characteristics of Object Oriented development are given as follows:

- Encapsulation: it refers to mechanisms that allow each object to have its own data and methods. The idea of encapsulating data together with methods existed before object-oriented languages were developed.
- Information Hiding: is a great programming technique because it reduces complexity.
- Associations and Inheritance: Inheritance is a kind of association in which a subclass extends the definition of its superclass. Inheritance is a mechanism of reusability.
- Polymorphism: Object oriented Computing allows different implementations of the same message through two or more separate classes.

The benefit of object orientation is that the software structures more easily map to real world entities. Today, object oriented technology is widely used and it is a dominant paradigm for developing application software.

Component Oriented Computing: It is a software engineering method that emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. Component-oriented programming is rapidly becoming a mainstream programming paradigm, offering higher reusability and better modular structure with greater elasticity than object-oriented approach. A software component is defined as a entity of composition with particular interfaces and precise context dependencies. “A software component can be deployed independently and is subject to composition by third parties” (Nawaz *et al.*, 2008). Components overlap the properties of object orientation, such as encapsulation and polymorphism, except it reduces the property of inheritance. In component thinking, inheritance is tightly coupled and unsuitable for most forms of packaging and reuse. Instead, components reuse the functionality by invoking other objects and components rather than inheriting from them.

Resource oriented computing (ROC): It is a simple fundamental model for describing, designing, and implementing software and software systems. “ROC is based upon the concept of resource; each resource is a directly accessible distributed component that is handled through a standard, common interface making possible resource handling” (Fielding, 2000). RESTful platforms (Richardson, 2007) based on

REST development technology enable the creation of ROC. “The main ROC concepts are the following:

- Resource: anything that is significant enough to be referenced as a thing itself.
- Resource name: unique identification of the resource.
- Resource representation: useful information about the current state of a resource.
- Resource links: link to another representation of the same or another resource.
- Resource interface: uniform interface for accessing the resource and manipulating its state”.

For detailed and exhaustive definition of the ROC’s main concepts, I invite the readers to refer to (Richardson, 2007). The resource interface semantics are based on the one of HTTP operations.

Service Oriented Computing (SOC): Today’s Web collects a group of computing methods. The main computing paradigm which survived for more than 10 years is the service oriented paradigm. “The Service-Oriented Computing (SOC) paradigm refers to the set of concepts, principles, and methods that represent computing in Service-Oriented Architecture (SOA) in which software applications are constructed based on independent component services with standard interfaces” (Tsai and Chen, 2006). Realizing the SOC promise requires the design of SOA that enable the development of simpler and cheaper distributed applications. SOA contains six elements, in its conceptual model, described as follows (McGovern, 2003): Service consumer, service provider, service registry, service contract, service proxy and service lease.

- **Service Consumer:** The consumer can be an application, another service, or some other type of software module that needs the service.
- **Service Provider:** It is the network target element that receives and performs the requests from consumers. It delivers the definite service description and the implementation of the service. The service provider is the side who is responsible of satisfying the service consumer's requirements.
- **Service Registry:** It is a meta-data store which can be accessible through the network and contains available service descriptions. Its main function is to store and publish service descriptions from providers and supply these descriptions to involved service consumers.

“A Web service is an abstract notion that must be implemented by a concrete agent” (W3C, 2004). The Web service has three parts: SOAP, WSDL and UDDI, which are summarized briefly as:

- SOAP – it is a network communication protocol used to exchange information over HTTP and over the internet. “The SOAP message body is designed to carry textual information. This is referred to as payload” (Panda, 2005).
- WSDL - The Web Services Description Language is an extension of the Extensible Markup Language (XML).
- UDDI – “Universal Description, Discovery and Integration are a specification for the XML-based registries to list and find services on the World Wide Web” (UDDI, 2011). lead

Cloud computing: The huge amounts of data guided web providers to employ larger web infrastructures. By distributing and replicating data across servers on demand,

resource utilization has been significantly improved. The term “Cloud” was firstly used by Amazon and associated with elastic infrastructures. Cloud computing is to employ computing resources that are delivered as a service over a network. “The Cloud computing is the future. It provides almost infinitely flexible and scalable external computing and processing services that not only offer significant cost benefits, but also provide the aptitude to connect with customers, partners and suppliers like never before” (Capgemini and HP, 2008).

“A Cloud is an elastic execution environment of resources involving multiple stakeholders and providing a metered service at multiple granularities for a specified level of quality of service” (Schubert, 2010). Cloud computing relies on sharing resources to achieve coherence and economies of scale similar to a utility over a network. Cloud providers typically center on one type of Cloud functionality provisioning: Infrastructure, Platform or Software. Cloud model offers multiple types of computing at the same time:

- *Infrastructure as a Service (IaaS)* offers resources as services to the cloud user. For example, *IaaS* providers may provide computers, servers or virtual machines as a service.
- *Platform as a Service (PaaS)* offers platforms based on computational resources in which applications and services can be used. For example, a cloud platform may include operating system, programming language execution environment, database, and Web server.

- *Software as a Service (SaaS)* offers simply an access to as a Service or Application on the cloud. For example, cloud users may use some application form cloud without installing them on their machines such as: Microsoft office application and many others.

“Cloud services may be hosted and employed in different ways based on the business model of the cloud provider. Some Clouds evolve from private solutions (private Clouds) to manage the local infrastructure and the amount of requests” (Schubert *et al.*, 2010). The other Cloud capabilities make use of these features for public purposes. Also Cloud providers find benefits from combining the public and private feature and emerge hybrid solutions.

1.2 CLOUD COMPUTING VS. SERVICE-ORIENTED COMPUTING

The very real risk for today’s organization is that while business and technical drivers will increase their need for web and Cloud services. “One of the latest challenges is how to work with service-oriented computing (SOC) in a Cloud computing environment. Traditional software Lifecycle models haven’t explicitly addressed this requirement for continuous integration of new capabilities” (Blake, 2007). “SOC aims to use services as basic blocks to construct rapid, low-cost, secure and reliable applications” (Papazoglou *et al.*, 2008). A web service is different from a traditional software artifact in that it is autonomous, self-described, reusable, and highly portable. Cloud Computing is the result of evolution and adoption of existing technologies and paradigms.

The below Venn diagram, in Figure 1.1, shows the relations among Web Services, SOA, and Cloud Computing. “Web Services encapsulate Cloud Computing in this diagram because Cloud Computing uses Web Services for connections.

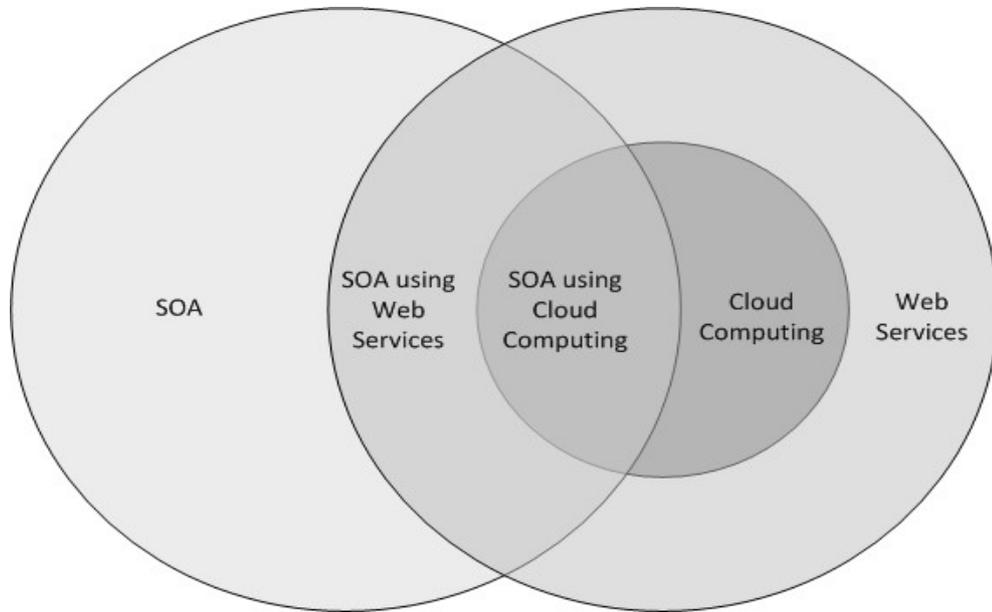


Figure 1.1 Cloud computing source SOA and web services [source: Service Architecture, Barry, 2013]

It is possible, however, to use Web Services in situations other than Cloud Computing. Such use of Web Services may be part of a service-oriented architecture, but it may not. Web Services could be simply be a connection. Therefore, it is possible to have a service-oriented architecture and not use Web Services for connections” (Barry, 2013).

1.3 PROBLEM SYNOPSIS

In the recent years, some of the technology companies deviated towards Cloud computing strategy. However, these organizations find it almost impossible to launch the

Cloud idea without adopting the old concepts and standards of service oriented paradigm despite of their problems. Consequently, the current Cloud faces many problems such as: availability, performance, security, composition, validation and compatibility. In addition to these problems, Cloud vendors are taking part in the Cloud platform battle where every Cloud company competes by its own Cloud platform. Some Cloud vendors (like: NASA) paid millions of dollars for building Cloud platforms, but suddenly the platform failed. The cloud paradigm expands sharply whenever no common Cloud platform structure exists among vendors. Indeed, Cloud Computing is often marketed as an efficient and cheap solution that will replace the client-server paradigm. It offers many strong points such as infrastructure flexibility, faster deployment of applications and data, cost control, adaptation of Cloud resources to real needs, improved productivity, etc. Most of Cloud companies are still concerned about Cloud issues such as are reliability, availability of services and data, security, complexity, costs, regulations and legal issues, performance, migration, reversion, the lack of standards, limited customization, and issues of privacy. Cloud problems are described in more details in the next three subsections.

1.3.1 CLOUD SERVICE PROBLEMS OVERVIEW

Some of Cloud service problems could be summarized by these questions: how to increase availability with less server data replication? How to increase Cloud performance? How to increase Cloud security and ensure privacy? How can trade-off decisions are accounted for during application design, how can they be modified during run-time? How Cloud services composition is dynamically achieved based on available

processes. As shown in Figure 1.2, security, availability and Performance lead Cloud challenges.

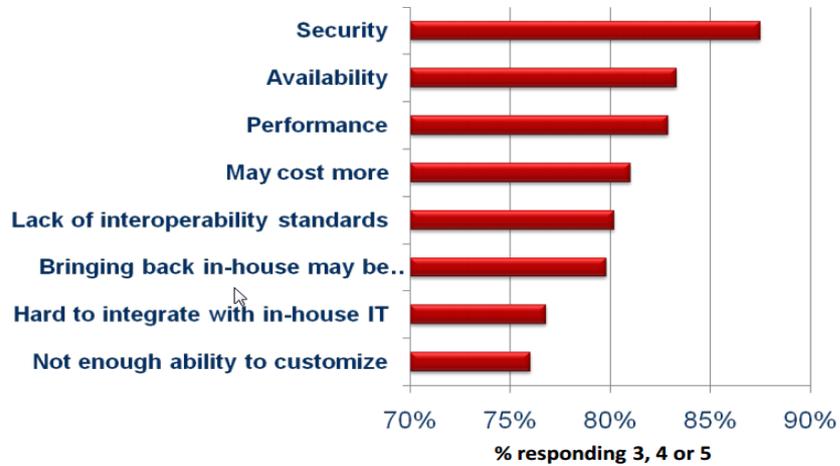


Figure 1.2 Security, Availability & Performance Lead Cloud Challenges (Source: IDC Enterprise Panel)

As an example about Cloud service problems, figure 1.3 below shows the number of problems of most integrated public Cloud providers, which is used by Ericsson; the Amazon Web Service (AWS). The figure 1.3 is taken from down-detector¹ and it shows that 10 reports, about service problems, are counted about AWS problems during 24 hours between 24 and 25 October 2013.

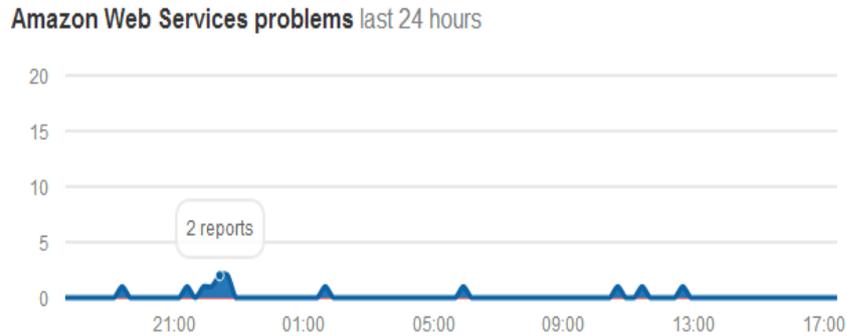


Figure 1.3 AWS problem reports number as a function of time between 24 and 25 October 2013

Many architectural properties need improvements to achieve perfect Cloud, such as: availability, performance, security, reusability, validity and compatibility.

Security: The world is moving toward a Cloud model for software-as-a-service. But the available Cloud service security approaches are not mature enough for the Cloud transformation. The two famous examples about Cloud security weakness are the security fails of Adobe² (2013) and Dropbox³ (2011). The security problems come from the adoption of the traditional SOC security concept of the Cloud. Current research approaches for Cloud security mainly focus on either the service providers' VMs or the host system. In the former area, integrity measurements are performed using the Cloud infrastructure's support. The Cloud infrastructure itself is not verified in these approaches.

1: www.downdetector.com

2: <http://venturebeat.com/2012/08/01/dropbox-has-become-problem-child-of-Cloud-security/>

3: <http://www.dpreview.com/news/2013/10/03/adobe-accounts-hacked-data-exposed-for-2-9-million-customers>

Availability: A perfect Cloud system is a reliable system that offers a group of business services with no accessibility limits. The importance of Cloud availability measure makes it fatal element to Cloud when case of failure. In modern Web environments, high availability often is a key requirement, as even the slightest outage can introduce significant financial consequences and impact customer trust. “High availability typically is addressed by means of replicating servers and storage” (Hauck *et al.*, 2010). But the more servers are replicated the more cost is added which is not the purpose of Cloud.

Figure 1.4 shows an example about the difference in availability among several Cloud providers. Indeed, replicated servers and storages cannot be always achieved by small companies since of their cost. Thereby, the current replication solution of Cloud availability causes a big data and network problems. Why not to build a new Cloud model that benefit from the diversity of Cloud services to achieve service availability instead of replication?

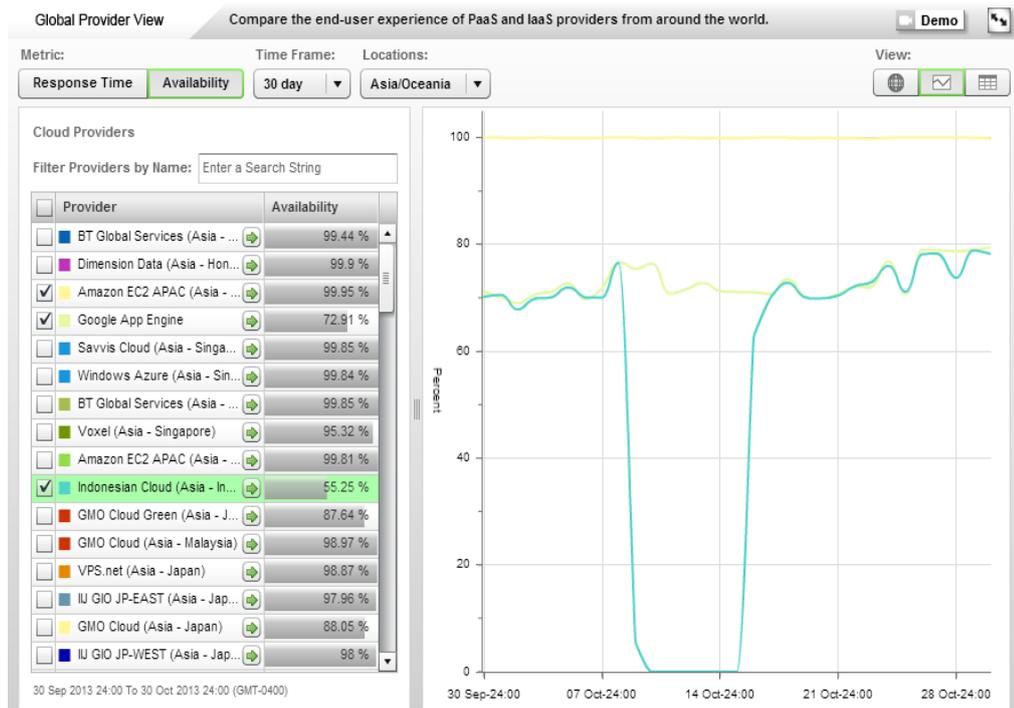


Figure 1.4 Availability variation of Amazon, Google & Indonesia Clouds (Source: CloudSleuth)

Indeed, currently the Cloud customers are dependent on availability of the Cloud provider that support them; when the availability of their Cloud decrease customers is affected negatively.

Performance: Reasoning about the performance of a Cloud service is a key factor that has to be taken into account in service development. Applications with less performance may cause lose customers, decrease employee efficiency, and add more costs on cloud companies. Because application performance can vary based on delivery environment, application performance must be optimized when written for deployment on the Cloud. Regarding service model, some performance approaches proposes solutions which allow the software architect to reason about the performance during the design-time (Hauck, 2010). However, these approaches have to be

enhanced to be used automatically in virtualized environments for the Cloud computing purpose.

Figure 1.5 divides the variations in performance among Cloud providers into three categories based on the service response time (green<13 s, 13 s<yellow<16 s, 16 s<red).

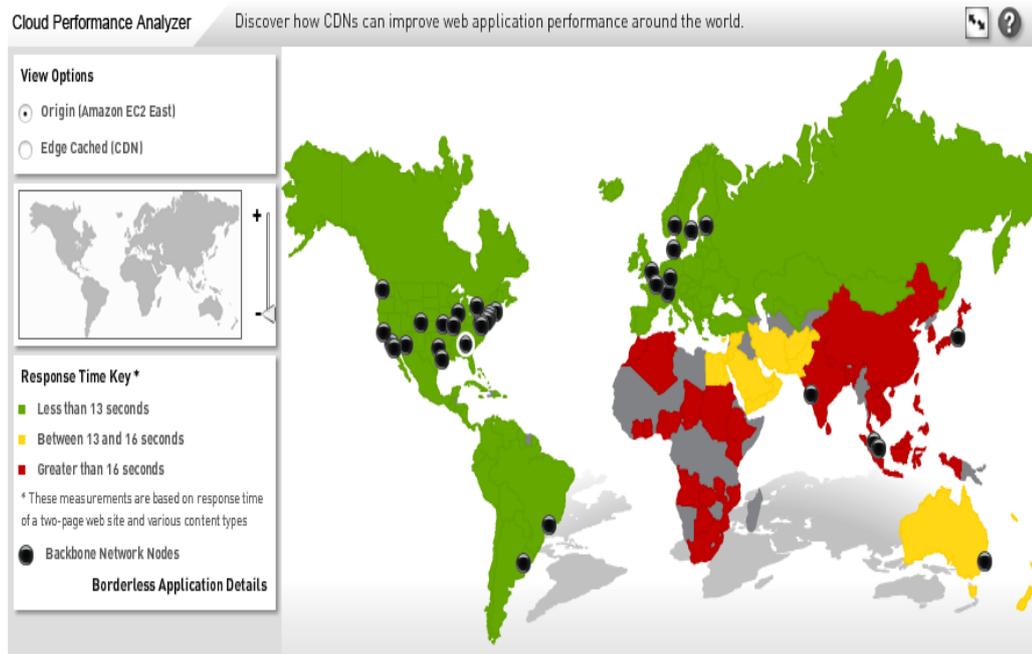


Figure 1.5 Analysis of cloud performance based on service response time (source: CloudSleuth)

Reusability and SOC: In spite of the fact that SOC is so complex, but almost available Cloud technologies are SOC dependent. For many years ago, before Cloud computing, SOC held the world business attention and maintenance, but it still suffers from several problems regarding the reusability of services. Since of SOC dependency, the Cloud faces the same questions before developing a new Cloud service: Which Cloud services can be adapted as components of a new software/service? How do we

evaluate existing services and measure their properties? Which is the exact service to be used in the Cloud application? These entire questions can be parts of service composition, discovery and selection problems.

- Service Composition: The main advantage of Web service composition is the possibility of creating value added services by combining existing ones. A great deal of recent Web related research has concentrated on dynamic Web service composition. Many dynamic composition models are proposed but the desired dynamic property is not achieved yet since Cloud services are not autonomous.
- Service Discovery: “Due to the increase of services, the discovery and selection of Web services meeting customer requirements become a very difficult operation” (Karray *et al.*, 2013). While the most used methods to discover a service is the search by word. Indeed the service discovery steps are complex because Cloud services are not categorized and classified.
- Service Selection: The service composition process depends mainly on the selection of available services on the Web but some services are equally qualified. “Identifying the optimal Web service, for each task that the application performs is a hard problem several services with equal qualities” (Nallur and Bahsoon, 2013).

Big data analysis: Big data is an inherent feature of the Cloud and provides unprecedented opportunities to use both traditional, structured database information and business analytics with social networking, sensor network data and far less structured multimedia (IBM, 2013). With these considerations, analyzing a large volume of data is not the only way to achieve value. In some Cloud software development there are so many changes with a given product on a regular basis, that using schema-based tools is not efficient. Each time there are new constraints on production users need to change the schema of their database. That is a tender procedure, especially with a big amount of data. Beside the constrained problem, there is also an analytic problem. The Cloud data increases sharply and it becomes hard to analyze stored data.

Validity: In the dynamic world of service-oriented computing, however, what is sure at design time, unluckily, may not be true at run time (Karawash *et al.*, 2013). Traditional approaches, which limit service composition validation to being a design time activity, are no longer valid in this Cloud dynamic setting. Besides performing design time validation, it is also necessary to perform precedent run-time validation to ensure that the required properties are maintained by the operating system. There is no way to validate service at the design phase regarding deadlock problems, but companies (like: IBM⁴ and Tibcommunity⁵) use the style of waiting deadlock errors to occur then apply fixing these errors.

-
- 4: <http://www-01.ibm.com/support/docview.wss?uid=swg1PM07820>
 - 5: <https://www.tibcommunity.com/message/70086>
 - 6: <http://www.w3.org/International/ws/>

Compatibility: According to W3C the current Web compatibility job is to achieve service internationalization⁶. The available Web and Cloud services do not support a global use of service, including all of the world's languages and cultures. Thereby, it is recommended to make these services compatible to every language. But, language compatibility demand added more difficulties on service developments and requires inserting new concepts to the service model since services are not categorized and dynamically developed.

1.3.2 CLOUD COMPUTING PLATFORMS CHALLENGES

There are several researches discussing approaches for the Cloud service problems. In the recent years, some of the technology companies deviated towards Cloud computing strategy because they predicted that Cloud computing is the solution of several web problems like availability. In order to launch Cloud computing faster, internet companies used the standards of the service paradigm. On the way to make more benefits, each of the web companies, which entered the Cloud world, built its own Cloud platform.

There are several Cloud computing providers, including Amazon, Google, Salesforce, Yahoo, Microsoft and others that are providing Cloud computing services. Currently, the Cloud vendors face a big problem which is summarized by the “Cloud Platform Battle” where every new Cloud company builds its own Cloud platform of special properties. This battle costs Cloud companies a lot of money because there is no approve on a standard Cloud platform. Passing through the Cloud battle some companies pay millions on building a Cloud platform but suddenly fails.

The Cloud problems are not simple because architect must solve first the problems of adopting service paradigm then to find a solution in the Cloud platforms battle.

The section highlights on some Cloud platforms and their issues such as:

Google Cloud Platform: it is a collection of Cloud computing products by Google. It enables developers to build, test and deploy applications on Google's highly-scalable and reliable infrastructure. Google Cloud platform supports a group of products such as: Application engine, Compute engine, Cloud storage, BigQuery and Cloud SQL. In order to displace other Cloud vendors (such as Amazon), Google Cloud Platform offers currently restored pricing, better testing and deployment tools. In 2014, Google's earnings report discovered that the company spent \$2.35 billion on infrastructure, which for Google means its data centers and all the IT gear that go with them. Recently, Google is on the way to spend \$10 billion building an outfitting data centers.

Microsoft Cloud Platform: It is a Cloud computing platform and infrastructure, created to deploy and control cloud services through a global network of Microsoft-managed data centers.

Microsoft Platform ensures a reliable hosting and scaling out of application codes. "In 2008, when Windows Azure was still known by its codename "Red Dog," Microsoft's message was that Windows Azure was a Cloud version of Windows Server. This twinning of its on-premises and Cloud offerings has been at the crux of Microsoft's private/public/hybrid Cloud messaging", Mary Jo Foley.

On 2014, Microsoft made \$5.66bn in financial profits. Azure revenue grew over 150% and there are 1000 Azure's new customers every day. "*Microsoft's Cloud growth really is impressive,*" said John Dinsdale, a chief analyst and research director at Synergy Research Group.

IBM Cloud Platform: SmartCloud Application Services is an IBM platform as a service offering that enables you to quickly and easily develop, test, deploy and manage applications within your IBM SmartCloud Enterprise account. It provides cloud clients with valuable platform services which can be consumed via different types of services.

IBM bets billions of dollars on Cloud computing. Recently, in 2014, it was announced that it is putting \$1 billion behind its platform-as-a-service strategy. On 2014, IBM shut down its SmartCloud Enterprise Cloud computing platform. IBM had migrated Cloud customers into its novel SoftLayer Cloud computing platform (Blue Mix), in July 2014, to better compete with other Cloud vendors.

1.3.3 DEFICIENCY OF CLOUD INTELLIGENCE

Companies, industry analysts, and customers have all expanded the meaning of the term to include a broad range of technologies and products. While a growth of Cloud vendors offers businesses more options, it also complicates the normal analysis and flow of the underlying technologies. Consequently, Cloud companies evaluating potential Cloud infrastructures should take a smart, consistent, realistic, business-minded approach in evaluating competing Cloud computing infrastructures.

The required Cloud is a holistic ecosystem of components, which has specific requirements to meet the needs of enterprise organizations. These requirements include intelligence, autonomy, scalability, adaptability and extensibility. In addition to that, the Cloud must reveal additional capabilities such as providing for security, real-time availability, and performance. To reach a Cloud rapidly, organizations find it almost impossible to achieve the methodological redirection without benefit from other web trusted architectures and models. In consequence, the SOA and service oriented computing paradigm formed a base to reach Cloud requirements (Ramana, 2011). This is because service oriented paradigm is mainly used paradigm in the web where it aims to achieve interoperability of remotely or locally located homogeneous and heterogeneous applications by utilizing reusable service logic.

Starting from the current Cloud, the ongoing evolution aims at transforming the today's inflexible distributed components in the future intelligent systems.

With the amassed success of Cloud computing, Cloud services have shined widely, both from Cloud startups and major industry vendors. Beyond porting intelligent features into the Cloud, some numerous issues must be solved (e.g., BigData analysis, cloud service security, cloud service performance, etc.). Also, the Cloud architect should not forget that Cloud poses new, broader challenges for making data analytics available to several enterprises and organizations, web communities, and even the average citizen; this idea probably requiring a combination of both private and open data.

Thus, Cloud intelligence is not considered as technological challenge only, but also an important general stake. Indeed, people increasingly demand open data and services in

which they need to access easily from the Web. Sometimes, in order to discover, select or analyze Cloud services clients utilizes intelligible on-line tools with advanced collaborative features and results are complex to be achieved. In addition to that, the reusability features of the Cloud components effect negatively on the Cloud clients, where a massive quantity of services is deployed while there is no autonomic way to overcome complexity issues of reusability. Sometimes, instead of providing simplicity, clarity and automaticity some computing approaches (like Cloud and web service approaches) invite clients to play the role of architect. Briefly, in this huge flow of data and overwhelmed web communications, Cloud clients look for the simplest and shortest method to reach their services and this cannot be reached without inserting intelligence property.

1.4 CONCLUSION

Cloud computing depends on sharing of resources to accomplish consistency over a network. It combines influential services management with rich business management tasks. It offers several fields of service models, including infrastructure as a service (*IaaS*), platform as a service (*PaaS*), and software as a service (*SaaS*). Clouds must be supported with more effective security, better service management and simpler lifecycle management. Despite the fact that Cloud model has already gained a lot of popularity and is considered the future in the IT industry, many companies are still concerned about Cloud issues. Some of these issues are: availability of services, security, complexity, legal issues, performance, validation and issues of reusability. However, Cloud Computing has many strong points: infrastructure flexibility, faster deployment of applications and data, cost control, adaptation of Cloud resources to real needs, improved productivity, etc.

The next chapter discusses the state-of-the-art of the previous approaches of cloud computing problems.

Chapter 2

RELATED WORKS

Cloud computing model faces several architectural challenges when the distributed system requirements are the goal. For now, reaching better clouds requires a restructure of distributed system strategy in the era of virtualized environments. This chapter discusses previous approaches of some Cloud computing problems.

2.1 INTELLIGENT DISTRIBUTED COMPUTING

Distributed systems have loosely-coupled components running on networked computers that communicate and organize their jobs by message transfer. Through the history of intelligent distributed computing, several models and strategies were proposed to achieve system intelligence and this section summaries some of these researches.

Davies *et al.*, in 1995, introduced Agent-K as a simple production rule mechanism for programming of agents (Finin *et al.*, 1993). This approach of knowledge integration turns out to be quite heavy in terms of computation. Seydim, in 1999, worked on an agent model in the domain of information retrieval, filtering, classification and learning along with a communication framework for the exchange of information between multiple mobile agents. These multiple agents worked over distributed systems for knowledge discovery (Kershenbaum, 1995).

Vassiliadis, in 2000, settled ARKTOS, an computerized tool for data cleaning and conversion in data warehouse environments. While specialized tools are already available designed by Data Mirror Corporation and others etc., this ARKTOS tool is good prepared with one graphical interface and two declarative languages closely related to XML and SQL. The tool covers primitive operations for Extraction-Transformation-Loading and more especially cleaning primitives like primary key violation, reference violation and others. Knoblock and Craig., in 2004, proposed the group of software agents for travel planning by retrieving information from web. These agents provide interactive interface as the user is provided with all the choices & monitors all the aspects of a trip. Finally, this software performs mining over all information to help the user in their decision making.

Zghal *et al.*, in 2005, introduced the agent framework for data mining of spatial data by combining the different algorithms of data mining & features offered by the multi-agent systems. Authors also resented the architecture of Computer Aided Spatial Agent Mining Mart Environment (CASAMME) and a CASE tool (2003) based on the multi-agents system. Ong *et al.*, in 2005, discussed the problems of wrong expectations associated with data mining algorithm. As a solution to this, he has introduced a new methodology of designing stream-based algorithms with mobile agents. The experimental results show the increased speed reached which is roughly closer to be linear.

Bach *et al.*, in 2005, proposed retrieval of public data exists over web with software agents for business intelligence. Software agent work for retrieval of data from the Data base of stolen cars in Croatia, the data thus collected is also analyzed and various reports are prepared stating risk involved in different classes and brands of cars to help for better

decision making in insurance company. Nurmi *et al.*, in 2005, presented architecture for distributed data pre-processing. Even though the developed methodologies of the context aware application mobile agents may be implemented at several software levels, the architecture proposed by the author needs more clarifications. The architecture framework offered by the authors is started with recognition phase, followed by decision making phase. At first peek, the architecture seems to emphasis on just preprocessing phase whereas it is also claimed that it may actually be implemented over distributed ubiquitous environment.

Tudor *et al.*, in 2009, emphasized the usage of software agents to figure out the relevant information so that academic groups may concentrate their activities on improving management quality based on knowledge. In this research, data mining & software agents are joint to work on knowledge management in academic. Moemeng *et al.*, in 2010, presented an agent-based distributed data mining platform named i-Analyst containing software packages & development kit for the better performance of data mining algorithms. The example outcomes itself discloses the significance of the agents in improving the execution performance. Singh *et al.*, in 2011, discussed and compared five agent development toolkits: JADE, VOYAGER, ZEUS, AGLET and ANCHOR developed by different groups. Their work has been drawn on the basis of standards followed, security mechanism, agent mobility and migration scheme etc. Authors realized that Jade agent development toolkit is most stable toolkit.

Jonsson *et al.*, in 2011 proposed an iterative strategy to better scale up with the number of agents and being able to compact with non-cooperative agents. In this approach,

typical off-the-shelf planning technology is used with a novel best-response planning method. In spite of the lack of convergence or optimality guarantees, this approach can be valuable to improve multi-agent plans. Jayabrabu *et al.*, in 2012, suggested the automated process of data mining for better visualization with the integration of multi-agent system to discover those new and hidden patterns. The automatic clustering of pertinent data set by these agents points to good input cluster to pit on, which ultimately returns the better correlated outputs. These outputs are shown by link charts instead of traditional data mining visualization methods like graphs, or histograms etc.

2.2 CLOUD SERVICES ISSUES

There are several previous researches that discuss approaches about the Cloud service problems. This section highlights on some approaches.

Web Service discovery: Service discovery is the process of locating Web service providers, and retrieving Web service descriptions that have been previously published. Banaei-Kashani *et al.* in 2004 and Toma *et al.* in 2005, attempted to bring Web service discovery mechanism on top of Peer-to-Peer network thereby reducing human intervention which is concerned with resource linking but nothing has been mentioned about the applications that process these resources. Wen-yue *et al.*, in 2010, divided search in three layers by applying filters at each layer and thus minimizing search area. They have applied this approach to intelligent automotive manufacturing system. Emekci *et al.*, in 2004, proposed a structured peer-to-peer framework for Web service discovery. As the format of sending a Web service request

is fixed, some information in user's request is lost during transforming user's request to formalized one. To overcome this limitation, Rong and Liu, in 2010, suggested a context aware Web service discovery approach.

Zhou *et al.*, in 2007, proposed a peer-to-peer framework for service discovery. To guarantee discovery efficiency, ServiceIndex schemed WSDL-S (Web Services Semantics) as Semantic Web Services description language and extracted its semantic attributes as indexing keys in Skip Graph. Kopeck, in 2007, intended to research an approach to the Semantic Web Service discovery to find the most appropriate Web services. Cardoso and Sheth, in 2002, presented a methodology and a set of algorithms for Web service discovery based on three dimensions: syntax, operational metrics, and semantics. Verma *et al.*, in 2003, presented METEOR-S Web Services Discovery Infrastructure (MWSDI) for semantic publication and discovery of Web services. Nawz *et al.*, in 2008, proposed a push model for Web service discovery where service requesters are provided with service notification prior to discovery.

Qiang *et al.*, in 2008, proposed a peer-to-peer based decentralized service discovery approach named Chord4S. Ge *et al.*, in 2006, presented a Web service discovery architecture by combining semantic Web service with P2P networks.

Keller *et al.*, in 2004, described different levels of service matching. It is understood that service matches are mandatory but not sufficient for Web service discovery. Deng *et al.*, in 2004, proposed a two-phase semantic-based service discovery mechanism to discover services in precise and automatic way.

Sivashanmugam *et al.*, in 2004, proposed METEOR-S Web Service Discovery Infrastructure (MWSDI), an ontology based infrastructure to provide access to private and public registries divided based on business domains and grouped into federations for enhancing the discovery process. Paolucci *et al.* focused on discovering Web services through a centralized UDDI registry. Centralized registries can provide effective methods for the Web service discovery, but they suffer from problems associated with having centralized systems such as a single point of failure, and delayed delivery of notifying updated service description.

Service selection: This part discusses the previous works which deal with the service selection process. Big efforts are done to define the *QoS* to be used in the service selection. The *QoS* has been received much interest in the Cloud service researches, because of the rapid increase of the number of services and the approximate equal qualities of the discovered services. Several re-search activities focused on how to benefit from the *QoS* in the service selection process. Some of these researches worked on extending the UDDI registry to support *QoS* information. At First, we mention the service selection algorithms used by the *QoS* broker for sequential composite flow models with only one *QoS* constraint (i.e. Throughput).

There are two main approaches we can use to select the optimal services for each component of a business process. The first approach is the combinatorial approach, by modeling the problem as a Multiple Choice Knapsack Problem (MCKP). In order to solve the MCKP, three methods are proposed: ex-haustive search, dynamic programming, a minimal algorithm for MCKP and performance study method. The

second approach is the graphical approach, by modeling the problem as the constrained shortest path problem in the graph theory. The proposed methods to solve the shortest path algorithm are: Constrained Bellman-Ford (CBF) method, Constrained Shortest Path (CSP) method and Breadth-First-Search (BFS) method. Also there is a set of other algorithms that deal with the service selection problem. In 2002, Maximilien and Singh proposed a Web service Agent proxy (WSAP) algorithm to access a service.

Shaikhali *et al.*, in 2003, extended in their UDDIe project the current UDDI registry by adding “blue pages” to record user defined properties associated with a service, such as *QoS* information, and to enable service discovery based on these properties. Also in 2003, Ran proposed a model for Web service discovery with *QoS* by extending the UDDI model with the *QoS* information, which is similar to UDDIe. Later, in 2004, Lee and Pan improved the fuzzy genetic algorithm (GA) that learns user preference related to *QoS*. In 2007, the algorithm of a personalized Web service selection UDDI with a fuzzy *QoS* attribute interface was proposed by Wang *et al.* Then Keskes *et al.*, proposed, in 2009, a model of automatic selection of the best service provider that is based on mixing context and *QoS* ontology for a given set of parameters of *QoS*.

In 2010, Raj and Saipraba proposed a service selection model that selects the best service based on *QoS* constraints. While Squicciarini *et al.*, studied, in 2011, the privacy implication caused by the exchange the large amount of sensitive data required by optimized strategies for service selection. In 2012, Mohebi proposed a

vector-based ranking model to enhance the discovery process of Web services. In 2013, a heuristic method called “Bee Algorithm” was proposed by Karry *et al.*, which helped to optimize the discovery and selection of Web service that meets customer requirements.

Service security: There are many security issues that affect the job of Web services. An encryption solution allows users to choose their preferred Web services, because the files are always encrypted and the keys are always their own. But this control of service protects is still not ideal and faces many problems. Basically, the security problems that are likely to affect Cloud services are the same as those that have affected the conservative Web-based systems. Security is significant to the adoption of Cloud services by enterprises, but, as it stands today, the Cloud service structure does not meet basic security requirements.

Service availability: Ensuring the availability of applications and data that run on the private Cloud is a very difficult task. High availability often is a key requirement, as even the slightest outage can introduce significant financial consequences and impact customer trust. High availability typically is addressed by means of replicating servers and storage. Since availability is a main challenge for enormous numbers of servers, replication of storages was the applied method. Also, replication techniques can potentially be implemented more cost-efficiently. How to build a new Cloud based applications that achieve aforementioned promises of improved scalability and availability?

Service performance: Cloud computing and Virtualization promise substantial reduction of IT operating costs resulting from higher energy efficiency and lower system management costs. However, the adoption of Cloud computing and Virtualization comes at the cost of increased system complexity and dynamicity. The increased complexity is caused by the introduction of virtual resources and the lack of direct control over the underlying physical hardware. In many cases, however, the underlying infrastructure of the Cloud platform may directly affect application performance (Joyent, White Paper).

Service validation: Some ways of dynamic system validation are discussed in this section. In 2006 Colombo *et al.*, commenced the topic of dynamic composition where the service parts do not always behave along expected lines. They provide an extension to the BPEL language in the form of the ‘SCENE platform’ which addresses this issue. The proposed platform was validated using a set of real services and observing the behavior of the application (Colombo *et al.*, 2006). In 2009, Silva *et al.* proposed the DynamiCos structure which response the requirements of different customers to dynamically put together personalized services. To confirm the proposed structure they set together an extensive model of the structure which enables services to be deployed and be published in a UDDI-like registry (Silva *et al.*, 2009).

In 2008, Eid *et al.* explained a set of scales alongside which to evaluate the various frameworks of dynamic composition. The set of scales was inclusive and classified into three parts: input subsystem, composition subsystem, and execution subsystem. To be considered good a composition model must achieve well against these scales. In

2007, Shen *et al.* found the Role and Coordinator (WSRC) model to hold dynamism in web service compositions. In this model, the development of service composition was divided into three layers: Service, Role, and Coordinator. To validate the model, they described a case-study of a vehicle navigation system which comprises a global positioning system and a traffic control service. These are a small list of the validation methods in use for dynamic composition models and structures. Some of these methods are quite complex like the model proposed by DynamiCos or that of the SCENE platform.

2.3 ENCOURAGING PROJECTS

Currently, there are some other projects that work on the idea of brain and cell in solving problems but from different perspective. Indeed, these projects obtained a high acceptance from companies and great funds.

K supercomputer project: The K Computer was made by the Japanese Ministry of Education, Culture, Sports, Science, and Technology (MoMESST) in union with the Fujitsu Corporation and specifically aimed towards breaking the 10 petaflop fence. The project is formed of 705,024 processor cores and 1.4 million GB of RAM, but still takes 40 minutes to crunch the data for just one second of brain activity. The K supercomputer has increased its computational output to 10.5 quadrillion calculations per second and making it the speediest number-crunching system on the planet. “According to industry benchmarks, the K computer is performing at 93 percent

efficiency. However, given that it burned through \$9.89 million of electricity yearly when it ran at just one petaflop” (Tarantola, 2011).

Brain cell database project: The National Institutes of Health (NIH) in USA announced that it will allow researchers to study brain cell activity in motor neuron disorders. For this purpose, it declared that it had been awarded the brain cell database project around \$8-million as a grant to establish one of six centers around the USA tasked with creating a database of brain cell activity, expected to help develop treatment for a number of diseases (Irvine, 2014). The project results will be used to identify cell targets for new drug treatments.

2.4 CONCLUSION

The Cloud computing paradigm has been receiving important interest in the recent years. Despite the difficulties and problems which face Cloud, there exist accumulations in the number of large companies that are offering Cloud computing infrastructure products. Cloud connects a network of virtualized computers that are dynamically provisioned as computing resources, based on contracts between service providers and users.

All that being said, Cloud computing made web more mainstream, the technical difficulties have begun to take new shapes along with its popularity. Several unresolved issues exist, particularly related to security and privacy, and reusability. Other open issues include data transfer bottlenecks, performance unpredictability, reliability, expensive availability, internationalization and big data analysis. Regarding this random Cloud growth, it is recommended to suggest approaches of major Cloud problem like: reusability

and less intelligence problems. The previous methods were based on the concept of service.

This chapter shows a state-of-the-art of previous approaches of Cloud service issues and highlights important research directions in this increasingly important area towards solving Cloud service problems. Through the next chapter, we are going to propose a new smart Cloud theory to enhance the traditional Cloud service model through adding some required properties like autonomy and intelligence.

Chapter 3

SMARTCELLS; A CELL-ORIENTED SMART CLOUD APPROACH

Recently, there has been a significant exploration of new ideas to take the technology towards a new scope. Cloud computing allows infrastructure, computational resources, databases, networks and services to be shared among many in an efficient and on-demand basis. Recently to achieve more facilities in managing the globe researches shows that everything deviated toward intelligence while the model that control the business world (like: Cloud computing) lack of the vital property. This has gained attention from both academia and industry and is considered to be one of the highly influential cases of study for effective sharing of different resources through intelligent distributed networking. This chapter shows the SmartCells approach as a new smart Cloud approach that applies new distributed computing concepts and algorithms to reach an intelligent Cloud system.

3.1 TOWARDS SMART CLOUDS COMPUTING

Distributed computing applications, communication tools, and mobile technologies are among the most influential innovations that shape our lives today.

“Every Cloud has to be managed by someone, even commodity Cloud infrastructure. You either outsource to a specialist company like Rackspace, or you pay handsomely to find and bring that talent in-house.” Said Dane Atkinson the CEO of SumAll.com

Several challenges may result from the integration across several types of computing models in the design and development phases. For instance, how to benefit from the wide use of service-oriented architecture in building intelligent architecture? How to avoid the complex selection process of the Web service model? How to achieve dynamic business process composition despite the variety of companies providing different types of service processing? How to use the intelligence of multi-agent systems as a control mode from the client side? How to reach the best non-functional properties of processes in an autonomic manner? How to avoid the security weaknesses resulting from mobile agent communications? How to prevent damage of services caused by internal and sub-service fail? Why not to separate software processes based on their purpose? How to arrange procedures of distributed computing in a way that evades big data analysis problems resulting from random connections among distributed systems? How to globally consistent solutions be generated through the dynamic interaction of distributed intelligent entities that only have local information? How can heterogeneous entities share capabilities to carry out collaborative tasks? How can intelligent distributed systems learn, improving their performance over time, or recognize and manage faults or anomalous situations? Why not to use dynamic online analysis centers that monitor the on-the-fly qualities of distributed software processes? How to validate the processes of distributed software at the design phase? How to accomplish the internal protection of distributed Cloud components based on the dual context-profile of both consumer and provider?

Due to the important maturity efforts invested in Cloud systems, there is a need to re-model existing designs so that Cloud-based services could take place in a better manner. There are many intelligent techniques that may be used or imitated to develop Cloud systems functionality. “A number of natural and artificial systems can be considered as intrinsically distributed and consisting of nodes presenting a certain degree of intelligence. Typical examples of distributed intelligent systems include human body, social insect colonies, flocks of vertebrates, multi-agent systems, transportation systems, multi-robot systems, and wireless sensor networks” (Martinoli, 2014).

If we consider the human body as a standard to be adopted, we find that every part of the human body is made up of cells. There is no such thing as a typical cell. “Our bodies are composed of different kinds of cells. The diverse types of cells have different, specialized jobs to do. Cell computing simulates the human cell tasks in the distributed systems environment. In fact, there are approximately 10 trillion cells in the human body” (Brain, 2013). Cells are the basic structural and functional units of the human body. Each cell has a specialized function and works in collaboration with other cells to perform a job. The cell acts like a mini computer. It is composed of a decision center (the nucleus), the protein industry (Mitochondria), store of human traits (Genes) and a defense system (cell membrane). All cells in the body are associated to brain Intelligence that controls their jobs.

Indeed, the human body system consists of huge cells network which is millions of times larger than the whole web networks. Each cell has a great capacity to receive and transmit information to every cell in the body. Each Cell remembers past actions, stores

information about our daily life also evaluates possibilities for the future. It has an internal defense system to face intruders when an external attack occurs. Adding intelligence to Cloud computing systems will make them more adaptive, flexible, and autonomic. Also, Cloud intelligence will improve the deficiency for cloud issues (such as security, availability and performance). The next section highlights some biological key features that are used in developing SmartCells components.

3.2 BIO-CELL IMITATED KEY-FEATURES

Building a smart distributed system model that controls the whole of Web communications needs to be based on an extremely consistent system. One of the best systems that can be adopted is the model of the human body system, specifically the body brain and cell.

HUMAN BRAIN

The brain is center of intelligence, performer of the senses, originator of body movement, and director of behavior. “The brain performs an unbelievable number of tasks including the following:

- It controls body temperature, blood pressure, heart rate and breathing.
- It accepts a flood of information about the world around you from your various senses (seeing, hearing, smelling, tasting and touching).
- It handles your physical movement when walking, talking, standing or sitting.
- It lets you think, dream, reason and experience emotions.
- All of these tasks are coordinated, controlled and regulated by an organ that is about the size of a small head of cauliflower” (Freudenrich and Boyd, 2013).

“The function of the brain is to apply centralized control over the other organs of the body (Figure 3.1). This centralized control allows rapid and coordinated responses to changes in the environment” (Brain, Stanford Wikipedia).

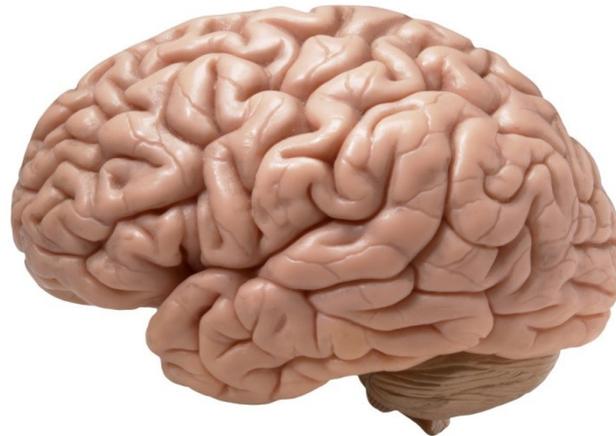


Figure 3.1 The human brain is a center of management of body organs

“Let's look at the brain using a different model. Let's look at the brain as an orchestra. In an orchestra, you have different musical sections. There are a percussion section, a string section, a woodwind section, and so on. Each has its own job to do and must work closely with the other sections. When playing music, each section waits for the conductor. The conductor raises a baton and all the members of the orchestra begin playing at the same time playing on the same note. If the drum section hasn't been practicing, they don't play as well as the rest of the orchestra. The overall sound of the music seems "off" or plays poorly at certain times. This is a better model of how the brain works. We used to think of the brain as a big computer, but it is really like millions of little computers all

working together” (Johnson, 2010). “The brain is like a committee of experts. All the parts of the brain work together, but each part has its own special properties” (NIH, 2014).

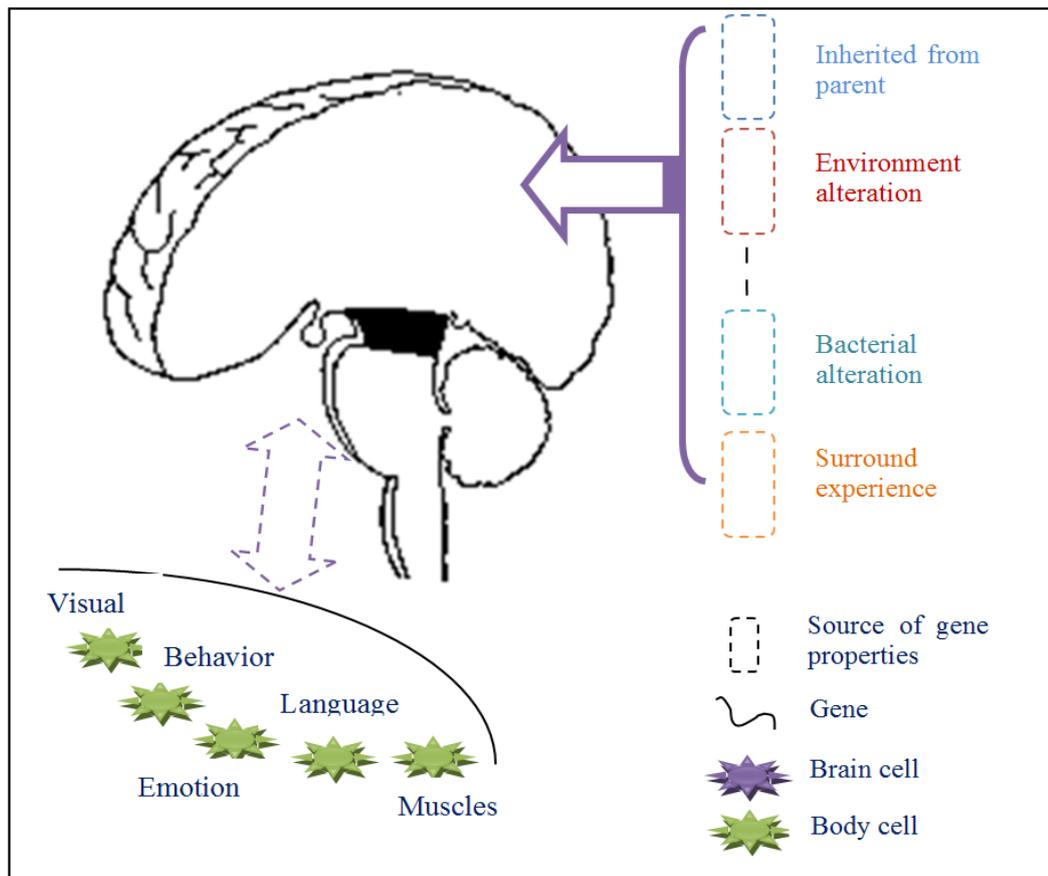


Figure 3.2 The brain in the SmartCells architecture

“The basic architecture of the brain is built through an ongoing procedure that begins before birth and continues into adulthood. Neural connections are formed first, followed by more complex circuits. In the first few years of life, 700 to 1,000 new neural connections form every second. Later, connections are reduced through a process called pruning, which allows brain circuits to become more efficient. Brain architecture is comprised of billions of connections between individual neurons across different areas of the brain. These connections enable lightning-fast communication among neurons that

specialize in different kinds of brain functions. The interactions of genes and experience shape the developing brain. Although genes provide the blueprint for the formation of brain circuits, these circuits are reinforced by repeated use” (NIH, 2014).

The brain is responsible of managing the whole body system. Brain cells ensure a well monitoring and direct for the body cells. The human body is built on cells and these cells are sources for every step done by the body. Body cells are divided into several types according to their jobs. Some of body cells jobs are expressed by the outer body signs like: visual, language, behaviour, emotion and muscles signs. The body cells depend on the instructions of the brain cells to finish their jobs. In their turns, the brain cells depend on specific elements in producing instructions. A gene stores a map of instructions to be executed by the brain cell and these instructions are dynamically changed. Genes build their instructions based on several sources such as parent genes, environment and bacteria. Indeed, the most known phenomena, about genes, is that they are properties inherited over generations. Besides, the changes in the person environment and the interference from an outer bacterial organism may alter the genes traits. As shown in the Figure 3.2 above, the brain architecture may be summarized by three components: A center of control and direction, a set of consumer cells and a set of sources of instructions. The center of control and direction manages the set of consumer body cells based on the genetic instruction collected from several sources.

BODY CELLS

Every minute billions of cells in our brains transmit signals that manipulate everything from our memories and emotions. Brain Cells monitor the changes in the body

and send commands to the other body cells in order to keep on normal body. As known cells are renewable, thousands of body cells are dying per day and new cells with same jobs are formed. The brain cells are not renewable and they are responsible for forming a new cell (Figure 3.3). The formation of new cells depends on gene properties of stem (or brain) cells. In the human body, genes are inherited from parents and they are flexible to be altered by an outer environment that surrounds the body.

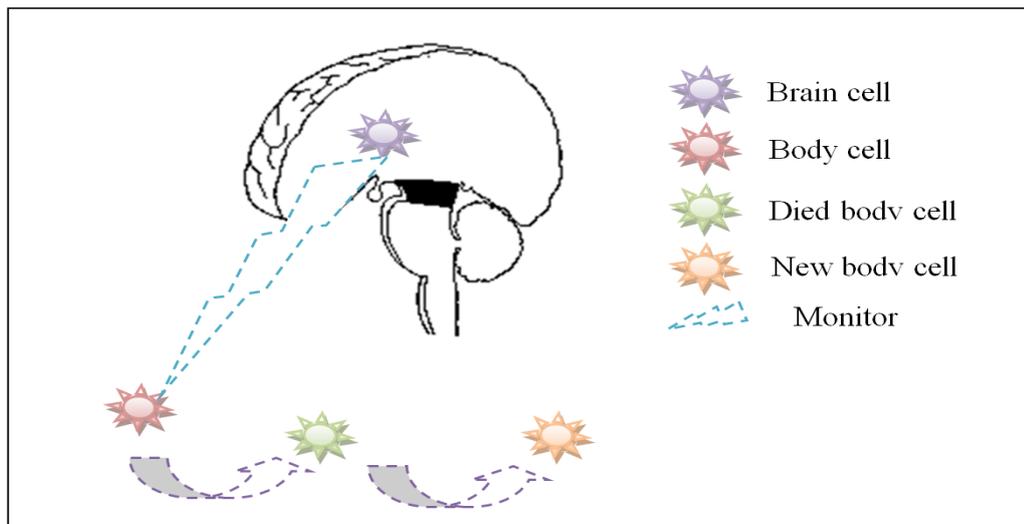


Figure 3.3 Cells renewal by the brain

“Communication between cells in the brain depends on specialized molecular receptors that conduct charged particles, or ions, between the outside and inside of cells. The brain is like an electrical circuit board, but it is very complicated to figure out how it all functions together. Memories are formed by strengthening the connections between brain cells, known as synapses. Specifically, memory requires the coordinated activation of many types of receptors at synapses” (Underwood, 2006).

The specialized, organized cells of our bodies are the product of millions of cycles of cell growth, and this growth may take two shapes: normal or cancerous.

Normal Cell: “The body is made up of tiny cells - for example, skin cells, muscle cells, heart cells, nerve cells, and bone cells. When a baby grows, the number of cells increases very quickly. A cell becomes a bit larger, and then divides into two "daughter" cells. After a period of time, each of these cells divides, and so on ...” (Larry, 2014). "Normal" cells stop dividing when they come into contact with like cells, (as shown in Figure 3.4) while cancerous cells lose this ability.

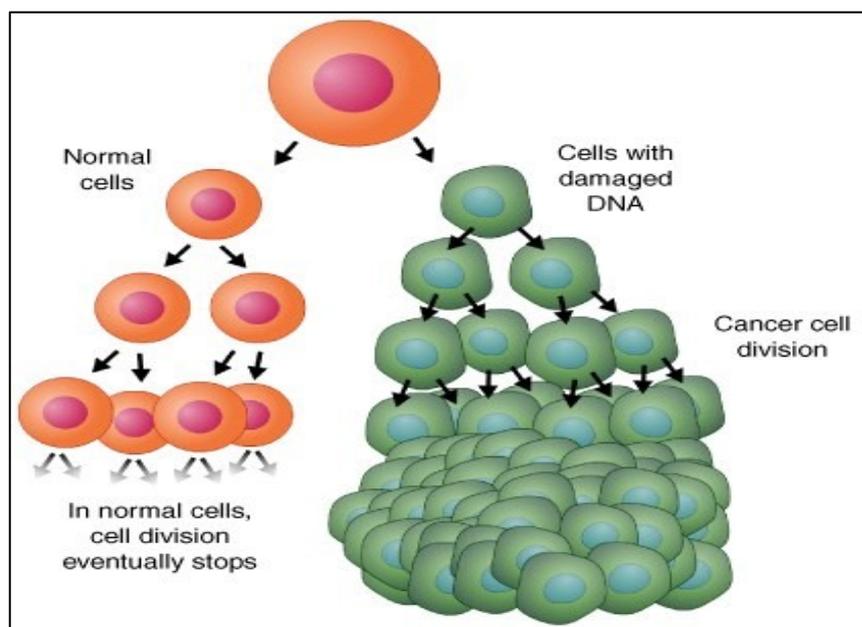


Figure 3.4 Normal vs. cancer cell growth

Cancerous Cell: “The immune system consists of a group of cells called white blood cells that destroy "foreign" material in the body such as bacteria, viruses, and unfamiliar or abnormal cells. Cancer cells somehow

manage to slip through this detection system without triggering the immune system to start fighting, either at the primary cancer site, in the blood vessels, or at the site of the distant spread” (Larry, 2014). “Cancerous cell are characterized by cell division, which is no longer controlled as it is in normal tissue. These cells have no normal checks and balances in place that control and limit cell division” (Chemocare, 2015); as shown in figure 3.4.

GENES

There are many diverse types of cells in the body. With the growth of the body, cells differentiate and become more specialized for specific functions. “Skin cells protect, muscle cells contract, and neurons, the most highly specialized cells of all, conduct messages. Every cell in our bodies contains a complete set of DNA. DNA, the "recipe of life," contains all the information inherited from our parents that helps to define who we are, such as our looks and certain abilities, such as a good singing voice”(NIH, 2014).

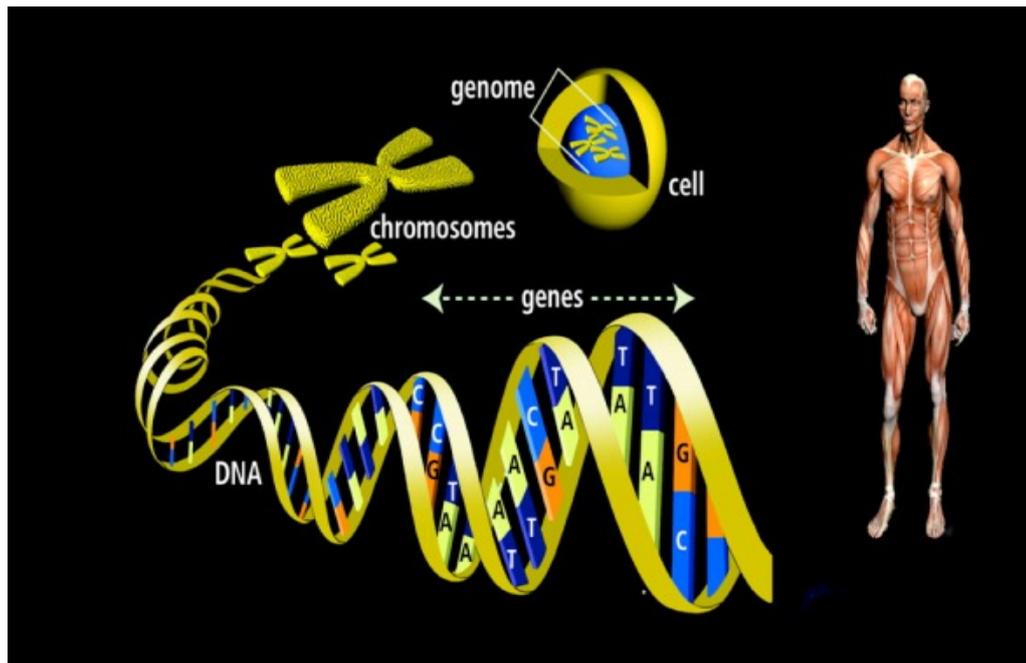


Figure 3.5 Genes contain the business process of a cell (source: U.S. energy Department)

As shown in Figure 3.5, a gene is a segment of DNA that contains internal codes about how to make proteins and other important body chemical components. DNA also control which genes are expressed and when, in all the cells of the body. “Genes do more than just determine the color of our eyes or whether we are tall or short. Genes are at the center of everything that makes us human. Genes are responsible for producing the proteins that run everything in our bodies. Some proteins are visible, such as the ones that compose our hair and skin. Others work out of sight, coordinating our basic biological functions. For the most part, every cell in our body contains exactly the same genes, but inside individual cells some genes are active while others are not. When genes are active, they are capable of producing proteins. At least one third of the approximately 20,000 different genes that make up the human genome are primarily located in the brain” (NIH, 2014).

3.3 BIO-CLOUD VS. BIO-CELL: MAPPING MODALITIES

In order to show briefly the main idea of the proposed SmartCells approach, this section discusses the common features between the Cloud computing strategy and the human body strategy. We start by discussing the general development of web computing methods then we compare the Cloud computing and human cell strategies. As shown in figure 3.6, the web had started as monolithic computing methods using a one simple machine (node).

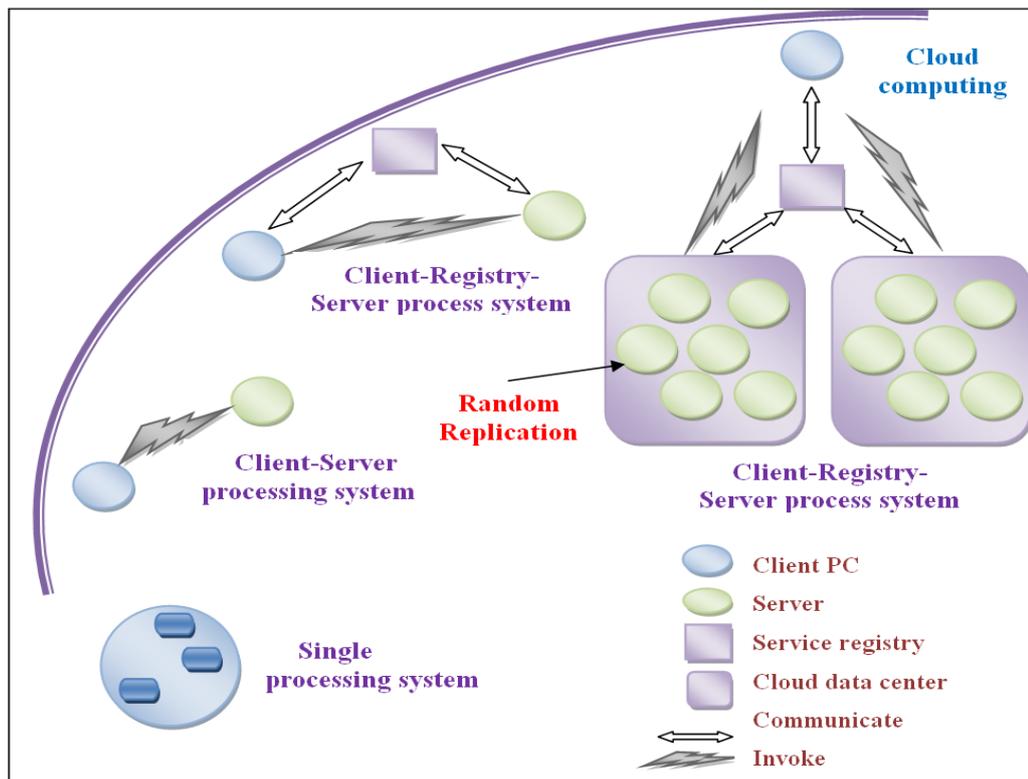


Figure 3.6 The development of distributed computing methods

After a while and because the client machine was of low properties, web companies suggested to add a complementary high quality machine and this computing method was known as client-server. Later on, the overwhelmed demands on the server side pushed companies to add more than one server to serve a client. Furthermore, they had added a third party (known as registry) to organize the communications between the client and invoked servers; this computing architecture is known by SOA. Recently, software engineering Architects adopted towards Cloud computing as a new strategy based on replicating the server side to achieve availability and some other benefits. But Cloud computing faces currently a very fast growth of web network where each Cloud company replicates its infrastructure tens of times.

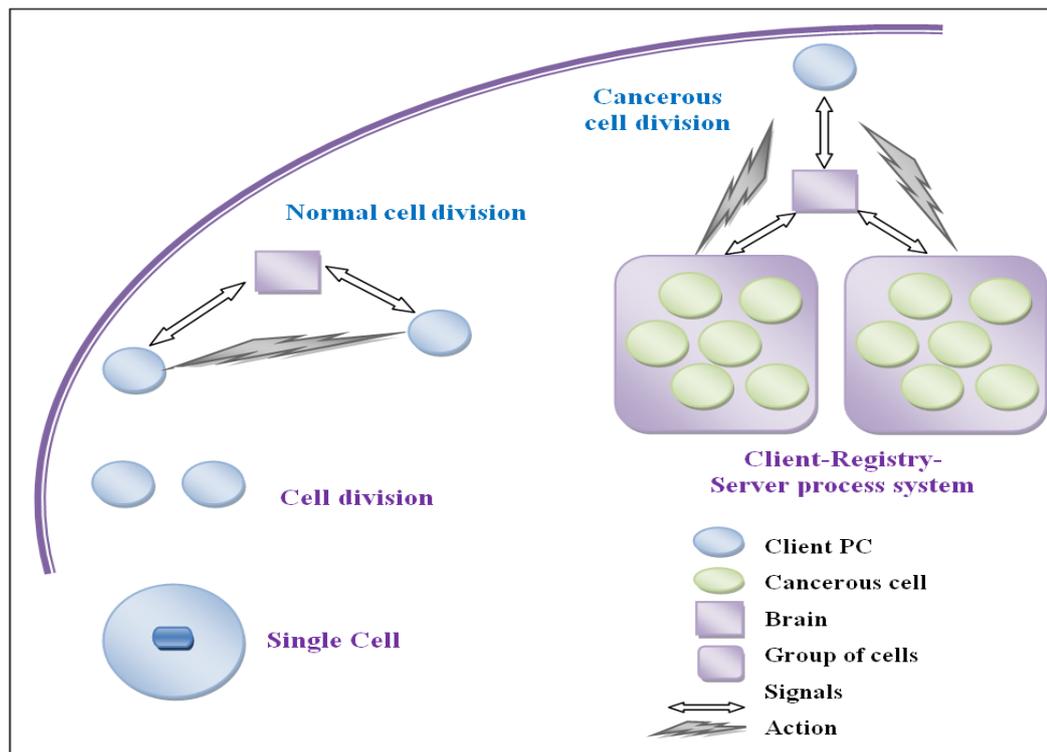


Figure 3.7 The development of body cells

Let us take a look to the growth strategy of cells inside the human body. As shown in figure 3.7, the main step in the growth is division step. Every task in the body starts with a unique cell which is divided or communicated later to give a collaboration of a group of cells. For example, if a skin cell senses a touch from an outer thing of the body it sends a signal to the brain, which in turns sends another signal to activate muscle cells. The organism system stays active through a cell division process. The cell division process is well organized and monitored by the brain in which a cell has a definite lifetime and can be divided into to give two other cells. A special case of division may occur without brain interference; which is the cancerous cell division. These cells divide randomly several times without stop and cause damage of body organs sometimes because they have own management system not following brain decisions.

If we compare the general views of both web development (till reaching Cloud) and the cancerous cell division, we notice that they are similar because both of them follow the same growth strategy. It is clear that in Cloud computing paradigm there is no centralized decision system for all Cloud platforms and the infrastructure of Cloud companies continue extends sharply; simply millions of servers are added every year. The case is similar for cancerous cell where it does not follow brain instructions and grow randomly with no limits. Also, Cloud model could reach the same result of that of cancerous cell and currently we see some of Cloud problems like big data, network overwhelm, etc.

For more accurate details, the table 3.1 highlights the common points between the Cloud and the human body strategies of work.

Table 3.1 Comparison between cloud and body strategies

Cloud Computing Strategy	Human body Strategy
<p>“Cloud computing is based on the centralization of resources. To the extent that content control are centralized” (De Filippi and McCarthy, 2012).</p>	<p>The brain’s centralized control allows rapid and coordinated responses to changes in the environment.</p>
<p>“Availability of Cloud systems is one of the main concerns of Cloud computing. In the Clouds, load balancing, as a method, is applied across different data centers to ensure the network availability” (Chaczko <i>et al.</i>, 2011).</p>	<p>The brain remains available and active while we are awake, sleeping, focused, or daydreaming.</p>
<p>A client invokes a Cloud service in order to perform a web job. For the next time use, the process of Cloud service may be altered internally and thus service becomes different, even if it has the same name and quality. Therefore, Cloud Architect keeps on Cloud services renewability.</p>	<p>Every day and to keep survival of human body, the brain manages the death and birth of thousands of body cells. A body cell lives for a period of time in order to serve a specific job in the human body.</p>
<p>A Cloud computing center is composed of thousands of servers that follow a load balancing</p>	<p>The brain consists of millions of interconnected neuron cells, which exchange</p>

strategy to serve clients. For example, Amazon Cloud uses 450,000 servers and Google Cloud uses around half million of servers.

signals with each other and with the rest of the body cells.

In order to enter the Cloud computing world, companies have to build Cloud centers. A Cloud center is composed of hundreds or thousands of servers that follow a load balancing strategy to serve clients. The Cloud services are invoked from within these servers. Cloud services are stored in the shape of business processes which control the quality and job of these services (figure 3.8).

The human body is composed of a set of organs. Each organ is built of group of cells. The main organ is the brain which composed of millions of nerve cells that control and manage the body jobs. Each body cell has a specific role and jobs to do when invoked by the body organs. A body cell works for a period of time, and then it is replaced by another cell. Body cells follow brain decisions and program stored in their genes. The gene is a part of DNA stored in the cell nucleus contains a work process map of a cell (figure 3.9).

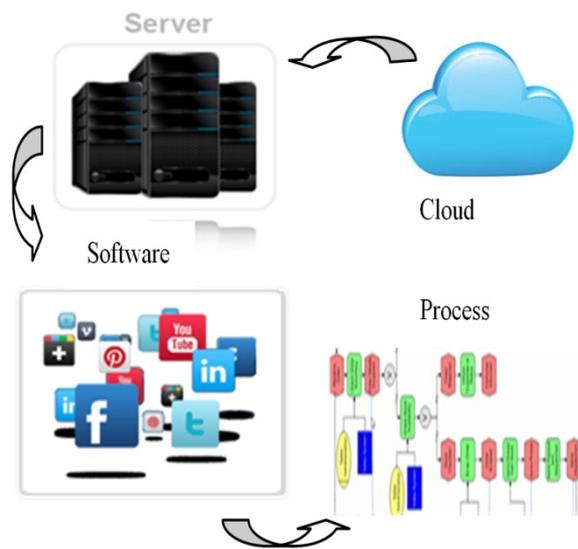


Figure 3.8 Service process is the last stone in building cloud service.

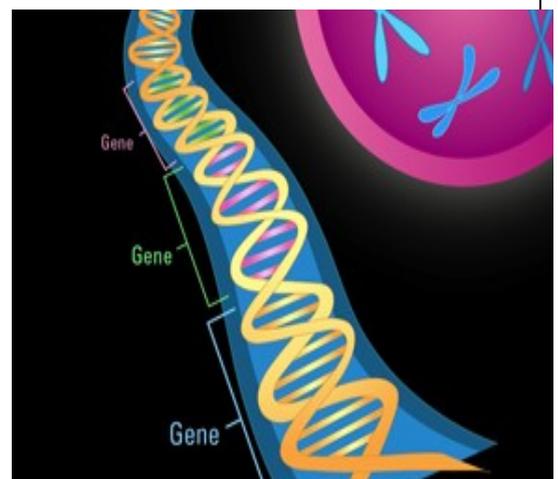


	Figure 3.9 A gene is a part of DNA
--	---

Almost every program takes input from a user in one form or another. A Cloud service is an online program which is invoked through sending input from the client to be used in fabricating the output (figure 3.10). Inputs and outputs may be several types such as: string, number, object, etc.

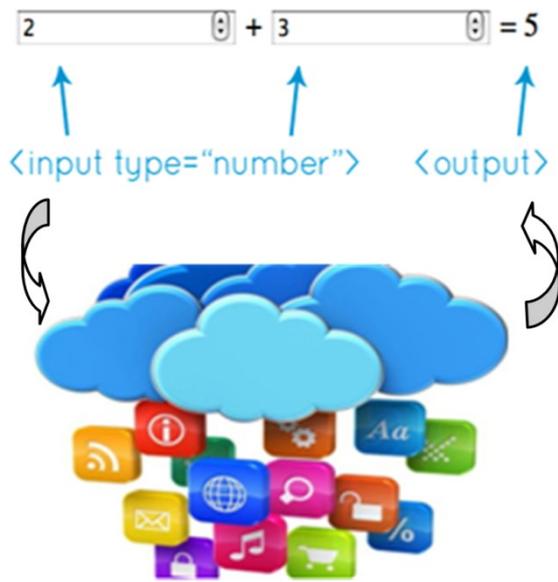
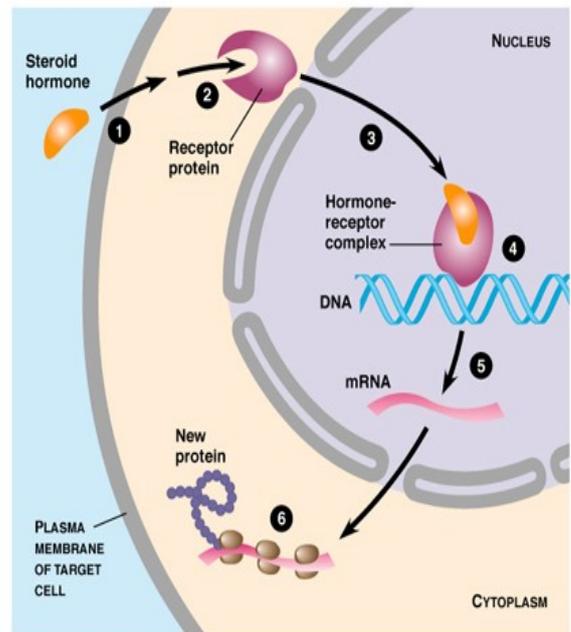


Figure 3.10 Example of a cloud service input and output

Body cell has a common property with Cloud service; it takes input and return output. The steroid hormone constitutes an input for a body cell in which it is fabricated in the nucleus and a new protein is fabricated as an output (see figure 3.11).



©1999 Addison Wesley Longman, Inc.

Figure 3.11 Human cell has inputs and outputs

Business Process Services make a stack of Cloud services and application available through Cloud providers. Most Cloud services are composite in which their business processes are composed of several subservices. As shown in the figure 3.12, a business process of a job application service calls out subservices like: Reject Application service

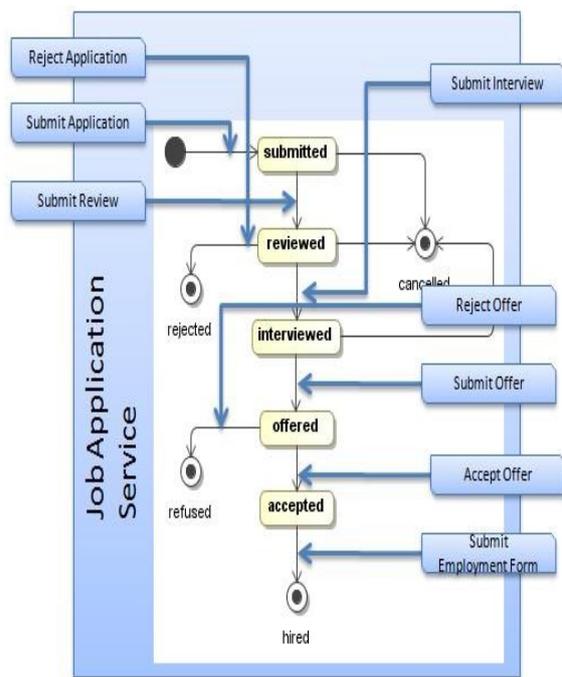


Figure 3.12 The job application service

The nucleus of a cell contains DNA, the genetic material of the cell (figure 3.13). “The DNA contains the information (genes) necessary for building the cell and directing the multitude of synthesis tasks performed by the cell in the process of life and reproduction” (Hickman *et al.*, 1995).

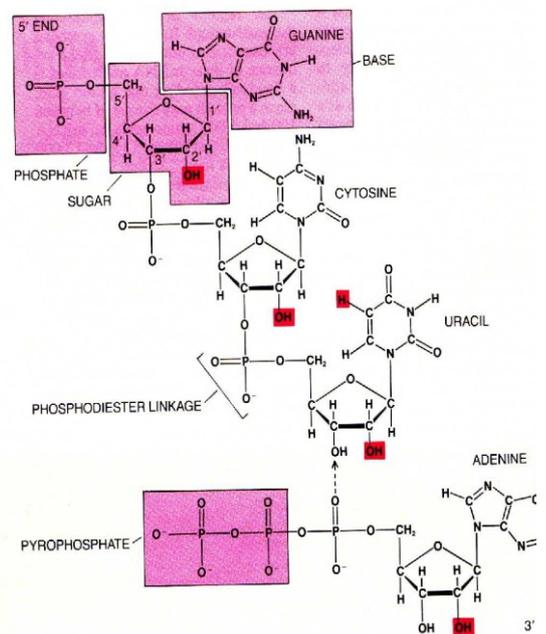


Figure 3.13 An example of DNA genetic material of a cell

As a result of comparing the current Cloud computing strategy to the human body strategy, we may conclude that the Cloud computing is a simulation of a body which

attacked by cancer. Indeed, all the organs in the normal body are directed by one manager which is the brain while in case of attacked by cancer some organs take their own decisions without following the brain's instructions. The same case appears in the Cloud computing world; there are several Cloud vendors (similar to body organs) which enlarged randomly through the web following their own decisions and there is no main common side (like the brain) to direct the services of these vendors. Therefore, Cloud's clients have to follow independently the rules and methods of various Cloud vendors to achieve a service. Also, each Cloud company tries to control web by building a huge infrastructures and poses millions of replicated services without knowing that their random tasks may cause death of the web.

On the way to improve the Cloud computing model we can benefit from the mentioned above common points with the brain models. But it is recommended to propose some approaches for the differences between these two models. These differences can be summarized as follows:

- The Cloud services are not managed by one central organization, but each group of services is directed by a specific company. Thus a lot of problems occur while discovering or composing a new Cloud service. The current way of the work of the Cloud services is similar to that of cancer cells inside the human body. These cells have its own management system which does not follow the brain commands and cause a random growth or death of body organs.

- There is a risky diversity of types of Cloud services. For example, a client could find himself in a place to select one of hundreds of services with similar quality. The problem is capable to be solved if Cloud architect follows the cell body strategy, in which the body has a definite number of cell types and under every one of these types a body cell can perform several jobs. Briefly, the body cell can be moderated with any new environment and for this purpose; it has already several actions compared to the quality of surround. Why not to make Cloud service moderated with client context and become capable to use internally all service process of similar type (for ex: hotel booking service) but appears as one unified type of service for the external clients.
- The recent history of Cloud companies proves that the Cloud services are of weak security properties. This is simply because the Cloud architect makes the client capable to access the Cloud vendor directly and invokes a service. Indeed, some of the security staffs of the previous web service model were destroyed by some hacker and the same thing happens in the Cloud paradigm because it follows the same standards. In the cell strategy, there is a block (cell membrane) prevent accessing the cell center (nucleus). To have a better security strategy for Cloud computing, why not to extend the Cloud architecture by adding a middle smart system, which can be invoked by client, instead of directly invoke Cloud servers.

The main goal of this work is to achieve a smart Cloud model similar in job to that of the brain model. But the existed approaches do totally support our goal, so we intend to develop a new paradigm of computing. The proposed cell oriented computing paradigm is a smart composition of several approaches to achieve a human cell simulation and it respects the web standards that Cloud service model follows.

3.4 SMARTCELLS APPROACH

As a step of adding intelligence to distributed computing systems, agent methodology proposed to send the whole object through distributed machines to be treated and produce a result. SmartCells approach offers solutions for some cloud problems like: availability cost of services and data, cloud security weakness, complexity of service composition, performance, service code validation, and service compatibility. SmartCells covers hybrid palette of methods and techniques derived from classical computational intelligence. SmartCells system is composed of a general *ecosystem* of components, not a point product or single vendor solution, and has basic, specific requirements to meet the intelligence needs of enterprise Cloud organizations. It is mainly based on building a standard center of instructions, known as a brain, which is capable to serve intelligently any type of request given by other machines (commanders) based on some criteria's. The SmartCells strategy is neither to do action via message transfer nor to send the complete object from one machine to another. It is simply following the robot strategy of work, briefly there are two types of system: commander and brain. The commander job is to order a service, while the brain has to provide an accurate and best selection of solutions.

In order to introduce the proposed cell theory, in the next sections we discuss the material of a new style of distributed computing: SmartCells approach then shows the basis of the Cell-Oriented computing strategy. For simplicity, we identify the architecture of the SmartCells as a Cell-Oriented Architecture (COA) and its functionality as Cell-Oriented Computing (COC) model.

3.4.1 CELL BASIS AND FOUNDATIONS

Cell theory is the modular representation of human cell characteristics from the perspective of computer science. It offers flexible and scalable virtual processing components that treat complex distributed computing smartly by controlled and precise decisions. A cell is a software object that:

- Is sited within a command/execution environment;
- Holds the following compulsory properties:
 - Collaborative: works in groups to finish a job;
 - Inheritance: serves clients according to their environmental profile if there is no specification in their requests;
 - Shares business processes: each cell business process represents a group of business processes of components with the same goal. However, every cell is open for collaboration with all other cells and can keep up best process quality via dynamic changes in process nodes. Thus, the cell has great processing power since all cells' business processes can be shared by one cell to serve the client;

- Uniqueness: each cell deals with a specific type of job;
- Reactive: cell senses modification in the environment and acts in accordance with those changes;
- Autonomous: has control over its own actions;
- Optimal: keeps to best functional and non-functional requirements;
- Federative: each cell has its own information resources;
- Self-error covering: monitors changes in the computing environment and applies improvements when errors are detected;
- Dynamic decision making: applies decision alteration based on the change of context;
- Learning: acclimatizes in accordance with previous experience;

Cell methodology uses commands among smart components: neither an invocation of non-smart component nor a migration of processes. It is based on cells that can benefit from the variety of already built Web components to achieve intelligent distributed computing. They have brains, decision support systems that can do the same jobs as a mobile agent. Furthermore, Cells has its own strategy to analyse and organize connections based on communications with the management and control center.

Cell methodology requires no discovery or selection steps to use a cell because it uses a new model of the composition process to realize the user's request. It participates in solving the big data problem by making a real time analysis of communications. It is

highly secure, since it uses a combination of context-aware and pervasive computing among cells.

CELL-ORIENTED COMPUTING

Cell-Oriented computing strategy allows sharing of the business processes to reach an output (Karawash *et al.*, 2015). This way of computing results, indirectly, in a shared resources environment similar to that of grid computing. Recursively, a client cell has access to all other executive cells as they are running on one machine. The cell network is organized, secure, reliable, scalable and dynamic. Cell computing strategy, as shown in Figure 3.15, is based on five main layers of computation: command layer, management layer, collaboration layer, analysis layer and feeding layer.

Command Layer: The command layer consists of proposals designed to make use of the smart selection of cells that can provide a specific service. It makes up the initial step of the exchange in cell architecture. An important role of the command layer is to allow for clear separation between available solutions and a logical methodology in producing a result based on the client's command.

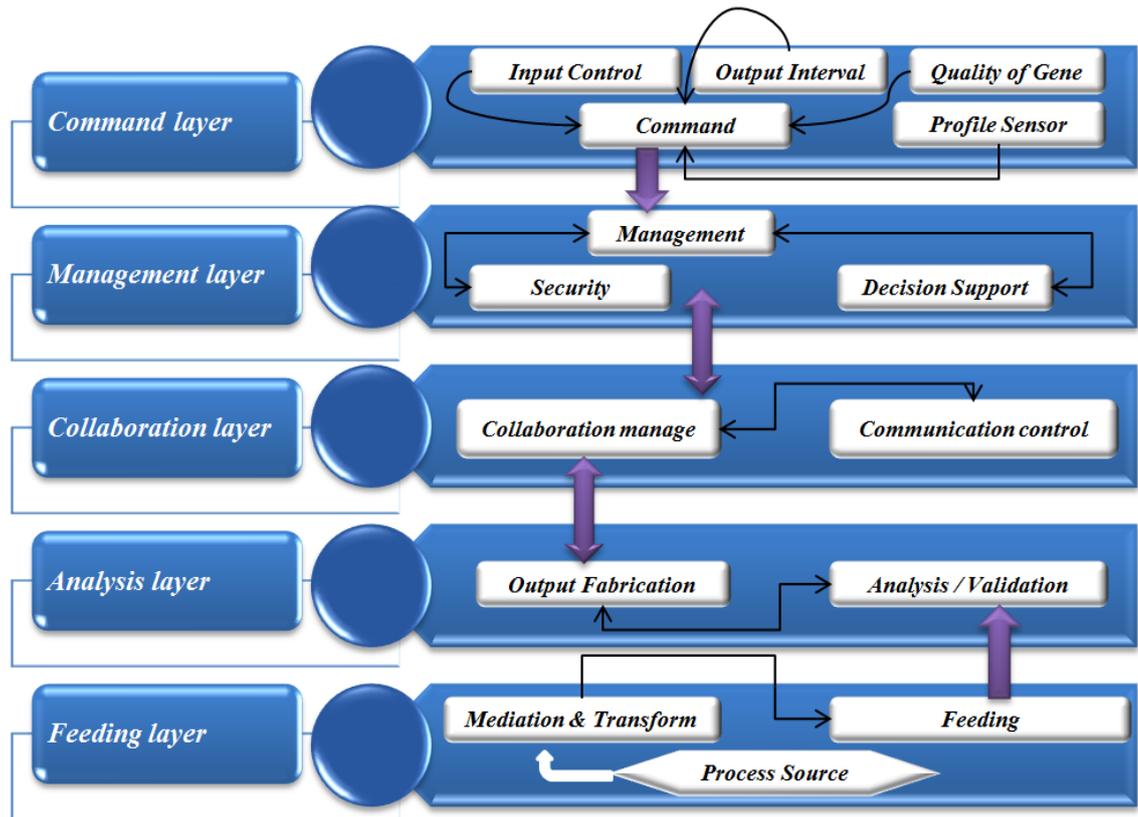


Figure 3.14 Strategy of Cell Computing

The traditional Web service methodology gives clients the right to select one of the pre-designed Web applications that will process their solutions depending on several qualities and a complex selection process. However, cell methodology has improved the process by making clients give commands and creating the application according to these commands. This approach enables a slew of new applications to be created that make use of the SmartCells's cooperative capabilities, without requiring in-depth knowledge of application processes, communication protocols, coding schemes or session management procedures; all these are handled by the upper layers of the cell strategy. This enables modular interfaces to incorporate new services via a set of commands composed of specifying inputs, output intervals, QoG requirements and the user profile.

Management Layer: this layer provides configurable controlling and reporting for client commands and server facilities at operational and services levels. It also provides visibility across physical, virtual-based layers, making it possible to govern the enforcement and migration of SmartCells across the distributed enterprise. The management layer of the cell-based architecture not only reduces deployment, maintenance and operation costs but also allows for the provision of better performance, scalability and reliability. Its agent-based capabilities provide for comprehensive management of all cell collaboration procedures. The management layer controls the start-up and status, the logging of maintenance events, the generation and processing of Cells, the supervision of security and the management of application failures.

The management layer supplies centralized way of control and monitoring every configured component, as well as activating and deactivating processes and single applications, including user-defined solutions. This layer additionally provides simple integration with a variety of enterprise-level business intelligence, reporting and event correlation tools for deeper analytics and insight. It automatically associates recovery method to results as active conditions in the system until they are removed by another maintenance event.

Collaboration Layer: in SmartCells, cells work with each other to perform a task and to achieve a shared goal. They utilize recursive processing and a deep determination to reach the client's objective. Most collaboration requires leadership; in SmartCells, each cell, by its decision system, can take the leading role. In particular, the collaborative property of the cells results in better processing power when facing competition for

complex jobs. SmartCells is based on specific rules of collaboration and manages the communications among cells. These rules characterize how a group moves through its activities. The desired cell collaboration aims to collect suitable sub-tasks that are composed to achieve a complete and efficient process in carrying out a specific job.

Analysis Layer: Through the analysis layer, providers of processes can be seen as a store of dynamic, organized quality of process, generating new cell processes. In this layer, the collected data of ontologies that represent business processes are analysed, validated and tested before operational use by cells. Two types of analysis are used in cell methodology. The first type studies the qualities of source processes; while the other type studies the graph analysis measures of the selected sub-processes.

Feeding Layer: this layer aims to find sources of business processes and tries to handle the complexity and diversity transforming business processes through a special mediator. The feeding process starts by fetching sources about process designs and results in a semantic design, as ontology, compatible with cell requirements.

Cell characteristics

Cells are smart components that combine a collection of characteristics from different environments. They apply autonomy and intelligence based on a mobile agent computational perspective. In addition, they map the human cell traits, such as inheritance and collaboration, into distributed computing. From the software engineering side, cells try to achieve best architecture properties such as security, availability, performance, etc.

Autonomy: The cell approach proposes that the problem space should be decomposed into multiple autonomous components that can act and interact in a flexible way to achieve a processing goal. Cells are autonomous in the sense that they have total control over their functions and have the right to take decisions without a third party intervention.

Inheritance: The commander cell inherits the profile property from its environment (company, university, etc.). However, the *Executer* cell can serve commanders according to their environmental profile (selection of suitable qualities of a process) or by special interference from the commander's side to specify more precisely the general design of a process and its qualities. This inheritance property in SmartCells is similar to the inheritance among generations of human beings. For example, babies inherit traits of their parents such that cells combine traits from the father and the mother, but the parent can ask a doctor for specific trait in a baby different from their own traits (blue eyes, brown hair, etc.). In this case, they have given more specifications to the cell in order to select suitable genes.

Internal Security: When application logic is spread across multiple physical boundaries, implementing fundamental security measures such as authentication and authorization becomes more difficult. In the traditional client-server model, the server is most responsible for any protection requirements. Well-established techniques, such as SSL (secure socket layer), granted a so-called transport level of security. Service and agent models emphasize the emplacement of security logic at the messaging level. Cell methodology applies an internal level of security in cells. Thus, command and

executive cells can communicate after the protection steps summarized by verifying the context profile of the cell that requests collaboration.

Availability: Availability of cells and data is an essential capability of cell systems; it is actually one of the core aspects giving rise to cell theory in the first instance. The novel methodology of cell theory decreases the redundancy of servers to ensure availability. Its strength lies in the ability to benefit from the redundancy of processes that serve similar goal, so failures can be masked transparently with less cost.

Collaborative: Collaborative components are need in today's primary resources to accomplish complex outcomes. Cell methodology depends on collaboration-by-command that enables coordination by one of the collaborative components. Collaboration allows cells to attain complex goals that are difficult for an individual cell to achieve. The cell collaborative process is recursive: the first collaborative agent makes a general command that is passed gradually through collaborative cells to more specific cells until reaching the desired results.

Performance: Distributed computational processes are disjointed; companies' coding is not ideal and it is difficult to monitor the complexity of every process. Thus, performance problems are widely spread among computational resources. Cell theory introduces an approach for performance problems in a distributed environment. The approach can be summarized as applying a permanent analysis of different processes aiming for the same goal, attached to a unified cell, then selecting the best process to do a job, based on basic properties such as response time and code complexity.

Furthermore, an increase in communication acquaintance can be a guide to an improvement in performance as it enables cells to communicate in a more efficient manner.

Federation: Cells are independent in their jobs and goals. However, all distributed processes that do same type of job are connected to a specific executive cell. Thus each executive cell is federated with respect to the commander cell's request. Cells map can be considered as a set of federated components that are capable of collaborating to achieve an output.

Self-Error Cover: There are two types of errors that can be handled by cell computing: structural and resource errors. The cell process is based on a combination of codes that are fabricated by different computational sides. These combinations may fail because of coding or system errors and fall in deadlock. The process validation system's job is to monitor changes in process and recover errors if detected. Resource errors are described as failure in providing a service from the computational resource. The proposed approach to these types of error is to connect spare procedures in each cell process to achieve the same quality of job from different sources.

Interoperability: Cell interoperability comes from the ability to communicate with different feeding sources and transform their business processes into cell business processes. For example, in spite of differences among business processes, such as BPEL and OWL-S, every provider of service is seen as a source of genes and as useful in cell computing. Based on cell interoperability, all procedures and applications used

by service providers can be unified under a unique type of process computing, the cell gene, with respect to cell provider.

3.4.2 SOFTWARE ARCHITECTURE AND REQUIRED INFRASTRUCTURES

Intelligent distributed computing is expected to create special challenges of adaptation and productive combination of results of several areas with a great impact on launching a new generation intelligent distributed information systems (Karawash *et al.*, 2015). The Cloud theory adopts the service concept when dealing with all the web resources such as: application as a service, platform as a service, etc. The predicted web is a smart and semantic web while the Cloud model lacks intelligence and autonomy. Also the service model faces some problems regarding reusability and security, which affect the Cloud negatively. Thereby, this project proposes to replace the traditional service-oriented concept by a cell-oriented concept without altering the Cloud service standard communication technologies (such as SOAP, XML, etc.).

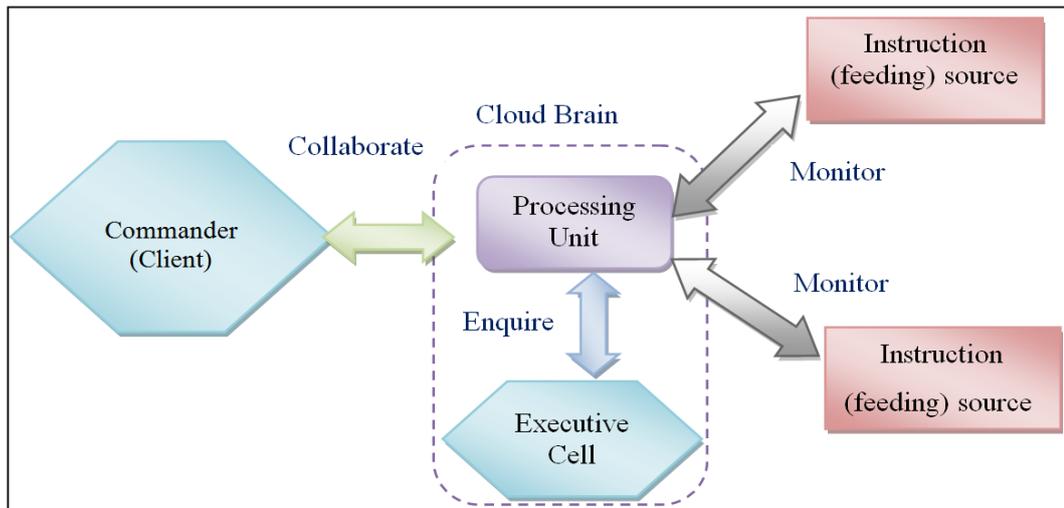


Figure 3.15 SmartCells Architecture

Figure 3.14 outlines the components of the architecture of SmartCells, its functionality and the operation of the underlying proprieties. It is composed of four main components: Commander Cell, Executive Cell, Instruction source, and Cloud Brain.

Commander (Client) Cell: This is a client side component that looks for a procedural module to accomplish a required function. The commander can be an application, another service, or some other type of software module that needs the service. The commander cell works like a brain cell in the human body; it deals with definite problems and suggests a general view of the solution to be realized by a specific type of executive cells.

Executive Cell: These Cells are intelligent components that are ready to serve commander Cells. Each Cell is characterized by: uniqueness of goal, self-governance, federated role, internal security, and interoperability. Cell business processes, which are called genes, are built directly by the cell designer or else and can be transformed by any

type of service business processes. Genes use the ontology of an abstract business process and link different processes with the same purpose into a specific node. Similar to the gene in human body, each artificial gene serves a specific type of job in a different style and no other gene is capable of doing the same job. Based on genetic characteristics, an executive Cell is unique in delivering a specific type of service; for example, if a client's cell requires a hotel service, there is only one, replicated, executive hotel cell to be invoked. Cell theory maintains diversity and competition between companies to serve clients; however, it hides complexity issues when selecting or composing Web processes. Outputs are prepared in an autonomic manner without any interference done by the client; that is why there is no complex discovery and selection of cells or processes of composition or intervention.

Instruction (Feeding) Sources: These are pre-built components used in building the genes of executive cells. The feeding source can be a Web service provider, a company, or any third party that is capable of supplying a process design. From the SOA perspective, these sources work like a service provider of instruction. These sources of instructions are network target elements that are hidden from Commander Cells and they supply Cloud Brain by processes to perform suitable for Executive Cell functions. They deliver definite service process descriptions and implementations. The Cloud service provider or other type of software system can be a component that fulfills the Cloud Brain by instruction. A cell's internal system can use a pre-designed business process or requires the building of new designs by process designers, making it suitable to be a cellular gene.

Cloud Brain (Executive Cell provider): It consists of a defined number of components that monitor and direct executive cells and make them up-to-date in serving Commander Cells. It follows every connection between cells and prepares all decisions, such as update requirement, communication logics, maintenance facilities, access control management, repository stores and backups, etc. The proposed brain could understand the complex associations of ongoing multidimensional changes in dynamics Cloud service operations with cognitive state. In other words, to optimally imitate the brain activities, it is important to take into consideration, as much as possible, the behavior that the brain is controlling. In the next chapter, we introduce in details the structure of the proposed Cloud brain.

SmartCells is mainly based on the imitation of the human cell methodology of work. It adds some properties as intelligence and autonomy to the known service-oriented characteristics and reaches a new homogenous system, the Cell system. The next section discusses the requirements and mechanisms of SmartCells.

SmartCells is a novel software design principle targeted generally at Web resource computing devices. The architecture allows users to engage in smart collaborations among devices during Web resource invocations. SmartCells is based on a center of intelligence, which collects cells in order to exchange data between participants and manage organized standard communication methods to obtain information. The architecture is designed to achieve smart Web goals and overcome the limitations of existing Web infrastructures. The cell architecture presented here is device, network and provider independent. This

means that SmartCells works across most computing machines and ensures a novel methodology of computing.

SmartCells is designed to cater to smart Web requirements and aims to achieve finally an ambient, intelligent Web environment. Cells in SmartCells are internally secured, sustain autonomic analysis of communications and are able to support the mechanism of collaborations through the following requirements:

- [R1]** Management & Communication: to establish local and remote sessions, the underlying infrastructure provides the ability to find any other cells in the network and then to establish a session with that cell.
- [R2]** Context-based Security: to enable secure interactions in the communication spaces among all connected participants.
- [R3]** Analysis: supporting analysis of data exchange among cells, plus encompassing the interior analysis of cell process infrastructure.
- [R4]** Validation: to verify cell components and ensure consistent process combinations among cells.
- [R5]** Output Calculation: to evaluate the suitable output results with less cost and minimal use of resources.
- [R6]** Trait Maintenance: to avoid and deal spontaneously with all sources of weakness in cells' communications.

To realize these goals, we developed a complete command-execute architecture, designed from the ground up to work over existing Web standards and traditional networks. SmartCells makes it possible to merge the material and digital worlds by incorporating physical and computing entities into *smart spaces*. Put simply, it facilitates the steps to achieving a pervasive form of computing. Cell theory is introduced to provide intelligence in distributed computing; however, it combines client/server and peer-to-peer

models at once. On one side, Cells follow a client/server representation because it presents two main components, a client component (the commander cell) and a server component (the *Executer* cells) to solve a problem. On the other hand, virtually, it is an illustration of peer-to-peer applications because we have two types of types of Cells communicating with each other.

3.5 CONCLUSION

“The rapid development of processing and storage technologies leads the internet resources to become cheaper, more powerful and more ubiquitously available than ever before. These technological progresses have enabled the realization of new computing models” (Zang et al., 2010). During the service model revolution, a group of weak points was discovered and marked as open problems such as: service composition, discovery, selection and security. When service model was adopted by Cloud computing, the service problems were transferred to the new computing method. This chapter opens the door for a new concept of Cloud modelling toward solving the Cloud problems. The main work in this chapter is to show the SmartCells architecture and to demonstrate its importance for the Cloud maintenance.

Chapter 4

CELL OPERATIONAL MODE

Distributed computing systems are of huge importance in a number of recently established and future functions in computer science. For example, they are vital to banking applications, communication of electronic systems, air traffic control, manufacturing automation, biomedical operation works, space monitoring systems and robotics information systems, and many more. As the nature of computing comes to be increasingly directed towards intelligence and autonomy, intelligent computations will be the key for all future applications. Building an intelligent style of distribution that controls the whole distributed system requires communications that must be based on a completely consistent system. We believe that human body system could be a good solution to build an intelligent distributed system, specifically the body's cells. As an artificial and virtual simulation of the high degree of intelligence that controls the body's cells, this chapter proposes a cell-oriented computing model, as an approach to achieve the desired intelligent distributed computing system. The components of SmartCells architecture are described and discussed in details in this chapter.

4.1 INTRODUCTION

Distributed computing (DC) is the consequence of permanent learning, the improvement of experience and the progress of computing knowledge. It offers advantages

in its potential for improving availability and reliability through replication; performance through parallelism; sharing and interoperability through interconnection; and flexibility and scalability through modularity. It aims to identify the distributable components and their mutual interactions that together fulfil the system's requirements.

With the extensive deployment of DC, the management, interoperability and integration of these systems have become challenging problems. Investigators have researched and developed important technologies to cope with these problems. One of the results of the continuous evolution of DC in the last decade is the service-oriented computing (SOC) paradigm, which offers an evolution of the internet-standards based DC model, an evolution in processes of architecting, design and implementation. The other key result is the mobile agent computing paradigm, which provides an alternative computing paradigm to the traditional client-server paradigm. Moreover, the latest DC technology is expressed by Cloud computing, which evolved from grid computing and provides on-demand resource provisioning. Grid computing connects disparate computers to form one large infrastructure, harnessing unused resources.

Trends in the future of the Web require building intelligence into DC; consequently the goal of future research is intelligent distributed computing (IDC). The emergent field of *IDC* focuses on the development of a new generation of intelligent distributed systems. *IDC* covers a combination of methods and techniques derived from classical artificial intelligence, computational intelligence and multi-agent systems. The field of DC predicts the development of methods and technology to build systems that are composed of collaborating components.

Building a smart distributed model that controls the whole of Web communications needs to be based on an extremely consistent system. The ultimate system that can be adopted in building IDC is the model of the human body system, specifically the body cell. Based on the high degree of intelligence that controls body cells, this chapter shows the components of the SmartCells.

4.2 STRUCTURE OF SMARTCELLS COMPONENTS

SmartCells is composed of three main components: Commander Cell, Cloud Brain and Cell Instructions Source. Cell theory is introduced to provide intelligence in distributed computing (Karawash *et al.*, 2015).

4.2.1 COMMANDER CELL STRUCTURE

The commander (Client) cell represents the client side in SmartCells and is the main requester of an output. This section discusses the structure of cells from the client side (Figure 4.1).

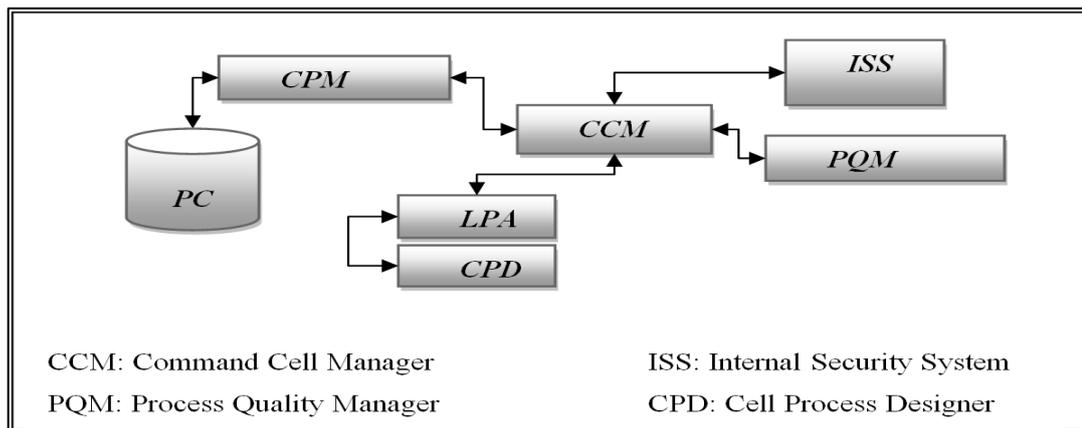


Figure 4.1 Structure of Client Cell

Command Cell Manager (CCM): the client cell's 'head' that is responsible of any external collaboration with the executive cells. It receives a client as a list of four components: proposed cell input, interval of output of executive cell result, proposed cell process's general design (if available) and the required cell process quality. Some of these components can be inherited from the client cell's environment. The command cell manager monitors the context profile of the commander cell via the profile manager. It also manages the access to the client cell by specified rules of internal security.

Internal Security System (ISS): this is protection software that is responsible of giving tickets for executive cells to access the command cell manager. It depends mainly on the analysis of the outer cell's context profile to ascertain whether it can collaborate with the client cell.

Process Quality Manager (PQM): software used by the commander cell to select the required quality of the cell process. For example, the client may need to specify some qualities such as performance, cost, response time, etc. If there is no selection of specific qualities, these qualities are inherited from the environment's qualities (as an employee may inherit a quality from his company).

Cell Process Designer (CPD): a graphical design interface that is used to build a general cell process flow graph or to select an option from the available process graphs. If there is no graph design or selection, the executive cell has the right to pick a suitable gene based on the commander profile.

Logic Process Analyser (LPA): after designing a general proposition for the executive gene design via the process designer, the job of the logic process analyser is to transform the graph design into a logical command to be sent to the executive side.

Context Profile Manager (CPM): this tool is responsible for collecting information about the commander cell profile, such as place, type of machine, user properties, etc. Since the commander profile is dynamic, several users may use the same commander cell; the profile information is instantaneously provided when needed.

Profile Core (PC): this storage is performed by a special database that stores information about the commander cell profile and allows the executive cell to tell whether there are several users utilizing the same commander cell.

4.2.2 CLOUD BRAIN STRUCTURE

This section discusses the structure of the Cloud Brian (Cell provider), as shown in in Figure 4.2.

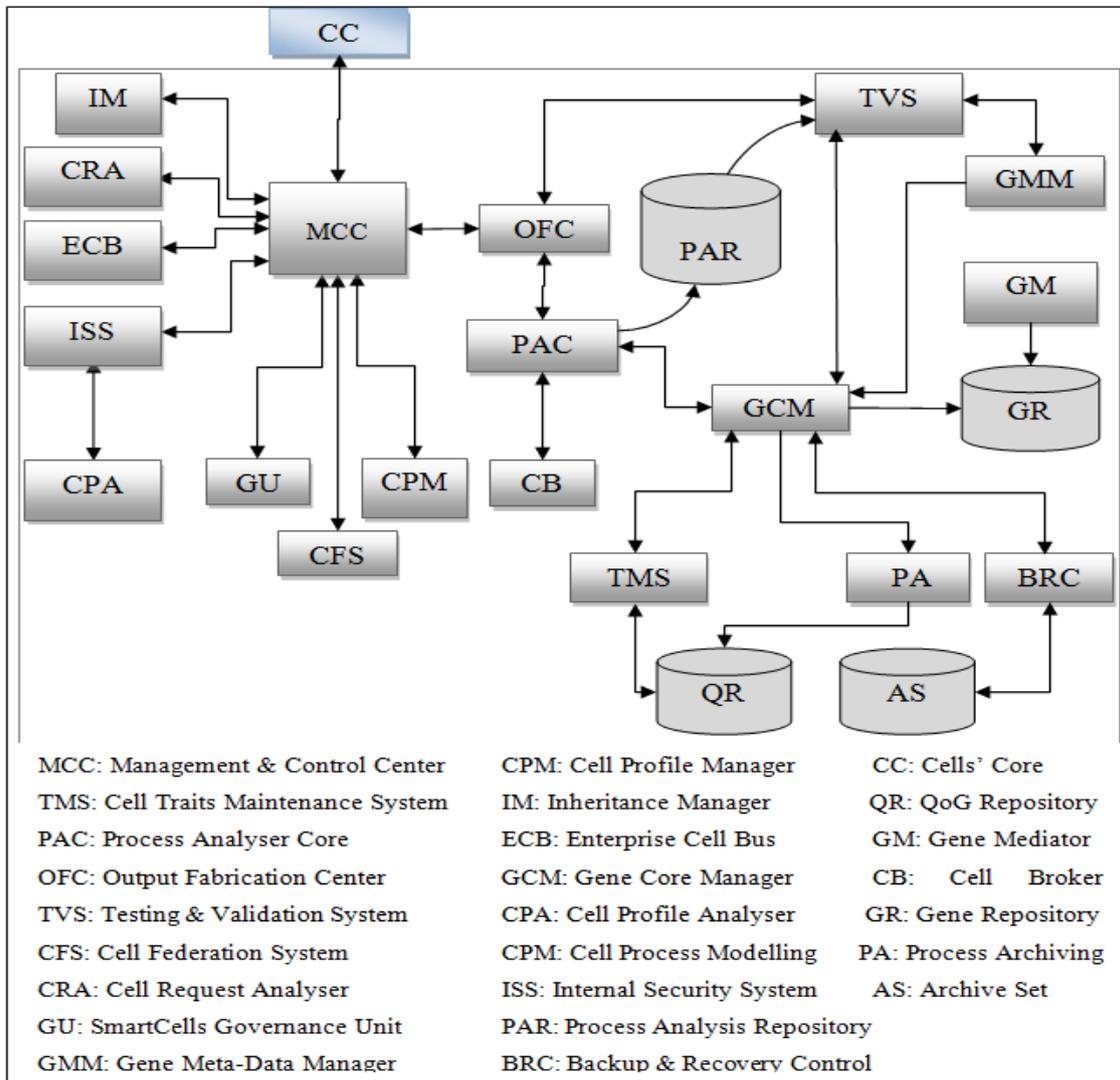


Figure 4.2 Structure of Cell Provider (Cloud Brain)

Management & Control Center (MCC): Smart software works like an agent and is considered to be similar to the brain of the SmartCells, in which it orchestrates the whole computing infrastructure. It is composed of a virtual processing unit that controls all the

internal and external connections. So, executive cells are supported and managed according to well-defined cell level agreements. It monitors every connection among cells and prepares all decisions, such as update requirement, communication logics, maintenance facilities, access control management, repository stores and backups, etc.

The SmartCells management and control center have stable jobs inside the cell provider. However, it cannot respond to an external job from other cells without security permission from the internal security system. Since one of the main principles of cell theory is availability, the management and control center is replicated in order that collaboration can be carried out to serve cells. Each cell uses its decision system to communicate with the SmartCells management center.

Testing & Validation System (TVS): the cell testing and validation system describes the testing of cells during the process composition phase of the executive cell. This will ensure that new or altered cells are fit for purpose (utility) and fit for use (warranty). Process validation is a vital point within cell theory and has often been the unseen underlying cause of what were in the past seen as inefficient cell management processes. If cells are not tested and validated sufficiently, then their introduction into the operational environment will bring problems such as loops, deadlocks, errors, etc.

In a previous book chapter (Karawash *et al.*, 2013), we have discussed a new model of how to validate the business processes of Web service; the concepts of the same validation method can be used to validate the cell business process (Gene). Cell validation and testing's goal means that the delivery of activities adds value in an agreed and expected manner.

Cell Traits Maintenance System (TMS): the challenge is to make cell technology work in a way that meets customer expectations of quality, such as availability, reliability, etc., while still offering executive cells the flexibility needed to adapt quickly to changes. Qualities of genes are stored in a *QoG* repository and the maintenance system has permission to access and monitor these qualities. *QoG* can be considered a combination of *QoS* with a set of Web services if the source of the cell is a Web service provider. *QoG* parameters are increasingly important as cell networks become interconnected and larger numbers of operators and providers interact to deliver business processes to executive cells.

Process Analyser Core (PAC): since a cell process map can be composed of a set of other components' business processes, there should be a method for selecting the best direction for the cell map. In addition to the context of environment dependency, cell theory uses a deep quality of service analysis to define a best process. This type of process map analysis is summarized by building a quality of process data warehouse to monitor changes in process map nodes. Every process component invokes a set of subcomponents, similar to sub services in a service model, in which all these subcomponents are categorized in groups according to goals. The process analyser core applies analysis to these subcomponents and communicates with the cell broker to achieve the best map of the executive cell process. In addition to analysing the executive cell process, the process analyser core also analyses and maps the invocations from the commander cells. This type of dual analysis results in an organized store of collaboration data without the need to re-analyse connections and without major data problems.

Output Fabrication Center (OFC): depending on the specific output goal, options may be available for executive cells to communicate with the output fabrication center. This center provides more control over the building of the executive cell process to serve the client cell. Based on the results of the process analyser core and the consequences of the test and validation system, executive cells, specifically their output builder systems, collaborate with the output fabrication center to return a suitable output to the commander cell.

Cell Profile Manager (CPM): traditional styles of client/server communications suffer from a weakness: the dominance of the provider. Indeed, a server can request information about client profiles for security purposes, but power is limited in the converse direction. In cell theory, every cell must have a profile to contact other cells. The cell profile manager works to build suitable profiles for executive cells to help in constructing a trusted cell instruction tunnel.

Cell Federation System (CFS): the system coordinates sharing and exchange of information which is organized by the cells, describing common structure and behavior. The prototype emphasizes the controlled sharing and exchanges of information among autonomous components by communicating via commands. The cell federation system ensures the highest possible autonomy for the different cooperating components.

Cells' Core (CC): this forms a center of executive cells. A cell is an item of smart software that performs a specific type of job. All cells have the same structure but different processes. Thus, the executive cell is considered an example of a general cell component. Each executive cell is composed of seven sub-components, as follows: decision system,

gene store system, trait maintenance system, output builder system, process validation system, process analyser system, defence system and gene storage. These sub-components communicate with the cell provider subsystems to carry out their jobs.

Inheritance Manager (IM): a client is observed as a commander cell so as to decide which types of cell inherit the properties of their environment. For example, if the commander is a professor, they can be seen a part of a university environment by executive cells. A commander can be part of more than one environment; and results in a hybrid profile of context. The inheritance manager maps the commander cell to its suitable environment. To serve a commander, the executive cell uses a quality of process compatible with its surroundings or follows the commander's requirements to build a suitable process.

Cell Request Analyser (CRA): cell theory is based on the concept of collaboration to serve the client. However, every client has a different request, so a computing component is needed to detect which cells will work in generating the answer. In general, the job of the cell request analyser is to map the commander cell to the appropriate *Executer* cells to accomplish a job.

Cell Profile Analyser (CPA): this component is related to the security of cells. One of the main concepts of cell theory is its context-based property. There are sensors for profile context collecting information about the commander at the client side. The cell profile analyser verifies the commander profile by a specific method before allowing access to *Executer* cells.

Internal Security System (ISS): since some commander cells can access sensitive data, stringent protection must be provided from the server side. The available security methods follow two types of protection: network and system protection. In network protection, the data among nodes is encrypted to hide the content from intruders. In system protection, a token (username and password), antivirus application and firewall are used. Cell theory proposes a new type of protection which is specific to the application itself. It is described as an internal system protection that verifies the profile of the user by several methods before allowing access.

Cell Process Modelling (CPM): a procedure for mapping out what the executive cell process does, both in terms of what various applications are expected to do and what the commander cells in the provider process are expected to do.

Enterprise Cell Bus (ECB): The enterprise cell bus is the interaction nerve core for cells in cell-oriented architecture. It has the propensity to be a controller of all relations, connecting to various types of middleware, repositories of metadata definitions and interfaces for every kind of communication.

Cell Broker (CB): analytical software that monitors changes in cell processes and evaluates quality of processes according to their modifications. The evaluation of quality of process is similar to that of quality of service in the service model. However, the new step can be summarized as the building of a data warehouse for quality of process that permits an advance online process analysis.

QoG Repository (QR): a data warehouse for the quality of cell process. It collects up-to-date information about process properties, such as performance, reliability, cost,

response time, etc. This repository has an *OLAP* feature that support an online process analysis.

SmartCells Governance Unit (GU): the SmartCells governance unit is a component of overall IT governance and as such administers controls when it comes to policy, process and metadata management.

Process Analysis Repository (PAR): a data warehouse of all cells' process connections. It stores information about cell processes in the shape of a network graph, in which every sub unit of a process represents a node. The collected data summarizes analytical measures such as centrality.

Gene Core Manager (GCM): software responsible of gene storage, backups and archiving. It receives updates about business processes from sources and alters the gene ontology, backs up the gene when errors occur and archives unused genes.

Gene Mediator (GM): the problem of communication between the gene core manager and the sources of business processes may be complex, so GM defines an object that encapsulates how a set of objects interact. With the gene mediator, communication between cells and their sources is encapsulated by a mediator object. Business process sources and cells do not communicate directly, but instead communicate through the mediation level, ensuring a consistent mapping of different business process types onto the gene infrastructure.

Gene Meta-Data Manager (GMM): genes are complex components that are difficult to analyse, so for analysis and validation purposes, the gene meta-data manger

invokes gene meta-data from the gene repository and supplies gene core data through this process.

Gene Repository (GR): ontologies are used as the data model throughout the gene repository, meaning that all resource descriptions, as well as all data interchanged during executive cell usage, are based on ontologies. Ontologies have been identified as the central enabling technology for the Semantic Web. The general use of ontologies allows semantically-enhanced information processing as well as support for interoperability. To facilitate the analysis of the gene map, meta-data about each gene is also stored in the gene repository.

Backup & Recovery Control (BRC): this refers to the different strategies and actions occupied in protecting cell repositories against data loss and reconstructing the database after any kind of such loss.

Process Archiving (PA): the archiving process helps to remove the cell process instances which have been completed and are no longer required by the business. All cell process instances which are marked for archiving will be taken out from the archive set database and archived to a location as configured by the administrator. The job of the process archiving component includes the process-, task- and business log-related content from the archive database.

Archive Set (AS): a database for unused genes that is accessed and managed by the process archiving component.

4.2.3 CELL SOURCE

Cell source can be any kind of code that can be reused and follows specific composition rules. Generally, the first sources of cells are Web service business processes (such as BPEL and OWL-S) or reusable code (Java, C# etc.). This section discusses the structure of the sources that feed executive cells (Figure 5.3).

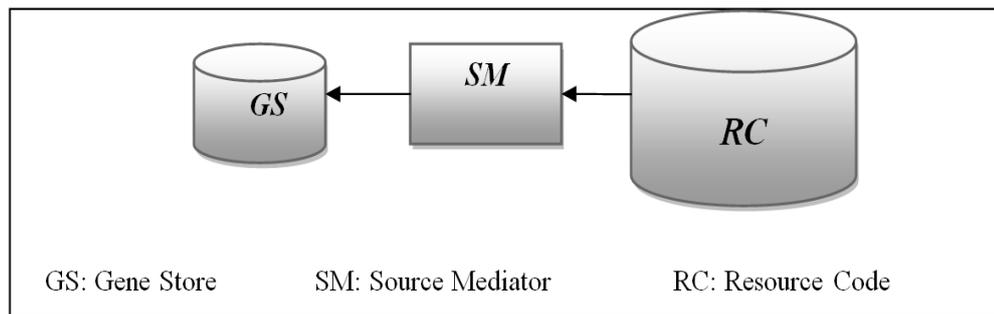


Figure 4.3 Structure of Cell Source

Resource Code (RC): a store of cell sources, such as business processes or reusable code. If the cell source is a Web service provider, then its business process may be BPEL, OWL-S, or another. Further, the cell source may be a reusable programming code for a combination of objects (in Java, C#, etc.).

Source Mediator (SM): transformer software that maps the process of a cell's source into a gene. The mediator's job is similar to that of the BPEL parser in a Web service provider, which maps BPEL code into a WSDL code. In SmartCells, every source business process is converted into OWL-S ontology. However, the obtained OWL-S ontology has a special property: the extension of OWL-S' business process.

Gene Store (GS): a store that is composed by mapping the source business process. This is an abstract of a source process in shape of ontology, organized in a structure compatible with the cell's job.

4.3 DEFINITIONS AND NOTATIONS

Definition 1 Let $W(P, Q, T)$ be a finite nonempty set that represents Web infrastructure, where: $P = \{p_1, p_2, \dots, p_n\}$ represents the set of feeding sources of Web applications, $Q = \{q_1, q_2, \dots, q_m\}$ represents the set of consumers of Web sources and $T = \{t_1, t_2, \dots, t_k\}$ represents the set of tools that are used by Web providers to serve Web customer, where $n, m, k \in \mathbb{N}$.

Definition 2 Let set $J = \{\cup_m^z j_m / j_m \text{ is specific goal and } j_m \neq j_n\}$ and set $S = \{\cup_m^z s_m / s_i \text{ is structure of component}\}$.

As with most things in the business world, the size and scope of the business plan depend on specific practice. A specific practice is the description of an activity that is considered important in achieving the associated specific goal. Set J represents a group of components, each of which supports a specific computing goal based on a particular practice. However, the structure of the studied components is denoted by set S .

Proposition 1 A set $\sigma = \{\cup_i^n L_i / \sigma_i \text{ denote a Cell}\} \subset T$, is a finite and ordered set such that $J_{\sigma_i} \cap J_{\sigma_j} = \emptyset$ and $S_{\sigma_i} = S_{\sigma_j}$, where $i, j, n \in \mathbb{N}$.

In all other computing models, different components may perform similar jobs. For example, two classes, in the object-oriented model, can utilize similar inputs and return the

same type of output but using different coding structures. Furthermore, in the discovery phase of service-oriented computing, service consumers receive a set of services that do the same job before selecting which one of them to invoke.

The main advantage of Web service theory is the possibility of creating value-added services by combining existing ones. Indeed, the variety involved in serving Web customers is useful in that it gives several aid choices to each one of them. However, this direction in computing failed since service customers found themselves facing a complex service selection process. One of the main properties of cell methodology is the avoidance of the ‘service selection’ problem.

The cell model is developed to provide highly focused functionality for solving specific computing problems. Every cell has its own functionality and goal to serve, so one cannot find two different cells which support the same type of job. However, all cells are similar in base and structure: they can sense, act, process data and communicate. That is to say, regarding cell structure there is only one component to deal with, while in function there are several internal components, each with a different computing method and resource.

Definition 3 Let φ be a property that expresses the collaboration relation such that $\alpha\varphi\beta$ where $\alpha, \beta \in \sigma$.

Business collaboration is increasingly taking place on smart phones, computers and servers. Cells in COC are intelligent components that are capable of collecting information, analysing results and taking decisions and identifying critical Web business considerations in a collaborative environment.

Proposition 2 A collaboration relation φ defined on the set σ is transitive, in which, if $\alpha\varphi\beta$ and $\beta\varphi\gamma \Rightarrow \alpha\varphi\gamma$, where $\alpha, \beta, \gamma \in \sigma$.

Transitive structures are building blocks of more complex, cohesive structures, such as response-cliques, which facilitate the construction of knowledge by consensus (Aviv *et al.*, 2003). The collaboration among cells follows a transitive mechanism to provide consistency. Transitivity among cells can be summarized by this example: if we consider three cells X, Y, Z and if X collaborates with Y, Y collaborates with Z, then indirectly X collaborates with Z.

Proposition 3 $\forall c_i \in \sigma \ \& \ \forall q_e \in Q, \ \exists \sigma_i, \sigma_j, \dots, \sigma_n$ s.t. $\cup_{i,j}(\sigma_i\varphi\sigma_j) \Rightarrow q_e$, where $i, j, e, n \in IN$.

COC's goal is to be introduced to serving Web customers with minimal cost, lower resource consumption and optimal results. For every customer request (t_e), there exists a cell collaboration ($\cup_{i,j}(\sigma_i\varphi\sigma_j)$) to return the appropriate answer. Cell collaboration is dynamic; results are produced without delay. Any future error in the proposed results generated by COC is corrected by an automatic repairing mechanism.

Definition 4 (cell subsystems) An executive cell system is an ordered set $C = (DS, GSS, TMS, OBS, PVS, PAS, DFS)$ such that:

- DS set builds and manages cell decisions.
- GSS set is responsible for cell process storage.

- *TMS* set monitors the cell's characteristics.
- *OBS* set maintains best output results of cells.
- *PVS* set is responsible for cell process validation.
- *PAS* set analyses the cell's business process.
- *DFS* set is responsible for cell security.

Proposition 4 A relation between cell subsystems is managed according to a set of mathematical mappings $M (\mu, \pi, \rho, \tau, \gamma, \delta, \varepsilon, \theta, \vartheta)$ such that:

$$\begin{array}{l} \mu: DFS \rightarrow DS \\ x \rightarrow \mu(x) \end{array} \quad \text{F.1}$$

$$\begin{array}{l} \pi: OBS \rightarrow DS \\ x \rightarrow \pi(x) \end{array} \quad \text{F.2}$$

$$\begin{array}{l} \rho: TMS \rightarrow OBS \\ x \rightarrow \rho(x) \end{array} \quad \text{F.3}$$

$$\begin{array}{l} \tau: PAS \rightarrow OBS \\ x \rightarrow \tau(x) \end{array} \quad \text{F.4}$$

$$\begin{array}{l} \gamma: PVS \rightarrow OBS \\ x \rightarrow \gamma(x) \end{array} \quad \text{F.5}$$

$$\begin{array}{l} \delta: GSS \rightarrow TMS \\ x \rightarrow \delta(x) \end{array} \quad \text{F.6}$$

$$\begin{array}{l} \varepsilon: GSS \rightarrow PAS \\ x \rightarrow \varepsilon(x) \end{array} \quad \text{F.7}$$

$$\begin{array}{l} \theta: GSS \rightarrow PVS \\ x \rightarrow \theta(x) \end{array} \quad \text{F.8}$$

Theorem: *If q denotes a commander cell request and x denotes a cell gene,*
then:

$$\mu(q) \equiv \pi(\rho(\delta(x)) \cap \tau(\varepsilon(x)) \cap \gamma(\theta(x))).$$

The management of a cell's internal system is divided among its subsystems according to a definite number of roles. In order to invoke a cell, a client request (q) must pass the cell's security system (F.1). After ensuring a secure cell invocation, DS begins the response process. It demands building output by the OBS (F.2). OBS output is based on a deep cell process analysis (F.4), a precise cell process validation (F.5) and assessing relevant cell characteristics (F.6). Tests (analysis and validation) are applied to cell process storage through GSS (F.6, F.7 and F.8).

4.4 COMPONENTS OF EXECUTIVE CELL

The proposed executive cell in cell theory is composed of (Figure 4.4): decision system (DS), gene store system (GSS), trait maintenance system (TMS), output builder system (OBS), process validation system (PVS), process analyser system (PAS), defence system (DFS) and gene storage.

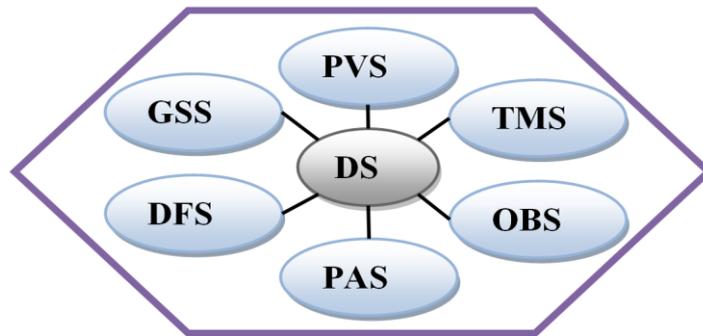


Figure 4.4 Components of Executive Cell

4.4.1 DECISION SYSTEM (DS)

The decision system is the brain of the cell in COC. It is controlled by the management and control center and is responsible for taking decisions and directing other components of the cell. Cell inputs are received by the DS which study the client request and emit suitable outputs. Cell computing is characterized by two levels of collaboration that are managed through DSs.

The first collaboration level is expressed by internal cooperation among cell subsystems, while the second level of collaboration is applied among cells to build a complete answer for cell customers. In the case of a customer request, the DS asks the defence system to verify the customer identity and request before starting the answer process. If the customer request is safe, DS sends the input to the OBS and waits for the answer. Sometimes, one cell is not sufficient to serve a customer. In this case, the DS asks for collaboration from other cells to produce an answer.

4.4.2 DEFENSE SYSTEM (DFS)

Cell computing aims to decrease the number of problems resulted from the adoption of the service model. One of the main service-oriented computing problems is security. Security weakness is less of a danger in the case of Web service, but currently most Cloud services are public and store sensitive data, so that any security fault may be fatal to some institutions. As a way of obtaining strict computing resource protection, COC introduces internal cell protection. As is well known, there are two main steps to protecting the Web. The first step is to achieve a network protection via several encryption methods. However, the second step is characterized by server resources protection via user tokens and security tools. Indeed, regarding security any distributed system is affected by the environment and architect job is to decrease the degree of loss in security and to remove it absolutely. For example, the migration of processes and the control of that migration, together with their effect on communication and security, was a problem for mobile agents. In case of intelligent distributed system the security risk decreases because the user action could be given a strict level of administration. But in case of agent giving an application the ability to move among distributed systems and choose the place to make execution may cause critical security problems. Agent methodology has several advantages; however, it can destroy human control if it is not covered by rules and limits.

The proposed COC security technique ensures protection against any internal or external unauthorized access to a cell. In addition to network and system protection, the cell defence system aims to introduce a double verification method. This is a hidden type of cell protection that verifies, on one side, if a customer has the right to invoke a cell,

while it also checks, on the other side, if a customer's machine is capable of receiving an output from such a cell. COC aims to make the distributed Web application as secure as possible.

4.4.3 GENE STORE SYSTEM (GSS)

There are several combinations of processes that return the same results in a distributed application. Some of these applications are Web services that are divided into a set of groups, such that in each group all the applications can do the same jobs. The problem for service theory is summed up by the question of how to select the best service from an ocean of similar job services? Through COC we have mentioned an approach to the service selection problem. Simply, why not transform all the Cloud services processes into a new structure to be used by a novel model like COC?

In order to obtain a successful COC model, we need to build a suitable business process (gene) for each cell. The first step in building cell genes is to transform the service business processes and their combinations into a graph (or map) of abstract business processes. The obtained graph has no abstract information about any service business process. For example, if several services make a *division* job, then all of their *abstract business processes* are linked to a *division* node of the gene graph. Each cell uses a specific part of the obtained abstract graph and is known as a cell business process or gene. The gene store system's job is to store the genes and classify them, shaped by logical rules in a database to be easily used by cell subsystems.

4.4.4 PROCESS ANALYSER SYSTEM (PAS)

Changes allow companies to improve processes, to expand in new directions and to remain up-to-date with the times and technology. A business process is a sequence of steps performed for a given purpose. Business process analysis is the activity of reviewing existing business practices and changing these practices so that they fit a new and improved process. The role of PAS is to keep up-to-date analysis of the cells' business processes. A cell's business process design is based on a composition of process combinations transformed from service business processes. In order to return the best cell output, PAS must select the best plan from these combinations.

By its very nature, Cloud network connection shares big data. The amount of data crossing networks will continue to explode. By 2020, 50 billion devices will be connected to networks and the internet (Cisco IBSG, 2011) and the absolute volume of digital information is predicted to increase to 35 trillion gigabytes, much of it comes from new sources including blog networks, social networks, internet search, and sensor networks. The network can play a valuable role in increasing big data's potential for enterprises. It can assist in collecting data and providing context at high velocity and it can impact the customer's experience.

As the number of online-network communications is increasing sharply, it is difficult to access or analyze relevant information from the web. One possible approach to this problem offered by Web 3.0 is web personalization (Eirinaki & Vazirgiannis, 2003). Personalization aims at alleviating the burden of information overload by tailoring the information presented to individual and immediate user needs (Mobasher *et al.*, 2000).

One of the personalization requirements, which can affect a large part of the network data, is the combination of user web accounts to constitute a personal profile for each user.

BIG DATA ANALYSIS PROBLEM

The huge number of random web and Cloud connections and the unorganized storage of big data in Web 2.0 motivated computer scientists to develop Web 3.0. The new web is based on a wide arrangement of data. One of the problems with Web 2.0 is the random distribution of multi-accounts of users (social, business or other). Web 3.0 proposed the idea of personalization that meant web concepts shifted from working with words to dealing with personal profiles. To achieve a personal profile, all the user's accounts are treated as one block (account aggregation). Although personalization concept can solve many problems, including random accounts and search engine difficulties, it could affect negatively in the analysis phase. Before personalization, analytical methods were easier to apply because the target was one network.

In the new web, however, the goal is multi-network analysis (or multidimensional network graph analysis). For example, in the social network case it is easy to apply analysis to one network as a calculation of centrality measures, but how can we analyze several graphs with a different purpose for one person at the same time (e.g. calculating the degree of centrality of a person in both Facebook and Twitter networks at the same time and with one request)?

Currently the available methods and tools deal with one-dimensional graphs. Thus, the challenge to the new web is to analyze the multi-network (multidimensional) graphs

simultaneously. What is the degree of online network analysis that can be achieved with Web 3.0?

MULTI-NETWORK GRAPH AND DATA MODEL (PROPOSED MODEL)

This section highlights the relationship between the graph model and the data model. The new web trend is to use a multi-network model instead of a graph model to deal with the explosive growth of online networks. A graph is a representation of a set of objects wherein some pairs of objects are connected by links. “The interconnected objects are represented by mathematical abstractions called vertices, and the links that connect some pairs of vertices are called edges. Typically, a graph is depicted in diagrammatic form as a set of dots for the vertices, joined by lines or curves for the edges” (Trudeau & Richard, 1993). The edges may be directed or undirected. A multi-network graph is generally understood to mean a graph in which multiple edges are allowed.

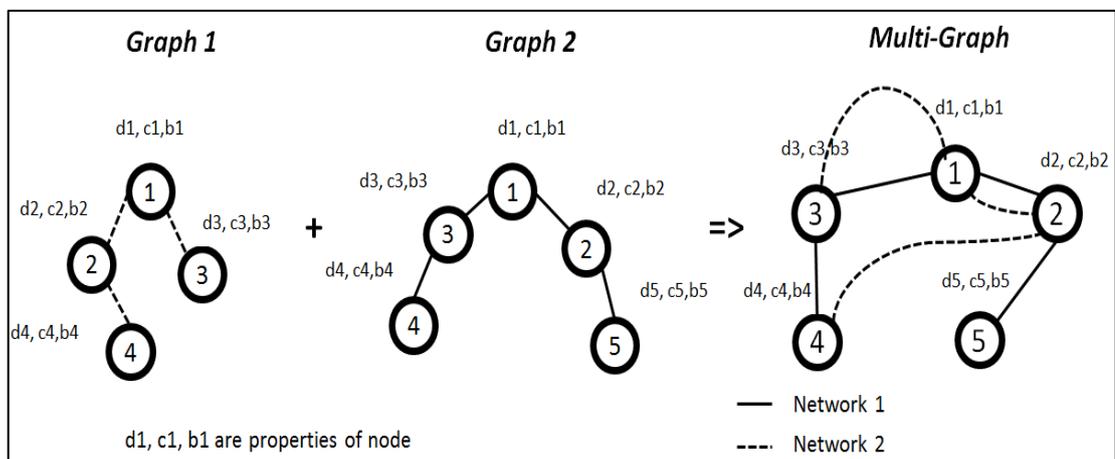


Figure 4.5 Merging multiple network graphs in one Multi-Network graph

Figure 4.5 shows an example of how a multi-graph is obtained from several graphs.

Graph1 and Graph2 represent the node connections in two different networks.

A multi-graph is based on *vertices*, *edges*, *belonging network* and *vertex properties*. A multi-graph is an ordered set $M = (V, L, N, P)$ such that:

V is a set of vertices,

$L = \{\{p, q\}: p, q \in V\}$ is a set of edges between two vertices which are subsets of V ,

$N = \{n1, n2, \dots, nk\}$ is the set of belonging networks that node belongs to and

$P = \{\text{degree centrality, closeness, betweenness, } \dots, \text{etc}\}$ is the set of properties of a node.

In order to talk about the relationship between the multi-graph model and the data model, it is necessary first to introduce the entity relationship (ER) model. ER is the most widespread semantic data model. It was first proposed by Chen in 1976 and has become a standard, extensively used in the design phase of commercial applications.

The entity relationship set $ER = (E, R, A)$ is composed of three basic types of sets: *entities*, *relationships*, and *attributes*. An entity set E denotes a set of objects, called *instances*, which have common properties. Element properties are modeled through a set of *attributes* A , whose values belong to one of several predefined domains, such as integer, string, or boolean. Properties that are caused by relations to other entities are modeled through the participation of the *entity* in *relationships*. A *relationship* set R denotes a set of tuples, each of which represents an association among a different combination of instances of the *entities* that participate in the *relationship*.

Let $g: V \rightarrow E$ and $h: N \rightarrow E$ be two functions mapping the values in set V and N to set E , in which if $x \in V$, then $g(x) \in E$. Facts $g(V)$ and $h(N)$, derived from the multi-graph M ,

are defined as follows: every *vertex* (node) x in the set of vertices V and every *belonging network* y in the set N is mapped by g and h respectively into *entities* in the set E .

Let $k: L \rightarrow R$ be a function such that $k(i) \in R$, where $i \in L$. This means that every *edge* belonging to set L is mapped to *relationship* by k .

Let $w: P \rightarrow A$ be a function such that $w(c) \in A$, where $c \in P$. This means that every *property* in the multi-graph is mapped in *attribute* in the **ER** diagram.

Figure 4.6 shows how a multi-network graph is mapped in the **ER** diagram. The multi-network graph consists of five nodes each with specific properties. Also, as in the graph in Figure 4.5, some of the nodes belong to network “1” (lined link) whereas others belong to network “2” (dotted line), and some may belong to both networks at the same time. As shown in Figure 4.5, the top **ER** diagram forms the result of the translation, in which nodes are translated to entities, properties to attributes and links to relationships.

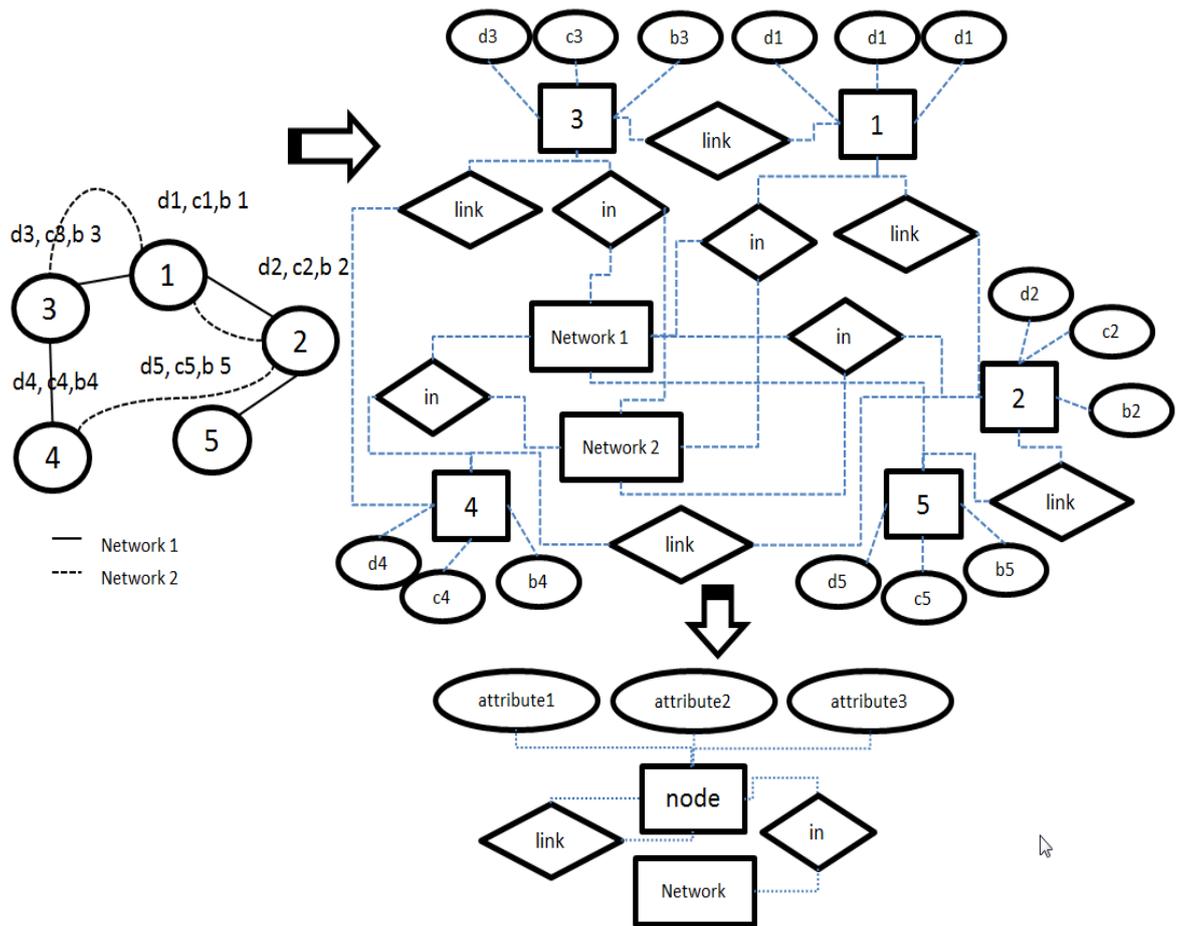


Figure 4.6 Mapping a multi-network graph into ER diagram

Because the same information is repeated (node name, network type and attributes) the top *ER* diagram is optimized into an optimized *ER* diagram at the bottom of the figure. The obtained *ER* diagram is the same for any multi-network graph (the number of attributes may vary).

MULTI-NETWORK GRAPH ANALYSIS

This section explains how to benefit from the mapping of the multi-network graph in the *ER* diagram in the network analysis. This part maps the obtained *ER* diagram in Figure

5.6 to a multi-dimensional database (cube). In this mapping, we study the three centrality measures (degree centrality, closeness and Betweenness).

BASIC CONCEPTS:

This section discusses some network analysis concepts. In graph theory and network analysis, there are several types of measures of the centrality of a vertex within a graph that determine the qualified status of a vertex within the graph. Many of the centrality concepts were first used for social network analysis, such as degree centrality, Betweenness, and closeness.

Degree Centrality: *The first and conceptually simplest concept, which is defined as the number of links incident upon a node. It is the number of nodes adjacent to a given node (sent = out a degree or received = in degree). The measure is entirely local, saying nothing about how one is positioned in the wider network. Degree centrality is defined by a degree of unit x : $c_D(x) =$ degree of unit x . Relative degree centrality is: $C_D(x) = c_D(x) / \text{highest degree} - 1 = c_D(x) / n - 1$, if n is the number of units in a network, the highest possible degree (network without loops) is $n-1$.*

Closeness Centrality: *Measures how many steps away from others one is in the network. Those with high closeness can reach many people in a few steps. Technically it is the sum of network distance to all others. This is not just a local measure, but uses information from the wider network. Sabidussi (1966) suggested a measure of centrality according to the closeness of unit x : $c_c(x) = 1 / \sum_{y \in U} d(x, y)$, where $d(x; y)$ is the length of the shortest path between units x and y , and U is the set of all units. Relative closeness centrality is defined by: $C_c(x) = (n - 1) * c_c(x)$, where n is the number of units in the network.*

Betweenness Centrality: *Betweenness centrality measures how often a given actor sits “between” others, “between” referring to the shortest geodesic. It detects the actor that has a higher likelihood of being able to control the flow of information in the network. Freeman (1977) defined the centrality measure of unit x according to Betweenness in the following way:*

$$c_B(x) = \sum_{y < z} \frac{\# \text{ of shortest paths between } y \text{ and } z \text{ through unit } x}{\# \text{ of shortest paths between } y \text{ and } z}$$

*Suppose that communication in a network always passes through the shortest available paths: the Betweenness centrality of unit x is the sum of probabilities across all possible pairs of units that the shortest path between y and z will pass through unit x . In network analysis, relative Betweenness centrality is used; it has two formulas according to the type of network. For undirected graphs of relative Betweenness, we use $C_B(x) = c_B(x) / ((n - 1) * (n - 2) / 2)$. For direct graphs of relative Betweenness, we use $C_B(x) = c_B(x) / (n - 1) * (n - 2)$.*

Every data analysis is based on a dataset, which is stored in a database. But in our case, we have a multi-dimensional graph. Therefore, we propose to map this type of graph in a multidimensional database. The functions and notations in this part depend on the previous definitions in previous section. Let $L_M(s, x)$ denote a link between s and x where $s, x \in V$ and $\varphi_s = |\sum_{i \in IN} L_M(s, x_i) / n - 1|$, where n is the number of nodes. Let function $d_M(s, t)$ calculate the shortest path distance between $s, t \in V(G_M)$.

Let $S_{st} = |n - 1/d_M(s, t)|$, where n the number of nodes is. Let P_{st} denote a set of different shortest paths between s and t (such that $s, t \in V$) and $\beta_{st} := |P_{st}|$. For every $v \in V$ let

$P_{st}(v)$ denote the set of different shortest paths containing v with $s \neq v \neq t$, & $\beta_{st} := |P_{st}(v)|$.

Let $D_{i*j*k}(R_{i*k}(D), C_{j*k}(D))$ be a multidimensional database (cube) of order 3, which represents a node in a multi-network graph, as shown in figure 4.7. $R_{i*k}(D)$ denotes the row i at the k level of the cube and $C_{j*k}(D)$ denotes the column j at the level k of the cube, and $i, j, k \in \mathbb{N}^+$.

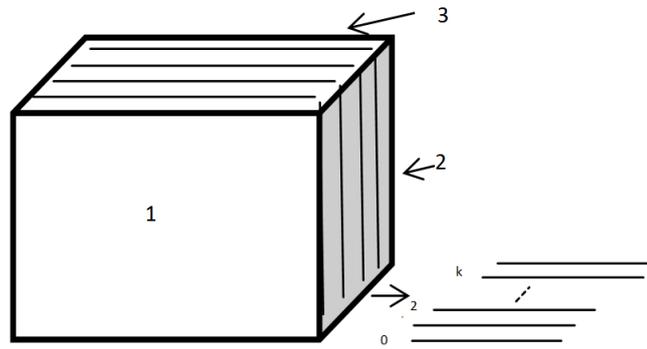


Figure 4.7 A structure of a cube with three faces and “k” levels of analysis measures

Let $M_k(a_{i,j})$ be a matrix of i, j dimensions, where $a_{i,j}$ is a value of the matrix entity at row i and column j with $i, j, k \in \mathbb{N}^+$. $M_k := \bigcup_{k,n \in \mathbb{N}} \bigcup_{0 \leq i \leq n} R_{i*k}(D) := \bigcup_{k,n \in \mathbb{N}} \bigcup_{0 \leq j \leq n} C_{j*k}(D)$, which means matrix M_k is formed by the union of cube rows or column at a specific level k . Let R_D denote the set of networks to be studied such that $R_D = \{R_0(D), R_1(D), \dots, R_N(D)\} = \{Networkname_1, Networkname_2, \dots, Networkname_N\}$.

Let set C_D denote the set of node names such that $C_D = \{C_{0*k}(D), C_{1*k}(D), \dots, C_{N*k}(D)\} = \{C_0(D), C_1(D), \dots, C_N(D)\} = \{node\ name_1, node\ name_2, \dots, node\ name_N\}$ (or $= \{A, B, \dots, Z\}$ sorted by first letter). Let set C_{D0} denote the set of number of links divided by $n - 1$ ($\varphi_{sx} =$

$|\sum_{i \in N} L_M(s, x_i) / n - 1|$) between a studied node and the other nodes named in C_D , such that $C_{D0} = \{C_{0,0}(D), C_{1,0}(D), \dots, C_{N,0}(D)\}$ or in other words C_{D0} represents the face of the cube at level zero.

Let set C_{Di} denote the set of the distances ($S_{xt^i} = |n - 1/d_{GM}(x, t^i)|$) from a studied node “x” to all the other nodes “tⁱ”, such that $C_{D1} = \{C_{0,1}(D), C_{1,1}(D), \dots, C_{n,1}(D)\}$. For all the other columns C_{Di} , where $i \geq 2$, let set C_{Di} denote the set of different paths between any two nodes passing through a specific node v which is studied by the cube ($\beta_{st} := |P_{st}(v)|$) divided by the sum of different paths between any two nodes ($\beta_{st} := |P_{st}|$), such that $C_{Di} = \{C_{0,i}(D), C_{1,i}(D), \dots, C_{n,i}(D)\}$.

Table 4.1 Representation of level 0 of the cube

	Nodename1 (n ₁)	Nodename2 (n ₂)	Nodename3 (n ₃)
Network1 (r ₁)	\mathcal{O}_{Sn1}^r	\mathcal{O}_{Sn2}^r	\mathcal{O}_{Sn3}^r
Network2 (r ₂)	\mathcal{O}_{Sn1}^r	\mathcal{O}_{Sn2}^r	\mathcal{O}_{Sn3}^r
Network3 (r ₃)	\mathcal{O}_{Sn1}^r	\mathcal{O}_{Sn2}^r	\mathcal{O}_{Sn3}^r

Table 4.1 explains the structure of node’s cube is structured as a three-dimensional cube of three faces that are divided into “K” number of levels (0,1,..., k).

Table 4.2 Representation of level 1 of the cube

	Nodename1 (n ₁)	Nodename2 (n ₂)	Nodename3 (n ₃)
Network1 (r ₁)	\mathcal{S}_{Sn1}	\mathcal{S}_{Sn2}	\mathcal{S}_{Sn3}
Network2 (r ₂)	\mathcal{S}_{Sn1}	\mathcal{S}_{Sn2}	\mathcal{S}_{Sn3}
Network3 (r ₃)	\mathcal{S}_{Sn1}	\mathcal{S}_{Sn2}	\mathcal{S}_{Sn3}

Table 4.2 represents the level 0 of the node’s cube “s” as a matrix, in which the columns show the other node’s name on the graph and the rows show the networks that a

node appears in. The values in the matrix entries contain the *degree of centrality* φ that node “s” has with the other nodes.

Table 4.3 Representation of level 2 of the cube

	Nodenname1 (n ₁)	Nodenname2 (n ₂)	Nodenname3 (n ₃)
et ₁ (r ₁)	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r1]}{\beta x_i y_i (n1)[r1]}$	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r1]}{\beta x_i y_i (n2)[r1]}$	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r1]}{\beta x_i y_i (n3)[r1]}$
et ₂ (r ₂)	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r2]}{\beta x_i y_i (n1)[r2]}$	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r2]}{\beta x_i y_i (n2)[r2]}$	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r2]}{\beta x_i y_i (n3)[r2]}$
et ₃ (r ₃)	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r3]}{\beta x_i y_i (n1)[r3]}$	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r3]}{\beta x_i y_i (n2)[r3]}$	$\sum_{\forall x_i, y_i \in V} \frac{\beta x_i y_i [r3]}{\beta x_i y_i (n3)[r3]}$

Table 4.3 represents level 2 of the node’s cube “s” as a matrix. The values in the matrix entries, however, contain the result of calculating the number of different paths between any two nodes passing through a node “s” ($\beta_{st}(s) := |P_{st}(s)|$) divided by the sum of different paths between any two nodes ($\beta_{st} := |P_{st}|$).

A database cube is obtained that represents a multi-network graph at the same time. As a result, it is easy to calculate centrality measures for each node depending on its cube ($D_{i,j,k}$) and by directly applying queries on cube values. In order to calculate the degree centrality and the closeness centrality, the contents of cube levels $k = 0$ and $k = 1$ are invoked, respectively. For Betweenness centrality, the cube level $k = 2$ is invoked. If the studied graph is undirected, then we divide the result by $((n - 1) * (n - 2) / 2)$; otherwise the result is divided by $(n - 1) * (n - 2)$.

4.4.5 PROCESS VALIDATION SYSTEM (PVS)

The cell business process, in COC, is built on a dynamic composition of a group of service business processes. If there are problems in one or more business applications that support a cell business process, then the consequences of disruption to the cell process can be serious. For example, some process compositions may result in infinite loops or deadlocks. The process validation system's job is to monitor and validate the changes in altered or new composition processes.

VALIDATION OF PROCESS COMPOSITION

“Web services are designed for interaction in a loosely coupled environment, and therefore are an ideal choice for companies seeking inter or intra business interactions that span heterogeneous platforms and systems” (Li, 2005).

Sometimes a single service is not sufficient to perform client's requirements and often services composition strategy is used as a solution. Designing a new composite service requires a discovery stage in which a set of candidate services are highlighted. But nothing notifies that the obtained composite service resulted from a set discovered services will work normally or not.

In the dynamic world of service-oriented architectures, however, what is sure at design time, unluckily, may not be true at run time. The actual services, to which the workflow is bound may change dynamically perhaps in an unexpected way, and therefore may cause the implemented composition to deviate from the assumptions made at design time. Besides performing design-time validation, it is also necessary to perform continuous

run-time validation to ensure that the required properties are maintained by the operating system. The compiler is the only way to validate the sequence of service process. Thus, PVS is built on a distributed dynamic compiler that compiles the composition of every new composite service. When a client designs a new composite service, the related compiler Grammar rules, of the invoked services, are sent to him as XML files then combined together to constitutes a local compiler that validate new service composition at design phase.

BASIC FEATURES:

This section discusses some basic features which are used in the proposed Cloud service process validation model.

Business process execution language (BPEL) - *BPEL is a language created to compose, orchestrate and coordinate web services. It allows the creation of composite processes with all its related activities.*

Compiler – *“is a program that takes a source program typically written in a high-level language and produces an equivalent target program in assembly or machine language” (Aho et al., 2007). A compiler performs two major tasks: analysis of the source program and synthesis of the target-language instructions. In order to build a compiler, there are six phases to follow as in figure 4.8: i) scanning the input program will be grouped into tokens, ii) parsing or syntax analysis, iii) building a Context-Free-Grammar, iv) applying semantic analysis to keep on mapping between each identifier of data structure (symbol table) and all its information and ensure consistent, v) extracting assembly code generation and vi) finally realizing code optimization.*

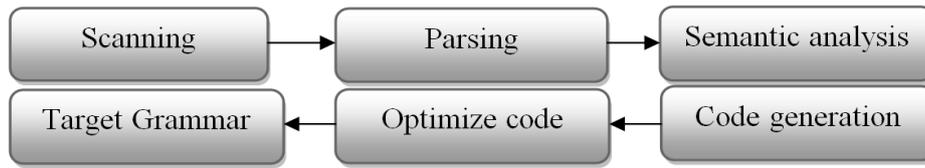


Figure 4.8 Phases to build a compiler

Depth First Traversal (DFS) – it is a graph theory algorithm for traversing a graph. It is a generalization of preorder traversal. It starts from a vertex and recursively it build a spanning forest that determine if the graph is cyclic (contain cycle loop) or acyclic.

SERVICE COMPOSITION PROBLEM

In order to highlight on the service composition problem and simplify the idea for the reader, this section gives two examples about service composition. The first example reflects a simple normal composition while the second shows an abnormal service behavior.

Simple Services Composition Example:

Figure 4.9 shows a simple example of how Providers of web services are communicated to achieve a composed service. Let $Client_2$ has to solve two mathematical formulas: “ $F1: A = 2*x + 3*y$ ” & “ $F2: B = 2*x$ ”.

In order to achieve his goal $Client_2$ will design a new composite web service. First of all, he searches in $UDDI_2$ which gives him a summary about the services that are existed in the $Provider_2$. $UDDI_2$ has two services that solve two equations:

$$EQ_1: “2*x” \quad \& \quad EQ_2: “3*y”.$$

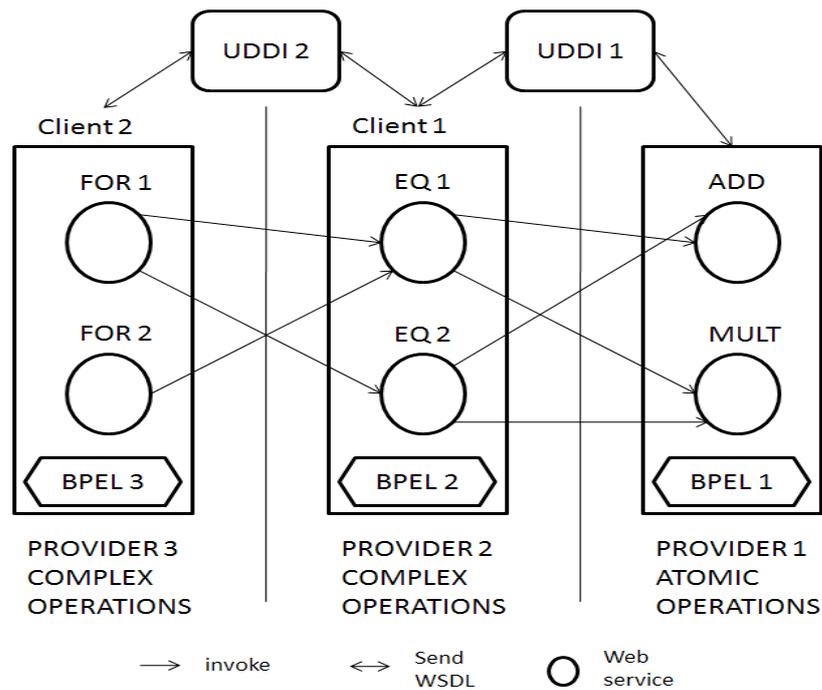


Figure 4.9 Example of composite services

Using the information given in the *WSDL* file by *UDDI₂*, *Client₂* invokes *Provider₂* operations. But the two services *EQ₁* and *EQ₂* invoke other services *ADD* & *Multiply* from the *Provider₁* to complete the required answer. This is a simple idea about how service composition works.

Deadlock Example:

Cells are based on Web and Cloud services that are distributed through the whole internet and controlled by various sides. In the modern state, services are dynamically managed. Because the most used services are huge and composite, the states of failure and infinite loop can be detected sometimes. Failure of composite services results from an obstacle in one of its parts, while infinite loop exists as a result of wrong process flow design.

A real example of the service composition problem (infinite loop) is the TIBCO web service (<https://www.tibcommunity.com/message/70086>). Figure 4.10 shows an infinite loop (or cycle) while executing composite service.

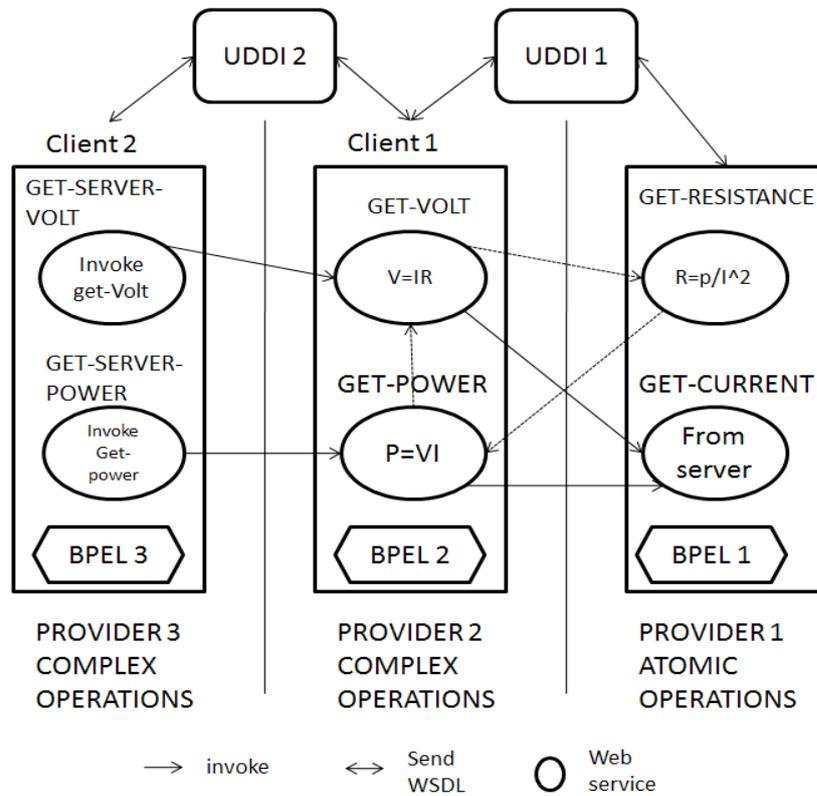


Figure 4.10 Infinite loop of web service

Let voltage represented by V , current by I , resistance by R and represent power by P .

We have a set of service to use:

- Client2 build two services GET-SERVER-VOLT & GET-SERVER-POWER
- Provider2 provides two services Get-Volt ($V= I \cdot R$) & Get-Power ($P=V \cdot I$)
- Provider1 provides two services Get-Resistance & Get-Current ($R=p/I^2$)

*Client*₂ wants to calculate the consumption of Voltage and Power of the last service provider machine during a composite service process. To complete the needed service, *Client*₂ invokes services from *Provider*₂ while *Provider*₂ invokes other services of *Provider*₁ to answer the question of *Client*₂. To build his own services (*GET-SERVER-VOLT & GET-SERVER-POWER*), *Client*₂ firstly searches in UDDI₂ about services and invokes *Get-Volt & Get-Power* from *Provider*₂.

Regarding the service “*Get-Volt*”, it invokes *Provider*₁ (the last service provider in this process) services specifically the “*Get-Resistance*” service to calculate resistance ‘*R*’ and it invokes the “*Get-Current*” service to calculate current ‘*I*’.

From the other side, the service “*Get-Resistance*” invokes “*Get-Power*” service from *Provider*₂ in order to calculate power ‘*P*’. But the service “*Get-Power*” invokes “*Get-Volt*” service to calculate voltage *V*.

Indeed, the “*Get-Server-Volt*” service falls into an *infinite loop* as seen above (figure 5.10) in red color. The “*Get-Volt*” node invokes the “*Get-Resistance*” node which needs results from “*Get-Volt*”. Thus “*Get-Volt*” invokes itself indirectly. There are also other types of errors may occur because of partial fail or bad service communications.

DISTRIBUTED GLOBAL SERVICE COMPILER (DGSC)

DGSC model consists of extracting compiler Context-Free-Grammar rules of the service business process of a web service. Then save these rules in the UDDI registry. Grammar rules are used later by the client when he fetches the registry to build a new composite service. Cell validation part (PVS) use DGSC to verify a new composite service before the execution.

Cell's goal is to discover design errors in the design phase of composite service without knowing the exact flow of service process. In fact, there are many obstacles facing DGSC because the service design takes place in the client side and the content of web services is dynamically edited from several sides. Cells search the Cell data center in order to build a new composite service. But nothing verifies that the new combination of service, that may also invoke other services, is free of errors and infinite loops. Also even if a correct composition of a complex service is achieved, this action may fail later because services are dynamically edited. In order to show a simulation of our proposed model, we will apply DGSC on BPEL; the mostly used business process.

The proposed validation approach uses two phases of compiler design (scanning and parsing). This phase of the compiler is applied in the business process (BPEL) of service that contains the internal service design. A grammar rules drive similar to the case of the third phase of compiler design (Context-Free-Grammar). These rules are sent to the storage of Cell data center in XML format. Cells use validation rule files of services to design a new composite service. Thus depending on DGSC model, a Cell downloads the rules file, from the UDDI, with the WSDL file and he uses these rules to compile a new design of composite service. Locally on the client side, a mathematical algorithm (Depth

First Search) is applied in these rules to detect if the new design of composite service contains infinite loop before service deployment.

EXTRACTION OF SERVICE PROCESS GRAMMAR

For every programming language there exists a compiler Grammar that is used to verify the steps of building a new program. But in DGSC model, scanning and parsing stages are applied to the business process files and Context-Free-Grammars are deduced about the business process of each service. In order to achieve our BPEL parser, the BPEL grammar of BP4WS is used. The BPEL parser is implemented using Java code. The outcome of the parser is a database table.

Each row entry represents the details of an individual activity which provides information about the current state name, current state properties (as My Role, Partner Role), PartnerLink (which represents the associated web service), name of the operation being invoked, condition of a looping structural activity, current state number, and next possible state numbers. The result of parsing BPEL file is saved in an Excel file.

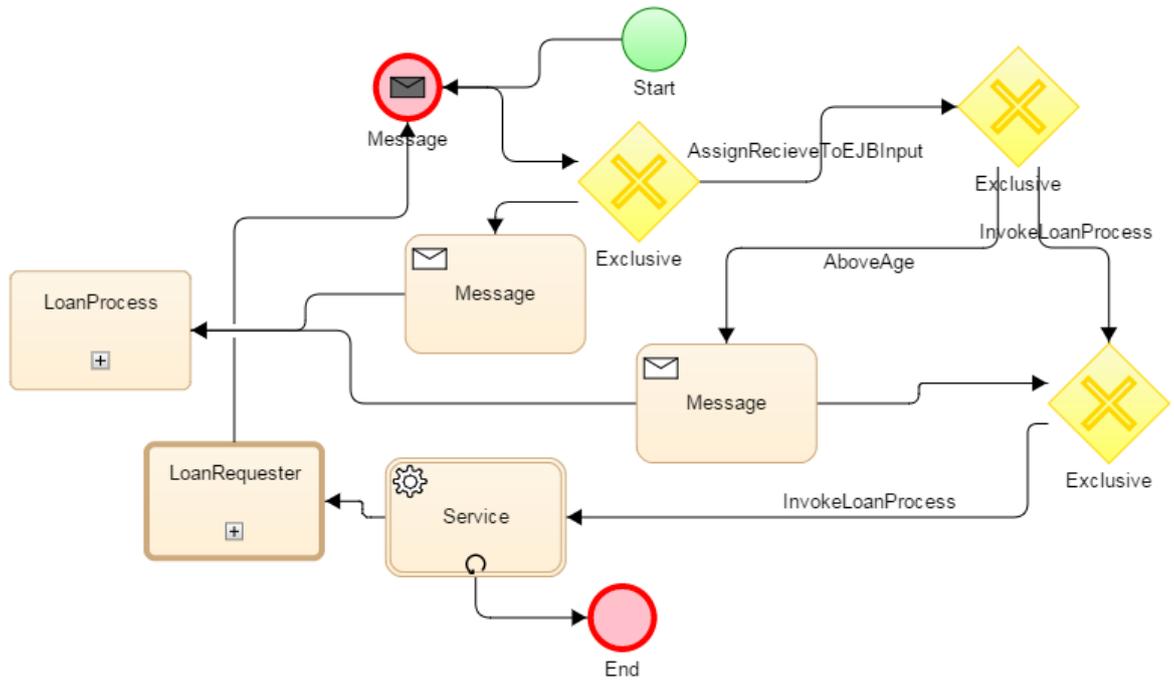


Figure 4.11 Loan BPEL example

Table 4.4 below contains the output of parsing Loan BPEL file of the BPEL design shown in figure 4.11.

Table 4.4 The Output result of parsing the Loan BPEL code

Activity name	0	1	2	3	4	5
Activity Name	receive	if	invoke	if	invoke	Reply
Current State	ReceiveFrom-Customer	nul 1	invokeloan-Process	nul 1	invokeloan-Process	ReplyToCustomer
Conditions	LoanRequestorOpera.	nul 1	ProcessApplication	nul 1	ProcessApplication	LoanProcessor
Partner Link	LoadRequestor	nul 1	LoanProcessor	nul 1	LoanProcessor	LoanProcessor
Operations	LoanRequestorOpera.	nul 1	ProcessApplication	nul 1	ProcessApplication	LoanProcessor
Next Activity	1	2,3	5	4,5	5	6

DETECT CYCLE BY DFS

According to DGSC, Validation rules are requested from Cell side. The result of parsing the business process file is considered as an input that is transformed into a directed graph (arcs between nodes have sense).

Now the problem is changed from programming into a graph theory problem (figure 4.12). Instead of checking if the new design of composite service falls in infinite loop or not, we can verify if that the obtained graph is directed cyclic or acyclic. Depth First Search algorithm is used to detect if the graph is acyclic.

Indeed, DFS starts from the root node and explores siblings as far as possible along each branch before backtracking and if it arrive a visited node again it will notify that the graph is cyclic (it contains cycle). But sometimes the service designer need to have a cycle like while-loops, for-loops or even reply to node that sends a request. Thus in all cases we give the designer the permission to discard the detected harmful loops.

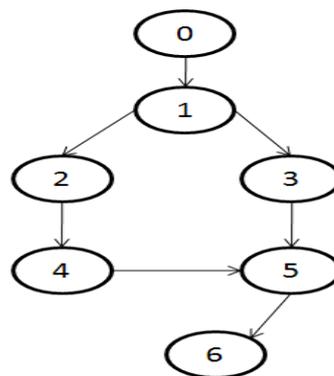


Figure 4.12 The directed graph of the BPEL example

Composite services are built on invoking other already implemented services, but these services are dynamic and able to be edited at random time. In order to achieve a smart component a Cell that is capable to validate composite services, we have built a Cell validation subsystem (PVS) that is based on decentralized compiler. When a new composite service is built, the validation rules are collected from the Cell data center. These rules are combined and DFS is applied to detect errors (infinite loops, errors...etc.). If the result returns an error then a notification appears to alter the wrong service design.

DGSC deals with the existing implemented services as standards, which have correct design, for new composite service. In other words, if a Cell is developing a new composite service called *XY* then all invoked services stay non-editable at the last stage of designing this service. Cell Data Center sends updates to a designer Cell regarding any changes occur in the shared services of the new service composition. Also the cell data center prevents changes in the shared services while the deployment phase of the new composite service takes place.

4.4.6 TRAITS MAINTENANCE SYSTEM (TMS)

A cell business process is a dynamically coordinated set of collaborative and transactional activities that deliver value to customers. Cell process is complex, dynamic, automated and long running. One of the key characteristics of a good cell business process is continuous improvement. These improvements ensure a constant flow of ideal traits into the cell process. Cell computing is built upon achieving a group of architectural traits such as: performance, reliability, availability and security. These qualities require stable monitoring to maintain the supply of customers.

Cells in COC apply internal and external efforts to maintain best traits. External efforts are achieved via cell collaboration, while internally the job is done by TMS. Indeed, TMS analyses the quality of process (QoG) combinations of a cell; these are combinations of the traditional quality of service (QoS) analysis. It uses a data warehouse of QoG to accomplish this type of analysis.

CLOUD INFRASTRUCTURE

Cloud computing is defined as a model for enabling expedient, on-demand network access to a mutual group of resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. A Cloud environment is characterized by system level, Cloud Broker level and user middle-ware level.

The user Middle-ware level includes the software frameworks such as Web 2.0 Interfaces and provides the programming environments and composition tools that ease the creation, deployment and execution of applications in Clouds.

The system level is composed of thousands of servers, each with its own service terms management systems, operating platforms and security levels. These servers are transparently managed by the higher level virtualization (Smith and Nair, 2005) services and toolkits that allow sharing their capacity among virtual instances of servers.

The Cloud Broker level implements the platform level services that provide runtime environment enabling Cloud computing capabilities to build Cloud services. The Cloud Service Broker performs several management operations to deliver personalized services to consumers. These operations are: security and policy management, access and identity

management, SLA management, provision and integration management. The security and policy manager is responsible for managing different kinds of policies such as authorization policies and *QoS*-aware selection policies of service providers.

The access and identity manager is responsible for the accessing services and respect the identity rules of services. The SLA Manager directs the concession process between a consumer and a selected *SaaS* provider in order to reach an agreement as to the service terms and conditions. The provision and integration manager is responsible for implementing different policies for the selection of suitable *SaaS* providers, based on the consumer's *QoS* requirements and the *SaaS* providers' *QoS* offerings. The back-end database stores sustain information about service policies, consumer profiles, SLAs, Registry and dynamic *QoS* information.

Cloud broker layer works to identify the most appropriate Cloud resource and maps the requirements of application to customer profile. Its job can also be dynamic by automatically routing data, applications and infrastructure needs based on some *QoS* criteria like availability, reliability, latency, price, etc. On the Broker side, service properties are stored as a combination of functional and non-functional properties. The functional properties relate to the external behavior of a service such as: service inputs and outputs, service type and the information required for connecting to the service. However, the non-functional properties are summarized by the *QoS*. By dynamically provisioning resources, Cloud broker, as shown in Figure 4.13, enables Cloud computing infrastructure to meet arbitrary varying resource and service requirements of Cloud customer applications.

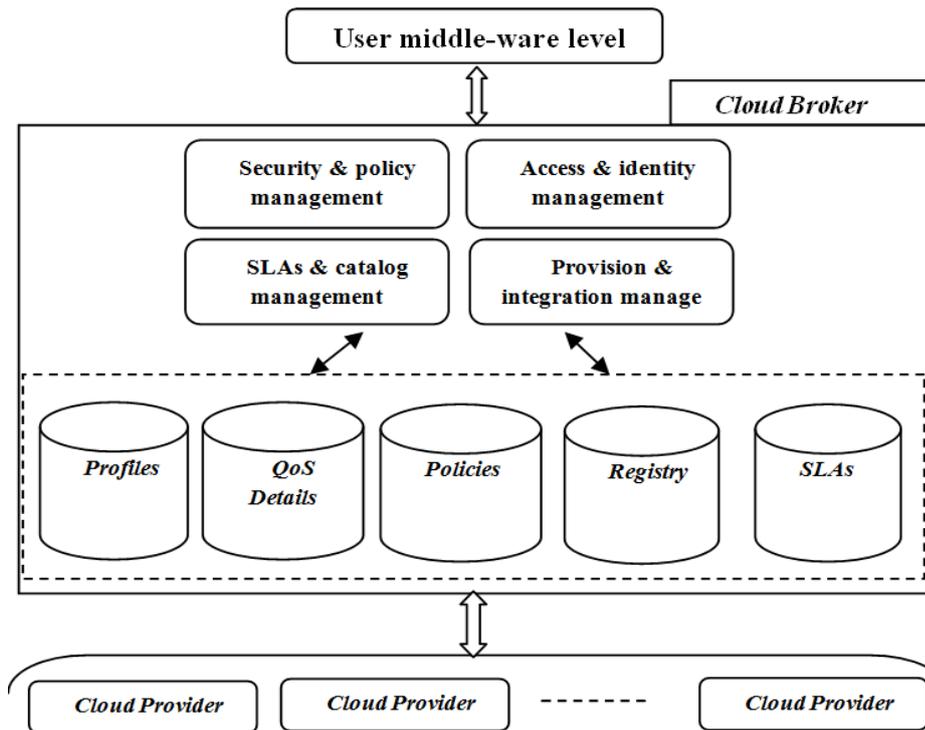


Figure 4.13 Main Layers of cloud infrastructure

However, there are still imperfections regarding service matching based on available services and customer profile requirements. The services selection problem is identified by an inaccurate *QoS* dependency and the utility of the imprecise domain of results suggested by *QoS* broker. As in (Al-Masri and Mahmoud, 2007), services are ranked into many levels such as Poor, fair, Good, Excellent or Bronze, Platinum, Silver and Gold, based on Web Service Relevancy Function (WsRF), which is measured based on the weighted mean value of the *QoS* parameters.

The *QoS* broker orchestrates resources at the end-points, coordinating resource management across layer boundaries. Based on the available technology, Service consumer is still incapable of a real analysis of the *QoS* based on the internal structure of complex service. Today's service selection solutions do not focus on *QoS* support from the

service requester view point, but they depend on service provider interpretation. Indeed, the current form of service selection is provider driven (Liu, 2005). A consumer may interact with a composite service without knowing much about the qualities of the services that underlie it (Yu and Bouguettaya, 2010).

QOSDW MODEL

Nowadays, the Cloud is full of a large number of Cloud services. Some of these services are similar in goal and quality.

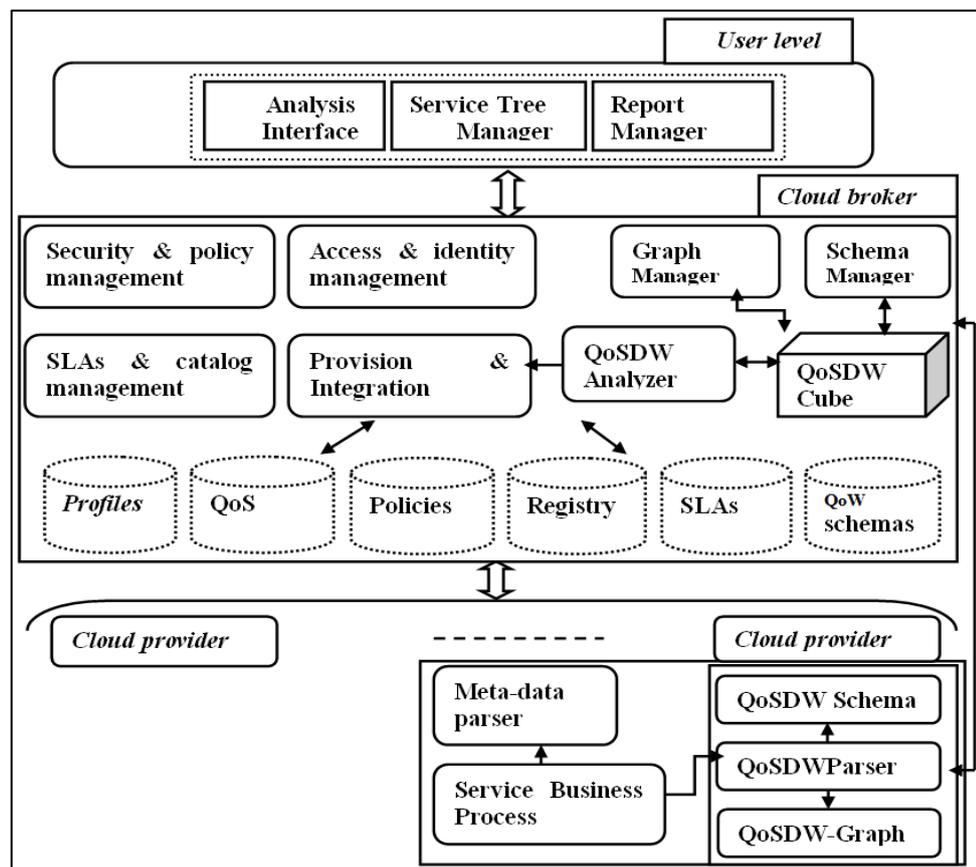


Figure 4.14 QoSDW model components

Therefore, it is difficult to select best service depending on the traditional *QoS* methods. In order to improve the selection of a complex service, we propose the *QoS* Data Warehouse (*QoSDW*) model which go deeper and study the quality each sub-service. The *QoSDW* model (described in Figure 4.14) supports a better analysis of services before taking a selection decision. The *QoSDW* model extracts details about services stored in the service provider, and gives the service's consumer the ability to discover the hidden facts about the properties of these services.

QOSDW COMPONENTS

This section describes a model for the selection of a Cloud service that can fulfil the service consumer request. In addition to the main Cloud framework elements discussed in the previous section, the proposed *QoSDW* model adds a group of other components such as: *QoSDW Parser*, *Schema Manager*, *Graph Manager*, *QoSDW Analyser*, *QoSDW Cube*, *Analysis Interface*, *Service Tree Manager* and *Report Manager*.

QoSDW Parser: *QoSDW Parser* is simply a *service business process* parser. Based on the parsers outputs and the *QoS* service provider, *QoSDW schema* and *QoSDW graph* are extracted and transported into the Cloud broker to be stored in a specific database. Regarding the database tables, each row entry collects details about service activities. It provides information about the current state name, current state properties (as *My Role*, *Partner Role*), *PartnerLink*, name of the operation being invoked, condition of a looping structural activity, current state number and next possible state numbers.

Schema Manager is responsible for managing the *QoSDW* schemas. The *QoSDW Schema* is a star schema which is composed of a set of organised tables, and which has a main fact table and set dimensional tables. *QoSDWSchema* consists of 22 dimensional tables as follows: *Quality, Availability, ResponseTime, Documentation, BestPractice, Throughput, Latency, Successability, Reliability, Compliance, property, ServiceType, ServiceName, ExpiryDate, CreationDate, ServiceFlow, Loop, Sequence, AndSplit, XorSplit, AndJoin and XorJoin table.*

QoSDW Cube is a Data Warehouse of quality and structure of both a service and its sub-services. It is accessed as a Cloud service and supports users by details about the quality and flow of service through a special Analyser. It maps the idea of the multidimensional data model to service selection model, through which it gives the service's user the ability to apply a multidimensional query on the discovered set of services.

QoSDW Analyser works like an analysis tool. It monitors *QoS* changes and prepares analytical reports about *QoS* information stored in the *QoSDWCube*. It gives the service consumer the right to query the *QoSDWCube* through its interface.

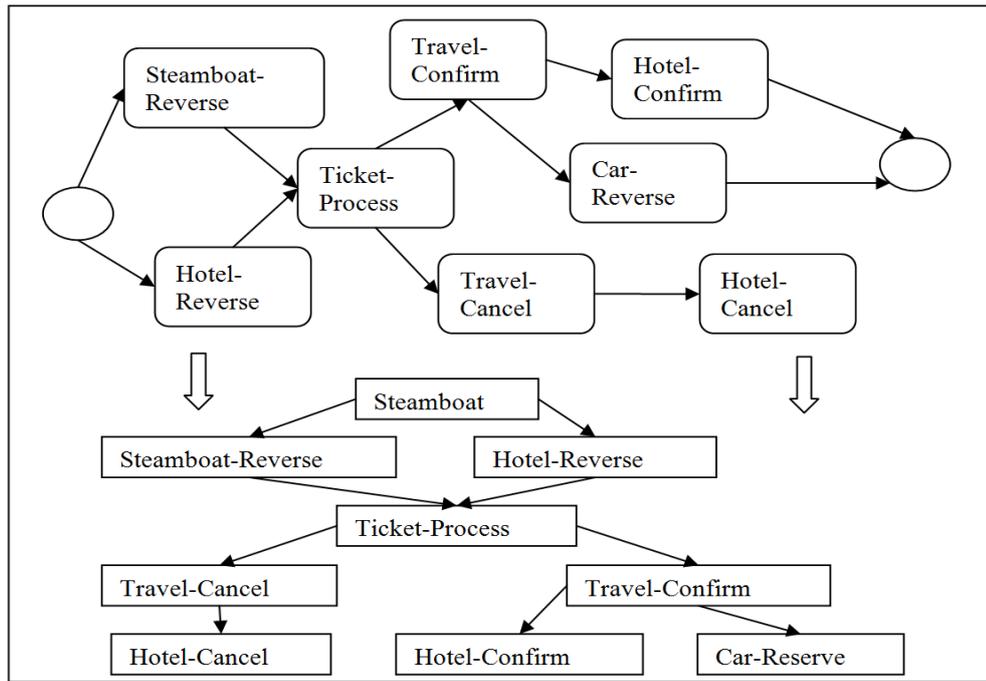


Figure 4.15 Transforming SteamBoat service business process into a tree of sub-services

Graph Manager ensures transforming the output of parsing the service business into a directed acyclic graph. Also, it converts the obtained graph into a service tree. For example, Figure 4.15 shows how *Steam Boat* service process diagram is transformed into a service tree. The service tree inserts a semantic layer into the service selection process.

Analysis Interface is a user interface application utilized to select Cloud services (*SaaS*). It consists of a statistical form which allows a user to deal easily with large statistical data, through slice, dice, Drill Down/Up and Roll-up the statistical results. It communicates with the *QoS* Analyser and allows users to connect to the *QoS* data warehouse, at the Cloud broker, and apply queries. When a service is selected, the *Selection Interface* connects the user to the required service via the SOAP/HTTP protocol.

Service Tree Manager supports a visual representation of the service's tree. It communicates with the *Graph Manager* indirectly through the *QoSDW Analyser*. Based on the service graph, the Analyser supplies the user by the service tree.

Report Manager: Sometimes the service's consumer needs ready reports that support their analysis. *Report Manager* allows requesting two types of reports: the primary report gives analysis results about the quality of first level sub-services, and the advanced report supports a deep service tree analysis to detect a weak quality subservice (or fatal subservice). Both reports are requested from the *QoSDW Analyser*.

FORMAL DEFINITIONS OF QOSDW MODEL

The main objective of a *QoSDW* model is to provide efficient analytical reporting on the quality of service. In order to qualify a service, the *QoSDW* depends on analysing the quality of its sub-services. *QoSDW* depends on the service business process to specify the structure of subservices.

Definition 1: A service business process is a tuple $K = (A, E, C, L)$ where:

- A is a set of activities,
- E is a set of events,
- C is a set of conditions and L is a set of control links.

Let $f: A \rightarrow B$ be a function that assigns activities to types, where activities are extracted from the set of activity $A = \{sequence, flow, pick, switch, while, scope, invoke, receive, reply, wait, assign, empty, throw, compensate, and exit\}$. Let I be a set of service information, where $I = \{service\ name, service\ type, service\ creation\ date, service\ expiry\ date\}$.

Let $g: P \rightarrow I$ be a function which assigns service information to properties.

QoSDW utilises an On-line Analytical Processing (*OLAP*) approach and performs analysis in conjunction with the operational database on a constant basis. The basic concept of *OLAP* model is to map the initial database schema to a multidimensional model. The *QoSDW* schema is structured as star (or snowflake) schemas.

Definition 2: A *QoSDW* schema is a tuple $S = (Q, P, B)$ where:

- Q is a set of *QoS*, such that:
 - $Q = \{Response\ time, Availability, Throughput, Successability, Reliability, Compliance, Best\ Practice, Latency, Documentation\}$.

- P is a set of service properties, such that $P = \{ServiceType, ServiceName, ExpiryDate, CreationDate\}$.
 - B is a set of activity type, where $B = \{Loop, Sequence, AndSplit, XorSplit, AndJoin, XorJoin\}$.
- Let h be a function which assigns the values of QoS to elements of set Q .

The $QoSDW$ graph adds a type of semantic knowledge when analysing the quality of sub-services and covers indirectly the hidden service business process vague.

Definition 3: A $QoSDW$ graph is a tuple $G = (Ni, Nf, N, F)$, where:

Ni is the node of the input,

Nf is the node of output,

N is the set of names of sub-services

and F is the set of service integration models. $F = \{Sequence, ANDSplit, XORSplit, loop, ANDJoin, XORJoin\}$.

Let $m: B \rightarrow F$ be a function that maps service activities to integration models.

The operations which are applied in the analysis phase of the $QoSDW$ model are summarized by: Composition, Pairing, Projection and Restriction.

Composition takes as input two functions f and g , such that $range(f) \subset def(g)$, and returns a function $g \circ f: def(f) \rightarrow range(g)$, defined by:

$$(g \circ f)(x) = g(f(x)) \text{ for all } x \text{ in } def(f).$$

Pairing takes as input two functions f and g , such that $def(f) = def(g)$, and returns a function $f \wedge g: def(f) \rightarrow range(f) \times range(g)$, defined by:

$$(f \wedge g)(x) = \langle f(x), g(x) \rangle, \text{ for all } x \text{ in } def(f).$$

Projection is the usual projection function over a Cartesian product. Take function $f: X \rightarrow Y$ and $g: X \rightarrow Z$ with common domain X , and let π_y and π_z denote the projection functions over $Y \times Z$:

$$f = \pi_y \circ (f \wedge g) \text{ and } g = \pi_z \circ (f \wedge g).$$

Restriction takes as argument a function $f: X \rightarrow Y$ and a set D , such that $D \subseteq X$, and returns a function $f/D: D \rightarrow Y$, defined by:

$$(f/D)(x) = f(x), \text{ for all } x \text{ in } D.$$

BUILDING QoSDW SCHEMA

The base of *QoSDW* schema is a finite labeled diagram whose nodes and arrows satisfy the following conditions: there is only one root, at least one path from the root to every other node and all arrow labels are distinct. Our goal from the obtained *QoSDW* schema is to have an organized store of service qualities, properties and structure in which multidimensional queries can be applied.

The proposed *QoSDWSchema* consists of the following tables:

- *Fact table: Fact (service_id*, URI_type);*
- *Table of dimension Quality: Quality (Quality_id*, Quality_value, foreign_service_id);*
- *Tables of dimension Quality attributes:*
- *Availability: Availability (avail_id*, avail_value, foreign_Quality_id);*
- *Response time: ResponseTime (response_id*, response_time_value, foreign_Quality_id);*
- *Documentation: Documentation (Doc_id*, Documentation_value, foreign_Quality_id);*

- *BestPractice: BestPractice (practice_id*, practice_value, foreign_Quality_id);*
- *Throughput: Throughput (throughput_id*, throughput_value, foreign_Quality_id);*
- *Latency: Latency (Latency_id*, Latency_value, foreign_Quality_id);*
- *Successability: Successability (Successability_id*, Successability_value, foreign_Quality_id);*
- *Reliability: Reliability (Reliability_id*, Reliability_value, foreign_Quality_id);*
- *Compliance: Compliance (Compliance_id*, Compliance_value, foreign_Quality_id);*
- *Table of dimension property: property (property_id*, property_value, foreign_service_id);*
- *Tables of dimension property attribute:*
- *Type: ServiceType (ser_type_id*, type_value, foreign_property_id) /value: service or sub-service*
- *Name: ServiceName (ser_name_id*, ser_value, foreign_property_id);*
- *ExpiryDate: ExpiryDate (ExpiryDate_id*, ExpiryDate_value, foreign_property_id);*
- *CreationDate: CreationDate (CreationDate_id*, CreationDate_value, foreign_property_id);*
- *Table of dimension flow: ServiceFlow (flow_id*, service_flow_value, foreign_service_id);*
- *Tables of dimensional flow attribute:*
- *Loop: Loop (loop_id*, input_service, output_service, service_stage, foreign_flow_id) / stages: start node, normal node or end node.*
- *Sequence: Sequence (sequence_id*, input_service, output_service, service_stage, foreign_flow_id);*
- *AndSplit: AndSplit (AndSplit_id*, input_service, output_service, service_stage, foreign_flow_id);*

- *XorSplit*: *XorSplit* (*XorSplit_id**, *input_service*, *output_service*, *service_stage*, *foreign_flow_id*);
- *AndJoin*: *AndJoin* (*AndJoin_id**, *input_service*, *output_service*, *service_stage*, *foreign_flow_id*);
- *XorJoin*: *XorJoin* (*XorJoin_id**, *input_service*, *output_service*, *service_stage*, *foreign_flow_id*);

SERVICE SELECTION BASED ON QOSDW

Based on the *QoSDW* schema, the *QoS* Data Warehouse is built. Similar to the traditional discovery method, the service consumer requests a service and the service registry replies by a set of related service. If the *QoS* is not helpful to select the best service, the service consumer requests an OLAP analysis report about the quality of the discovered set of services. The *QoSDW* model consists of a special *QoSDWAnalyser* which supports two types of reports about *QoS*. The first type is a preliminary report which provides information about the quality of first level sub-services. Figure 4.16 shows a visual representation given by the *QoSDWAnalyser* about *QoS* of sub-services.

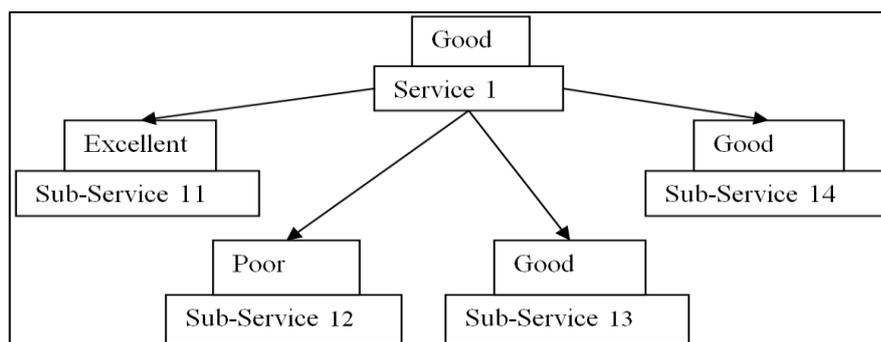


Figure 4.16 Representation of the initial report by *QoSDWAnalyser*

Sometimes the result of the initial report is not beneficial in designing a new composite service of better quality. Thus, the advanced *QoS* report is demanded by the service designer. As regards building the required report, the *QoSDW* Analyzer applies some queries on Data warehouse, which results different shapes of service's tree (as in figure 4.15). Then, Analyzer utilizes a tree search algorithm to detect fatal sub-services (see *Algorithm 1*). The implementation and efficiency of *Algorithm 1* is discussed in my previous book chapter (Karawash *et al.*, 2014b).

Algorithm 1: Detection of infected services

Input: A tree graph,
A set of start nodes,
Boolean procedure *undercritical* (*n*), that tests if the *QoS* of a tree node '*n*' is under critical values.
Frontier: = {<*s*>: *s* is a start node};
Fatalist: = {<*r*>: *r* is a sub-service of weak *QoS*};
Filter (*x*): a procedure that removes the node duplications from arraylist *x*.
While *frontier* is not empty:
 Select and remove path <*n0*; ...; *nk*> from *the frontier*;
 If *undercritical* (*nk*)
 Add node *nk* to the *FatalList*
 Forevery neighbor *n* of *nk*
 Add<*n0*; ...; *nk*> to *frontier*;
Endwhile
Filter (*FatalList*)
Output: Return the filtered set of *FatalList*

The fatal service is a weak quality sub-service (its *QoS* is below the critical values), which causes weakness in the quality of the parent service. The existence of fatal sub service is sufficient for the service consumers not to select the parent service, because they pay their money for utilising an infected service. Thus, the *QoSDW* models added a new quality attribute in the selection process – the number of fatal sub-services. Indeed, if there

is a group of discovered services of equal *QoS* level, the service which has the least number of infected sub-services must be selected. In terms of infected services detection, the service designer is capable of rebuilding improved versions of these services, free of fatal sub-services. Also, if the *QoSDWAnalyser* reports are not helpful in selecting the best service, service consumers can apply their own queries on the Data Warehouse as described in the next section.

QoSDW MODEL BENEFITS

Because the *QoS* of sub-services are now accessible through *OLAP* queries, some hidden facts, about *QoS*, can be discovered. In the previous approaches, the discovered services are only qualified with no information about its internal sub services. Based on the *QoSDW* model, the weak sub-services which lead to bad parent service qualities can be studied and treated, in each part of the complex service, before the selection process. Compared with the traditional selection process, *QoSDW* is more advanced. Both service consumers and service providers can benefit from the *QoSDW* model. Service consumers are capable of applying a deep analysis concerning the service component before selection, using *QoSDW* Analyser reports and *OLAP* queries. The *QoSDW* is also beneficial for Cloud service companies, because service designers are capable of analysing the fatal sub-services that cause a weak service and redesigning a similar service with better quality.

In order to show the advantages of the *QoSDW model* from the queries prospective, we present an *OLAP* example, which is simulated as graph and algebraic queries. Consider a schema S , an *OLAP* Query over S is a triple $Q = (x, y, z)$, satisfying the following conditions:

- x and y are path expressions such that the source (x) = source (y) = *root object*.
- z is an operation over the target of y .
- The expression x will be referred to as the classifier of Q and the expression v as the measure of Q .

Figure 4.17 presents the *QoS* schema as an acyclic graph, such that the root is the object of an application, while the remaining nodes model the attributes of the object.

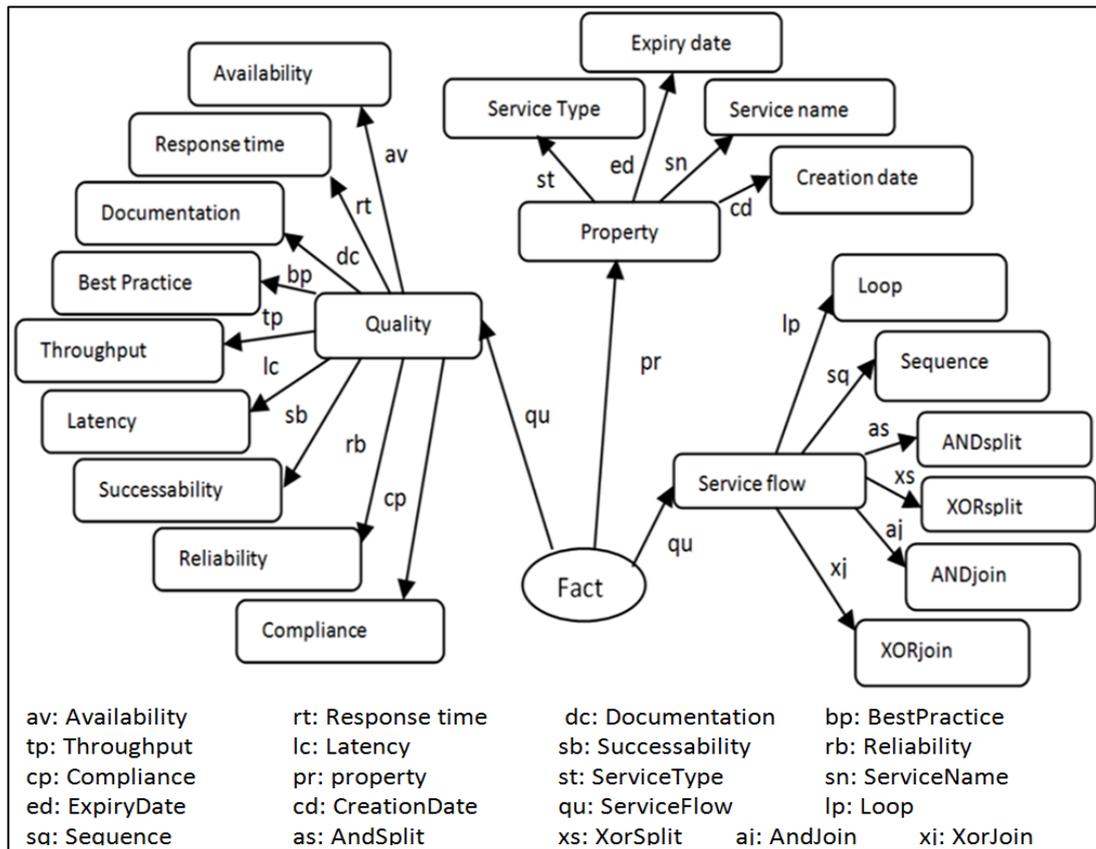


Figure 4.17 The proposed *QoS* schema

Through queries, some functions (such as *av*, *rt* and *dc*) are used when invoking object. Concerning the online *QoS* analysis through *QoS*, *OLAP* queries are prepared using paths starting at the root object (*Fact*). Through *OLAP*, service consumers can apply an advanced query such as:

QI: Ask for sub-services which utilize *XORjoin* integration when invoking other services and their *Response Time* greater than 80 (*ms*) sorted by name of service.

Let us divide the query *QI*:

- Ask for sub-services: *pro st.value == 'sub-service'*
- Which utilizes *XORjoin* integration when invoking other services: *quoxj*
- Their *Response Time* greater than 80 (*ms*): *quort. value>80*
- Sorted by name of service: *(prosn)^ (pr o st.value == 'service')*

QI = < *(prosn)^ (pr o st.value== 'service')*, *((pro st.value== 'sub-service') ^ (quoxj) ^ (quort. value>80))*, *sum*>

4.4.7 OUTPUT BUILDER SYSTEM (OBS)

Cells in COC are considered as intelligent modular applications that can be published, located and invoked across the Web. They are intended to give the client best results by composing their distributed business processes dynamically and automatically based on specific rules. Based on the service model, companies only implement their core business and outsource other application services over the Internet. However, no single Web service can satisfy the functionality required by the user; thus, companies try to combine services together in order to fulfil the request. Indeed, companies face a major problem: Web service composition is still a highly complex task and it is already beyond human capability to deal with the whole process manually. “The complexity, in general, comes from the following sources. First, the number of services accessible over the Web

has increased radically during recent years and a huge Web service repository to be searched is anticipated. Second, Web services can be formed and updated during normal processing; thus the composition system needs to detect this updating at runtime and make decisions based on the up-to-date information. Third, Web services can be developed by different organizations, which use different conceptual models to describe the services; however, there is no unique business process to define and evaluate Web services. Therefore, building composite Web services with an automated or semi-automated tool is critical” (Portchelvi *et al.*, 2012).

As an approach to the service composition problem, cell theory proposes a cell that is capable of achieving an automated composition of its business process. At instant, after analyse, validation and ensure the good characteristics of business process choices to be used by a cell by PAS, PVS and TMS, OBS selects and executes the best process plan based on the user’s request. The role of OBS is to apply a dynamic and autonomic pruning of the selected business processes of the collaborating cells.

The data related to virtually all features of stored proposed genes is a valuable resource if the right tools are available for putting it to use. Machine learning algorithms are a set of techniques that automatically build models describing the structure at the heart of a set of data. Such models have two important applications. First, if they accurately represent the structure underlying the data, they can be used to predict properties of future data points. Second, if they summarize the essential information in human-readable form, people can use them to analyze the domain from which the data originates. To be useful for analysis, a model must be an accurate representation of the domain. To avoid

superfluous complexity, an efficient mechanism is needed for determining when a particular effect is due to chance alone. Given such a mechanism, those parts of a model that describe chance effects can be eliminated. The process of cutting off non-predictive parts of a model is called “pruning.” To prune means, among other things, “to remove as superfluous.” By removing superfluous structure, pruning mechanisms reduce the size of a model and often improve its accuracy. Ideally, pruning should only discard those parts of a model that are due to noise, and never eliminate any structure that is truly predictive. This decision must be based on the data at hand, such as the proposed genetic data stored in the Cloud brain.

4.5 CONCLUSION

Intelligent distributed computing will become the base for the growth of an innovative generation of intelligent distributed systems. Nowadays, research centers require the development of architectures of intelligent and collaborated systems; these systems must be capable of solving problems by themselves to save processing time and reduce costs. Cell oriented computing is a demonstration of human cell characteristics from the computer science viewpoint. It is a flexible and scalable virtual processing unit that treats intricate distributed computing by structured and precise decisions. The cell computing imitates the human cell situation in the distributed systems world.

Chapter 5

VALIDATION; A CASE STUDY

“Anonymous email is both very easy do to, and yet also extremely difficult. The level of difficulty involved depends on the chance that someone would go the extra mile to identify you” (Notenboom, 2005). In order to show the importance of SmartCells approach, this chapter discusses an Identity Cell scenario as a technique that contribute in solving the anonymous email problem. The proposed technique is summarized by automatic identity detector service that aggregate the sender geographical context-profile to his message. Simulations applied through this scenario show the differences between the Service-Oriented and Cell-Oriented approaches.

5.1 CASE STUDY: AN IDENTITY DETECTOR CELL OUTLINES

In the early Internet period, one of the key features was anonymity. No one online knew who you were if you did not want them to. Naturally, this causes some problems with trolls and other troublemakers ruining online discussions, sending hateful emails to people and generally being unlikable. Nowadays, anonymity is a little tougher to come by, because a quick Google search turns up your entire life. So, the anonymity is gone, but somehow the unlikable people are still with us and causing problems. Still, being anonymous is not impossible. In fact, if you want to send an anonymous email or message,

it is entirely possible. One of the good features about free email from Google, Yahoo, Outlook and other providers is that you can have as many accounts as you want. In addition, there are few services that let you send and receive email without giving out any personal information. For sending email you just put in: the recipient's email address, the subject and the message, while it is impossible for anyone to reply to your email.

Every day thousands of anonymous emails flow through web networks, some of them cause problems for email users. One way that many people advocate sending anonymous communications is via an anonymous proxy or gateway. In addition to actual networking proxy services there are online services that will send messages on your behalf, presumably without any information that could be used to personally identify you. The biggest concern with any anonymous proxy or service is security and privacy.

In fact, correlating the characteristics of one anonymous email message with an email message from a known source is the traditional common way to identify the source of the message. Throughout this chapter, I suggest to insert geographical information about sender inside the anonymous message. Consequently, an email receiver deals with anonymous message based on the sender context-profile.

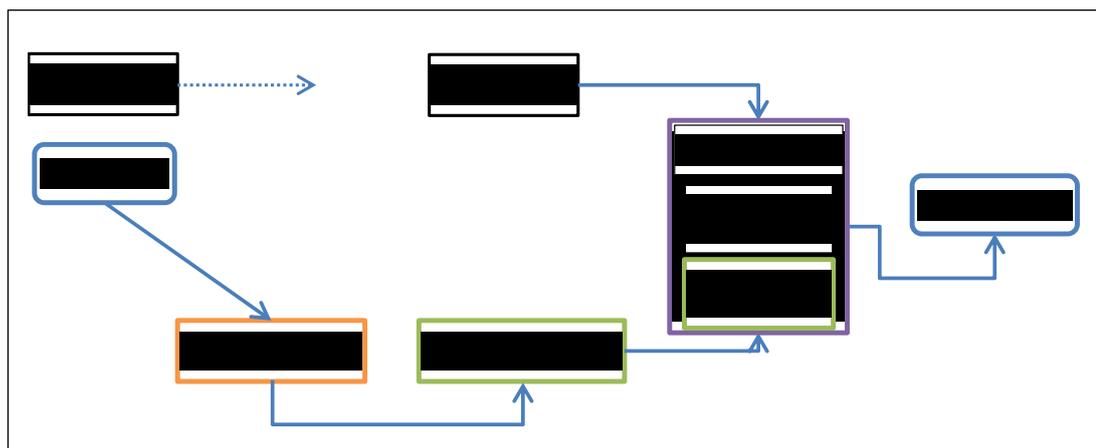


Figure 5.1: The process of sending a verified anonymous email

Briefly, as shown in figure 5.1, an anonymous email sender is monitored by his closest mail server. When a message is prepared to be sent from such machine, a cloud service localized on the nearest server records the IP address of that machine. Based on the IP address, a second cloud service provides a set of geographical properties about the sender. Later, a third cloud service plays a final role by sending the email composed of the actual message and the geographical context.

The main goal of this chapter is build a SmartCells simulation and to reach this goal I show, through the next sections, how to build the Identity Mail Cell. Also, I compare the proposed Cell-Oriented approach to the Service-Oriented approach through two different simulations.

5.2 IMPLEMENTATION: TOOLS AND PLATFORMS

Throughout the SmartCells simulation, I have used several machines, tools and coding libraries in order to show beneficial results.

The machines used in this simulation are:

- *A Thinkpad Desktop Lenovo PC, Intel(R) Core(TM) i3 CPU, 8.00 GB RAM, 64-bit Operating System, Windows 7.*
- *A Thinkpad Lenovo tablet x201, Intel(R) Core(TM) i7-640LM CPU, 4.00 GB RAM, 64-bit Operating System, Windows 7.*

The tools used in this simulation are:

- *Apache HTTP Web Server Version 2.4*
- *MySQL Server 5.6*
- *Microsoft SQL Server 2012*
- *Ucinet version 6*
- *Eclipse for PHP Developer*
- *Google Chrome*

- *Server Monitor program*
- *Notepad++*
- *Msodbcsql*
- *sqlncli_x64*

The libraries used in this simulation are:

- *php_pdo_sqlsrv_55_nts.dll*
- *Nusoap PHP Library*

The Web services used in this simulation are:

- *ExternalAddress web service, source: <http://icanhazip.com>*
- *RemoteAddress web service, source: <http://icanhazip.com>*
- *RealIP web service, source: <http://localhost/>*
- *DBIP_Client web service, source: <http://api.db-ip.com/>*
- *Geoplugin web service, source: <http://www.geoplugin.net/php.gp>*
- *Melissa web service, source: <https://iplocator.melissadata.net/v2/SOAP/Service.wsdl>*
- *PHPMailer web service, source: <https://github.com/PHPMailer/PHPMailer>*
- *Swiftmailer webservice, source: <https://github.com/swiftmailer/swiftmailer>*
- *LocalMail web service, source: <http://localhost/myworks/Simulation/SendMail/MailwithLocalPHP/MailwithLocalPHP.php>*
- *Google SMTP Cloud service, source: www.google.com*
- *Yahoo SMTP Cloud service, source: www.yahoo.com*
- *Hotmail SMTP Cloud service, source: www.hotmail.com*

The analysis algorithms used in this simulation are:

- *Breadth-first search (BFS) algorithm that is a graph search algorithm that traverse graph shortest paths.*
- *Dijkstra's algorithm solves the shortest path from one node to all the other nodes in a weighted graph with no negative weight edges.*
- *A topological ordering algorithm for Direct Acyclic Graph. It is an ordering of the DAG's nodes, such that each node comes before all nodes to which it has outbound edges.*
- *Prims's algorithm that finds the minimum spanning tree of a graph.*

- *Degree centrality algorithm that detect the most important node in a Gene map.*
- *Closeness centrality algorithm that shows which Gene node is closer to more nodes than any other node.*
- *Betweenness centrality algorithm that views a node as being in a favored position to the extent that the node falls on the geodesic paths between other pairs of nodes in the network.*

The real implementation of SmartCells requires building a data warehouse to monitor quality of Gene's changes. In this simulation response time is studied as parameter of Gene's quality, therefore I show in a simple way how Cells subsystems apply analysis on Gene databases. The used database tools are MySQL server and Microsoft SQL server 2012. To study Cell's response time, the SmartCells PHP code connects to the Server_Monitor database on the MySQL server. While to study the Cell's performance based on distance factor, the SmartCells PHP code connects to the Cell_algorithms database on the SQL server.

5.3 SERVICE-ORIENTED SIMULATION

The Service-Oriented approach follows the web standard that requires three main components: Service Client, Service Registry and Service Provider. Service composition steps are done on the provider side, and then a meta-data about the service are stored at the Registry side. When a client wants to utilize a service, a Registry sends a meta-data of a set of discovered services. Based on *QoS* criteria, the client select the best service to be invoked. Services are ranked in many levels, such as Poor, Good and Excellent. It is based on Web Service Relevancy Function (WsRF), which is measured based on the weighted mean value of the *QoS* parameters. Services are classified according to user's invocations as follows:

- **Excellent:** users accept to pay lower cost regarding better service qualities.
- **Good:** users pay normal cost for normal service qualities.
- **Poor:** users accept worse cost with lower service qualities.

CLOUD SERVICES:

In our case, we have three different cloud services that can return the sender's geographical identity via email. The three used services are:

- ProfileInMail service that is marked as good service and figure 5.2 shows the code of its business process.

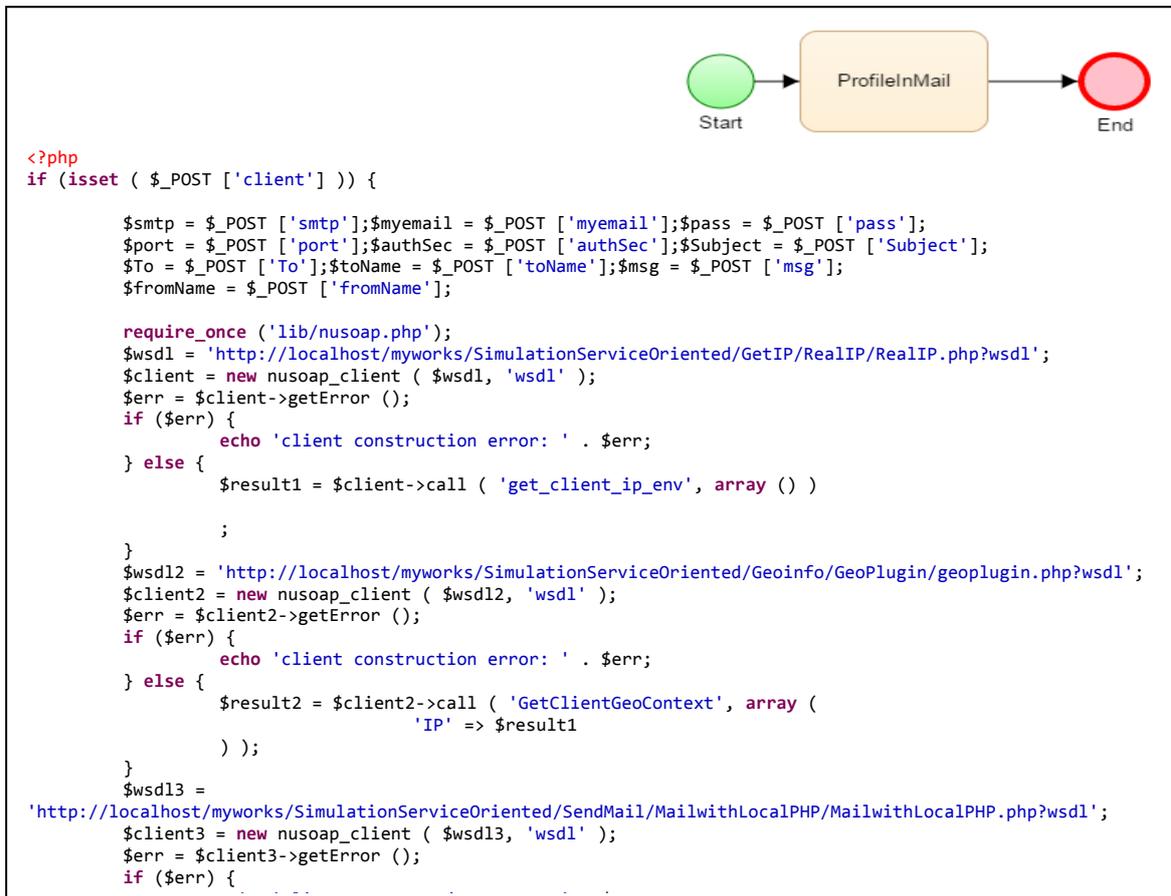


Figure 5.2 : ProfileInMail service process description

- VerifyMailer service that is marked as excellent cloud service and figure 5.3 shows the code of its business process.

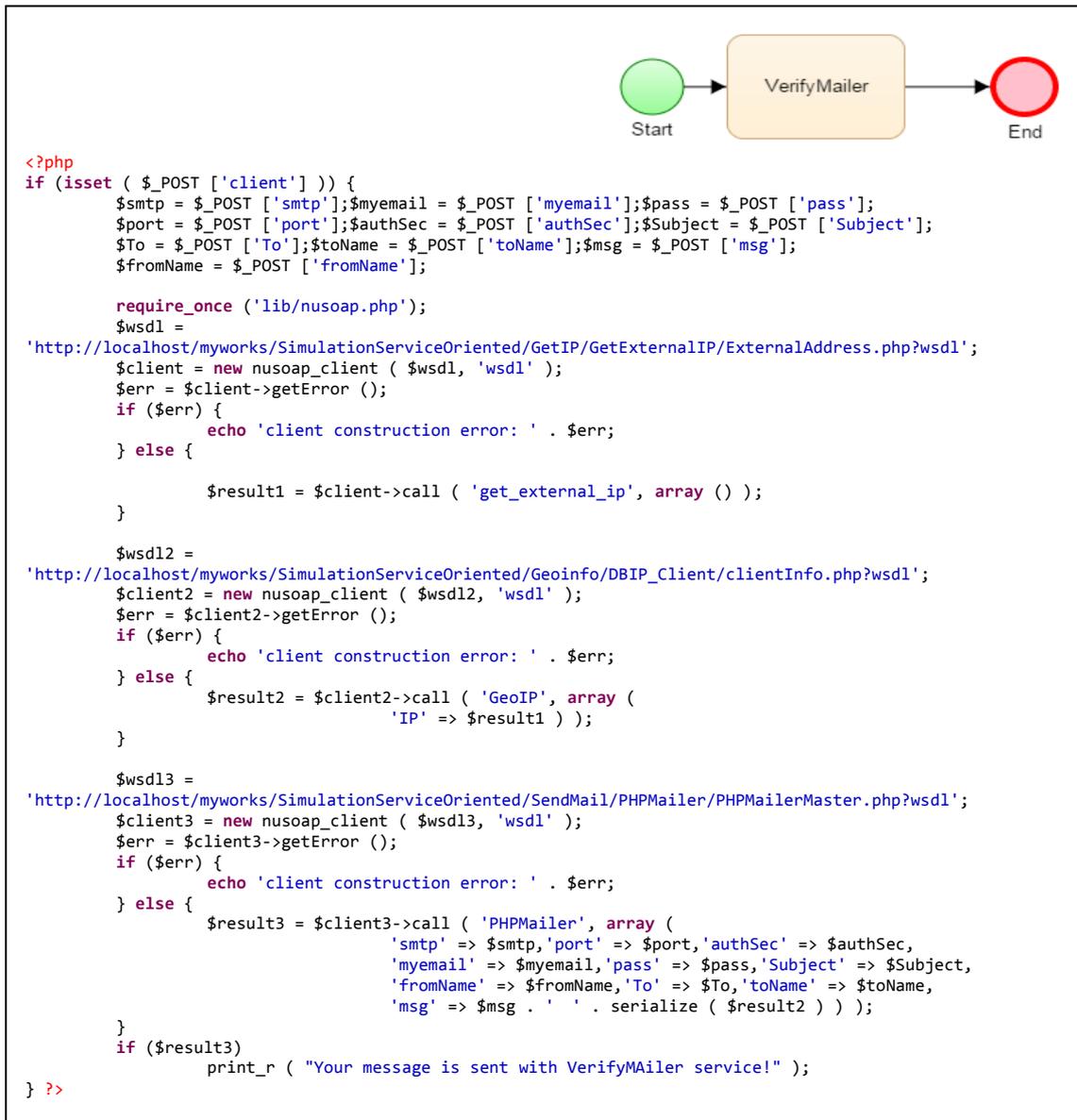


Figure 5.3: VerifyMailer service process description

- SenderDetector service that is marked as good cloud service and figure 5.4 shows the code of its business process.

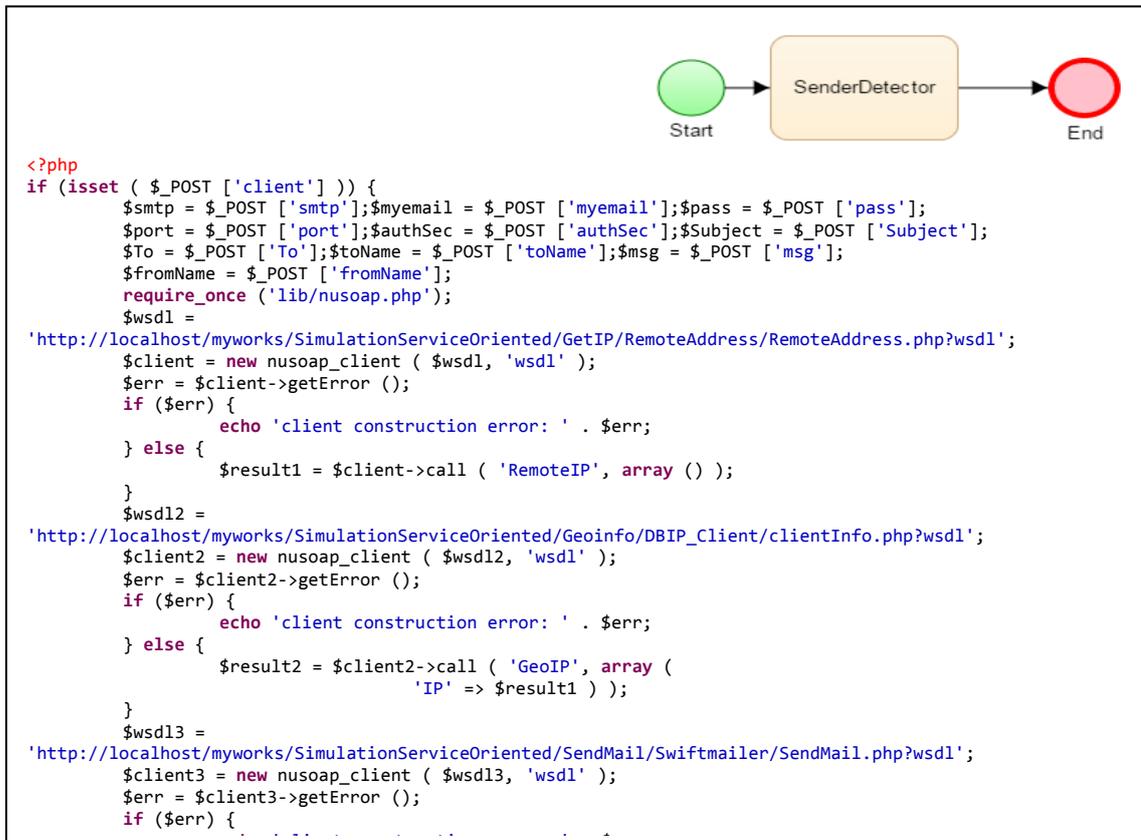


Figure 5.4: SenderDetector service process description

SERVICE SELECTION:

In order to show how the Service-Oriented approach works, Latency (The period of time that one component in a system is spinning its wheels waiting for another component) is used as *QoS* criteria. The response time of each used cloud service is studied for a specific period and the Service monitor returned the following results:

- Figure 5.5 shows the ProfileInMail service Latency during a period of one hour (12:55 to 13:55). The graph shows the variation of Latency as a function of time. The average Latency reached is 1.4859 with 100% uptime.

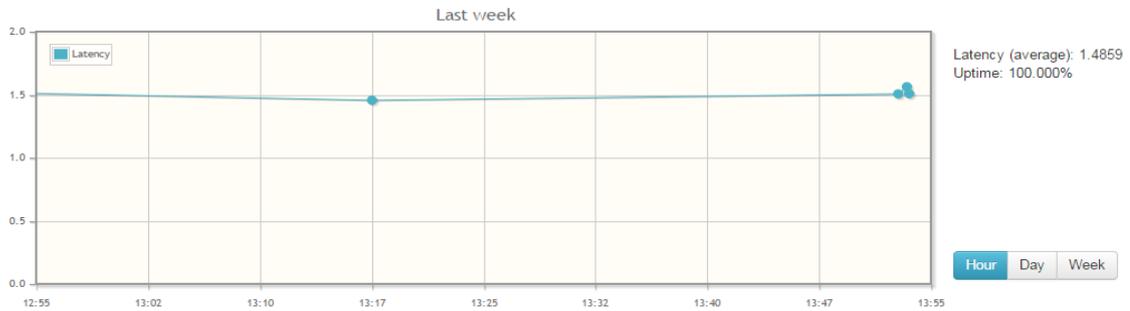


Figure 5.5: The variation of ProfileInMail latency as a function of time

- Figure 5.6 shows, also, the VerifyMailer service Latency. The average Latency reached is 1.4605 with 99.997% uptime.



Figure 5.6: The variation of VerifyMailer latency as a function of time

- Figure 5.7 shows, also, the SenderDetector service Latency. The average Latency reached is 1.5292 with 100% uptime.

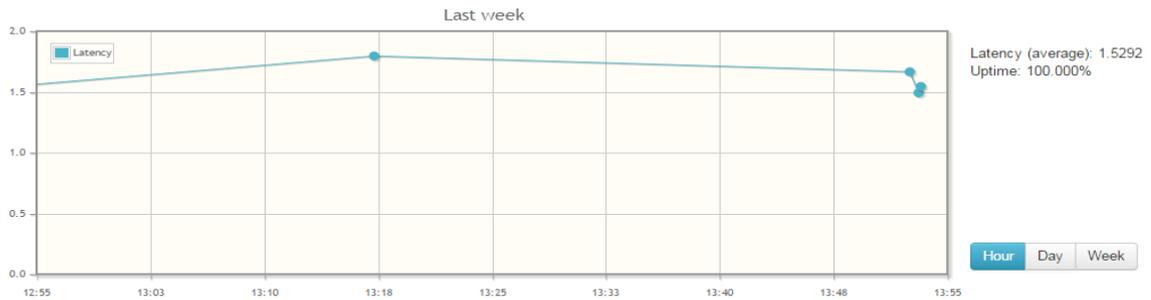


Figure 5.7: The variation of SenderDetector latency as a function of time

Based on the QoS variation, the best service is selected. As known, the best service has the minimum response time consequently the least Latency. In our example, the best service is VerifyMailer that has the least Latency value. Thus, a client selects the VerifyMailer service, as a best choice, to be used in his process.

CLOUD SERVICE INVOCATION

After the termination of the selection process, the client uses the description given from the Registry side about to invoke a VerifyMailer service. Figure 5.8 shows a web page that contains a description about VerifyMailer service and an online invocation form. The filled form means that a sender (of email: ahmad_karawash@hotmail.com) sends an email to a receiver (of email: ahmad.karawash1@uqac.ca). As a hidden step, the VerifyMailer service collects the Geo-profile of the sender and integrates it to his message.

Service Oriented Simulation

Current Cloud service Strategy

Service Name	VerifyMailer
Service Description	Service job is to send mail composed of a message and sender Geo context profile.
Input:	Service Complextype:'smtp' => 'xsd:string', 'port' => 'xsd:integer', 'authSec' => 'xsd:string', 'myemail' => 'xsd:string', 'pass' => 'xsd:string', 'Subject' => 'xsd:string', 'fromName' => 'xsd:string', 'To' => 'xsd:string', 'toName' => 'xsd:string', 'msg' => 'xsd:string'
Output:	string: result

smtp server	smtp.gmail.com	authSec	tls	port	587
email	ahmad.karawash@gmai	password	Subject	salam
fromName	ahmad	To	ahmad.karawash1@uqa	toName	ahmad

Hi,

This is the service oriented simulation of the cloud service way of work.

Regards,
Ahmad Karawash

[Invoke »](#)

Copyright© 2015 Ahmad Karawash

Figure 5.8: Invoking form of VerifyMailer

The result of this simulation is an email containing the sender message and his Geo-Profile. The online result of this simulation is shown in figure 5.9 below.

IMPLEMENTATION RESULTS

The prepared SmartCells website provides a simple use of Cells and its content is understandable and navigable. It includes not only clear and simple forms, but also providing understandable mechanisms for navigating within and between pages. Not all users can make use of Cells randomly but each user should pass the registration step to have access to his account. Users provide some of their contextual information through registration while provider job is to discover the full context profile of the user. Figure 5.10, shows welcome page of the SmartCells website. This website is based mainly on PHP, JavaScript, HTML, MySQL and SQL coding languages.

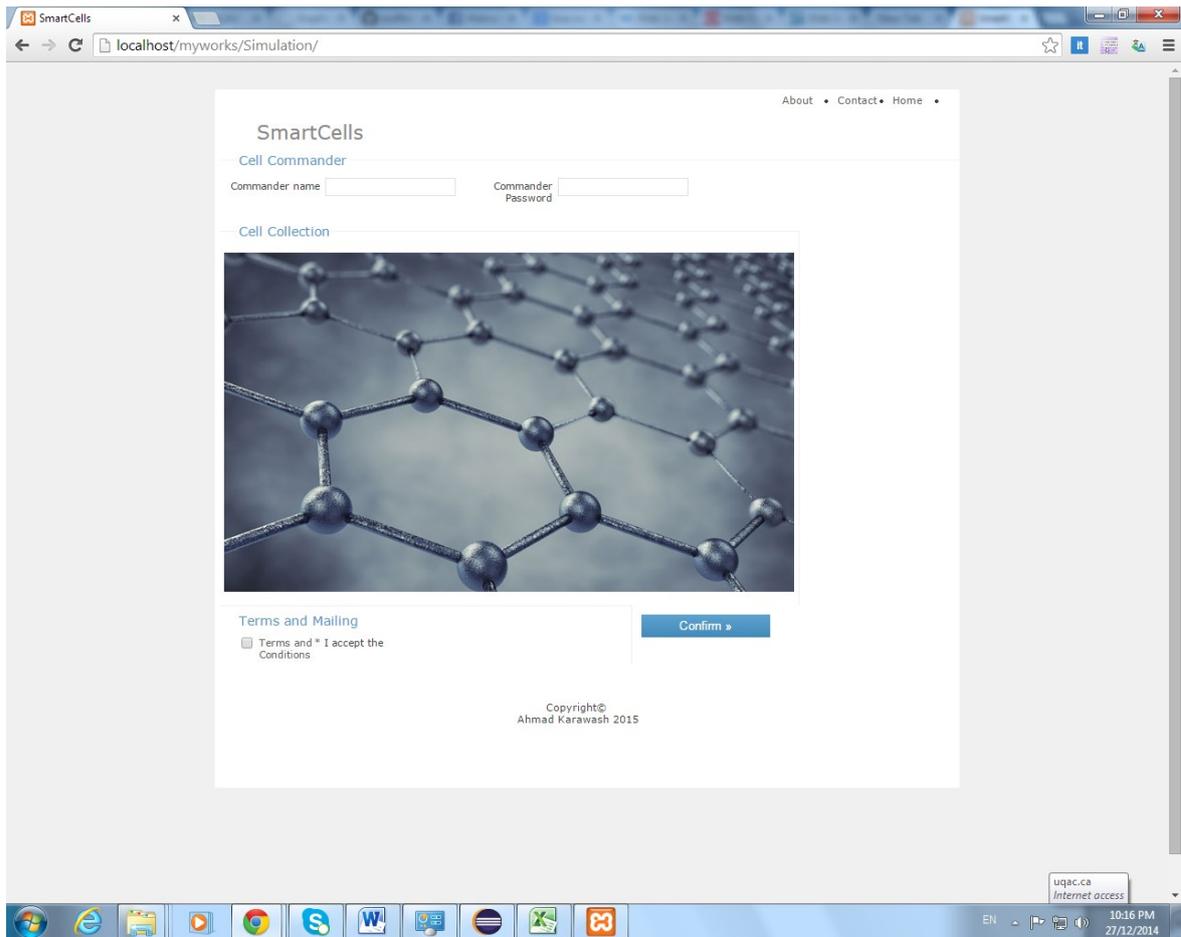


Figure 5.10: SmartCells website

Currently as shown in the SmartCells selection page in figure 5.11, the SmartCells website covers four type of Cells of names: IdentityMail, GetIP, GetGeoProfile and SendMail.

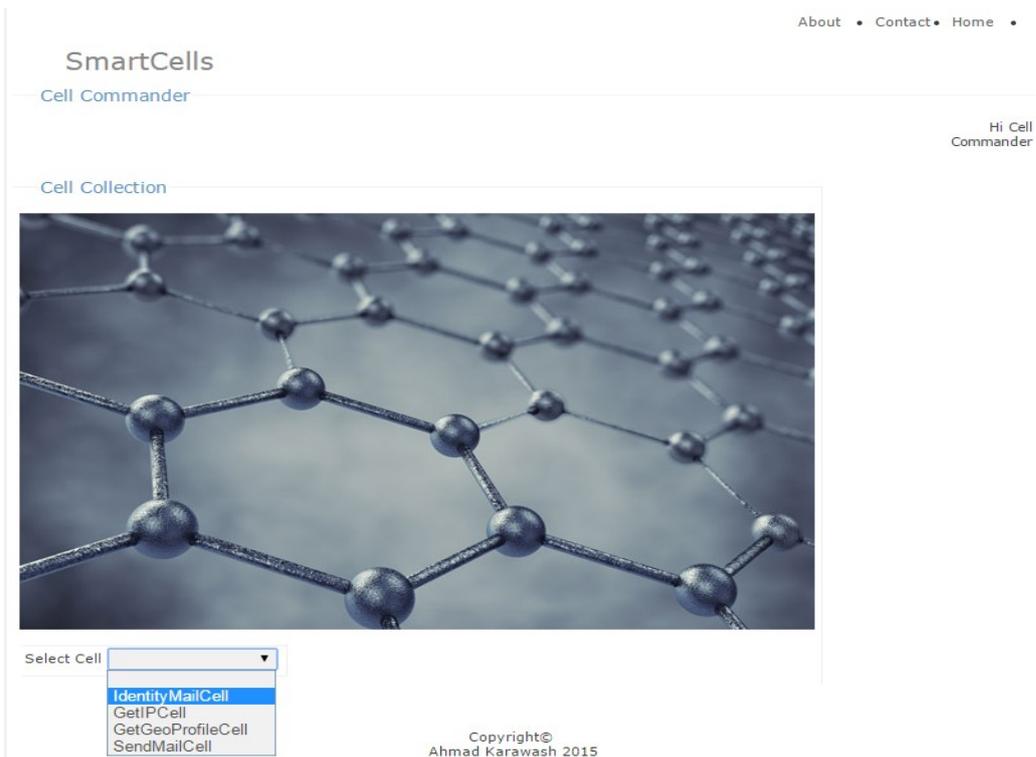


Figure 5.11: SmartCells selection method

IDENTITYMAIL CELL:

IdentityMail Cell is responsible of providing a “*secure*” email based on the context-profile of the Cell commander. Figure 5.12 shows an online interface of commanding IdentityMail Cell. Through that interface commanders insert a set of creditional inputs as: email, password, etc and the email content.

About • Contact • Home •

SmartCells

Cell Details

Cell Name	IdentityMailCell
Cell Description	GetIPCell job is to send mail composed of a message and sender Geo context profile.
Input:	IdentityMailCell Complextype:'smtp' => 'xsd:string', 'port' => 'xsd:integer', 'authSec' => 'xsd:string', 'myemail' => 'xsd:string', 'pass' => 'xsd:string', 'Subject' => 'xsd:string', 'fromName' => 'xsd:string', 'To' => 'xsd:string', 'toName' => 'xsd:string', 'msg' => 'xsd:string'
Output:	string: result

smtp server	smtp.gmail.com	authSec	tls	port	587
email	ahmad.karawash@gmai	password		Subject	salam
fromName	ahmad	To	ahmad.karawash1@uqa	toName	ahmad
msg	<pre> Hello sir, This is the result of my simulation. Regards, Ahmad Karawash. </pre>				

Command »

Copyright© Ahmad Karawash 2015

Figure 5.12: Commander's page of IdentityMail Cell

The result of commanding the IdentityMail Cell is an email composed of :
 Commander message and Commander geographical context-profile as shown in figure
 5.13.

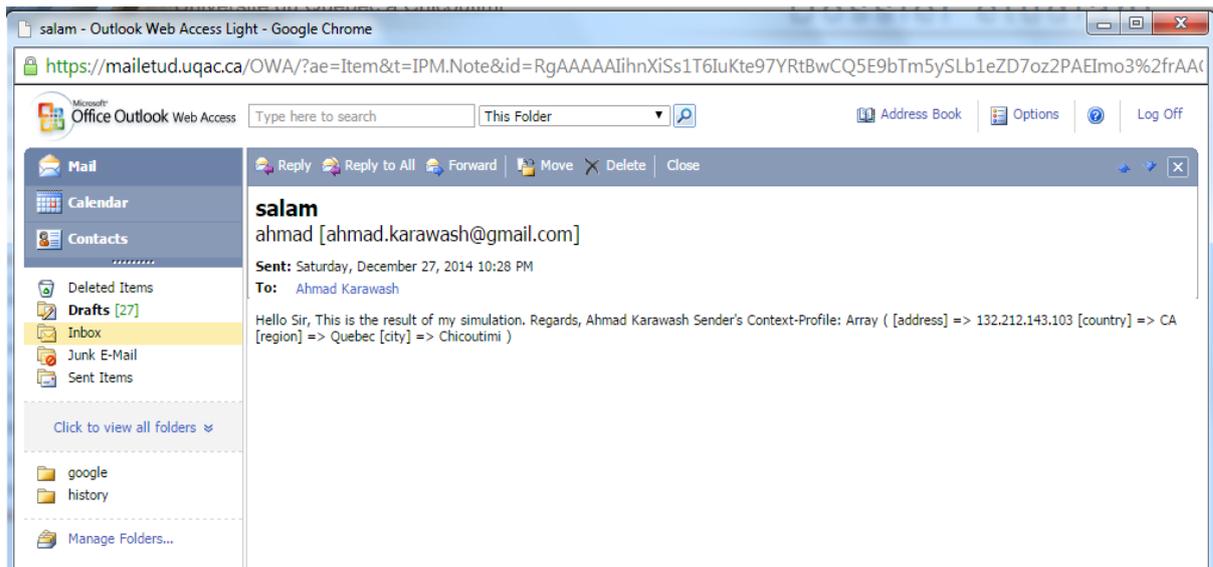


Figure 5.13: Result of Commanding IdentityMail Cell

GETIP CELL:

GetIP Cell detects the real IP of the Commander on the web network. Figure 5.14 below shows the web interface used to command this Cell.

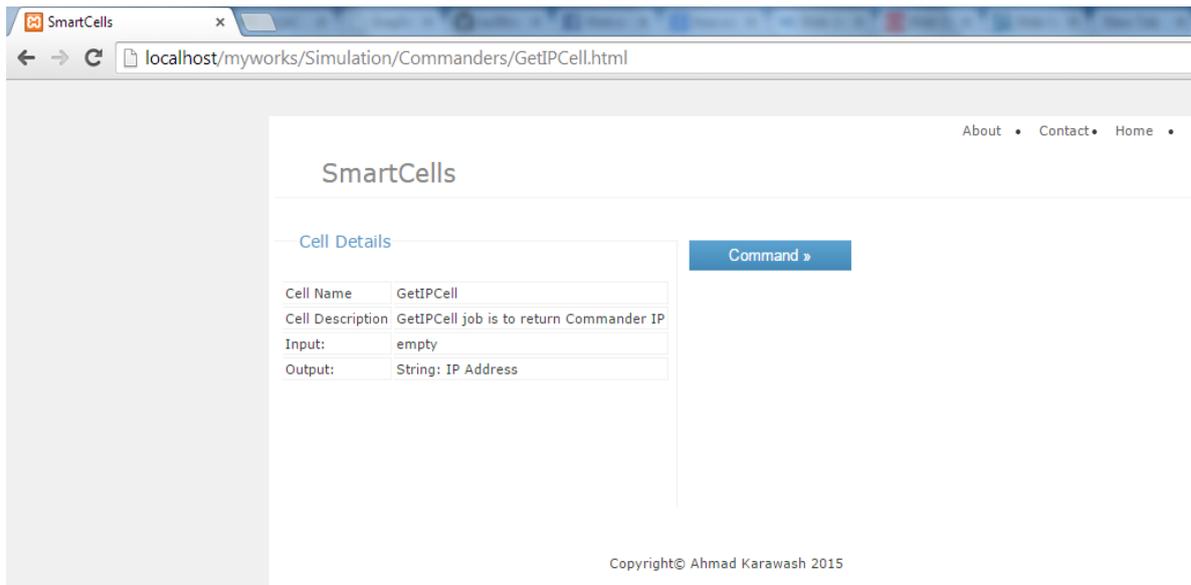


Figure 5.14: Commander’s page of GetIP Cell

As a result of commanding GetIP Cell, figure 5.15 shows the output result that is consist of client IP address.

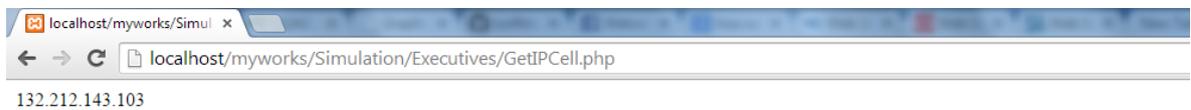


Figure 5.15: Result of Commanding GetIP Cell

GETGEOPROFILE CELL:

GetGeoProfile Cell provides a set of geographical information about the Cell Commander. It depends mainly on IP to analyse the Commander’s geographical characteristics. Figure 5.16 represent the web interface used to command GetIP Cell.

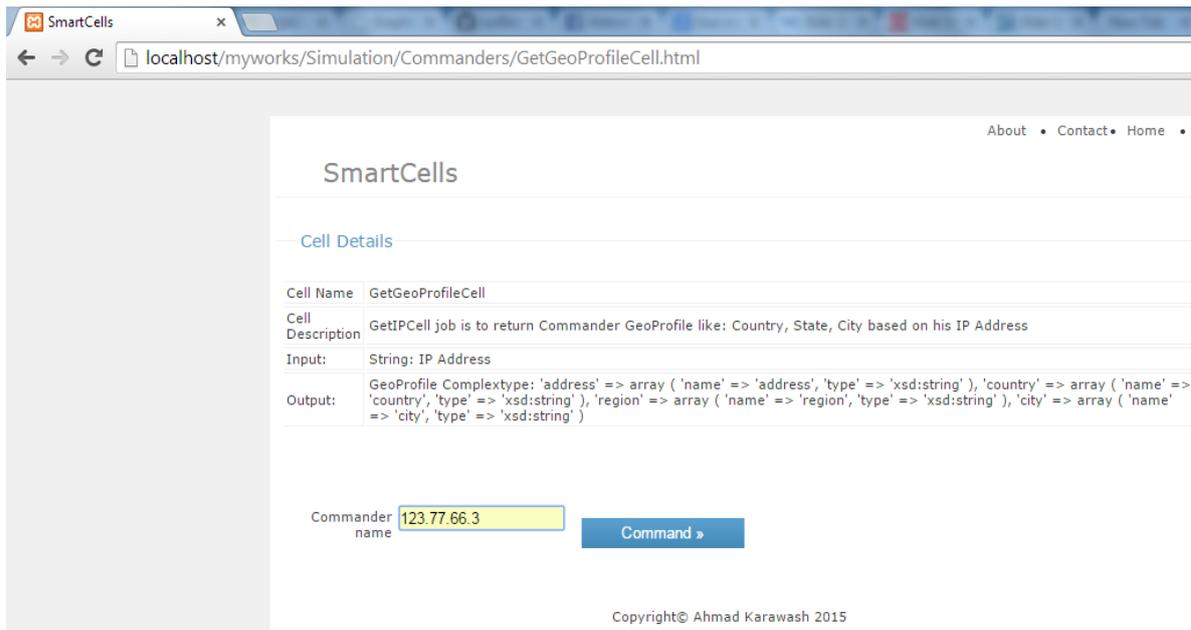


Figure 5.16: Commander’s page of GetGeoProfile Cell

As shown in figure 5.17, the output of GetIP Cell is an array of four components: address, country, region and city.

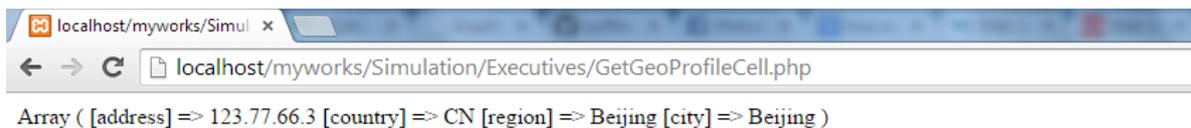


Figure 5.17: Result of Commanding GetGeoProfile Cell

SENDMAIL CELL:

SendMail Cell is responsible of sending anynomous emails based on PHP code in colloboration with cloud mail vendor such as Google SMTP, Yahoo SMTP or Hotmail SMTP. Figure 5.18 shows an online interface of commanding SendMail Cell. As in

IdentityMail interface, commanders insert a set of credtional inputs as: email, password, etc and the email content.

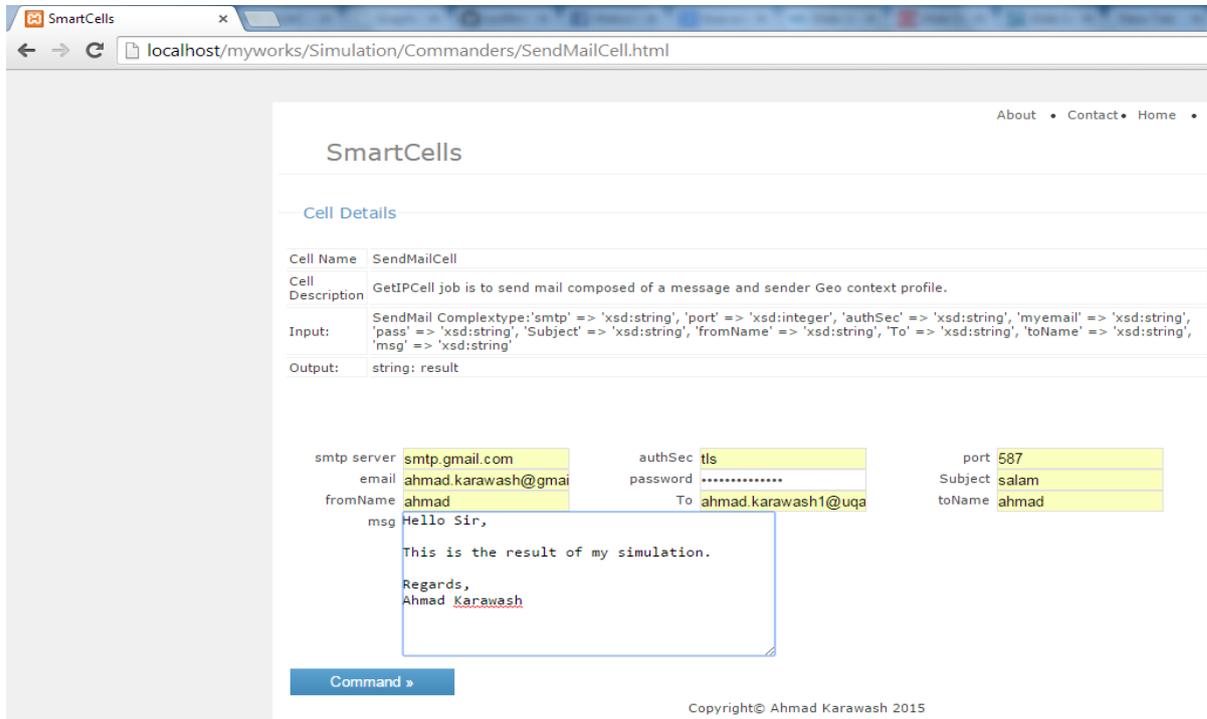


Figure 5.18: Commander’s page of SendMail Cell

As shown in figure 5.19, the result of commanding SendMail Cell is an anonymous email that has no information about the sender.

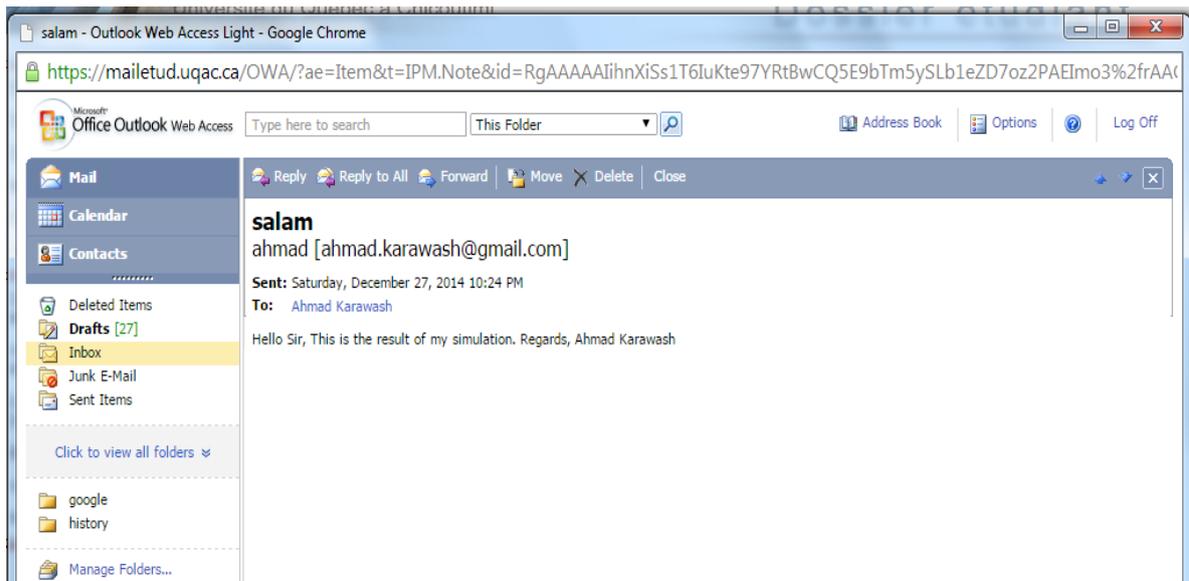


Figure 5.19: Result of Commanding SendMail Cell

PROCESS ANALYSIS AND SELECTION

In contrast to Service-Oriented approach that is based on the quality of service, the Cell-Oriented approach goes further to analyse the quality of subservices. If the same services, which used in the Service-Oriented simulation, are studied, the quality of subservices reflects a better analysis of parent services. To show one of the composition strategies of SmartCells, let us study again the same services but with the analysis of their

subservices.



Figure 5.20: Analysis of SenderDetector service process based on quality of subservices

As shown in figure 5.20, SenderDetector is composed of three subservices defined by their names as follows: RemoteAddress, ClientGeoInfo and SwiftMailer.

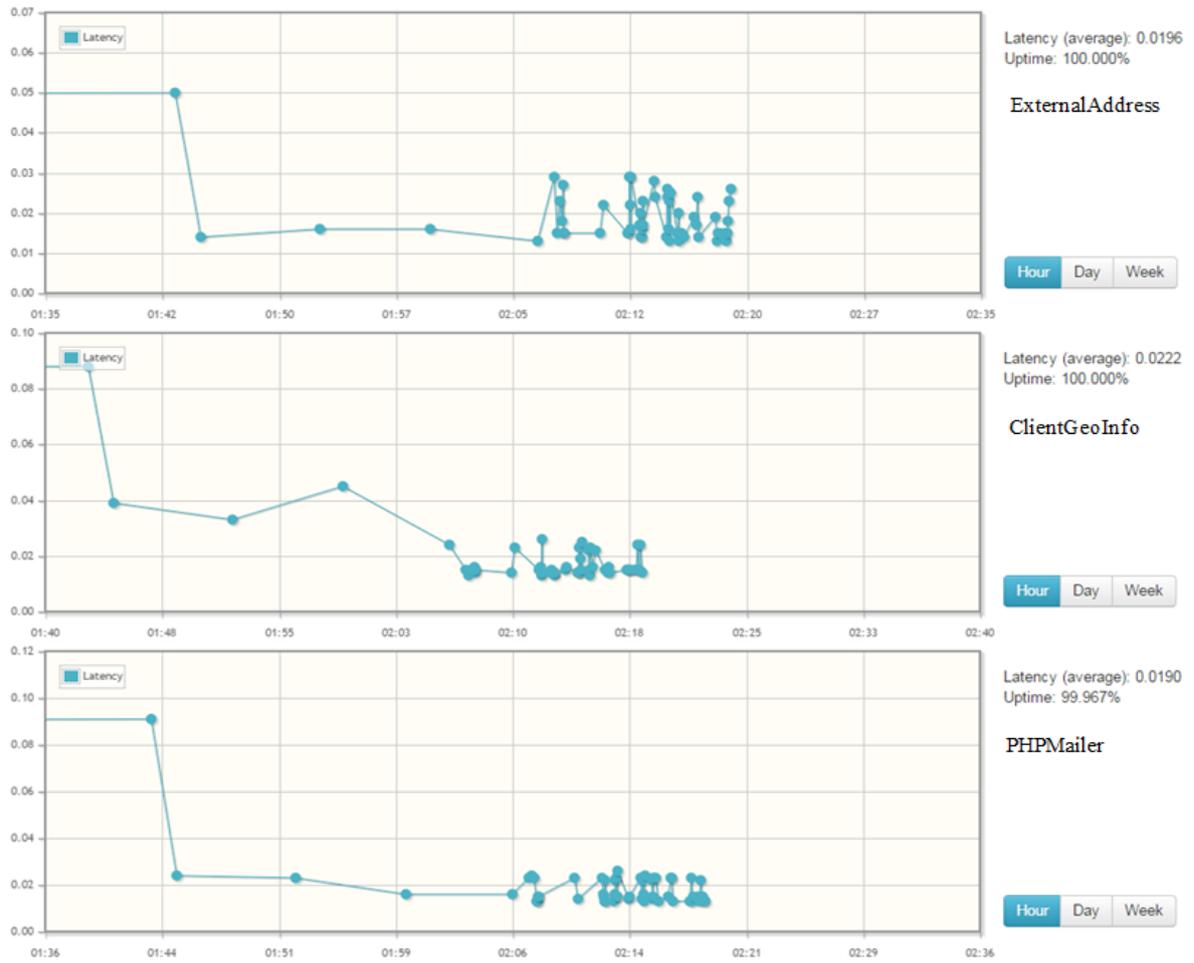
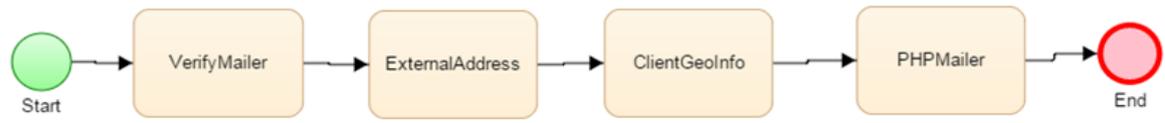


Figure 5.21: Analysis of VerifyMailer service process based on quality of subservices

As shown in figure 5.21, VerifyMAiler is composed of three subservices defined by their names as follows: ExternalAddress, ClientGeoInfo and PHPMailer.



Figure 5.22: Analysis of ProfileInMail service process based on quality of subservices

As shown in figure 5.22, ProfileInMail is composed of three subservices defined by their names as follows: RealIP, GeoPlugin and MailLocalPHP.

As discussed in Chapter 4 (section 4.4.6), we can detect the weakness points of the composition of each service using on the quality of subservices.

GENE MAP

In SmartCells, the cloud services are replaced by a cloud Cells and Cell process is defined by a Gene. The main characteristic of Cell is the uniqueness in which there are no different Cells provide the same type of service. Also, Genes map is dynamically changed in order to “*provide permanent availability and best performance*”. Each Gene covers all web and cloud components that give the same type of service. Regarding the anonymous email scenario we deal with four types of Cells as follows: GetIPCell, GetGeoProfileCell, SendMailCell and IdentityMailCell.

GetIPCell: The job of this Cell is to return the real IP address of the commander. It takes no input and it returns a string IP address. As shown in figure 5.23, it covers, in its Gene, all the processes cloud services that return the IP

address.

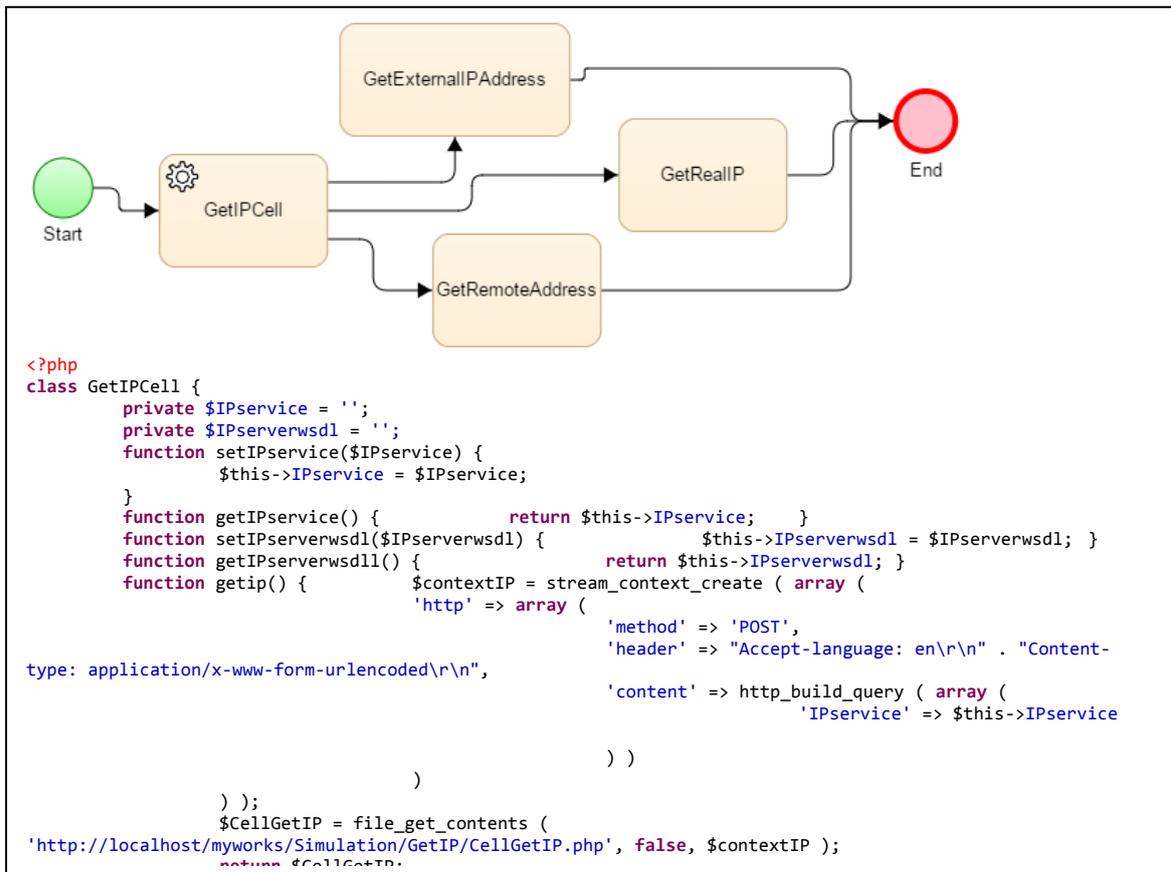


Figure 5.23: Gene map of the GetIP Cell

GetGeoProfileCell: The job of this Cell is to return the geographical context-profile of the commander. It takes the IP address as input and it returns a String array of geographical information. As shown in figure 5.24, it covers, in its Gene, all the processes cloud services that return the Geo-profile information.

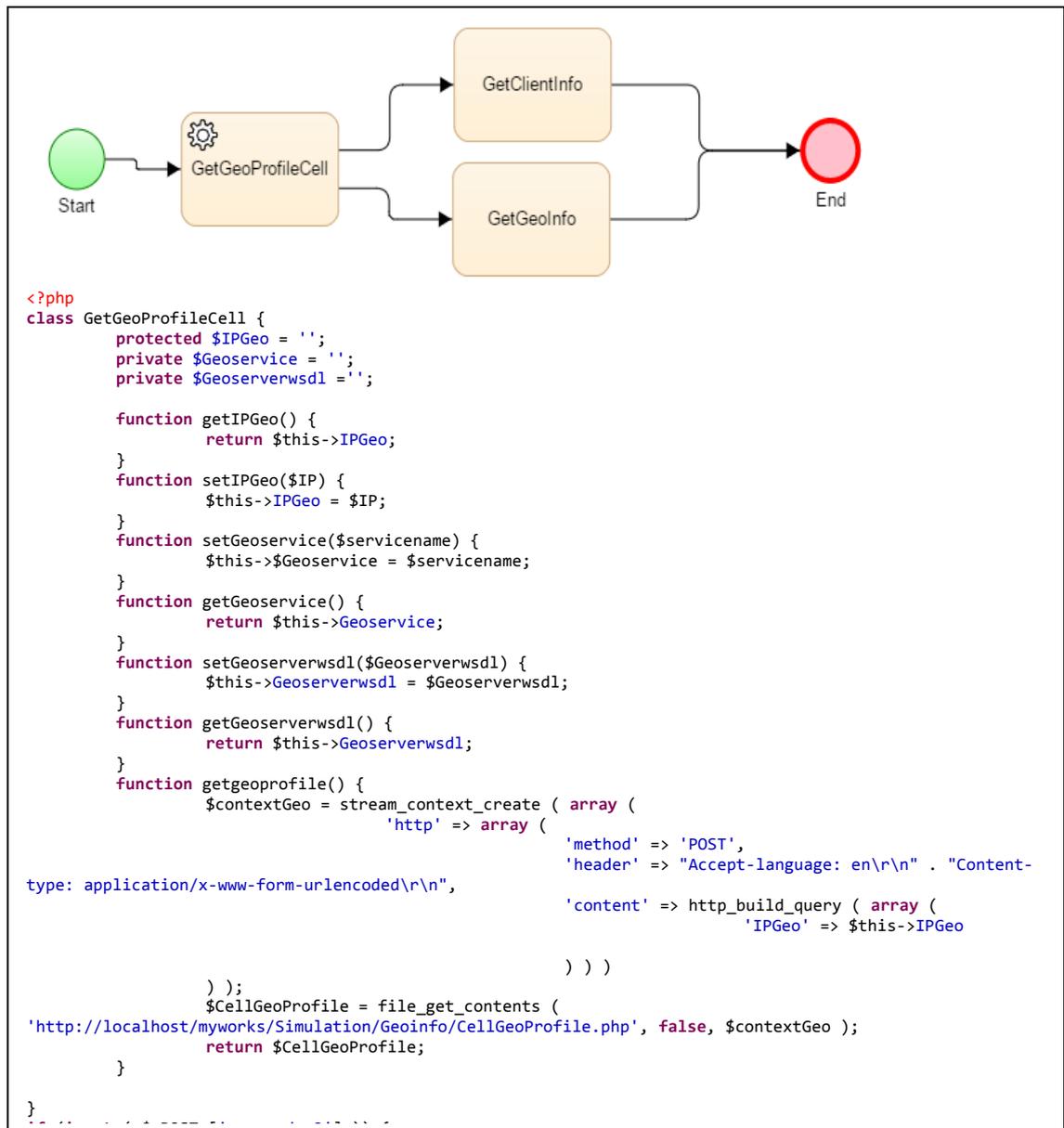


Figure 5.24: Gene map of the GetGeoProfile Cell

SendMailCell: The job of this Cell is to send anonymous email. It takes a set of String parameters as input as follows: SMTP server name, valid email feeds the SMTP server, password, port number, authentication method, subject of message, receiver email, receiver person name, message, and sender name. As shown in figure 5.25, it covers, in its Gene; all the processes cloud services that send anonymous emails.

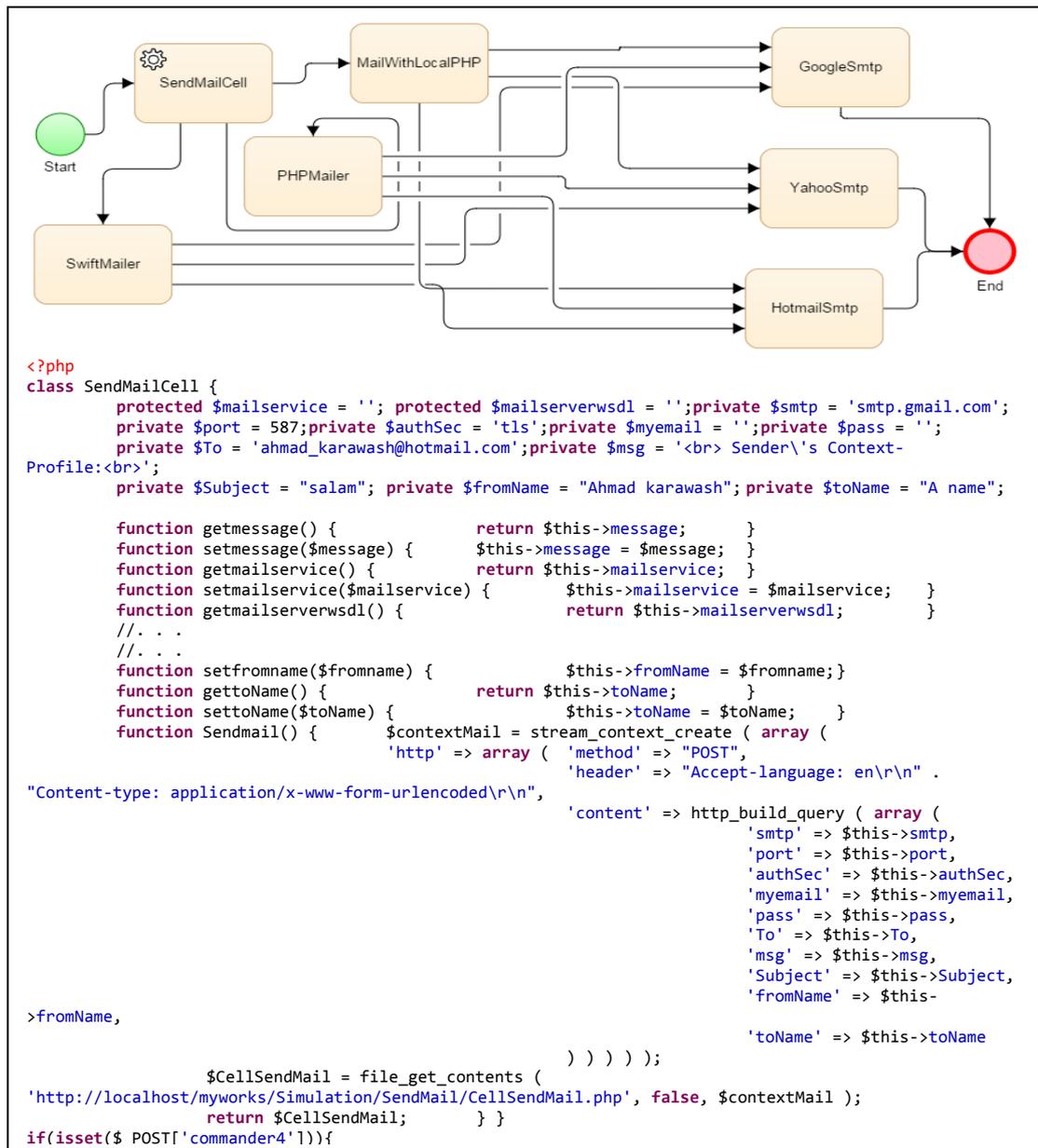


Figure 5.25: Gene map of the SendMail Cell

IdentityMailCell: The job of this Cell is to send anonymous email Integrated with a commander geo-profile. It takes the same set of String parameters of SendMailCell as input while it adds the commander geographical identity to the message. As shown in

figure 5.26, this Cell depends on collaboration with the other three Cells to complete its job.

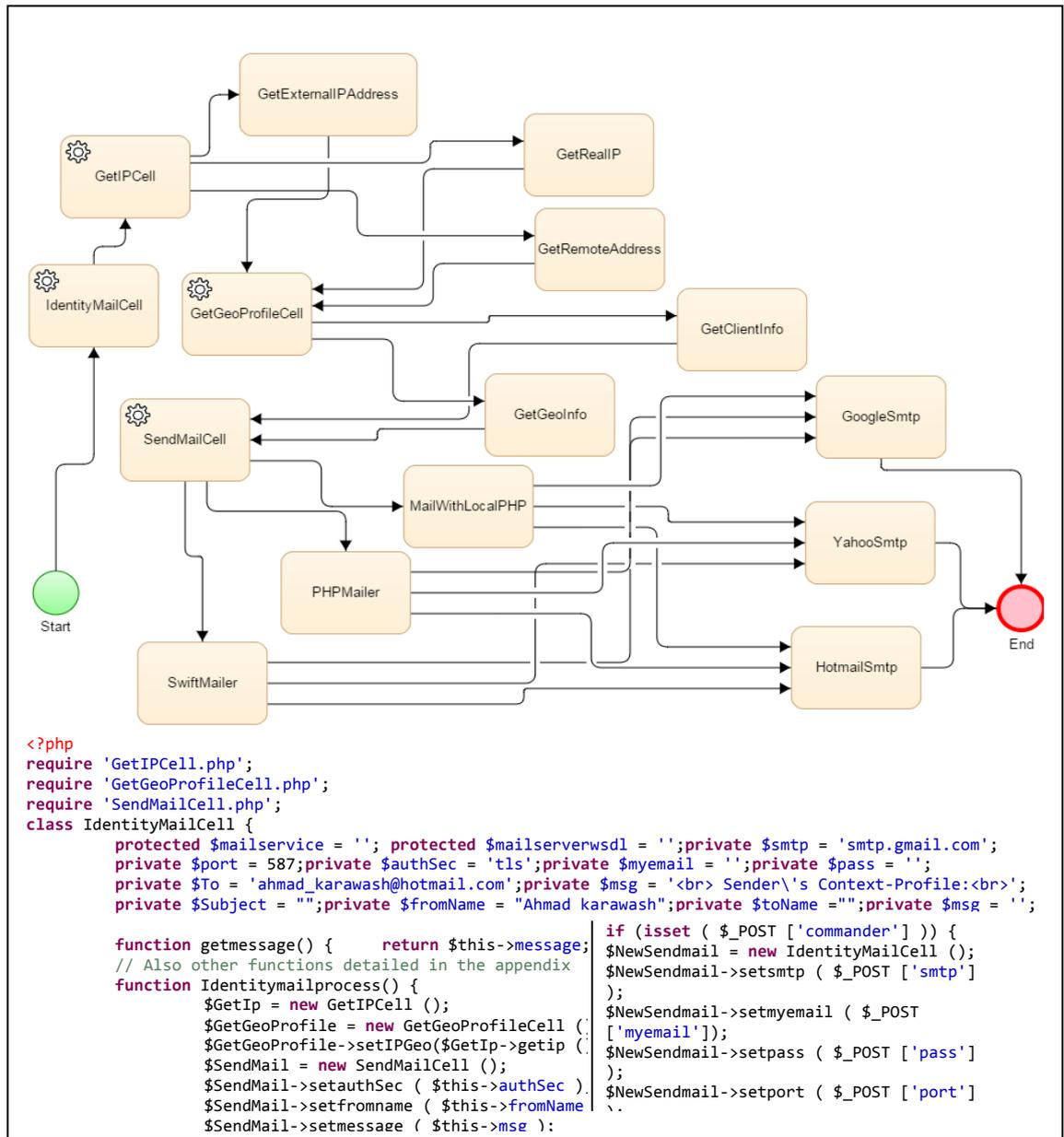


Figure 5.26: Gene map of the IdentityMail Cell

CELL FUNCTIONS

For simplicity, let us show how Gene is stored and how Cell “*analysis*” is done using an Cell_algorithms database. The screenshots of database tables, queries and outputs are collected from SQL server 2012. Figure 5.27 shows how Gene nodes (or Gene components that can be analyzed) are stored in the dbo.Node table as one block.

The figure displays the initial SQL database structure for a gene map. It includes two screenshots of SQL Server Query Designer and a large block of SQL INSERT statements.

The top screenshot shows the 'dbo.Node' table with the following data:

Id	Name
1	IdentityMailCell
2	GetIPCell
3	GetGeoProfileC...
4	SendMailCell
5	GetExternalIPA...
6	GetRealIP
7	GetRemoteAdd...
8	GetClientInfo
9	GetGeoInfo
10	MailWithLocalP...
11	PHPMailer
12	SwiftMailer
13	GoogleSmtp
14	YahooSmtp
15	HotmailSmtp

The bottom screenshot shows the 'dbo.Edge' table with the following data:

FromNode	ToNode
1	2
2	5
2	6
2	7
3	2
3	8
3	9
4	3
4	10
4	11
4	12
5	3
6	3
7	3
8	4
9	4
10	13
10	14
10	15
11	13
11	14
11	15
12	13
12	14
12	15
13	16
14	16
15	16

The SQL code on the right defines the structure and data for both tables:

```

INSERT dbo.Node (Id, Name) VALUES (1, 'IdentityMailCell')
INSERT dbo.Node (Id, Name) VALUES (2, 'GetIPCell')
INSERT dbo.Node (Id, Name) VALUES (3, 'GetGeoProfileCell')
INSERT dbo.Node (Id, Name) VALUES (4, 'SendMailCell')
INSERT dbo.Node (Id, Name) VALUES (5, 'GetExternalIPAddress')
INSERT dbo.Node (Id, Name) VALUES (6, 'GetRealIP')
INSERT dbo.Node (Id, Name) VALUES (7, 'GetRemoteAddress')
INSERT dbo.Node (Id, Name) VALUES (8, 'GetClientInfo')
INSERT dbo.Node (Id, Name) VALUES (9, 'GetGeoInfo')
INSERT dbo.Node (Id, Name) VALUES (10, 'MailWithLocalPHP')
INSERT dbo.Node (Id, Name) VALUES (11, 'PHPMailer')
INSERT dbo.Node (Id, Name) VALUES (12, 'SwiftMailer')
INSERT dbo.Node (Id, Name) VALUES (13, 'GoogleSmtp')
INSERT dbo.Node (Id, Name) VALUES (14, 'YahooSmtp')
INSERT dbo.Node (Id, Name) VALUES (15, 'HotmailSmtp')
INSERT dbo.Node (Id, Name) VALUES (16, 'Result')

INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (1, 2, 1306.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (2, 5, 1507.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (2, 6, 919.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (2, 7, 629.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 8, 613.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 9, 435.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 2, 537.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 3, 265.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 10, 1983.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 11, 325.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 12, 765.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (5, 3, 2161.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (6, 3, 1225.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (7, 3, 1483.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (8, 4, 1258.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (9, 4, 2661.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (10, 13, 1532.000)

```

Figure 5.27: initial SQL database by gene map

DETECT GENE MAP SHAPE:

Because Gene map changes dynamically based on quality parameters, so I use Prim's algorithm to get an up-to-date Gene shape. Figure 5.28 below shows the output of applying Prime algorithm on the IdentityMail Gene.

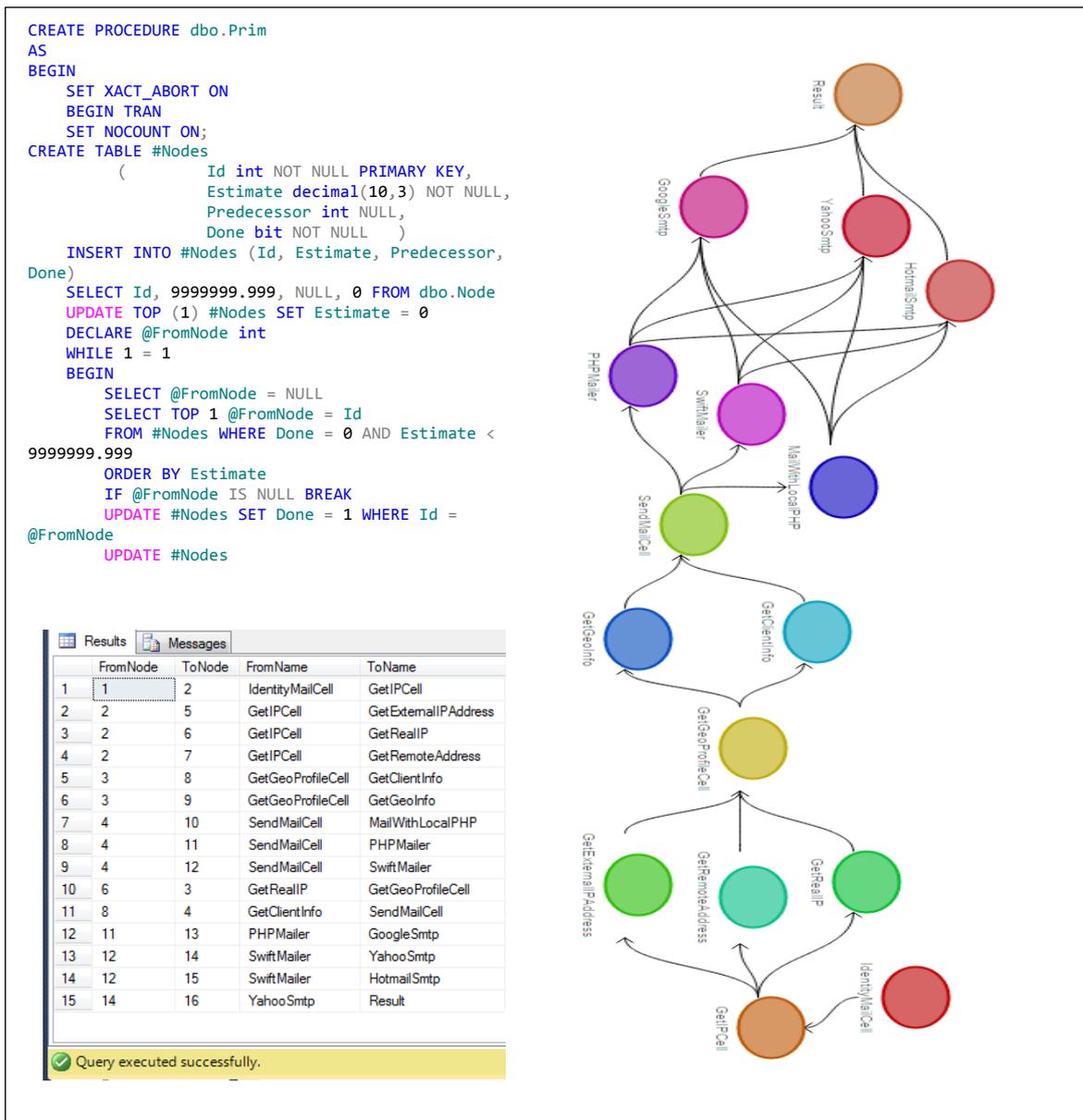


Figure 5.28: Trace the map of IdentityMail Gene

TRACE POSSIBLE COMPOSITIONS:

SmartCells approach follows an automatic and “*dynamic composition*” of a Cell Gene and the Gene is considered as a directed graph. So, it is recommended to cover all the possibility of best quality compositions among Gene nodes. For this purpose, I trace all paths between start and end nodes using Ucinet tool. As shown in figure 5.29 below, the result means that there are 54 composition paths are capable to be applied in the IdentityMail Gene.

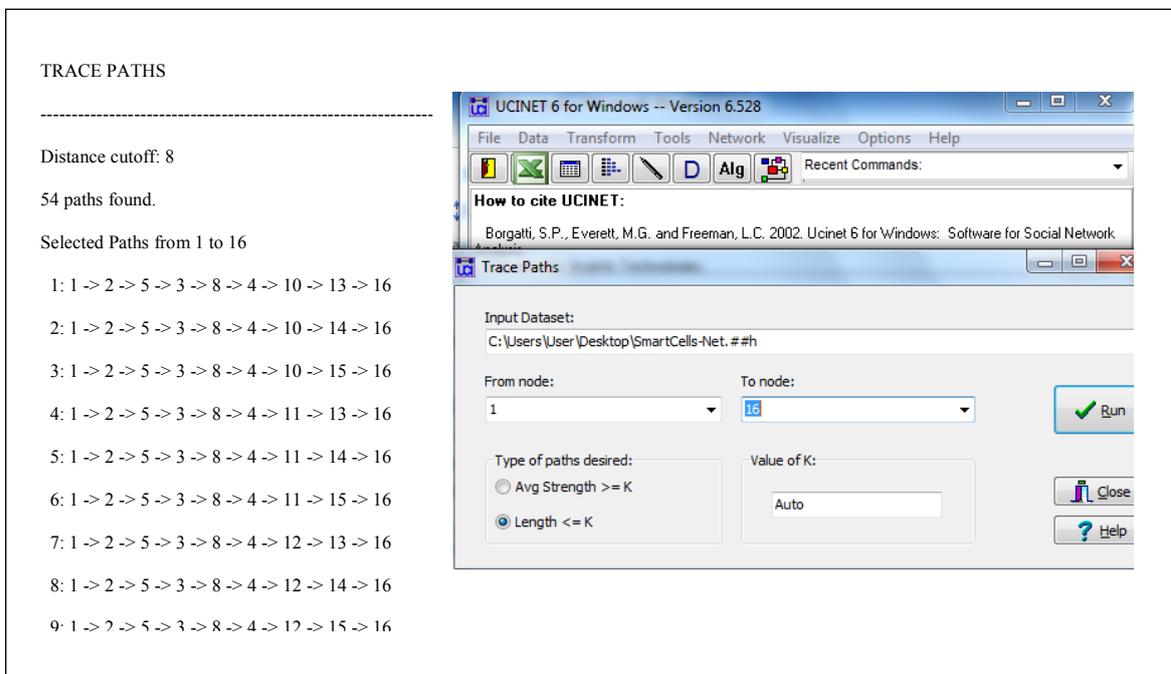


Figure 5.29: Trace the possible compositions of IdentityMail Cell

VALIDATE POSSIBLE COMPOSITION:

In order to “*validate*” a Gene graph map, I use a topological ordering based on Depth First Search algorithm. A topological ordering is an ordering of the directed acyclic graph (DAG) nodes, such that each node comes before all nodes to which it has outbound

edges. To be able to produce a topological ordering, the graph cannot have any cycles (that cause a code deadlock).

Consequently, I try to alter the correct follow of Identity Mail gene by inserting to fake edges that will cause cycles as follows shown in figure 5.30 below:

```

CREATE PROCEDURE dbo.TopologicalSort
AS
BEGIN
SET XACT_ABORT ON
BEGIN TRAN
SET NOCOUNT ON;
CREATE TABLE #Order
(
    NodeId int PRIMARY KEY,
    Ordinal int NULL
)
CREATE TABLE #TempEdges
(
    FromNode int,
    ToNode int,
    PRIMARY KEY (FromNode, ToNode)
)
INSERT INTO #TempEdges (FromNode, ToNode)
SELECT e.FromNode, e.ToNode
FROM dbo.Edge e
INSERT INTO #Order (NodeId, Ordinal)
SELECT n.Id, NULL
FROM dbo.Node n
WHERE NOT EXISTS (
SELECT TOP 1 1 FROM dbo.Edge e WHERE e.ToNode = n.Id
)
DECLARE @CurrentNode int,
        @Counter int = 0

WHILE 1 = 1
    BEGIN
        SET @CurrentNode = NULL
        SELECT TOP 1 @CurrentNode = NodeId
        FROM #Order WHERE Ordinal IS NULL
        IF @CurrentNode IS NULL BREAK
        UPDATE #Order SET Ordinal =
        @Counter, @Counter = @Counter + 1
        WHERE NodeId = @CurrentNode
        INSERT #Order (NodeId, Ordinal)
        SELECT Id, NULL
        FROM dbo.Node n
        JOIN #TempEdges e1 ON n.Id =
        e1.ToNode
        WHERE e1.FromNode = @CurrentNode
        AND
            NOT EXISTS (
                SELECT TOP 1 1
                FROM #TempEdges e2
                WHERE e2.ToNode
                = n.Id AND e2.FromNode <> @CurrentNode)
        DELETE FROM #TempEdges WHERE
        FromNode = @CurrentNode
        END
        IF EXISTS (SELECT TOP 1 1 FROM #TempEdges)
            BEGIN
                SELECT 'The graph contains cycles
    
```

Fake edges:

```

INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 2, 537.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 3, 265.000)
    
```

	FromNode	ToNode
1	2	5
2	2	6
3	2	7
4	3	2
5	3	8
6	3	9
7	4	3
8	4	10
9	4	11
10	4	12
11	5	3

Query executed successfully. DIM-P3-5060A

Figure 5.30: Validate of possible Gene composition

DISTANCE AS PERFORMANCE MEASURE:

Let us provide the distances among Cell nodes, so each node's link, in the graph, is now given weight and the problem becomes a network analysis problem as shown in figure 5.31:

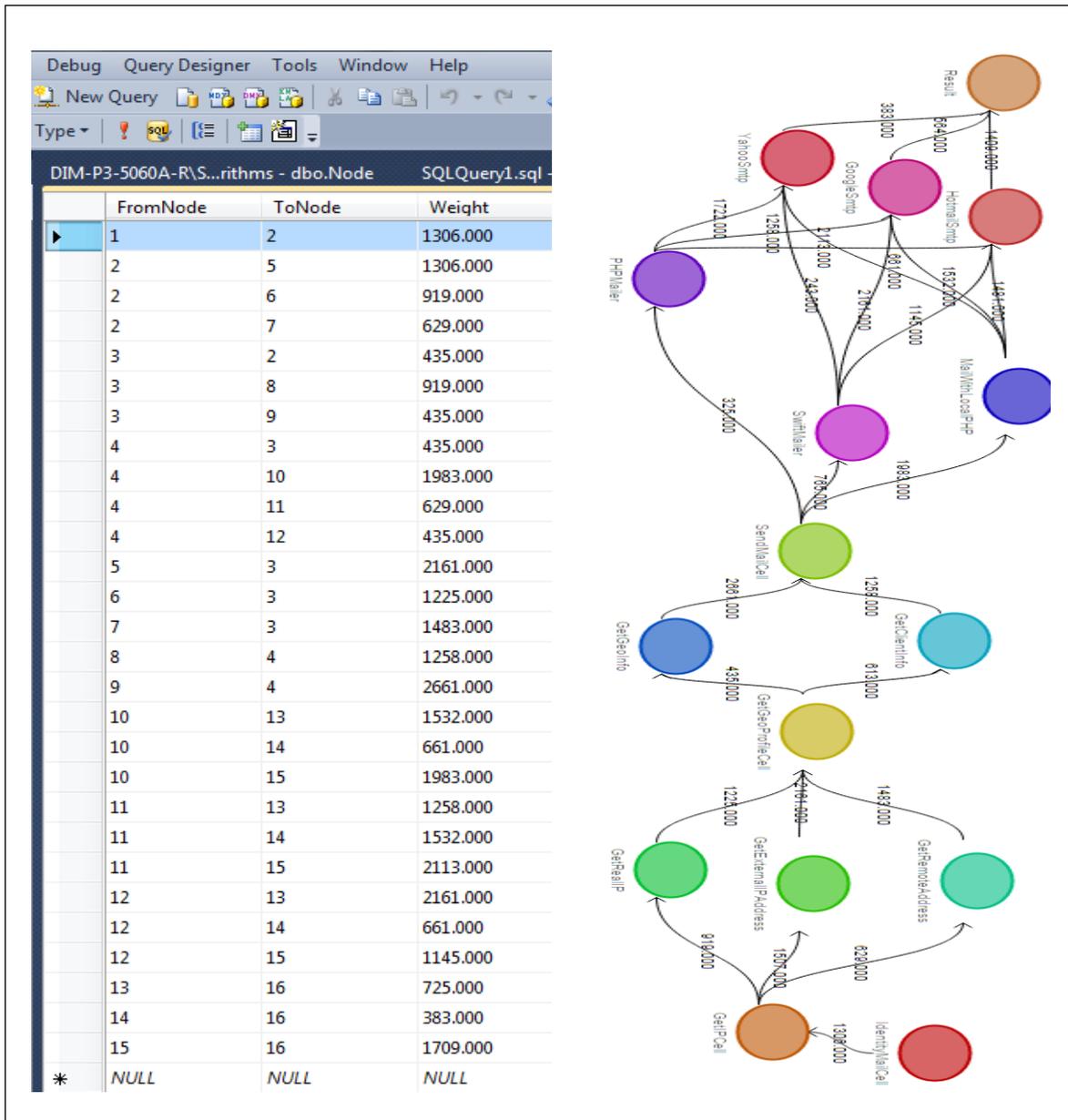


Figure 5.31: Improving the performance based on distance criteria

FLEXIBLE GENE ANALYSIS:

If Cell designer tries to use apply analysis about links among nodes before a validation step, the Cell system avoids automatically the edges that may cause cycles. Furthermore, Dijkstra algorithm is used to detect the shortest path among Gene nodes and it avoids paths that cause cycles. As shown in the figure 5.32, Dijkstra provides the shortest path but it avoids using the two edges (3 to 2 and 4 to 3) that cause cycles.

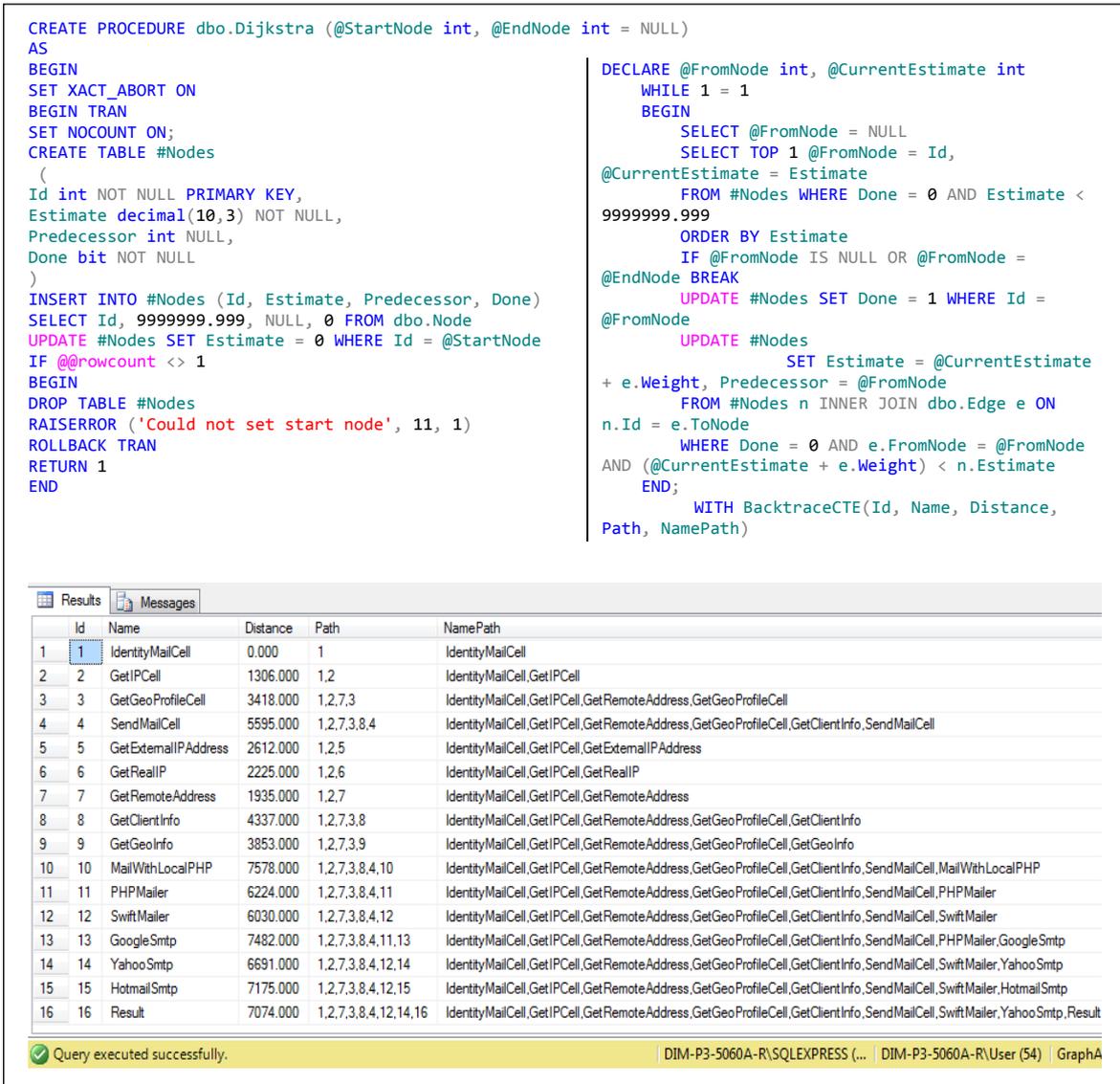


Figure 5.32 : Flexible Gene map analysis

MINIMAL PROCESS COMPOSITION:

To achieve the best Gene composition, a Cell requires the best response time composition that leads to best performance of distributed code. Consequently, the shortest geographical path among Gene nodes should be used. For example, if a Cell provides a service in Canada and it sends commands to another Cell in China this will affect the

performance of that Commander Cell. As shown in figure 5.33, Cell utilizes a Breadth First Search algorithm to detect the shortest geographical path among IdentityMail nodes.

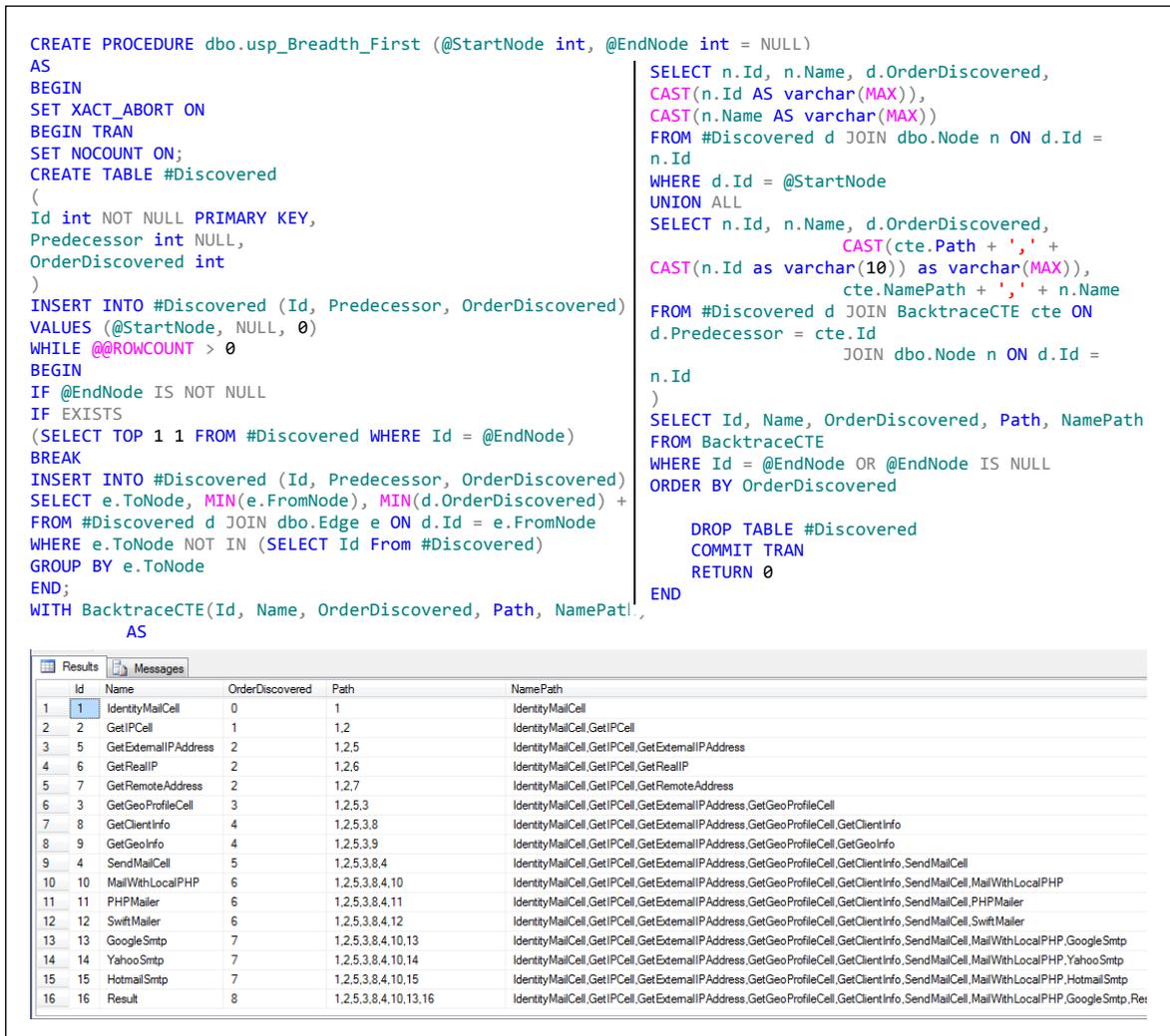


Figure 5.33: Minimal composition of Gene map

CENTRALITY MEASURES:

Since the service composition problem is transferred into a graph analysis problem, thus several graph measure can be applied. Using Ucinet tool, I show in this section how

IdentityMail Cell analyzes the Gene graph through some centrality measures, such as: Degree centrality, Closeness and Betweenness.

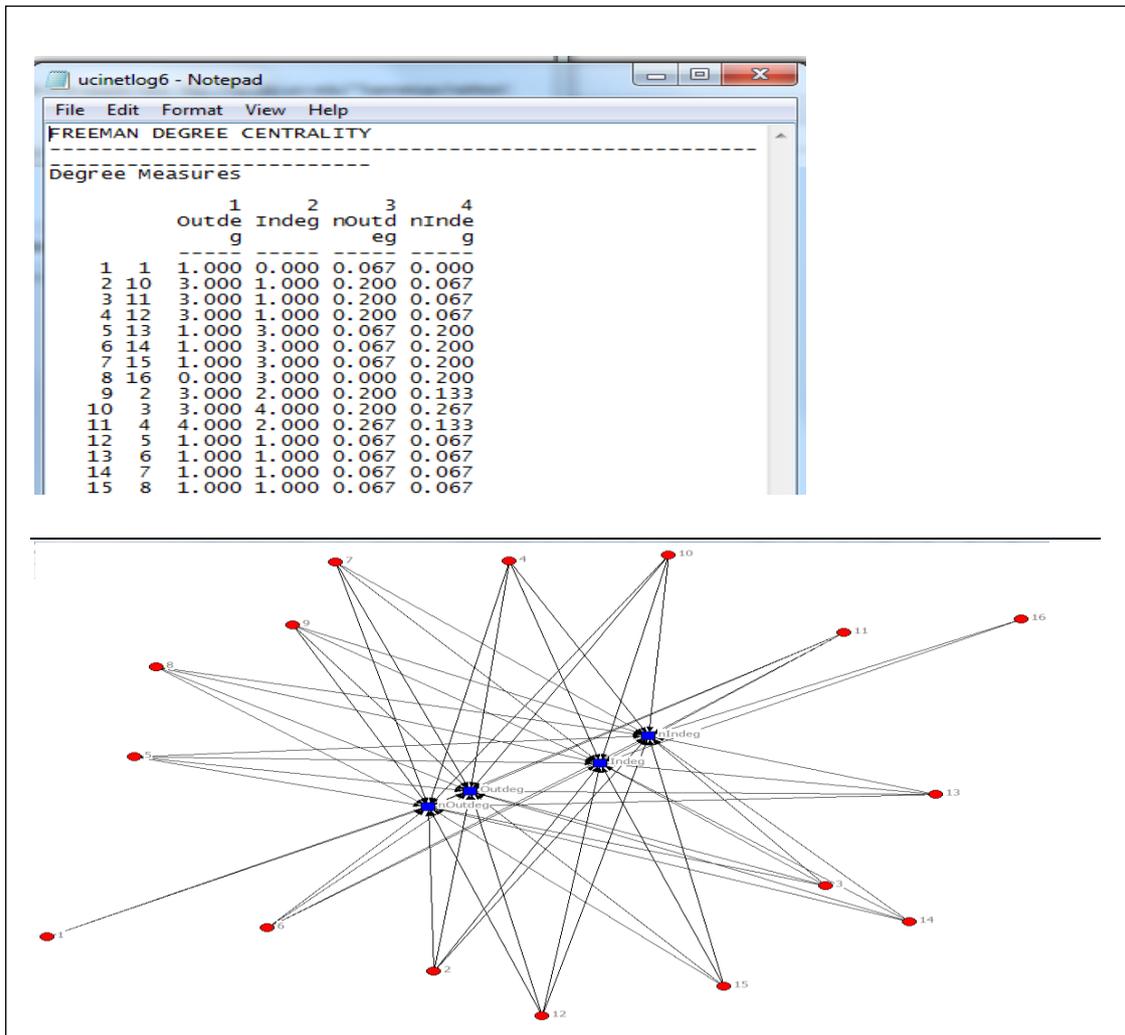


Figure 5.34: Gene analysis based on Degree Centrality measure

Figure 5.34 above shows the result of applying degree centrality measure on the Identity Mail Gene.

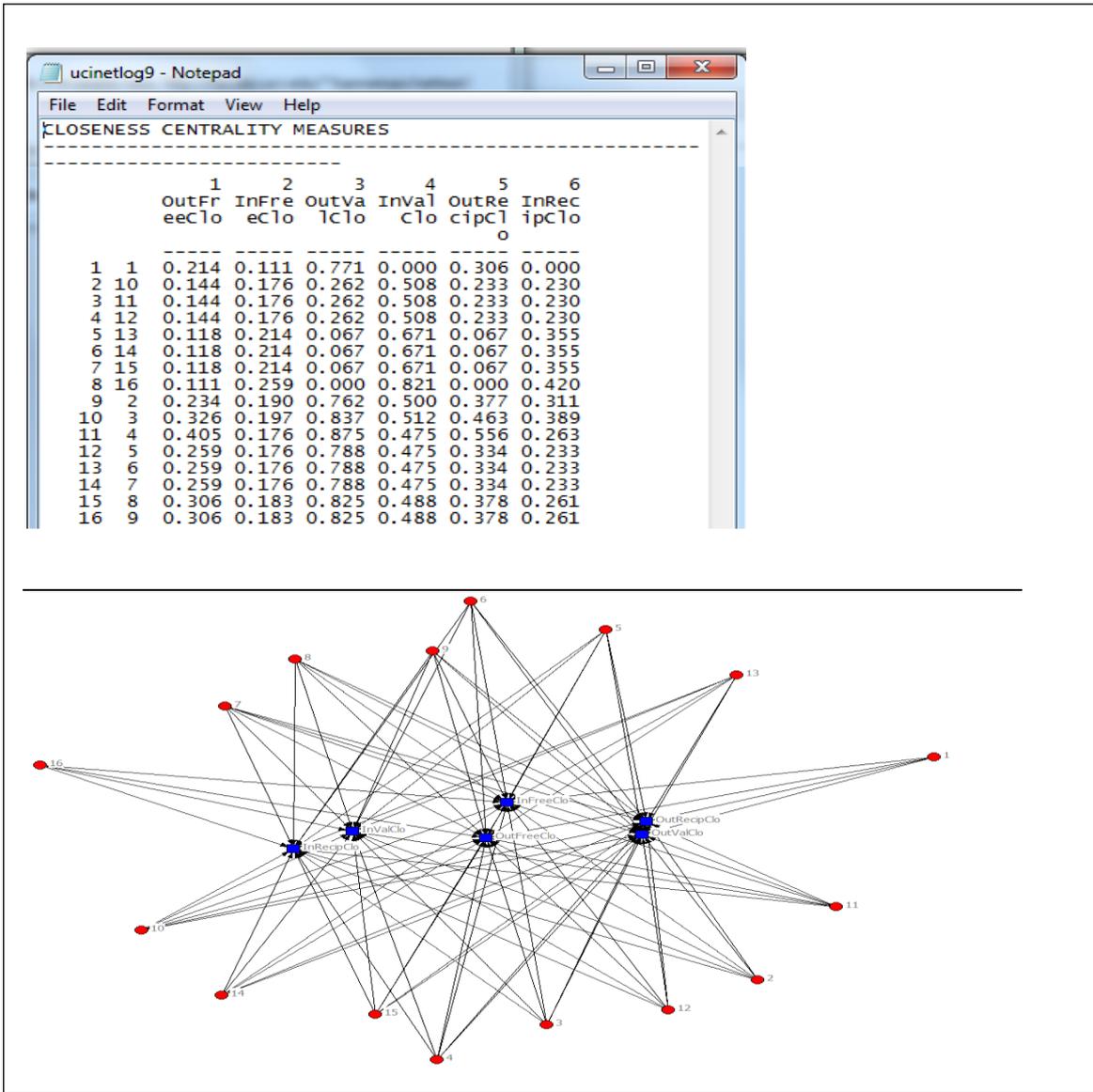


Figure 5.35: Gene analysis based on Closeness Centrality measure

Figure 5.35 above shows the result of applying Closeness centrality measure on the Identity Mail Gene.

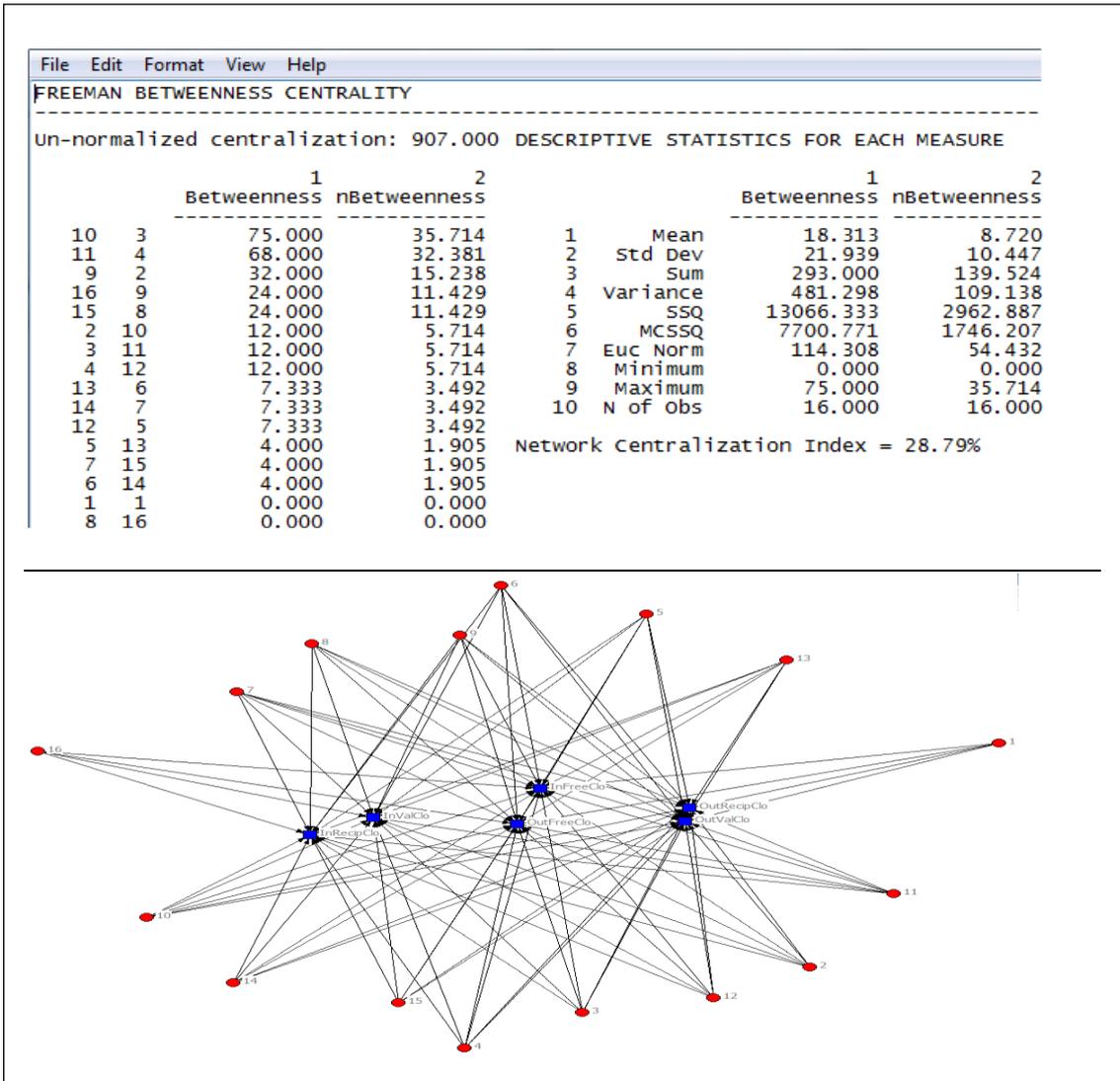


Figure 5.36: Gene analysis based on Betweenness Centrality measure

Figure 5.36 above shows the result of applying Betweenness centrality measure on the Identity Mail Gene.

5.5 OBSERVATION AND CRITICISM

Sometimes, we receive fake emails from people we don't know them and emails are automatically distributed to our contact list in which the primer sender is unknown. The aggregation of the Geographical context-Profile scenario, which is discussed through this chapter, is a new idea to decrease the severity of anonymous email. Also, it is good option to be added to emails by mails vendors such as: Hotmail, Gmail, yahoo, etc. Furthermore, it can improve email security by blocking email based on the sender context-profile. Help to know how to benefit from context-aware in the cloud environment.

The anonymous scenario is used to show the difference in using two different approaches; the traditional Service-Oriented approach and the proposed Cell-Oriented approach. Regarding Service-Oriented approach, I discuss first the anonymous email scenario as a cloud service simulation based on SOA. Then, I discuss the same scenario as a cloud Cell simulation using the SmartCells methodology. The Cell simulation shows how Cloud Brain's Cells are capable to analyze and offer the best result to Commanders. The two simulations are built as a PHP websites linked to both SQL and MySQL servers and they are capable to run as real cloud services.

CONCLUSIONS AND PERSPECTIVES

With the extensive deployment of Cloud computing, the management, reusability, security, interoperability and integration of these distributed systems have become challenging problems. Cloud architects try to develop important technologies to cope with these problems. Consequently, Cloud companies must work on some useful architectural properties (such as: service intelligent, process consistency) in order to achieve better Clouds.

PROBLEM STATEMENT REVIEW

Cloud computing paradigm, as one of the results of the continuous evolution of distributed computing, depends on sharing of resources to accomplish consistency over a network. Also, it is based on open standards, robust security, governance, compliance and privacy capabilities. However, without previous paradigms such as the service-oriented paradigm, organizations find it almost impossible to reach the Cloud. One of the latest challenges is how to avoid the disadvantages of the utilized paradigms in the Cloud computing environment. Indeed, the service concept was used for more than 10 years before rendering Clouds. During the service revolution, a group of weak points was discovered and marked as open problems such as service reusability, service validation, random performance, expensive availability, analysis of service process and service

compatibility. In addition to these facts, the Cloud infrastructure grows sharply without having used a unified autonomic or smart system to cover all the technological enlargement of this growth. That is why Cloud architecture needs some enhancements to avoid such vital issues of inherited properties.

PROSPECTED GOALS

Cloud model combines a powerful automation and services management with rich business management functions for fully integrated. It supports the full spectrum of Cloud service models, including infrastructure as a service, platform as a service, and software as a service. Whereas, technological evolutions and challenges require Clouds that are permanently up-to-date with more effective security, resiliency, service management, governance, business planning and life cycle management. Due to the significant development effort invested in these Cloud systems, there is a pressing need to revisit existing design, development, and management strategies so that dynamic adaptability, rapid delivery, and efficient access to Cloud-based services could take place in a seamless manner. Also, the desired web is a smart and semantic web and it is recommended to insert new essential properties to the Cloud paradigm such as autonomy and intelligence. In order to achieve a perfect Cloud paradigm, this Cloud must adopt a new well-organized system. After taking a look on my thesis research, the proper approach consists in rebuilding the Cloud paradigm to recover weakness points of some adopted approaches by Cloud.

CONTRIBUTIONS

Achieving perfect Cloud and competitiveness requires that companies continually modify their IT systems by adding new features or updating old ones in a relatively short period of time. Recently in Bio-Informatics domain, thousands of researches were applied to discover facts about the human body Genes. We propose the SmartCells Concept to benefit from the way Genes work to achieve a better Cloud computing model?

Through my research, and in contrast to the Bio-Informatics strategy that utilize technologies to solve and discover biological facts (like Gene map), I suggest to imitate the functions of these biological evidences in solving Cloud computing issues.

Indeed, I have developed a novel style of intelligent distributed computing proper for the Cloud technology challenges; called “SmartCells”. It ensures intelligent approach for Cloud service problems without altering the Cloud or web standards. The Cloud problems that treated through SmartCells are: Cloud computing security, service process validation, quality of Cloud services, multi-dimensional analysis of big data, expensive availability, random performance, and internationalization. This research builds basis for a new intelligent Cloud model, through combing the advantages of previous approaches and introducing new concepts and methodologies. The main effort in this thesis is to replace the old service concept by a new Cell notion. Indeed, Cells are smart components developed to provide highly focused functionality for solving specific distributed computing problems. Every cell has its own functionality and goal to serve, so one cannot

find two different cells to do same type of jobs. However, all cells are similar in base and structure, they can also sense changes, act, analyze data, and communicate.

As a result, my contribution through my researches is a novel intelligent distributed computing theory, the SmartCells theory, which benefits from biological strategies to upgrade Cloud computing paradigm. This theory is composed of some new computing models and concepts that are expressed in details through significant book chapters (Karawash *et al.*, 2015; Karawash *et al.*, 2014a; Karawash *et al.*, 2014b; Karawash *et al.*, 2013).

LIMITATIONS

Despite the promising results obtained during the experimental phase by testing Cloud Cells, the proposed approach faces some limitations.

WORK LIMITATIONS:

As discussed in Chapter 3, SmartCells is developed to manage the whole Cloud systems in a new style of distributed computing. Consequently, the works in this thesis, including book chapters, are preliminary steps comparing to project general goals. Thus, the important ideas of this thesis need to be studied deeply by Cloud Computing experts, developed more by researchers and implemented by companies. Despite the fact that I have dealt with several architectural computing level (as security, network, analysis, etc.) to achieve good results, SmartCells theory still requires more efforts because each of its levels forms a domain of study. Indeed, through this thesis I have built the bases of new intelligent Cloud systems.

INFRASTRUCTURAL LIMITATIONS:

As discussed in Chapter 3 (section 3.4.1), SmartCells Architecture is composed of four main components: Demander Cell, Executive Cell, Instruction source, and Cloud Brain. In comparison with SOA, Demander Cells replace Cloud service client and Instruction source replaces the Cloud service provider, while the Cloud Brain that supplies Executive Cells is a new component. As shown in Chapter 4 (section 4.2.2), Cloud Brain infrastructure is similar to that of Cloud data center and requires a collaboration between companies to be built and supported by Cloud data. This limitation could be solved if one of the Cloud vendors such as Google that has experience in building Cloud data centers, adopts the SmartCells project.

FUTURE WORKS

Cloud computing systems are of huge importance in a number of recently established and future functions in computer science. For example, they are vital to banking applications, communication of electronic systems, air traffic control, manufacturing automation, biomedical operation works, space monitoring systems, robotics information systems and many more. As the nature of computing comes to be increasingly directed towards intelligence and autonomy, intelligent computations will be the key for all future applications. Intelligent Cloud computing will become the base for the growth of an innovative generation of intelligent distributed systems. Research centers require the development of architectures of intelligent and collaborated systems; these systems must be capable of solving problems by themselves to save processing time and reduce costs. Based on SmartCells, my future goal is to achieve an intelligent Cloud

computing system that controls the whole distributed system based on completely consistent rules. Specifically, as a future work project, I aim to develop a perfect distributed system which operates similar to the human Cell system. To achieve this purpose, I will try to follow the recent and future researches about artificial and virtual simulation of body Cells.

REFERENCES

- (Agrawal *et al.*, 2012): Agrawal, D., El Abbadi, A., Das, S., and Elmore, A., Database Scalability, Elasticity, and Autonomy in the Cloud, University of California at Santa Barbara, 2012.
- (Ahuja and Moore, 2013): Ahuja, S. & Moore, B., State of Big Data Analysis in the Cloud, Network and Communication Technologies; Vol. 2, No. 1; 2013, ISSN 1927-064X E-ISSN 1927-0658, Published by Canadian Center of Science and Education
- (Al-Masri and Mahmoud, 2007): Al-Masri, E., Mahmoud, QH., Discovering the best web service, Proceedings of the 16th International Conference on World Wide Web, 2007, pp. 1257-1258. <http://dx.doi.org/10.1145/1242572.1242795>.
- (Altamash and Niranjana, 2013): Altamash, M. S., Niranjana, P. Y., A Survey of Identifying Key Challenges of Performance Modeling in Cloud Computing International Journal of Computer Science and Information Technology Research (IJCSITR), Vol. 1, Issue 1, pp: (33-41), December 2013.
- (Tarantola, 2011): Tarantola, A., Japan's K Computer Is the Fastest of Them All, Topping 10 Petaflops, gizmodo, 2011, <http://gizmodo.com/5856272/japans-k-computer-is-the-fastest-of-them-all-topping-10-petaflops>

- (Mohebi, 2012): Mohebi, A., An efficient *QoS*-Based Ranking Model for Web Service Selection with Consideration of User's Requirement; Thesis and dissertations; Ryerson University; Ontario, Canada (2012).
- (Azab, 2009): Azab, A., & Meling, H. (2009), Cloud computing (Vol. 5931, pp. 200–211), doi: 10.1007/978-3-642-10665-1
- (Azar and Sundarapandian, 2015): Azar, A., Vaidyanathan, Sundarapandian (Eds.), Computational Intelligence Applications in Modeling and Control, Studies in Computational Intelligence, Vol. 575, 2015, Springer International Publishing.
- (Babcock, 2011): Babcock, C, "When Amazon's Cloud Turned On Itself", Information Week (2011), <http://www.lexisnexis.com.proxy.lib.uwaterloo.ca/hottopics/lnacademic/>
- (Babeetha *et al.*, 2013): Babeetha, S. et Muruganatham, B., An Efficient Approach for Web Service Composition Using Semantic based Web Service Discovery, International Journal of Computer Trends and Technology, Volume.4, Issue.2- 2013.
- (Baburajan, 2011): Baburajan, R., The Rising Cloud Storage Market Opportunity Strengthens Vendors, infoTECH, August 24, 2011, It.tmcnet.com, 2011-08-24, Retrieved 2011-12-02.
- (Bach *et al.* 2005): Bach, M. P., Vlahovic, N, & Knezevic, B 2005, September, "Public data retrieval with software agents for business intelligence", in proceedings of the 5th wseas int. Conf. On Applied Informatics, pp. 15-17

- (Barry, 2013): Barry, D. K., & Dick, D. (2013), Web services, service-oriented architectures, and Cloud computing: The savvy manager's guide. Waltham, MA: Morgan Kaufmann.
- (Barry, 2013): Barry, D., Web Services, Service-Oriented Architectures, and Cloud Computing, Second Edition: The Savvy Manager's Guide (The Savvy Manager's Guides), 2013, Edition: 2nd, ISBN-13: 978-0123983572.
- (Bell, 2008): Bell, M., Service Oriented Modeling, John Wiley & Sons, Feb 25, 2008 - 384 pages.
- (Bento, 2013): Bento, A. M., & Aggarwal, A., Cloud computing service and deployment models: Layers and management. Hershey, 2013, PA: Business Science Reference.
- (Bessis and Ciprian, 2014): Bessis, N., Dobre, C., (Eds.), Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence, Vol. 546, 2014, Springer International Publishing.
- (Bieber and Carpenter, 2001): Bieber, Carpenter et. al., "Jini Technology Architectural Overview", Sun Microsystems, 2001 (online) Available: <http://www.sun.com>
- (Bryant, 2008): Bryant, R. E., Katz, R. H., & Lazowska. E. D., Big-data computing: Creating revolutionary breakthroughs in commerce, science, and society. In Computing Research Initiatives for the 21st Century, Computing Research Association, 2008.
- (Brain, Stanford Wikipedia): Brain, Stanford Wikipedia, Symsys100, <http://web.stanford.edu/class/symsys100/Brain-Wikipedia.pdf>
- (Buyya, 2011): Buyya, R., Broberg, J., & Gościński, A., Cloud computing: Principles and paradigms. Hoboken, N.J: Wiley, 2011.

- (Capgemini and HP, 2008): Capgemini in collaboration with HP, The Cloud and SOA Creating an Architecture for Today and for the Future, 2008.
- (Cardoso and Sheth, 2002): Cardoso, J., and Sheth, A., Semantic e-Workflow Composition, Technical Report# 02-004, LSDIS Lab, Computer Science, 2002.
- (Chaczko *et al.*, 2011): Chaczko, Z., Mahadevan, V., Aslanzadeh, S., and Mcdermid, C., Availability and Load Balancing in Cloud Computing, International Conference on Computer and Software Modeling, IPCSIT vol.14 (2011), IACSIT Press, Singapore.
- (Chow *et al.*, 2009): Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., & Molina, J. (2009), Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control, In Proceedings of the 2009 ACM workshop on Cloud computing security (pp. 85–90). doi:10.1145/1655008.1655020
- (Chuob, 2011): Chuob, S., Pokharel, M., Park, J.S., "Modeling and Analysis of Cloud Computing Availability Based on Eucalyptus Platform for E-Government Data Center," Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on , vol., no., pp.289,296, June 30 2011-July 2 2011, doi: 10.1109/IMIS.2011.135
- (Chemocare, 2015): Cancer Cells and Chemotherapy, Chemotherapy, Chemcore.com, <http://www.chemocare.com/chemotherapy/what-is-chemotherapy/cancer-cells-chemotherapy.aspx>
- (Coulouris et al, 2011): Coulouris, G., Dollimore, J., Kindberg, T., Blair, G., Distributed Systems: Concepts and Design (5th Edition). Boston: Addison-Wesley, ISBN 0-132-14301-1, 2011.

- (Florescu et al, 2002): Florescu, D., Grünhagen, A., Kossmann, D., XL: An XML programming language for Web service specification and composition, in: Proceedings of the Eleventh International World Wide Web Conference (WWW), Honolulu, Hawaii, USA, 7–11 May 2002.
- (Darekar, 2013): Darekar, S., Ingle, D.R., Service-Oriented Architecture For Enterprise Application, INTERNATIONAL JOURNAL OF ADVANCED AND INNOVATIVE RESEARCH, ISSN: 2278-7844, 2013
- (Dastjerdi et Buyya, 2014): Dastjerdi, A. V., & Buyya, R., Compatibility-Aware Cloud Service Composition under Fuzzy Preferences of Users. IEEE Transactions on Cloud Computing, 2, 1–13. doi:10.1109/TCC.2014.2300855, 2014.
- (Davies *et al.* 1995): Davies, W. H., & Edwards, P., “Agent-based knowledge discovery”, In Working Notes Of The Aaai Spring Symposium On Information Gathering From Heterogeneous, Distributed Environments, Stanford University, Stanford, Ca Winton, 1995.
- (De Filippi and McCarthy, 2012): De Filippi, P., McCarthy, S., Cloud Computing: Centralization and Data Sovereignty, European Journal of Law and Technology 3, 2012, http://hal.archives-ouvertes.fr/docs/00/74/60/65/PDF/07_-_2012_EJLT_-_Cloud_Computing_and_Data_Sovereignty.pdf
- (Erl, 2012): Erl, T., What is SOA: an Introduction to Service Oriented Computing, <http://www.whatissoa.com>, SOA System, 2012.
- (Banaei-Kashani *et al.*, 2004): Banaei-Kashani, F., Chen, C., and Shahabi., C., Web services peer-to-peer discovery service, International Conference on Internet Computing, 2004.

- (Emekci *et al.*, 2004): Emekci, F., Sahin, O., Agrawal, D., El Abbadi, A., A Peer-to-Peer Framework for Web Service Discovery with Ranking, IEEE International Conference on Web Services, 2004.
- (Fielding, 2000): Fielding, R.T., Architectural Styles and the Design of Network-based Software Architectures, Ph.D. dissertation, in University of California, Irvine. 2000.
- (Freudenrich and Boyd, 2013): Freudenrich, C., and Boyd, R., 2013. How Your Brain Works, journal of how stuff works, <http://science.howstuffworks.com/life/inside-the-mind/human-brain/brain.htm>
- (Ge *et al.*, 2006): Ge, X., Yu, S., Zhang, J., Wu, G., Web Service Discovery in Large Distributed System Incorporating Semantic Annotations, 2006.
- (Haibo, 2011): Mi, H., Wang, H., Yin, G., Cai, H., Zhou, Q., Sun, T., "Performance problems online detection in Cloud computing systems via analyzing request execution paths," Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on , vol., no., pp.135,139, 27-30 June 2011, doi: 10.1109/DSNW.2011.5958799.
- (Hang *et al.*, 2011): Hang C.W., and Singh M. P., 2011, Trustworthy service selection and composition. ACM Trans. Auton. Adapt. Syst. 6, 1, Article 5 (2011), 17 pages. DOI = 10.1145/1921641.1921646 <http://doi.acm.org/10.1145/1921641.1921646>
- (Hauck *et al.*, 2010): Hauck, M., Huber, M., Klems, M., Kounev, S., Muller-Quade, J., Pretschner, A., Reussner, A., Tai, S., Challenges and opportunities of Cloud Computing, Karlsruhe Institute of Technology Technical Report Vol., 2010.

- (Hayes, 2008): Hayes, B., & Computing, C. (2008), Cloud Computing, Communications of the ACM, 51, 9–11, doi:10.1145/1364782.1364786
- (He *et al.*, 2008): He, Q., Yan, J., Yang1, Y., Kowalczyk, R., Jin, H., Chord4S: A P2P-based Decentralised Service Discovery Approach, 2008 IEEE International Conference on Services Computing, 2008.
- (Hickman *et al.*, 1995): Hickman, C., Roberts, L., and Larson, A., Integrated Principles of Zoology, 9th Ed., Wm C., Brown, 1995.
- (Hong-Linh, 2011): Hong-Linh, T., Gangadharan, G.R., Comerio, M., Dustdar, S., De Paoli, F., "On Analyzing and Developing Data Contracts in Cloud-Based Data Marketplaces," Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific , vol., no., pp.174,181, 12-15 Dec. 2011, doi: 10.1109/APSCC.2011.65
- (Huang and Sandia, 2013): Huang, W., et Sandia, I, Cloud Computing with Intelligent Agents to Support Service Oriented System Control and Management, International Journal of E-Business Development, 2013, Vol. 3 Iss. 4, PP. 165-173.
- (IBM, 2013): Siewert, S., Big data in the Cloud Data velocity, volume, variety, veracity, IBM- Developer works, 2013, ibm.com/developerWorks/.
- (IBM, 2014): IBM SmartCloud Platform, <http://www.ibm.com/cloud-computing/>
- (Is, 2010): Is, W., & Computing, C. (2010), The Rise of Cloud Computing. Computing, doi: 10.2307/2186304
- (Su *et al.*, 2003): Su, J., Hull, R., Bultan, T., Fu, X., Conversation specification: a new approach to design and analysis of E-service composition, in:

Proceedings of the Twelfth International World Wide Web Conference (WWW), Budapest, Hungary, 20–24 May 2003.

(Kopeck, 2007): Kopeck, J., (2007), Semantic Web Service Offer Discovery. OTM 2007 Ws, Part I, LNCS 4805.

(Jayabrabu *et al.* 2012): Jayabrabu, R., Saravanan, V., & Vivekanandan, K., “Software agents paradigm in automated data mining for better visualization using intelligent agents”, Journal Of Theoretical And Applied Information Technology, 39(2), 2012.

(Johnson, 2010): Johnson, G., Understanding How the Brain Works, Tbiguide, <http://www.tbiguide.com/howbrainworks.html>

(Jonsson *et al.*, 2011): Jonsson, A., Rovatsos, M., Scaling up multiagent planning: A best-response approach. In: Proceedings of the 21st International Conference on Automated Planning and Scheduling. ICAPS (2011)

(Joyent, White Paper): A Joyent White Paper, Performance and Scale in Cloud Computing, <http://www.joyent.com/content/06-developers/01-resources/07-performance-and-scale-in-Cloud-computing/performance-scale-Cloud-computing.pdf>

(Karawash *et al.*, 2013): Karawash, A., Mcheick, H., Dbouk, M., Intelligent Web Based on Mathematic Theory, Springer book (SCI), Springer International Publishing Switzerland 2013, R. Lee (Ed.): Computer and Information Science, SCI 493, pp. 201–213. DOI: 10.1007/978-3-319-00804-2_15

(Karawash *et al.*, 2014a): Karawash, A., Mcheick, H., & Dbouk, M., Simultaneous Analysis of Multiple Big Data Networks: Mapping Graphs into a Data Model.

Chapter, Springer's Studies in Computation Intelligence (SCI), DOI: 10.1007/978-3-319-05029-4_10, Volume 546, 2014, pp 243-257.

- (Karawash *et al.*, 2014b): Karawash, A., Mcheick H., Dbouk, M., Quality-of-service data warehouse for the selection of Cloud service: a recent trend, Springer's Studies in Computation Intelligence (SCI), Computer Communications and Networks 2014, pp 257-276, DOI: 10.1007/978-3-319-10530-7_11.
- (Karawash *et al.*, 2015): Karawash, A., Mcheick H., Dbouk, M., Towards Intelligent Distributed Computing: A Cell Oriented Computing, Springer's Studies in Computation Intelligence (SCI), 2015.
- (Karn, 2010): Karn, B., Security Issues to Cloud Computing. In Cloud Computing (Vol. 0, pp. 271–288), doi: 10.1007/978-1-84996-241-4, 2010.
- (Karray *et al.*, 2013): Karray, A., Teyeb, R., Jemma, M., A Heuristic Approach for Web-service discovery and selection, International Journal of Computer Science & Information Technology (IJCSIT) Vol 5, No 2, April 2013, DOI: 10.5121/ijcsit.2013.5210.
- (Keller *et al.*, 2004): Keller, U., Lara, R., Polelres, A., Toma, I., Kifer, M., and Fensel, D., WSMO Web Service Discovery. WSMO Working Draft, v0.1, 2004.
- (Keskes, 2009): Keskes, N., Web Services Selection Based on Context Ontology and Quality of Services; Management information system Department, King Faisal University, Saudi Arabia; Lehireche, A.; Rahmoun, A.; Computer science Department, Uni-versity of Sidi Bel Abbes, Algeria, 2009.
- (Knoblock and Craig 2004): Knoblock, A., “Building Software Agents For Planning, Monitoring, And Optimizing Travel”, University Of Southern California Marina Del Rey Information Sciences Inst, 2004.

- (Lee, 2013): Lee, L., Computer and Information Science, Studies in Computational Intelligence, Vol. 493, Springer International Publishing, 2013.
- (Li *et al.*, 2012): Li, C., Zhu, Z., Li, Q., Yao, X. (2012), Study on semantic web service automatic combination technology based on agent, Springer Berlin Heidelberg, Lecture Notes in Electrical Engineering Volume 227, 187-194.
- (Liu *et al.*, 2008): Liu, D. and Deters, P., “Management of service-oriented systems” in Journal of Service Oriented Computing and Applications, Volume 2, Special Issue 2-3, pp. 51-64, Springer-Verlag, July 2008
- (Liu *et al.*, 2013): Liu, Q., Wang, G., & Wu, J., Consistency as a service: Auditing Cloud consistency. IEEE Transactions on Network and Service Management, 11, 25–35. doi:10.1109/TNSM.2013.122613.130411, 2013.
- (Liu, 2005): Liu, W., Trustworthy service selection and composition - reducing the entropy of service-oriented Web, INDIN, 2005, 3rd IEEE Int. Conf. Ind. Informatics, 2005.
- (Schubert, 2010): Schubert, L., The Future of Cloud Computing Opportunities for European Cloud Computing Beyond 2010, SAP Research, 2010.
- (Blake, 2007): Blake, M., “Decomposing Composition: Service-Oriented Software Engineers,” IEEE Software, vol. 24, no. 6, 2007, pp. 68–77.
- (Papazoglou *et al.*, 2008): Papazoglou *et al.*, “Service-Oriented Computing: A Research Roadmap,” Int’l J. Cooperative Information Systems, vol. 17, no. 2, 2008, pp. 223–255, 2008.
- (Mahmood, 2011): Mahmood, Z., & Hill, R., Cloud computing for enterprise architectures. London: Springer, 2011.

- (Mahmood, 2014): Zaigham, M. (Ed.), *Cloud Computing Challenges, Limitations and R&D Solutions, Studies in Computational Intelligence, Computer Communications and Networks*, Springer International Publishing, 2014.
- (Martinoli, 2014): Alcherio Martinoli, *Distributed Intelligent Systems, Distributed Intelligent Systems and Algorithms Laboratory Disal*, Ecole Polytechnique federal de Lausanne, 2014.
- (Maximilien and Singh, 2002): Maximilien, E. M., & Singh, M. P.; Conceptual model of Web service reputation; *ACM SIGMOD Record*, 31(4), 36-41, 2002.
- (McGovern *et al.*, 2003): James McGovern, Sameer Tyagi, Michael E. Stevens, Sunil Mathew *Java Web Services Architecture*, Morgan Kaufmann Publishers, 2003.
- (Mehrotra, 2011): Mehrotra, N., *Cloud-Testing vs. Testing a Cloud*. 10th Annual International Software Testing Conference (p. 8), 2011, Retrieved from <http://www.infosys.com/engineering-services/white-papers/documents/Cloud-testing-vs-testing-Cloud.pdf>
- (Hauck *et al.*, 2010): Hauck, M., Huber, M., Klems, M., Kounev, S., Muller-Quade, J., Pretschner, A., Reussner, R., Tai, S., *Challenges and Opportunities of Cloud Computing: Trade-off Decisions in Cloud Computing Architecture*, Karlsruhe Institute of Technology, Technical Report, Vol. 2010-1, 2010.
- (Microsoft blog, 2010): "Windows Azure General Availability - The Official Microsoft Blog - Site Home - TechNet Blogs", [Blogs.technet.com](http://blogs.technet.com), 2010.
- (Moemeng *et al.* 2010): Moemeng, C., Zhu, X., Cao, L., & Jiahang, C., "I-Analyst: An agent-based distributed data mining platform", In *Data Mining Workshops*

(ICDMW), 2010 IEEE International Conference, IEEE, pp. 1404-1406, 2010.

(Mohanty *et al.*, 2012): Mohanty, R.; Ravi, V.; Patra, M. R.; Classification of Web Services Using Bayesian Network; Journal of Software Engineering and Applications, 291-296; doi:10.4236/jsea.2012.54034, 2012.

(Monaco, 2012): Monaco, A., "A View inside the Cloud", 2012, <http://theinstitute.ieee.org/technology-focus/technology-topic/a-view-inside-the-Cloud>, theinstitute.ieee.org (IEEE).

(Mulholland *et al.*, 2008): Mulholland, A., Daniels, R., Hall, T., Capgemini and HP, The Cloud and SOA: Creating an Architecture for Today and for the Future, 2008.

(Nallur and Bahsoon, 2013): Nallur, V., and Bahsoon, R., A Decentralized Self-Adaptation Mechanism for Service-Based Applications in the Cloud, IEEE Transactions on Software Engineering, Vol. 39, No. 5, pp.591-612, 2013.

(Nawaz *et al.*, 2008): Nawaz, F., Qadir, K., Ahmad, F., "SEMREG-Pro: A Semantic based Registry for Proactive Web Service Discovery using Publish Subscribe Model", Fourth International Conference on Semantics, Knowledge and Grid, IEEE Xplore, 2008.

(Nguyen, 2011): Nguyen, C., Marchetto, A., Tonella, P., Challenges in audit testing of web services. In Proceedings - 4th IEEE International Conference on Software Testing, Verification, and Validation Workshops, ICSTW 2011 (pp. 103–106), doi:10.1109/ICSTW.2011.104, 2011.

(Niloofer, 2013): Khanghahi, N., and Ravanmehr, R., Cloud Computing Performance Evaluation: Issues and Challenges, International Journal on Cloud

Computing: Services and Architecture (IJCCSA), Vol. 3, No.5, October 2013.

- (NIH, 2014) NIH Publication No.11-3440a, Brain Basics: know Your Brain, National Institute of Neurological Disorders and Stroke, 2014.
- (Ning, 2012): Li, N., Fan, P., Lv, H., "The construction of Cloud data analysis platform and its application in intelligent industrial park," Advanced Communication Technology (ICACT), 2012 14th International Conference, vol., no., pp.860, 863, 19-22, 2012
- (NIST definition, 2010): NIST definition of Cloud computing, <http://www.nist.gov/itl/Cloud/>.
- (Notenboom, 2005): Notenboom, L., How can I send anonymous email?, Making Technology Work For Everyone, 2005, https://askleo.com/how_can_i_send_anonymous_email.
- (Nurmi *et al.*, 2005): Nurmi, P., Przybilski, M., Linden, G. & Floreen, P., An Architecture For Distributed Agent-Based Data Preprocessing, Springer, Autonomous Intelligent Systems: Agents and Data Mining, Lecture Notes in Computer Science Volume 3505, pp 123-13, 2005.
- (Ong *et al.*, 2005): Ong, K. L., Zhang, Z., Ng, W. K., & Lim, E. P., "Agents and stream data mining: a new perspective", Intelligent Systems, IEEE, 20(3), pp. 60-67, 2005.
- (Onwubiko, 2010): Onwubiko, C, Security Issues to Cloud Computing. In Cloud Computing (Vol. 0, pp. 271–288), doi: 10.1007/978-1-84996-241-4, 2010.
- (Panda, 2005): Panda, D., An Introduction to Service-Oriented Architecture from a Java Developer Perspective. <http://onjava.com/pub/a/onjava/2005/01/26/soa-intro.html>, O'Reilly, 2005.

- (Portchelvi *et al.*, 2012): Portchelvi, V., Prasanna Venkatesan, V., Shanmugasundaram, G., Achieving web services composition – a survey, Scientific and Academic publishing, 2(5): 195-202, 2012.
- (Raj *et al.*, 2010) : Raj, R.J.R.; Sasipraba, T. ; Web service selection based on *QoS* Constraints; Sathyabama Univ., Chennai, India (2010).
- (Ran, 2003): Ran, S., (2003), A model for Web services discovery with *QoS*, ACM SIGecom Ex-changes, 4(1), 1-10.
- (Riungu *et al.*, 2010): Riungu, L. M., Taipale, O., & Smolander, K. (2010), Software testing as an online service: Observations from practice. In ICSTW 2010 - 3rd International Conference on Software Testing, Verification, and Validation Workshops (pp. 418–423), doi:10.1109/ICSTW.2010.62
- (Rong and Liu, 2010): Wenge R., Kecheng L., (2010), “A Survey of Context Aware Web Service Discovery: From User’s Perspective”,Fifth IEEE International Symposium on Service Oriented System Engineering.
- (Rountree, 2013): Rountree, D., and Castrillo, I., (2013), The basics of Cloud computing: Understanding the fundamentals of Cloud computing in theory and practice. Burlington: Elsevier Science.
- (Narayanan et al, 2002): S. Narayanan, S. McIlraith, (2002) Simulation, verification and automated composition of Web services, in: Proceedings of the Eleventh International World Wide Web Conference (WWW), Honolulu, Hawaii, USA.
- (Sarnovsky *et al.*, 2012): Sarnovsky, M.; Butka, P.; Pocsova, J., "Cloud computing as a platform for distributed fuzzy FCA approach in data analysis," Intelligent Engineering Systems (INES), 2012 IEEE 16th International Conference

on , vol., no., pp.291,296, 13-15 June 2012, doi:
10.1109/INES.2012.6249847

- (Schlosser *et al.*, 2002): M. Schlosser, M. Sintek, S. Decker, W. Nejdl, (2002), A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services.
- (Security challenge, 2005): Security Challenges, Threats and Countermeasures, Web Services-Interoperability Organisation (WS-I), 2005. <http://www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0.pdf>
- (Seydim, 1999): Seydim, A Y 1999, "Intelligent agents: A data mining perspective", Southern Methodist University, Dallas.
- (Shaikh and Haider, 2011): Shaikh, F. B. F., & Haider, S. (2011). Security threats in Cloud computing. 2011 International Conference for Internet Technology and Secured Transactions, 214–219.
- (Shaikhali *et al.*, 2003): Shaikhali, A.; Rana, O.F.; Al-Ali,R.J.; & Walker, D.W.; UDDIe: An extended registry for Web services; Symposium on Applications and the Internet Work-shops; SAINT 03 Workshops (2003).
- (Singh *et al.* 2011): Singh, A, Juneja, D, & Sharma, A K 2011, "Agent development toolkits".
- (Sivashanmugam *et al.*, 2004): K. Sivashanmugam, Kunal Verma, Amit Sheth (2004): Discovery of Web Services in a Federated Registry Environment.
- (Smith and Nair, 2005): J. Smith, R. Nair, (2005), Virtual Machines: Versatile Platforms For Systems And Processes, book, Morgan Kaufmann.
- (SOA Definition, 2012): Service-oriented architecture (SOA) definition, <http://www.servicearchitecture.com/Web->

services/articles/serviceoriented_architecture_soa_definition.html,Barry & Associates, 2012.

- (Squicciarini *et al.*, 2011): Squicciarini, A.; Carminati, B. ; Karumanchi, S.; A Privacy-Preserving Approach for Web Service Selection and Provisioning, *Inf. Sci. & Technol.*, Penn-sylvania State Univ., University Park, PA, USA (2011).
- (Sun *et al.*, 2012): Sun, D., Chang, G., Jin, L., Sun, L., & Wang, X. (2012), Analyzing and modeling data center availability from replication perspective in Cloud computing environments *Information*.
- (Sun, 2012): Sun, D.-W., Chang, G.-R., Gao, S., Jin, L.-Z., & Wang, X.-W. (2012). Modeling a dynamic data replication strategy to increase system availability in Cloud computing environments. *Journal of Computer Science and Technology*, 27, 256–272, doi: 10.1007/s11390-012-1221-4
- (Schubert *et al.*, 2010): Schubert, L., Jeffery, K., & Neidecker-Lutz, B. (2010). The Future of Cloud Computing. Opportunities for European Cloud Computing Beyond 2010. European Commission, the Cloud Expert Group (p. 66). doi:10.1016/B978-1-59749-537-0.00012-0
- (Toma *et al.*, 2005): I. Toma, B. Sapkota, J. Secuila, J. M. Gomez, D. Roman, and C. Bussler (2005), P2p discovery mechanisms for Web service execution environment, Second WSMO Implementation Workshop.
- (Tsai and Chen, 2006): W.T. Tsai and Yinong Chen, Introduction to Service-Oriented Computing, Technology Based Learning and Research, Arizona State University (2006).
- (Tsai, 2011): Tsai, W.-T. b, Zhong, P. ., Balasooriya, J. ., Chen, Y. ., Bai, X. ., & Elston, J. . (2011), An approach for service composition and testing for

Cloud computing, In Proceedings - 2011 10th International Symposium on Autonomous Decentralized Systems, ISADS 2011 (pp. 631–636).

(Tudor *et al.* 2009): Tudor, I, & Ionita, L 2009, “Intelligent agents as data mining techniques used in academic environment”, In The 4th International Conference On Virtual Learning ,vol. 156, pp. 380-384.

(Irvine, 2014): Irvine (2014), UCI awarded \$8 million for creation of brain cell database, Los Angeles Times journal, <http://www.latimes.com/local/lanow/la-me-ln-uci-brain-cell-database-20141008-story.html>

(UDDI, 2011): Universal Description, Discovery, and Integration (UDDI) definition, http://www.servicearchitecture.com/Webservices/articles/universal_description_discovery_and_integration_uddi.html, 2011 Barry & Associates.

(Underwood, 2006): Learning How Brain Cells Communicate. Psych Central. Retrieved on October 11, 2014, from <http://psychcentral.com/blog/archives/2006/10/31/learning-how-brain-cells-communicate/>

(Venkatraman, 2014): Archana Venkatraman, Cloud impact can be as big as the advent of computing itself: EIU report, ComputerWeekly.com journal, June 2014.

(Verma *et al.*, 2003): K. Verma, K. Sivashanmugam, Amit Sheth, Abhijit Patil, Swapna Oundhakar, John Miller : METEOR–S WSDI (2003), A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services.

(w3c, 2004): Web service definition, <http://www.w3.org/TR/wsa-reqs/>, (2004).

- (Wang and Qian, 2005): A. Wang and K. Qian, Component-Oriented Programming, Wiley book, ISBN: 978-0-471-64446-0, 336 pages, March 2005.
- (Wang, 2012): Wang, L. (2012), Cloud computing: Methodology, systems, and applications. Boca Raton, FL: CRC Press.
- (Wang, 2007): Wang, H.; Combining subjective and objective *QoS* factors for personalized Web service selection; Institute of Information Management, National Cheng Kung University, 1st University Road, Tainan 701, Taiwan; Lee, C.; Ho, T.; Department of Computer Science, National University of Tainan, Taiwan; Expert Systems with Applications 32, 571–584 (2007).
- (WebService, 2012): Web Services explained. http://www.service-architecture.com/Webservices/articles/Web_services_explained.html, 2012 Barry & Associates. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, September 2012.
- (Wei and Blake, 2010): Yi Wei and M. Brian Blake, Service-Oriented Computing and Cloud Computing: Challenges and Opportunities, Published by the IEEE Computer Society, 2010, 1089-7801/10/.
- (Wen-yue *et al.*, 2010): Guo Wen-yue, Qu Hai-cheng, Chen Hong, (2010), “Semantic Web service discovery algorithm and its application on the intelligent automotive manufacturing system”, International Conference on Information Management and Engineering, IEEEExplore.
- (Wu et Yang, 2007): Wu J. et Yang F., *QoS* Prediction for Composite Web Services with Transactions, Lecture Notes in Computer Science, Springer Berlin Heidelberg, DOI: 10.1007/978-3-540-75492-3_8, 2007, pp 86-94.

- (Xiao *et al.*, 2001): Xiao R, Dillon T., Chand E, and Feng L., “Modelling and Transformation of Object Oriented Conceptual Models into XML Schema”. Lecture Notes in Computer Science, vol2113, Springer-Verlag, pp. 795-804, 2001.
- (Ylianttila, 2012): Ylianttila, M., Riekki, J., Zhou, J., Athukorala, K., & Gilman, E. (2012). Cloud Architecture for Dynamic Service Composition. *Int. J. Grid High Perform. Comput.*, 4, 17–31. doi:10.4018/jghpc.2012040102
- (Yu and Bouguettaya, 2010): Yu Q, and Bouguettaya A, (2010), Guest Editorial: Special Section on Query Models and Efficient Selection of Web Services, *IEEE Transactions on Services Computing*, Vol. 3, No. 3.
- (Zang *et al.*, 2010): Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1, 7–18. doi:10.1007/s13174-010-0007-6
- (Zeng *et al.*, 2009): Zeng, C., Guo, X., Ou, W., & Han, D. (2009). Cloud Computing Service Composition and Search Based on Semantic, 1st Int Conference on Cloud Computing CloudCom09, 5931, 290–300, doi: 10.1007/978-3-642-10665-1_26
- (Zghal *et al.* 2005): Zghal, H B, Faiz, S, & Ghezala, H B 2005, “A framework for data mining based multi-agent: An application to spatial data”, World Academy of Science, Engineering and Technology.
- (Zhao, 2012): Zhao, L., Sakr, S., Fekete, A., Wada, H., & Liu, A. (2012). Application-managed database replication on virtualized Cloud environments, In *Proceedings - 2012 IEEE 28th International Conference on Data Engineering Workshops, ICDEW 2012* (pp. 127–134), doi: 10.1109/ICDEW.2012.77

(Zhou *et al.*, 2007):

Zhou, G., Yu, J., Chen, R., Zhang, H., Scalable Web Service Discovery on P2P Overlay Network, 2007 IEEE International Conference on Services Computing, 2007.

APPENDIXES

Figure 5.2:



Figure 5.3:



Figure 5.4:

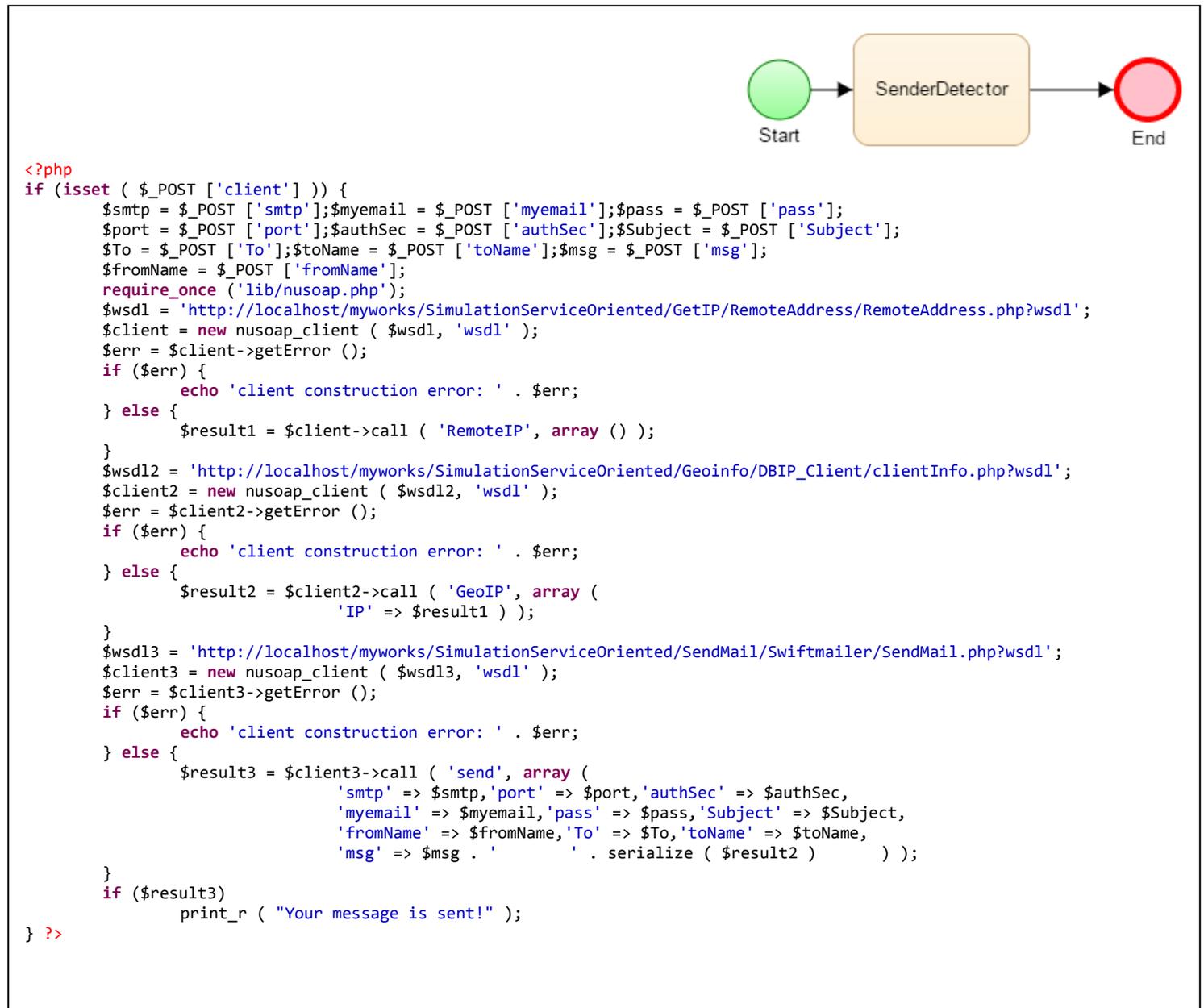


Figure 5.5:

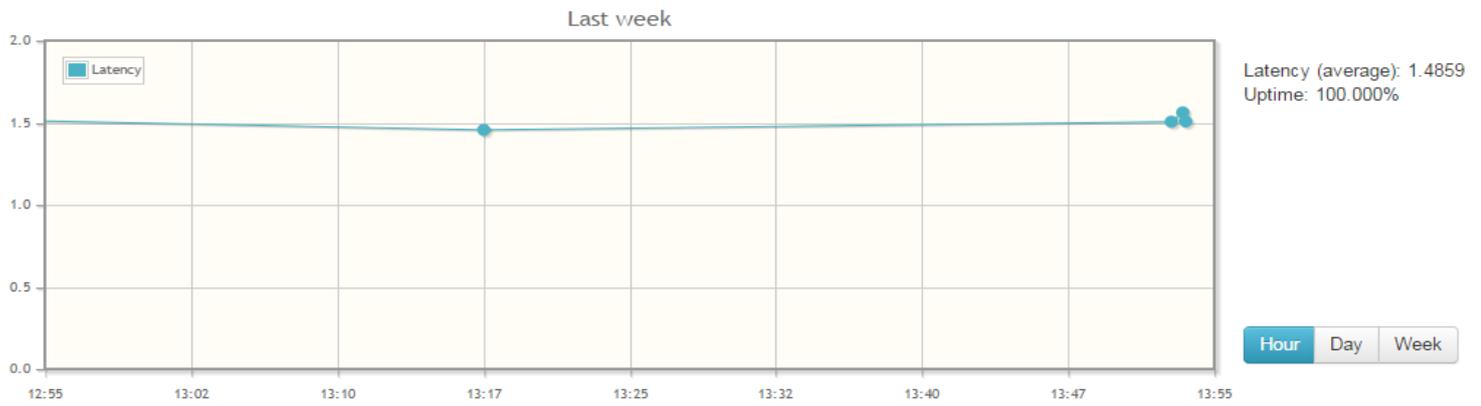


Figure 5.6:

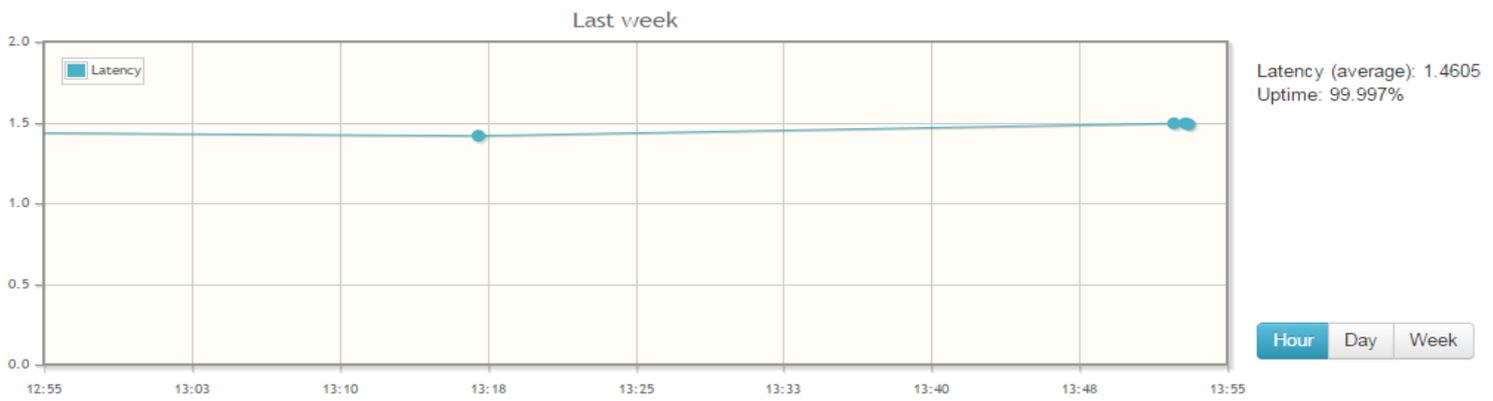


Figure 5.7:

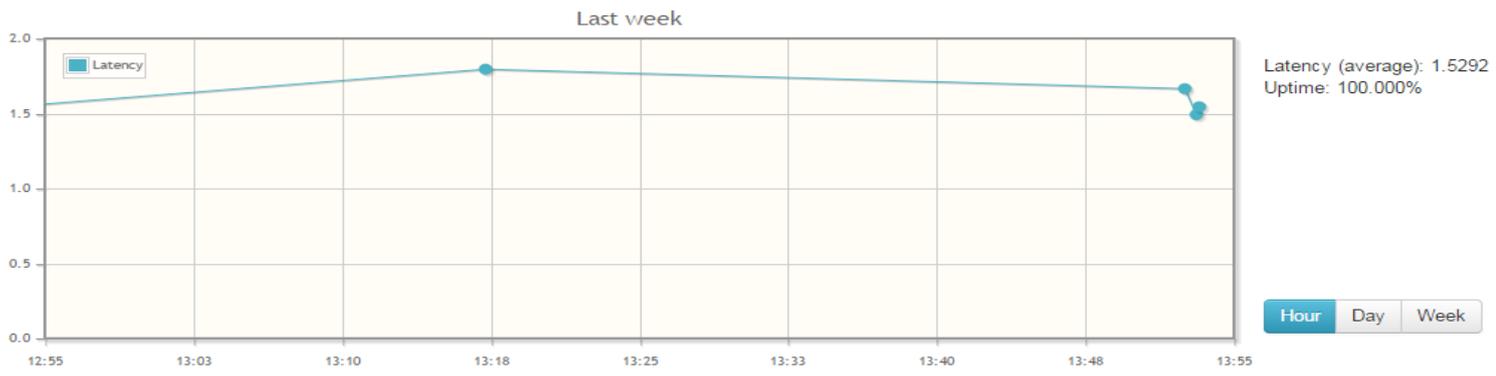


Figure 5.20:

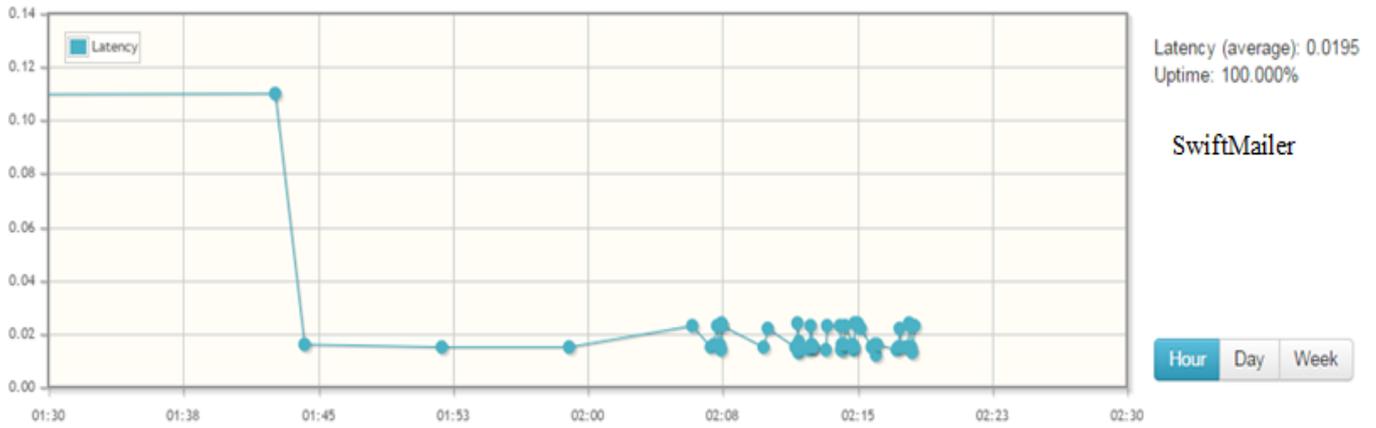
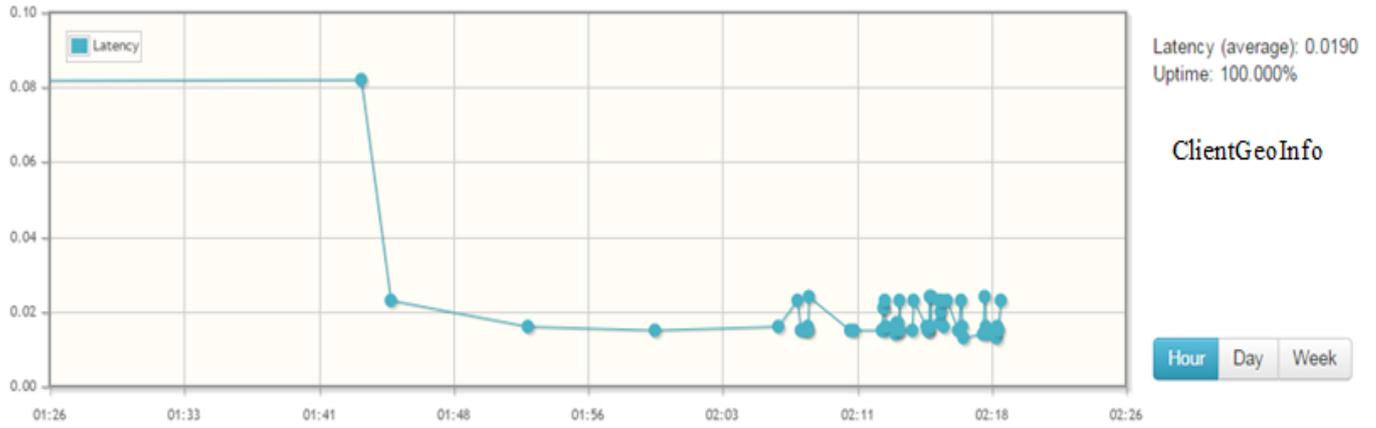
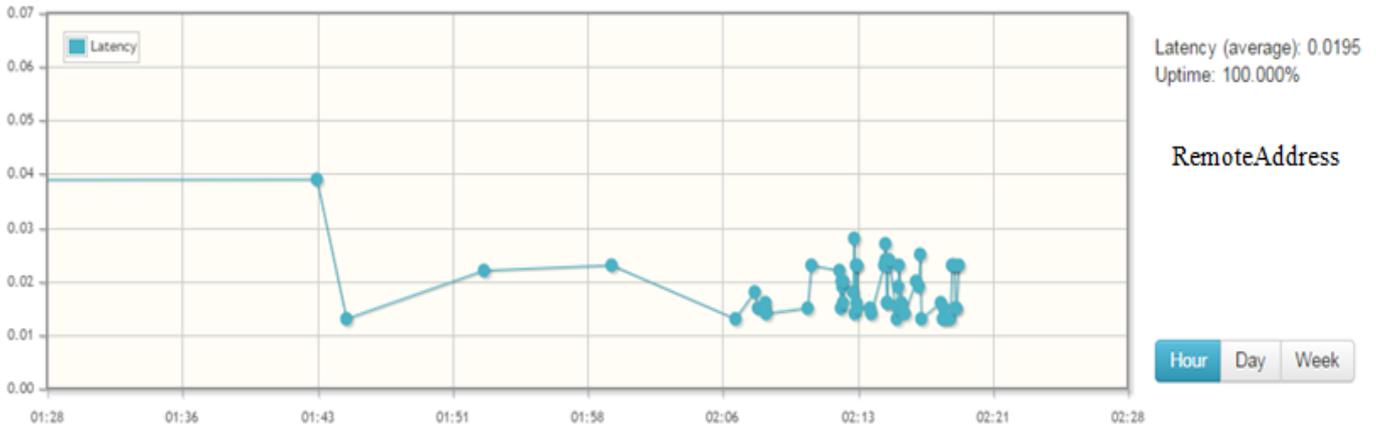
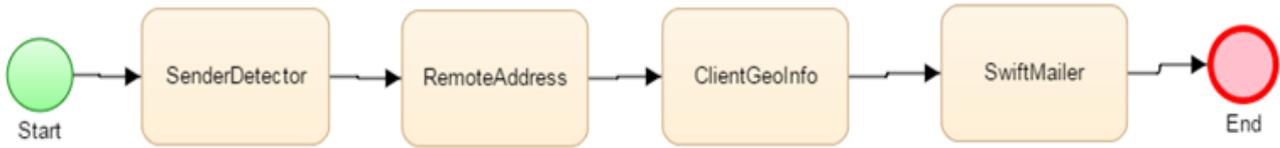


Figure 5.21:

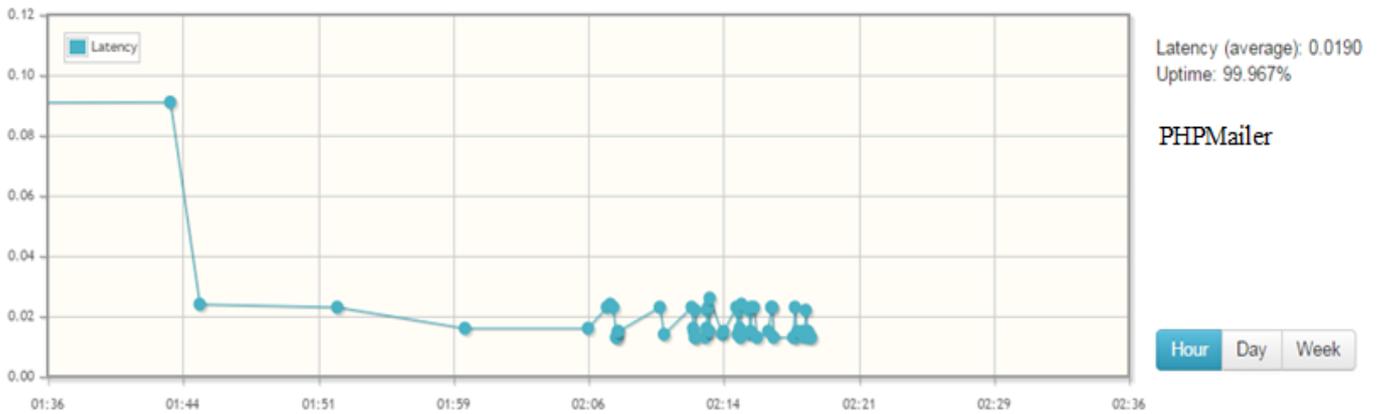
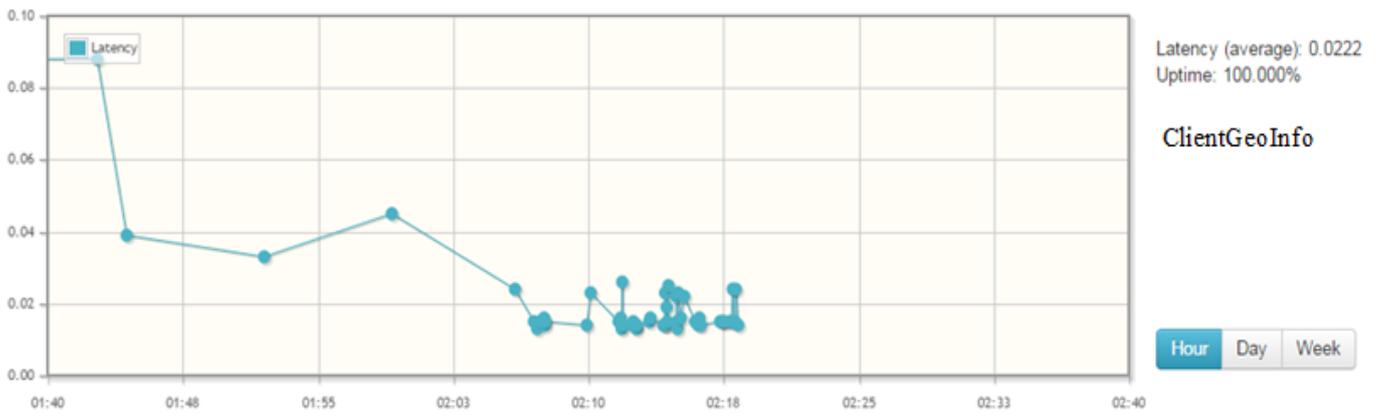
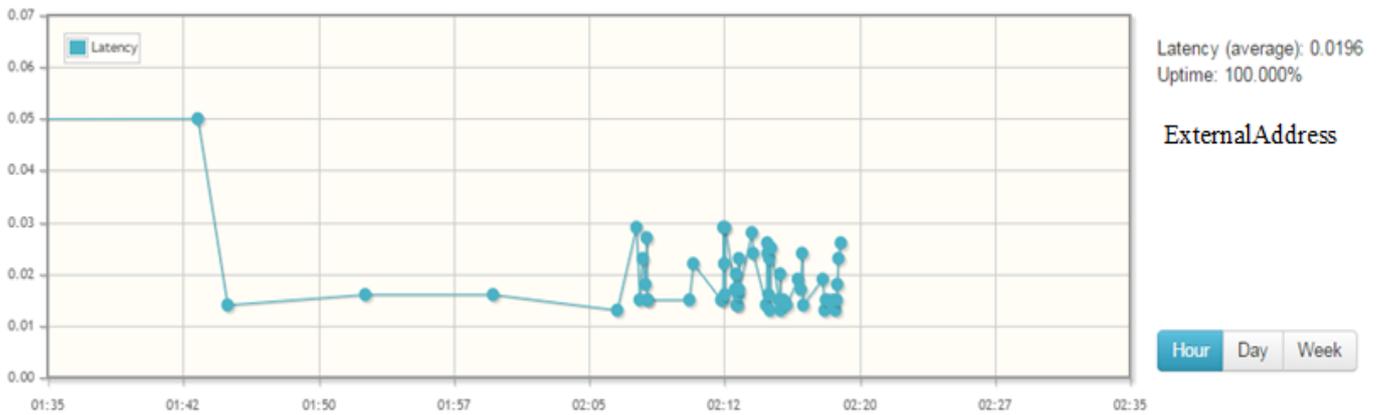
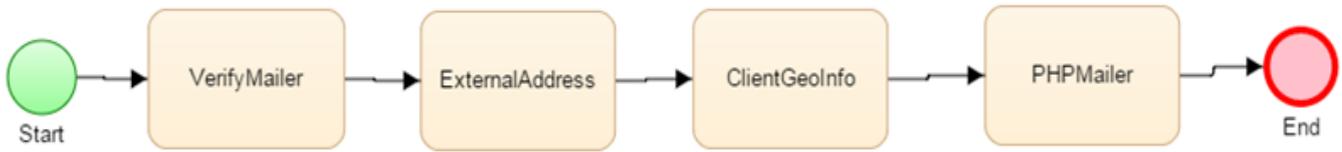


Figure 5.12:

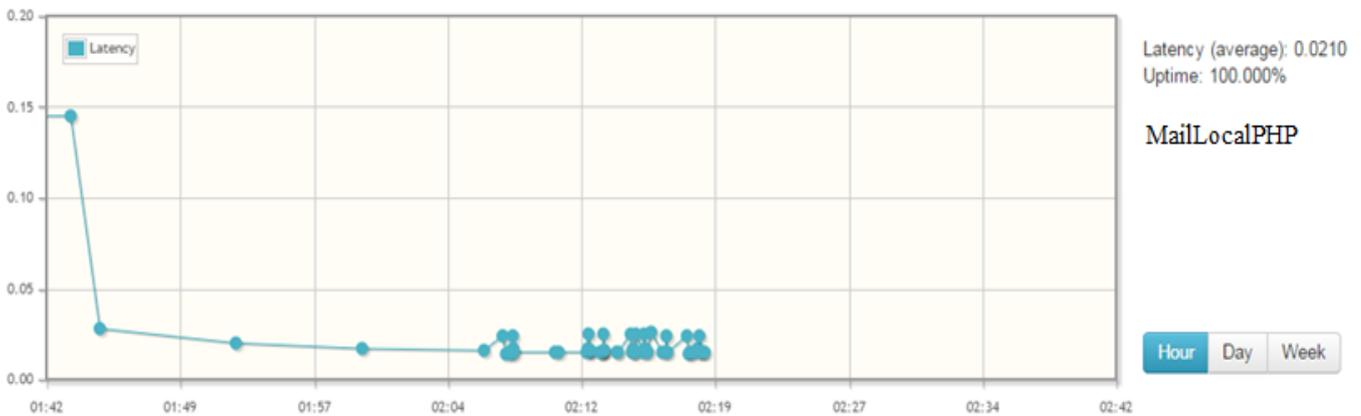
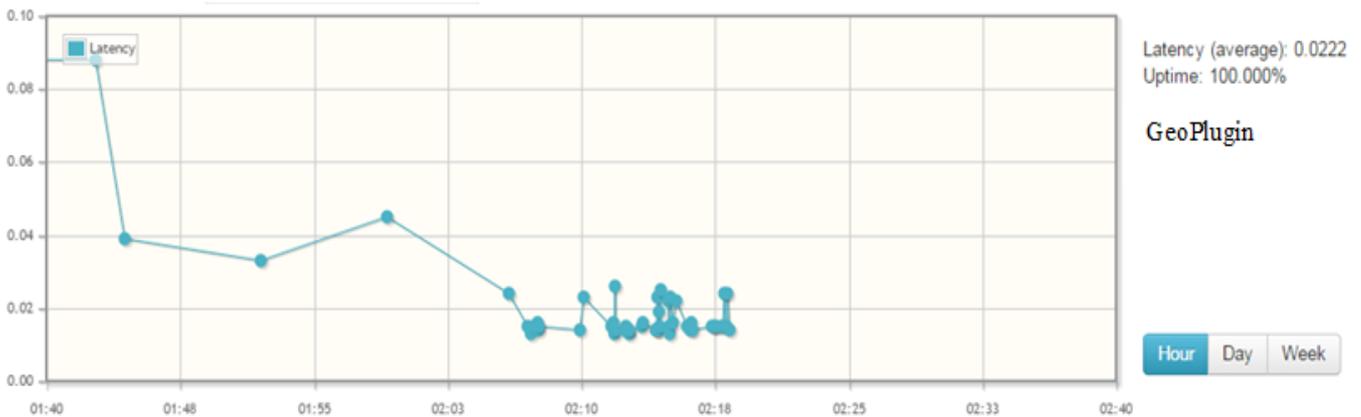
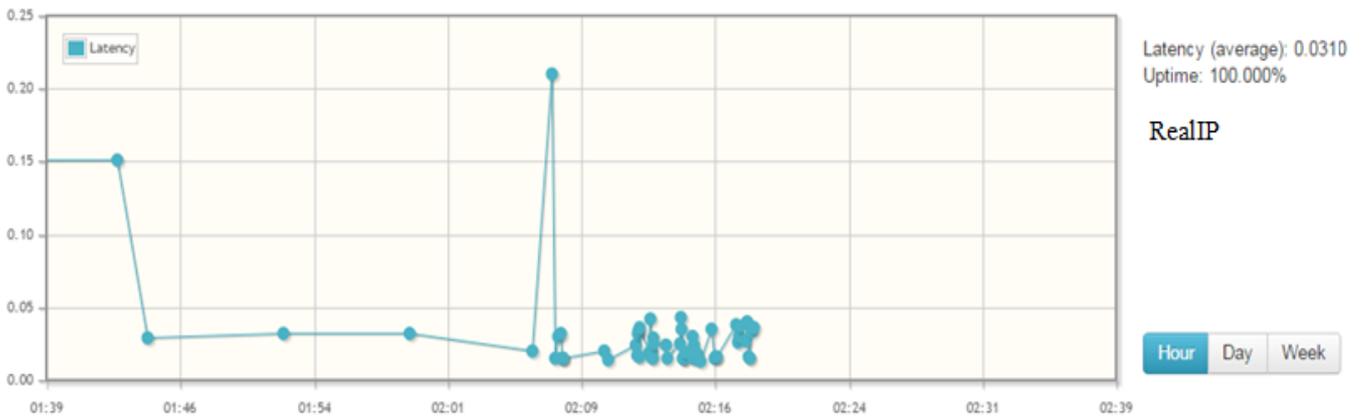
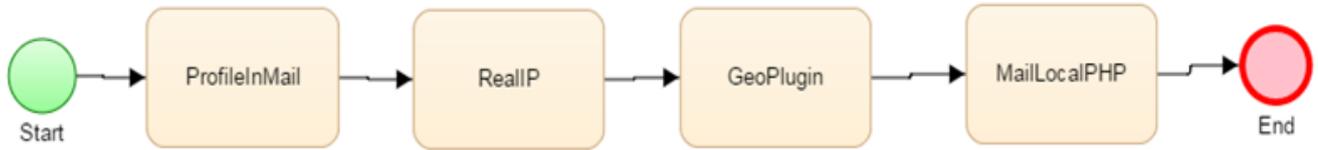


Figure 5.23:

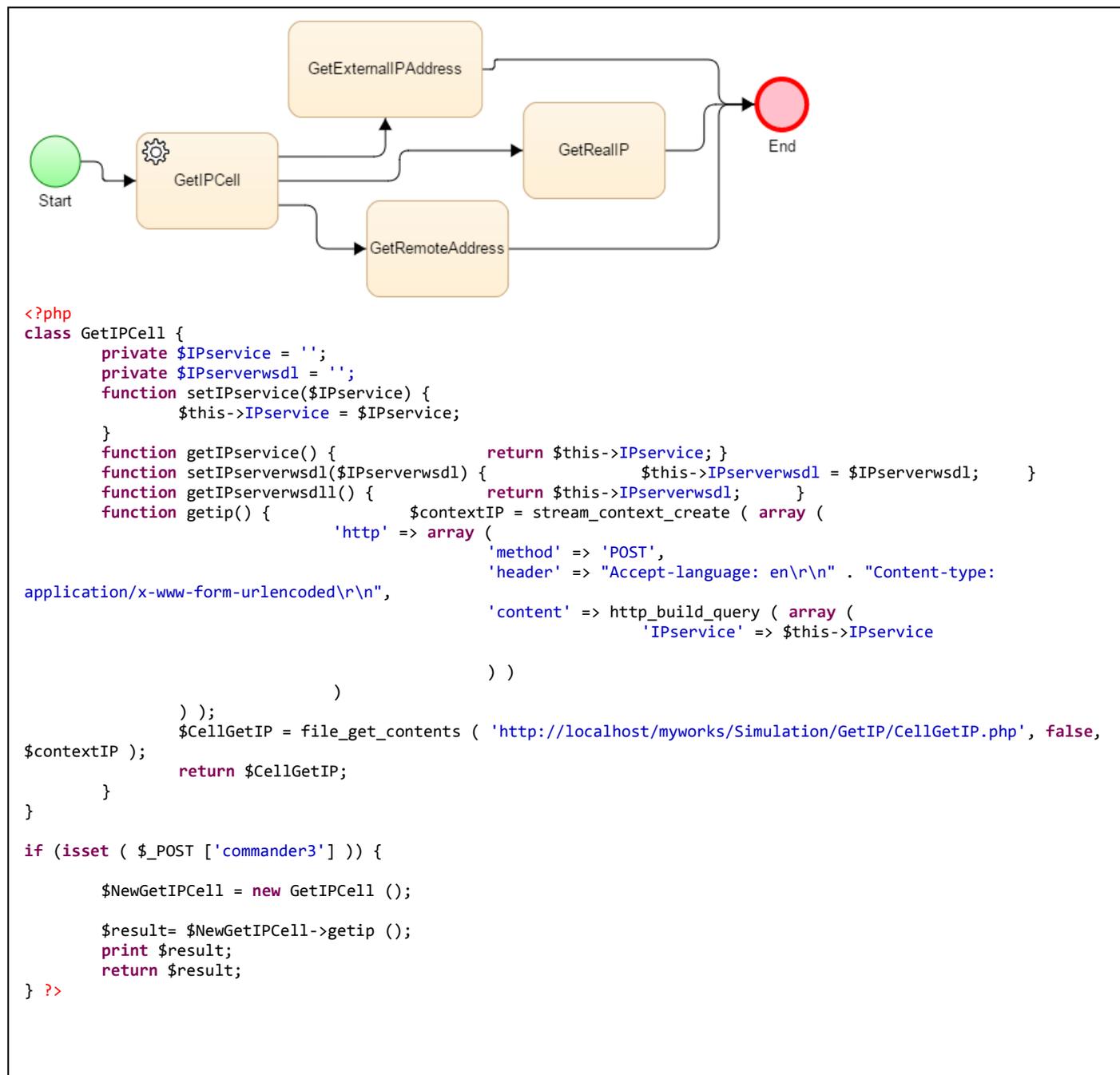
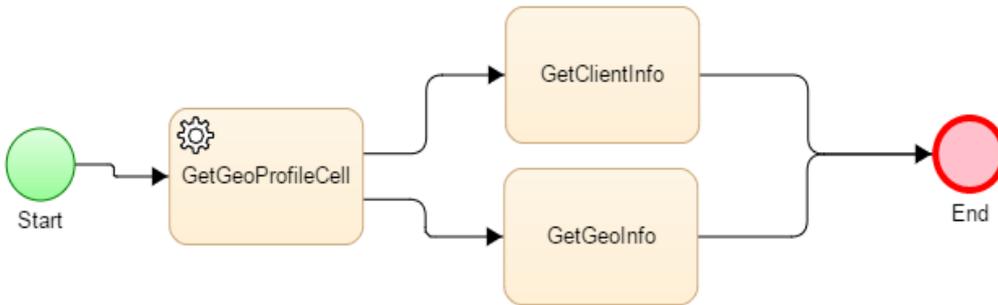


Figure 5.24:



```

<?php
class GetGeoProfileCell {
    protected $IPGeo = '';
    private $Geoservice = '';
    private $Geoserverwsdl = '';

    function getIPGeo() {
        return $this->IPGeo;
    }
    function setIPGeo($IP) {
        $this->IPGeo = $IP;
    }
    function setGeoservice($servicename) {
        $this->$Geoservice = $servicename;
    }
    function getGeoservice() {
        return $this->Geoservice;
    }
    function setGeoserverwsdl($Geoserverwsdl) {
        $this->Geoserverwsdl = $Geoserverwsdl;
    }
    function getGeoserverwsdl() {
        return $this->Geoserverwsdl;
    }
    function getgeoprofile() {
        $contextGeo = stream_context_create ( array (
            'http' => array (
                'method' => 'POST',
                'header' => "Accept-language: en\r\n" . "Content-type: application/x-
www-form-urlencoded\r\n",
                'content' => http_build_query ( array (
                    'IPGeo' => $this->IPGeo
                ) )
            )
        ) );
        $CellGeoProfile = file_get_contents ( 'http://localhost/myworks/Simulation/Geoinfo/CellGeoProfile.php',
false, $contextGeo );
        return $CellGeoProfile;
    }
}

if (isset ( $_POST ['commander2'] )) {
    $NewGetGeoProfileCell= new GetGeoProfileCell();
    $NewGetGeoProfileCell->setIPGeo($_POST['ipaddress']);
    $result= $NewGetGeoProfileCell->getgeoprofile();
    print $result;
    return $result;
} ?>
  
```

Figure 5.25:

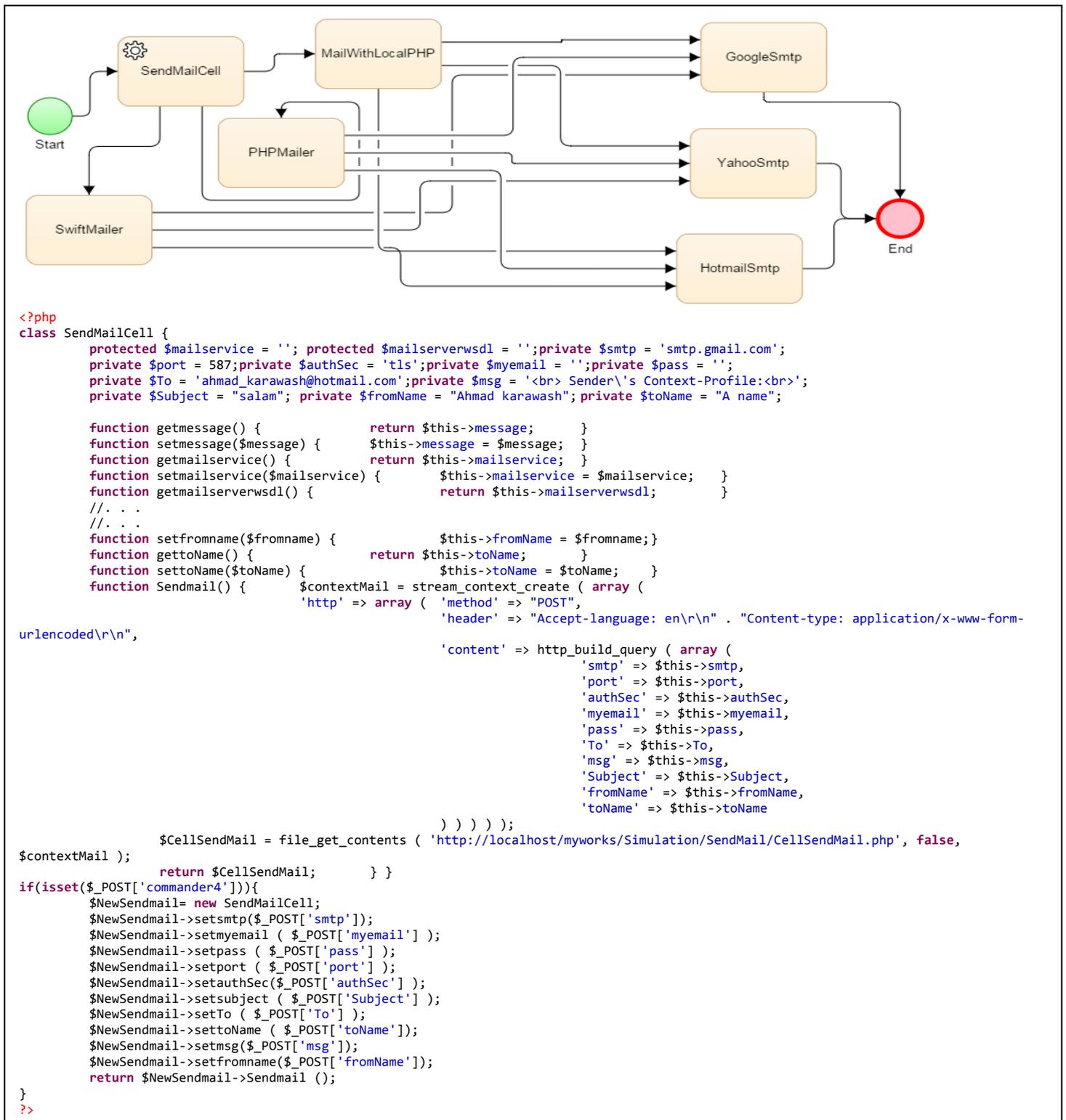
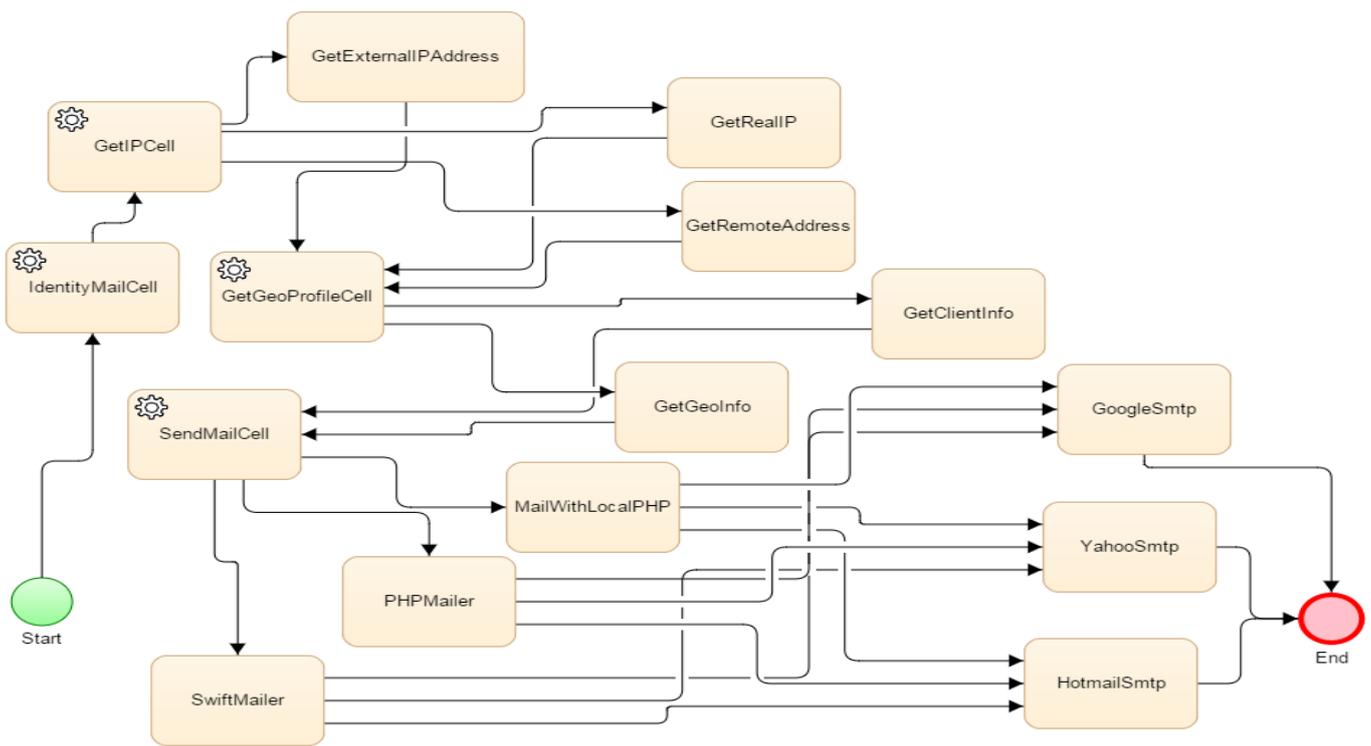


Figure 5.26:



```

<?php
require 'GetIPCell.php';
require 'GetGeoProfileCell.php';
require 'SendMailCell.php';
class IdentityMailCell {
    private $smtp = 'smtp.gmail.com';
    private $port = 587;
    private $authSec = 'tls';
    private $myemail = '';
    private $pass = '';
    private $To = 'ahmad_karawash@hotmail.com';
    private $Subject = "salam";
    private $fromName = "Ahmad karawash";
    private $toName = "A name";
    private $msg = '';

    function getmessage() {
        return $this->message;
    }
    function setmessage($message) {
        $this->message = $message;
    }
    function getmailservice() {
        return $this->mailservice;
    }
    function setmailservice($mailservice) {
        $this->mailservice = $mailservice;
    }
}

```

```

function getmailserverwsdl() { return $this->mailserverwsdl; }
function setmailserverwsdl($mailserverwsdl) { $this->mailserverwsdl = $mailserverwsdl; }
function getsmtpp() { return $this->smtpp; }
function setsmtpp($smtpp) { $this->smtpp = $smtpp; }
function getport() { return $this->port; }
function setport($port) { $this->port = $port; }
function getauthSec() { return $this->authSec; }
function setauthSec($authSec) { $this->authSec = $authSec; }
function getmyemail() { return $this->myemail; }
function setmyemail($email) { $this->myemail = $email; }
function getpass() { return $this->pass; }
function setpass($pass) { $this->pass = $pass; }
function getTo() { return $this->To; }
function setTo($to) { $this->To = $to; }
function getmsg() { return $this->msg; }
function setmsg($msg) { $this->msg = $msg; }
function getsubject() { return $this->Subject; }
function setsubject($subject) { $this->Subject = $subject; }
function getfromname() { return $this->fromName; }
function setfromname($fromname) { $this->fromName = $fromname; }
function gettoName() { return $this->toName; }
function settoName($toName) { $this->toName = $toName; }
function Identitymailprocess() {
    $GetIp = new GetIPCell ();
    $GetGeoProfile = new GetGeoProfileCell ();
    $GetGeoProfile->setIPGeo ( $GetIp->getip () );
    $SendMail = new SendMailCell ();
    $SendMail->setauthSec ( $this->authSec );
    $SendMail->setfromname ( $this->fromName );
    $SendMail->setmessage ( $this->msg );
    $SendMail->setmsg ( ' ' .

    $SendMail->getmessage () . '

    Sender\'s Context-Profile:

    ' . $GetGeoProfile->getgeoprofile () . ' ' );
    $SendMail->setmyemail ( $this->myemail );
    $SendMail->setpass ( $this->pass );
    $SendMail->setport ( $this->port );
    $SendMail->setsmtpp ( $this->smtpp );
    $SendMail->setsubject ( $this->Subject );
    $SendMail->setTo ( $this->To );
    $SendMail->settoName ( $this->toName );
    if($SendMail->Sendmail())
    return 'done';
}
}
if (isset ( $_POST ['commander'] )) {
    $NewSendmail = new IdentityMailCell ();
    $NewSendmail->setsmtpp ( $_POST ['smtpp'] );
    $NewSendmail->setmyemail ( $_POST ['myemail'] );
    $NewSendmail->setpass ( $_POST ['pass'] );
    $NewSendmail->setport ( $_POST ['port'] );
    $NewSendmail->setauthSec ( $_POST ['authSec'] );
    $NewSendmail->setsubject ( $_POST ['Subject'] );
    $NewSendmail->setTo ( $_POST ['To'] );
    $NewSendmail->settoName ( $_POST ['toName'] );
    $NewSendmail->setmsg ( $_POST ['msg'] );
    $NewSendmail->setfromname ( $_POST ['fromName'] );

    if($NewSendmail->Identitymailprocess ())
        print 'Message is recieved and the Sender context-Profile is detected';
}
}

```

Figure 5.27:

The image displays two screenshots from SQL Server Enterprise Manager. The top screenshot shows the 'dbo.Node' table with 16 rows. The bottom screenshot shows the 'dbo.Edge' table with 25 rows. To the right of these screenshots is a large block of SQL code for inserting data into these tables.

Id	Name
1	IdentityMailCell
2	GetIPCell
3	GetGeoProfileC...
4	SendMailCell
5	GetExternalIPA...
6	GetRealIP
7	GetRemoteAdd...
8	GetClientInfo
9	GetGeoInfo
10	MailWithLocalP...
11	PHPMailer
12	SwiftMailer
13	GoogleSmtmp
14	YahooSmtmp
15	HotmailSmtmp
16	Result
*	NULL

FromNode	ToNode	Weight
1	2	1306.000
2	5	1507.000
2	6	919.000
2	7	629.000
3	8	613.000
3	9	435.000
3	2	537.000
4	3	265.000
4	10	1983.000
4	11	325.000
4	12	765.000
5	3	2161.000
6	3	1225.000
7	3	1483.000
8	4	1258.000
9	4	2661.000
10	13	1532.000
10	14	661.000
10	15	1481.000
11	13	1258.000
11	14	1722.000
11	15	2113.000
12	13	2161.000
12	14	243.000
12	15	1145.000
13	16	564.000
14	16	383.000
15	16	1409.000
3	2	537.000
4	3	265.000

```

INSERT dbo.Node (Id, Name) VALUES (1, 'IdentityMailCell')
INSERT dbo.Node (Id, Name) VALUES (2, 'GetIPCell')
INSERT dbo.Node (Id, Name) VALUES (3, 'GetGeoProfileCell')
INSERT dbo.Node (Id, Name) VALUES (4, 'SendMailCell')
INSERT dbo.Node (Id, Name) VALUES (5, 'GetExternalIPAddress')
INSERT dbo.Node (Id, Name) VALUES (6, 'GetRealIP')
INSERT dbo.Node (Id, Name) VALUES (7, 'GetRemoteAddress')
INSERT dbo.Node (Id, Name) VALUES (8, 'GetClientInfo')
INSERT dbo.Node (Id, Name) VALUES (9, 'GetGeoInfo')
INSERT dbo.Node (Id, Name) VALUES (10, 'MailWithLocalPHP')
INSERT dbo.Node (Id, Name) VALUES (11, 'PHPMailer')
INSERT dbo.Node (Id, Name) VALUES (12, 'SwiftMailer')
INSERT dbo.Node (Id, Name) VALUES (13, 'GoogleSmtmp')
INSERT dbo.Node (Id, Name) VALUES (14, 'YahooSmtmp')
INSERT dbo.Node (Id, Name) VALUES (15, 'HotmailSmtmp')
INSERT dbo.Node (Id, Name) VALUES (16, 'Result')

INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (1, 2, 1306.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (2, 5, 1507.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (2, 6, 919.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (2, 7, 629.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 8, 613.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 9, 435.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 2, 537.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 3, 265.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 10, 1983.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 11, 325.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 12, 765.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (5, 3, 2161.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (6, 3, 1225.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (7, 3, 1483.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (8, 4, 1258.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (9, 4, 2661.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (10, 13,
1532.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (10, 14, 661.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (10, 15,
1481.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (11, 13,
1258.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (11, 14,
1722.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (11, 15,
2113.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (12, 13,
2161.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (12, 14, 243.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (12, 15,
1145.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (13, 16, 564.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (14, 16, 383.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (15, 16,
1409.000)

// edges cause cycle
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 2, 537.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 3, 265.000)
//

```

Figure 5.28:

```

CREATE PROCEDURE dbo.usp_Prim
AS
BEGIN
    SET XACT_ABORT ON
    BEGIN TRAN
    SET NOCOUNT ON;
    CREATE TABLE #Nodes
    (
        Id int NOT NULL PRIMARY KEY,
        Estimate decimal(10,3) NOT NULL,
        Predecessor int NULL,
        Done bit NOT NULL )
    INSERT INTO #Nodes (Id, Estimate, Predecessor, Done)
    SELECT Id, 9999999.999, NULL, 0 FROM dbo.Node
    UPDATE TOP (1) #Nodes SET Estimate = 0
    DECLARE @FromNode int
    WHILE 1 = 1
    BEGIN
        SELECT @FromNode = NULL
        SELECT TOP 1 @FromNode = Id
        FROM #Nodes WHERE Done = 0 AND Estimate < 9999999.999
        ORDER BY Estimate
        IF @FromNode IS NULL BREAK
        UPDATE #Nodes SET Done = 1 WHERE Id = @FromNode
        UPDATE #Nodes
            SET Estimate = e.Weight, Predecessor =
@FromNode
        FROM #Nodes n INNER JOIN dbo.Edge e ON n.Id = e.ToNode
        WHERE Done = 0 AND e.FromNode = @FromNode AND e.Weight
< n.Estimate
        END
        IF EXISTS (SELECT TOP 1 1 FROM #Nodes WHERE Done = 0)
        BEGIN
            DROP TABLE #Nodes
            RAISERROR('Error: The graph is not
connected.', 1, 1)
            ROLLBACK TRAN
            RETURN 1
        END
        SELECT n.Predecessor AS FromNode, n.Id AS ToNode,
            node1.Name AS FromName, node2.Name AS ToName
        FROM #Nodes n
        JOIN dbo.Node node1 ON n.Predecessor = node1.Id
        JOIN dbo.Node node2 ON n.Id = node2.id
        WHERE n.Predecessor IS NOT NULL
        ORDER BY n.Predecessor, n.id
        DROP TABLE #Nodes
    COMMIT TRAN
    RETURN 0
END
GO

```

	FromNode	ToNode	FromName	ToName
1	1	2	IdentityMailCell	GetIPCell
2	2	5	GetIPCell	GetExtremalIPAddress
3	2	6	GetIPCell	GetRealIP
4	2	7	GetIPCell	GetRemoteAddress
5	3	8	GetGeoProfileCell	GetClientInfo
6	3	9	GetGeoProfileCell	GetGeoInfo
7	4	10	SendMailCell	MailWithLocalPHP
8	4	11	SendMailCell	PHPMailer
9	4	12	SendMailCell	SwiftMailer
10	6	3	GetRealIP	GetGeoProfileCell
11	8	4	GetClientInfo	SendMailCell
12	11	13	PHPMailer	GoogleSmtpt
13	12	14	SwiftMailer	YahooSmtpt
14	12	15	SwiftMailer	HotmailSmtpt
15	14	16	YahooSmtpt	Result

Query executed successfully.

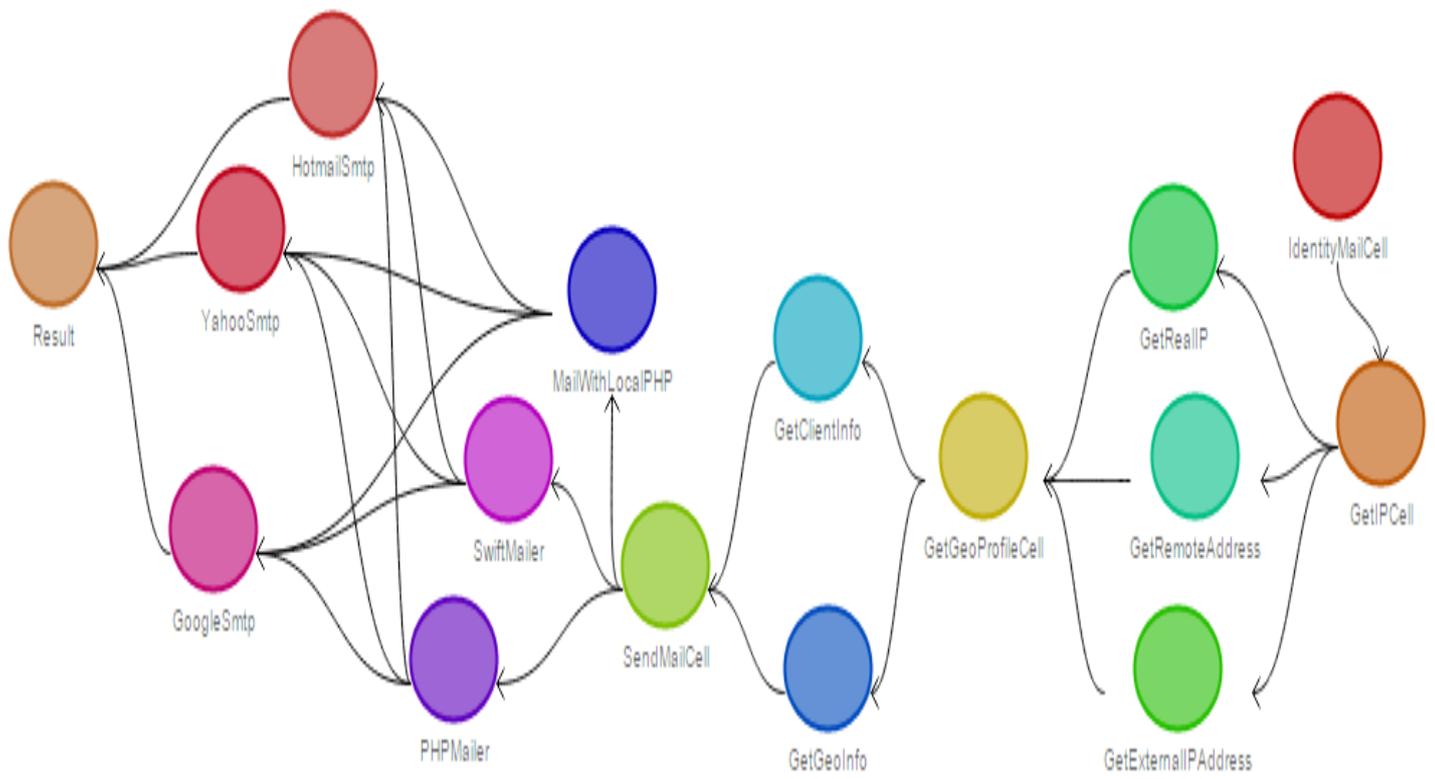


Figure 5.29:

TRACE PATHS

Distance cutoff: 8

54 paths found.

Selected Paths from 1 to 16

1: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 10 -> 13 -> 16	41: 1 -> 2 -> 7 -> 3 -> 8 -> 4 -> 11 -> 14 -> 16
2: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 10 -> 14 -> 16	42: 1 -> 2 -> 7 -> 3 -> 8 -> 4 -> 11 -> 15 -> 16
3: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 10 -> 15 -> 16	43: 1 -> 2 -> 7 -> 3 -> 8 -> 4 -> 12 -> 13 -> 16
4: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 11 -> 13 -> 16	44: 1 -> 2 -> 7 -> 3 -> 8 -> 4 -> 12 -> 14 -> 16
5: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 11 -> 14 -> 16	45: 1 -> 2 -> 7 -> 3 -> 8 -> 4 -> 12 -> 15 -> 16
6: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 11 -> 15 -> 16	46: 1 -> 2 -> 7 -> 3 -> 9 -> 4 -> 10 -> 13 -> 16
7: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 12 -> 13 -> 16	47: 1 -> 2 -> 7 -> 3 -> 9 -> 4 -> 10 -> 14 -> 16
8: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 12 -> 14 -> 16	48: 1 -> 2 -> 7 -> 3 -> 9 -> 4 -> 10 -> 15 -> 16
9: 1 -> 2 -> 5 -> 3 -> 8 -> 4 -> 12 -> 15 -> 16	49: 1 -> 2 -> 7 -> 3 -> 9 -> 4 -> 11 -> 13 -> 16
10: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 10 -> 13 -> 16	50: 1 -> 2 -> 7 -> 3 -> 9 -> 4 -> 11 -> 14 -> 16
11: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 10 -> 14 -> 16	51: 1 -> 2 -> 7 -> 3 -> 9 -> 4 -> 11 -> 15 -> 16
12: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 10 -> 15 -> 16	52: 1 -> 2 -> 7 -> 3 -> 9 -> 4 -> 12 -> 13 -> 16
13: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 11 -> 13 -> 16	
14: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 11 -> 14 -> 16	
15: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 11 -> 15 -> 16	
16: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 12 -> 13 -> 16	
17: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 12 -> 14 -> 16	
18: 1 -> 2 -> 5 -> 3 -> 9 -> 4 -> 12 -> 15 -> 16	
19: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 10 -> 13 -> 16	
20: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 10 -> 14 -> 16	
21: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 10 -> 15 -> 16	
22: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 11 -> 13 -> 16	
23: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 11 -> 14 -> 16	
24: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 11 -> 15 -> 16	
25: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 12 -> 13 -> 16	
26: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 12 -> 14 -> 16	
27: 1 -> 2 -> 6 -> 3 -> 8 -> 4 -> 12 -> 15 -> 16	
28: 1 -> 2 -> 6 -> 3 -> 9 -> 4 -> 10 -> 13 -> 16	
29: 1 -> 2 -> 6 -> 3 -> 9 -> 4 -> 10 -> 14 -> 16	

The screenshot shows the UCINET 6 for Windows software interface. The main window is titled 'UCINET 6 for Windows -- Version 6.528'. Below the menu bar (File, Data, Transform, Tools, Network, Visualize, Options, Help) is a toolbar with various icons. A 'Recent Commands' section is visible. Below the toolbar is a 'How to cite UCINET:' section with a citation: 'Borgatti, S.P., Everett, M.G. and Freeman, L.C. 2002. Ucinet 6 for Windows: Software for Social Network Analysis.' The 'Trace Paths' dialog box is open, showing the following settings:

- Input Dataset: C:\Users\User\Desktop\SmartCells-Net. #.h
- From node: 1
- To node: 16
- Type of paths desired:
 - Avg Strength >= K
 - Length <= K
- Value of K: Auto

Figure 5.30:

```
CREATE PROCEDURE dbo.TopologicalSort
AS
BEGIN
    SET XACT_ABORT ON
    BEGIN TRAN
    SET NOCOUNT ON;
    CREATE TABLE #Order
    (
        NodeId int PRIMARY KEY,
        Ordinal int NULL
    )
    CREATE TABLE #TempEdges
    (
        FromNode int,
        ToNode int,
        PRIMARY KEY (FromNode, ToNode)
    )
    INSERT INTO #TempEdges (FromNode, ToNode)
    SELECT e.FromNode, e.ToNode
    FROM dbo.Edge e
    INSERT INTO #Order (NodeId, Ordinal)
    SELECT n.Id, NULL
    FROM dbo.Node n
    WHERE NOT EXISTS (
        SELECT TOP 1 1 FROM dbo.Edge e WHERE e.ToNode = n.Id)
    DECLARE @CurrentNode int,
            @Counter int = 0
    WHILE 1 = 1
    BEGIN
        SET @CurrentNode = NULL
        SELECT TOP 1 @CurrentNode = NodeId
        FROM #Order WHERE Ordinal IS NULL
        IF @CurrentNode IS NULL BREAK
        UPDATE #Order SET Ordinal = @Counter, @Counter = @Counter + 1
        WHERE NodeId = @CurrentNode
        INSERT #Order (NodeId, Ordinal)
        SELECT Id, NULL
        FROM dbo.Node n
        JOIN #TempEdges e1 ON n.Id = e1.ToNode
        WHERE e1.FromNode = @CurrentNode AND
            NOT EXISTS (
                SELECT TOP 1 1 FROM #TempEdges e2
                WHERE e2.ToNode = n.Id AND e2.FromNode <> @CurrentNode)
        DELETE FROM #TempEdges WHERE FromNode = @CurrentNode
    END
    IF EXISTS (SELECT TOP 1 1 FROM #TempEdges)
    BEGIN
        SELECT 'The graph contains cycles and no topological ordering can
            be produced. This is the set of edges I could not remove:'
        SELECT FromNode, ToNode FROM #TempEdges
    END
    ELSE
        SELECT n.Id, n.Name
        FROM dbo.Node n
        JOIN #Order o ON n.Id = o.NodeId
        ORDER BY o.Ordinal
    DROP TABLE #TempEdges
    DROP TABLE #Order
    COMMIT TRAN
    RETURN 0
END
```

Fake edges:

```
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (3, 2, 537.000)
INSERT dbo.Edge (FromNode, ToNode, [Weight]) VALUES (4, 3, 265.000)
```

The screenshot shows the SQL Server Enterprise Manager interface. At the top, there are tabs for 'Results' and 'Messages'. The 'Messages' tab is active, displaying a single error message: 'The graph contains cycles and no topological ordering can be produced. This is the set of edges I could not remove:'. Below the message, a table lists 11 edges with columns 'FromNode' and 'ToNode'. The first row of the table is highlighted. At the bottom of the window, a yellow status bar indicates 'Query executed successfully.' and the session ID 'DIM-P3-506'.

	FromNode	ToNode
1	2	5
2	2	6
3	2	7
4	3	2
5	3	8
6	3	9
7	4	3
8	4	10
9	4	11
10	4	12
11	5	3

Query executed successfully. DIM-P3-506

Figure 5.31:

The screenshot shows a SQL query result in a window titled "DIM-P3-5060A-R\S...rithms - dbo.Node" and "SQLQuery1.sql - DI". The table has three columns: FromNode, ToNode, and Weight. The data is as follows:

	FromNode	ToNode	Weight
▶	1	2	1306.000
	2	5	1306.000
	2	6	919.000
	2	7	629.000
	3	2	435.000
	3	8	919.000
	3	9	435.000
	4	3	435.000
	4	10	1983.000
	4	11	629.000
	4	12	435.000
	5	3	2161.000
	6	3	1225.000
	7	3	1483.000
	8	4	1258.000
	9	4	2661.000
	10	13	1532.000
	10	14	661.000
	10	15	1983.000
	11	13	1258.000
	11	14	1532.000
	11	15	2113.000
	12	13	2161.000
	12	14	661.000
	12	15	1145.000
	13	16	725.000
	14	16	383.000
	15	16	1709.000
*	NULL	NULL	NULL


```

CREATE PROCEDURE dbo.usp_Dijkstra (@StartNode int, @EndNode int = NULL)
AS
BEGIN
    SET XACT_ABORT ON
    BEGIN TRAN
    SET NOCOUNT ON;
    CREATE TABLE #Nodes
    (
        Id int NOT NULL PRIMARY KEY,
        Estimate decimal(10,3) NOT NULL,
        Predecessor int NULL,
        Done bit NOT NULL
    )
    INSERT INTO #Nodes (Id, Estimate, Predecessor, Done)
    SELECT Id, 9999999.999, NULL, 0 FROM dbo.Node
    UPDATE #Nodes SET Estimate = 0 WHERE Id = @StartNode
    IF @@rowcount <> 1
    BEGIN
        DROP TABLE #Nodes
        RAISERROR ('Could not set start node', 11, 1)
        ROLLBACK TRAN
        RETURN 1
    END
    DECLARE @FromNode int, @CurrentEstimate int
    WHILE 1 = 1
    BEGIN
        SELECT @FromNode = NULL
        SELECT TOP 1 @FromNode = Id, @CurrentEstimate = Estimate
        FROM #Nodes WHERE Done = 0 AND Estimate < 9999999.999
        ORDER BY Estimate
        IF @FromNode IS NULL OR @FromNode = @EndNode BREAK
        UPDATE #Nodes SET Done = 1 WHERE Id = @FromNode
        UPDATE #Nodes
            SET Estimate = @CurrentEstimate + e.Weight, Predecessor = @FromNode
        FROM #Nodes n INNER JOIN dbo.Edge e ON n.Id = e.ToNode
        WHERE Done = 0 AND e.FromNode = @FromNode AND (@CurrentEstimate + e.Weight) < n.Estimate
    END;
    WITH BacktraceCTE(Id, Name, Distance, Path, NamePath)
    AS
    (
        SELECT n.Id, node.Name, n.Estimate, CAST(n.Id AS varchar(8000)),
            CAST(node.Name AS varchar(8000))
        FROM #Nodes n JOIN dbo.Node node ON n.Id = node.Id
        WHERE n.Id = @StartNode
        UNION ALL
        SELECT n.Id, node.Name, n.Estimate,
            CAST(cte.Path + ',' + CAST(n.Id as varchar(10)) as varchar(8000)),
            CAST(cte.NamePath + ',' + node.Name AS varchar(8000))
        FROM #Nodes n JOIN BacktraceCTE cte ON n.Predecessor = cte.Id
        JOIN dbo.Node node ON n.Id = node.Id
    )
    SELECT Id, Name, Distance, Path, NamePath FROM BacktraceCTE
    WHERE Id = @EndNode OR @EndNode IS NULL
    ORDER BY Id
    DROP TABLE #Nodes
    COMMIT TRAN
    RETURN 0
END
GO

```

Results		Messages				
Id	Name	Distance	Path	NamePath		
1	1	IdentityMailCell	0.000	1	IdentityMailCell	
2	2	GetIPCell	1306.000	1,2	IdentityMailCell,GetIPCell	
3	3	GetGeoProfileCell	3418.000	1,2,7,3	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell	
4	4	SendMailCell	5595.000	1,2,7,3,8,4	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo,SendMailCell	
5	5	GetExternalIPAddress	2612.000	1,2,5	IdentityMailCell,GetIPCell,GetExternalIPAddress	
6	6	GetRealIP	2225.000	1,2,6	IdentityMailCell,GetIPCell,GetRealIP	
7	7	GetRemoteAddress	1935.000	1,2,7	IdentityMailCell,GetIPCell,GetRemoteAddress	
8	8	GetClientInfo	4337.000	1,2,7,3,8	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo	
9	9	GetGeoInfo	3853.000	1,2,7,3,9	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetGeoInfo	
10	10	MailWithLocalPHP	7578.000	1,2,7,3,8,4,10	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,MailWithLocalPHP	
11	11	PHPMailer	6224.000	1,2,7,3,8,4,11	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,PHPMailer	
12	12	SwiftMailer	6030.000	1,2,7,3,8,4,12	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,SwiftMailer	
13	13	GoogleSmtplib	7482.000	1,2,7,3,8,4,11,13	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,PHPMailer,GoogleSmtplib	
14	14	YahooSmtplib	6691.000	1,2,7,3,8,4,12,14	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,SwiftMailer,YahooSmtplib	
15	15	HotmailSmtplib	7175.000	1,2,7,3,8,4,12,15	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,SwiftMailer,HotmailSmtplib	
16	16	Result	7074.000	1,2,7,3,8,4,12,14,16	IdentityMailCell,GetIPCell,GetRemoteAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,SwiftMailer,YahooSmtplib,Result	

 Query executed successfully.
 DIM-P3-5060A-R\SQLEXPRESS (... | DIM-P3-5060A-R\User (54) | Graph

Figure 5.33:

```

CREATE PROCEDURE dbo.usp_Breadth_First (@StartNode int, @EndNode int = NULL)
AS
BEGIN
SET XACT_ABORT ON
BEGIN TRAN
SET NOCOUNT ON;
CREATE TABLE #Discovered
(
    Id int NOT NULL PRIMARY KEY,    -- The Node Id
    Predecessor int NULL,          -- The node we came from to get to this node.
    OrderDiscovered int -- The order in which the nodes were discovered.
)
INSERT INTO #Discovered (Id, Predecessor, OrderDiscovered)
VALUES (@StartNode, NULL, 0)
WHILE @@ROWCOUNT > 0
BEGIN
IF @EndNode IS NOT NULL
IF EXISTS (SELECT TOP 1 1 FROM #Discovered WHERE Id = @EndNode)
BREAK
        INSERT INTO #Discovered (Id, Predecessor, OrderDiscovered)
        SELECT e.ToNode, MIN(e.FromNode), MIN(d.OrderDiscovered) + 1
        FROM #Discovered d JOIN dbo.Edge e ON d.Id = e.FromNode
        WHERE e.ToNode NOT IN (SELECT Id From #Discovered)
        GROUP BY e.ToNode
END;
        WITH BacktraceCTE(Id, Name, OrderDiscovered, Path, NamePath)
        AS
        (
SELECT n.Id, n.Name, d.OrderDiscovered, CAST(n.Id AS varchar(MAX)),
CAST(n.Name AS varchar(MAX))
FROM #Discovered d JOIN dbo.Node n ON d.Id = n.Id
WHERE d.Id = @StartNode
UNION ALL
SELECT n.Id, n.Name, d.OrderDiscovered,
        CAST(cte.Path + ',' + CAST(n.Id as varchar(10)) as varchar(MAX)),
        cte.NamePath + ',' + n.Name
FROM #Discovered d JOIN BacktraceCTE cte ON d.Predecessor = cte.Id
        JOIN dbo.Node n ON d.Id = n.Id
        )
SELECT Id, Name, OrderDiscovered, Path, NamePath FROM BacktraceCTE
WHERE Id = @EndNode OR @EndNode IS NULL
ORDER BY OrderDiscovered
DROP TABLE #Discovered
COMMIT TRAN
RETURN 0
END
GO

```

Results		Messages			
Id	Name	OrderDiscovered	Path	NamePath	
1	1	IdentityMailCell	0	1	IdentityMailCell
2	2	GetIPCell	1	1,2	IdentityMailCell,GetIPCell
3	5	GetExternalIPAddress	2	1,2,5	IdentityMailCell,GetIPCell,GetExternalIPAddress
4	6	GetRealIP	2	1,2,6	IdentityMailCell,GetIPCell,GetRealIP
5	7	GetRemoteAddress	2	1,2,7	IdentityMailCell,GetIPCell,GetRemoteAddress
6	3	GetGeoProfileCell	3	1,2,5,3	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell
7	8	GetClientInfo	4	1,2,5,3,8	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo
8	9	GetGeoInfo	4	1,2,5,3,9	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetGeoInfo
9	4	SendMailCell	5	1,2,5,3,8,4	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo,SendMailCell
10	10	MailWithLocalPHP	6	1,2,5,3,8,4,10	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,MailWithLocalPHP
11	11	PHPMailer	6	1,2,5,3,8,4,11	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,PHPMailer
12	12	SwiftMailer	6	1,2,5,3,8,4,12	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,SwiftMailer
13	13	GoogleSmtп	7	1,2,5,3,8,4,10,13	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,MailWithLocalPHP,GoogleSmtп
14	14	YahooSmtп	7	1,2,5,3,8,4,10,14	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,MailWithLocalPHP,YahooSmtп
15	15	HotmailSmtп	7	1,2,5,3,8,4,10,15	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,MailWithLocalPHP,HotmailSmtп
16	16	Result	8	1,2,5,3,8,4,10,13,16	IdentityMailCell,GetIPCell,GetExternalIPAddress,GetGeoProfileCell,GetClientInfo,SendMailCell,MailWithLocalPHP,GoogleSmtп,Result

INDEX

- Analysis, 8, 10, 15, 39, 88, 96, 110, 149, 151, 186, 187, 188, 215, 218, 223
- applications, 2, 3, 12, 13, 14, 15, 22, 24, 28, 31, 32, 34, 43, 44, 45, 47, 51, 55, 60, 86, 87, 92, 97, 99, 109, 113, 120, 124, 133, 144, 145, 160, 162, 176, 213, 233
- approach, 3, 15, 16, 19, 23, 24, 27, 45, 48, 51, 52, 54, 60, 72, 82, 83, 86, 89, 91, 92, 99, 120, 122, 139, 152, 161, 164, 166, 169, 172, 176, 185, 197, 207, 210, 211, 212, 222, 230, 232
- autonomy, 3, 12, 17, 22, 45, 59, 88, 93, 95, 99, 107, 210, 213
- availability, 3, 12, 14, 17, 18, 21, 24, 33, 34, 36, 37, 38, 42, 45, 47, 55, 59, 63, 73, 75, 82, 89, 90, 100, 105, 106, 144, 145, 190, 209, 211, 231
- AWS, 8, 35, 36
- big data, 15, 19, 37, 59, 61, 75, 85, 121, 122, 211
- Bio-Informatics, 3, 20, 210
- Cell, 8, 9, 10, 17, 22, 23, 62, 63, 69, 83, 84, 85, 86, 88, 90, 91, 92, 93, 94, 95, 96, 97, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 112, 116, 118, 119, 139, 140, 142, 143, 162, 164, 166, 168, 179, 180, 181, 182, 183, 184, 185, 190, 191, 192, 193, 194, 195, 197, 198, 200, 201, 202, 204, 207, 211, 212, 213, 223
- Client*, 8, 25, 93, 102, 167, 169
- Cloud, 2, 3, 8, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 51, 53, 55, 56, 58, 59, 60,

61, 62, 63, 72, 73, 75, 76, 78, 79, 80, 81,
82, 92, 93, 95, 98, 100, 101, 104, 119, 120,
121, 122, 134, 137, 144, 145, 147, 149,
150, 151, 158, 162, 168, 207, 208, 209,
210, 211, 212, 213, 215, 216, 217, 218,
219, 220, 221, 222, 223, 224, 225, 226,
227, 228, 229, 230, 231, 232, 233, 234,
235

Cloud Computing, 2, 3, 11, 12, 32, 34, 47,
212, 219, 224, 225, 226, 228, 229, 232,
235

Cloud platforms, 12, 34, 43, 75

Cloud problems, 3, 11, 23, 34, 75, 98, 211

complexity, 18, 26, 34, 46, 47, 56, 82, 88, 91,
94, 161, 162

components, 3, 9, 16, 22, 27, 40, 45, 46, 48,
62, 63, 67, 71, 82, 83, 84, 88, 89, 90, 91,
93, 94, 95, 97, 99, 100, 101, 102, 106, 107,
108, 111, 113, 114, 115, 118, 148, 149,
169, 183, 190, 195, 211, 212

data, 13, 14, 15, 19, 24, 25, 26, 29, 30, 34,
37, 41, 43, 46, 47, 49, 50, 51, 55, 57, 59,
75, 82, 88, 90, 96, 106, 109, 110, 111, 114,
119, 121, 122, 123, 124, 129, 135, 139,
140, 143, 144, 145, 150, 151, 161, 168,
169, 211, 213, 216, 217, 222, 223, 226,
228, 230, 231, 232, 235

distributed, 61, 62, 63, 83, 87, 89, 91, 97,
101, 120, 160

distributed compiler, 17, 19

distributed computing, 8, 13, 16, 21, 22, 48,
60, 73, 82, 83, 84, 92, 99, 101, 162, 209,
211, 212

flexibility, 34, 47, 100, 106

human body, 3, 22, 62, 63, 67, 68, 72, 74, 75,
76, 79, 80, 94, 99, 101, 210

intelligence, 2, 3, 12, 13, 17, 19, 22, 45, 46,
48, 49, 59, 60, 61, 62, 63, 64, 82, 87, 89,
93, 95, 96, 97, 99, 101, 210, 213, 217

interoperability, 16, 45, 92, 94, 100, 111, 208

multi-agent, 3, 49, 51, 61, 62, 101, 235

multidimensional database, 19, 129, 130

network, 9, 11, 19, 29, 30, 37, 41, 43, 47, 51,
58, 63, 73, 75, 85, 95, 96, 109, 110, 119,

120, 121, 122, 123, 124, 125, 127, 128,
129, 130, 132, 144, 168, 181, 200, 209,
212

objects, 26, 27, 111, 112, 123, 125

OLAP, 19, 20, 110, 152, 156, 158, 159, 160

Performance, 8, 14, 35, 38, 91, 215, 220,
223, 228

privacy, 34, 55, 59, 165, 209

process composition, 18, 61, 105

provider, 12, 29, 31, 38, 54, 62, 92, 94, 95,
96, 104, 105, 106, 107, 108, 109, 112, 113,
138, 144, 145, 147, 148, 149, 169, 177,
212

replication, 3, 14, 17, 34, 37, 55, 100, 231,
235

Reusability, 14, 39

Security, 8, 14, 18, 35, 36, 55, 90, 96, 102,
109, 119, 224, 229, 230

server, 20, 25, 31, 34, 72, 73, 87, 90, 97, 100,
107, 109, 119, 166, 168, 192, 195

Service-oriented computing, 13

smart, 3, 11, 12, 20, 22, 45, 59, 60, 63, 81,
82, 84, 85, 88, 93, 96, 97, 101, 108, 115,
143, 209, 210, 211

SmartCells, 2, 3, 8, 9, 16, 20, 22, 23, 60, 63,
66, 72, 82, 83, 86, 87, 89, 93, 95, 96, 97,
98, 99, 101, 102, 104, 105, 110, 113, 164,
166, 168, 176, 177, 178, 179, 185, 190,
197, 207, 210, 211, 212, 213

SOC, 28, 32, 36, 39, 100

technology, 3, 11, 14, 16, 21, 27, 28, 33, 42,
51, 60, 100, 101, 106, 111, 121, 147, 211,
224, 227

Validation, 15, 23, 97, 105, 142, 227, 229

web service, 3, 9, 32, 46, 57, 81, 136, 137,
139, 140, 167, 168, 215, 224